Foundations of Human-Aware Explanations for Sequential Decision-Making Problems

by

Sarath Sreedharan

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

Approved June 2022 by the
Graduate Supervisory Committee:

Subbarao Kambhampati, Chair
Been Kim
David E. Smith
Siddharth Srivastava
Yu Zhang

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

Recent breakthroughs in Artificial Intelligence (AI) have brought the dream of developing and deploying complex AI systems that can potentially transform everyday life closer to reality than ever before. However, the growing realization that there might soon be people from all walks of life using and working with these systems has also spurred a lot of interest in ensuring that AI systems can efficiently and effectively work and collaborate with their intended users. Chief among the efforts in this direction has been the pursuit of imbuing these agents with the ability to provide intuitive and useful explanations regarding their decisions and actions to end-users. In this dissertation, I will describe various works that I have done in the area of explaining sequential decision-making problems. Furthermore, I will frame the discussions of my work within a broader framework for understanding and analyzing explainable AI (XAI). My works herein tackle many of the core challenges related to explaining automated decisions to users including (1) techniques to address asymmetry in knowledge between the user and the system, (2) techniques to address asymmetry in inferential capabilities, and (3) techniques to address vocabulary mismatch. The dissertation will also describe the works I have done in generating interpretable behavior and policy summarization. I will conclude this dissertation, by using the framework of human-aware explanation as a lens to analyze and understand the current landscape of explainable planning.

ACKNOWLEDGEMENTS

This work is the result of the kindness of numerous people. More people than I could list in this small section (ASU only allows three pages of acknowledgments). Regardless, I would like to take this opportunity to thank a few people who have made a tremendous impact on me and without whom I couldn't have written any of the works listed here, let alone the dissertation itself.

First, I would like to thank my mentors. I have been truly blessed when it comes to mentors. In my relatively short research career, I have worked with some truly brilliant and generous researchers and human beings. My list of mentors has to start with my amazing advisor Prof. Subbarao Kambhampati, to whom I would forever be indebted. I literally can't imagine having a better advisor. I have gained a lot from his tutelage and this space is too short to truly express how grateful I am for his mentorship. Next, I would like to thank Dr. David Smith. Each time I had a meeting with him, I ended the meeting being more excited about the problem than when I went in. I am always amazed by his ability to cut to the heart of the problem and see the possibilities I had overlooked.

I got the chance to work with Prof. Siddharth Srivastava at a very pivotal time during my Ph.D. My current research style owes a lot to my collaborations with him. It was my collaboration with him that convinced me of the importance of clean and crisp formulations. Prof. Yu Zhang has probably had a larger impact on my research agenda than anyone else. There isn't a single work in this thesis, which can't in some sense be traced back to a conversation we have previously had. Among my committee members, the person I have had the chance to work with least has been Dr. Been Kim. However, Dr. Kim has already had an enormous impact on both my work and my way to approach explainable AI. She has inspired me to look at the larger picture,

to look beyond the limited scenarios I have been considering, and to open up myself to the potential and the challenges related to modern AI systems.

I would also like to take the chance to thank Prof. Dimitri Bertsekas and Prof. Huan Liu. While they weren't a part of my thesis committee, both Prof. Bertsekas and Prof. Liu were extremely kind to me and I am quite thankful for all the advice and the discussions I have had with them. One of my regrets about my Ph.D. is the fact that I didn't get to work directly with them on any research problems. I would like to end my list of mentors with a person whom I debated whether to include in the list of mentors or the list of my student collaborators. I ended up including Dr. Tathagata Chakraborti in the list of mentors, as in every argument we ever had, he was usually right. Many of the works listed here wouldn't have happened if he hadn't pushed me to do them. He was the easiest person I have ever worked with as he always got exactly what I had in mind, regardless of how poor my explanations were.

If there was an aspect that I would say I have been luckier at than all my mentors, it would be with my labmates. I truly believe that I have had the opportunity to work at the greatest research group ever. I honestly can't remember a day when I wasn't truly excited to go to the lab, and it was in no small part due to my labmates. I have to start my acknowledgment with Anagha, who was my constant collaborator. We have spent a lot of long nights working together on papers and robot demos. She almost made the cut for being a mentor, but she always insisted that I was senior to her since I was at yochan already when she joined. However, she is still my go-to whenever I want some sobering piece of advice, and it won't be an exaggeration to say that she probably saved my life. The next person to mention is Sailik, who was my neighbor through most of my Ph.D. and the one with who I would be gossiping all the time. Sachin is truly one of the nicest people I know and the person I always contact first whenever I am in trouble. Lydia, for all her generosity and for being like a big

sister to me. Sriram, while I didn't quite enjoy our arguments about technical stuff, our Friday evening hangout sessions were truly one of the highlights of my week, and I will miss them a lot. Yantian, who has been my roommate through the entirety of my Ph.D., is truly an inspiration to me. He is one of the most driven people I know and I always strive to be even partly as ambitious as he is with the kind of technical problems I tackle. I am also thankful to him for putting up with me as a roommate for all these years.

I believe the current batch of yochanites are some of the most talented people I know. I want to thank Zahra for always being there and for being so helpful; Karthik for always being ready to help me with all my harebrained projects; Siddhanth for putting up with me as his nosy and loud neighbor; Alberto for always being a sobering and critical voice and trying to push me to be more healthy; Mudit for always asking important questions; Lin for teaching me how to strike a good work-life balance and Utkarsh for being so reliable and for always being open to collaborating with me.

Among Yochan seniors, I would like to thank Srijith Ravikumar for vouching for me; Kartik Talamadupula for telling me about the possible project related to PISA (which Tathagata was nice enough not to complete), and Tuan Nguyen for working with me on that project. I would also like to thank my non-yochanite collaborators (in no particular order); Dr. Christian Muise, Dr. Pascal Bercher, Dr. Sarah Keren, Dr. Yara Rizk, Dr. Emil Keyder, Dr. Matthew Davis, Dr. Heni Ben-Amor, Prof. Hankz Hankui Zhuo, Dr. Michael Katz, and Dr. Yasaman Khazaeni. I would also like to acknowledge all the help I have received from friends like Pullkit Verma, Naman Shah, Lu Cheng, Midhun Madhusoodhan, Gabe Saba, Daniel D'Souza, Mansooreh Karami, Monica Dugan, and Pamela Dunn.

Finally, I would like to thank all my family, without their patience none of this would have been possible. I know it has been a long road, but it is finally over.

TABLE OF CONTENTS

CHAPTER Page

<div align="center">LIST OF TABLES</div>

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Recent years have brought about a number of groundbreaking advances in the field of Artificial Intelligence (AI). From expert Go players to self-driving cars, the field has developed systems of such impressive capabilities that their descriptions would have found a comfortable home in the science-fiction books of just the past decade. However, we are still far from reaping the true transformational potential AI systems promise to our society. Today, we see most AI systems being used effectively in two modes, either in consumer facing applications with low impact and complexity or in mission critical applications where the AI system's user is specially trained to anticipate and work with the various idiosyncrasies of the system. We are currently far from capable of fielding complex AI systems that can work with users from different walks of life and our incapability stems not necessarily from a lack of progress in some core competencies related to the tasks themselves. In fact, I would argue that as of the time of writing this thesis, some of the most daunting challenges to building such systems belong to the realm of social intelligence. The problem of creating AI systems that can effectively and fluently work with humans from different backgrounds is one we must adequately address if we are to create truly transformative AI systems.

In this thesis, I will focus on one of the prerequisite skills required for such social agents, namely the ability to effectively explain and justify its decisions to the users they are interacting with. Note that the focus here will be on identifying the content of the explanation and not mode of communication and as such we will eschew from discussions on issues like natural language generation. Even when we will touch on the question of how to communicate the information, our focus will be on the specific vocabulary to use as opposed to generating an exact natural language text or visualization.

The problem of generating explanations for the decisions derived by AI systems has recently received a lot of attention from the wider AI research community. This has led to the rapid growth of a subfield of AI popularly referred to as **explainable AI** or **XAI**, that has led to the development of a number of useful explanatory tools. However, a large number of works outside the ones discussed in this thesis, tend to adopt a very myopic view of explanation generation problems. Most of these works consider a specific explanatory challenge and try to propose solutions that reflect the authors' intuition about potentially useful explanatory information that could be provided in this context. Even with all its successes, a shortcoming of such a fragmented approach to studying explanation generation, is a lack of coherent framework to understand the overall problem. This lack of an overarching framework becomes quite apparent when one looks at popular surveys of the field that tend to focus on clustering the works according to methodological similarities than any deeper connection between the methods (cf. (Lakkaraju *et al.*, 2020)).

The primary objective of this thesis is to provide a comprehensive and cogent discussion of my contributions related to the problem of explaining solutions to sequential decision-making problems. As a secondary objective, I hope to provide a computational and philosophical foundation to the problem of explanation generation by introducing the framework of 'Human-Aware Explanations'. This framework will essentially reduce the problem of explanation generation to a multi-agent reasoning problem. Through this thesis, I will use this framework as a lens to analyze my various works that contribute to the larger problem of explanation generation. As a coda, I will also turn this lens to the larger field of explanation generation for planning problems (popularly referred to as XAIP) and use it to provide a preliminary analysis of the field as a whole.

In the rest of the chapter, we will set the stage for the rest of the thesis by describing some of the previous works done in the space of explanation generation. This is meant to be purely a quick introduction, I have provided a more exhaustive characterization of the work done within the space of explainable planning in Chapter 22. We then introduce the basic explanatory setting, along with the framework of 'Human-Aware Explanations', which we will be using throughout the rest of the thesis. Next, we describe the three salient dimensions for explanations as identified by the framework, along with how the framework related to some of the other well-established concepts in the field. Next, we will provide an overview of the overall structure of the thesis and finally end the chapter with a discussion of the impact the works covered in this thesis have had.

1.1 A Brief History of Explanation Generation in AI

Generating effective explanations, and understanding the mechanisms and methodology behind generating effective explanations have been a part of the human intellectual enterprise for a very long time. As such, the study of explanations is a very rich field that has been looked at by experts in various areas, including psychology, philosophy, and other social sciences. Some of the earliest attempts at providing a unified account of explanation go back as far as the four cause model introduced by Aristotle (Hankinson, 2001). Since then, numerous models for explanation have been introduced including Hempel's theory of explanations (Hempel and Oppenheim, 1948) and statistical models of explanations (Kelley, 1967).

On the other hand, explanation generation in AI has been mostly driven by the need to create systems that can interact and successfully collaborate with their end-users. As such the literature on AI has been more focused on methods that address specific explanatory issues related to specific systems or decision-making approaches rather than proposing and formalizing overarching theories or models for explanations.

A particular subfield of AI that identified the need for effective explanations early on was Expert Systems (Chandrasekaran *et al.*, 1989; Swartout and Moore, 1993). This is not surprising given the early popularity of the field, the drive to make practical systems for complex domains, and the inclusion of mechanisms to support some forms of uncertainty. In these cases, the explanatory systems were meant to act both as a way for the system to walk the user through the reasoning behind their decisions and even potentially help the user identify cases where the system may be making faulty decisions. Explanations for most cases take the form of presenting the user with a human-understandable representation of the trace of their internal inference process and were expected to be translated into natural language. Going through the desiderata of explanations as identified by Swartout and Moore (1993), we already see initial versions of many of the ideas we will revisit in this document, including the need to adapt to specific user's background, use of abstractions, and being able to explain the preference of one recommendation over another.

Expert systems were not the only field in AI to look at the utility of explanations. Specifically for this document, there was also early interest in generating explanations in the context of automated planning (Kambhampati, 1990). This includes techniques like generating correctness explanations and the use of derivational traces. While these were not necessarily explanations targeted at the end-user and were for most cases, additional annotations were provided along with the plans to allow for faster retrieval and reuse of these plans in novel scenarios. One interesting point to see is that you can trace back many of the currently popular explanatory methods in automated planning to some of these earlier works (cf. (Seegebarth *et al.*, 2012)). Though in general, given these explanations were meant to be generated by and for the planner, they were, as Chakraborti *et al.* (2017) puts it "*soliloquys*" with no considerations provided to the users of the system (essentially assuming that the end-user has the same model as the planner).

As the recent developments in AI and in particular, those in deep learning have brought us one step closer to realizing practical AI-powered systems, we see again the resurgence of calls to create explainable systems (Gunning, 2017). Unsurprisingly most of the current interest is focused on generating explanations for single-shot decision-making systems (Lakkaraju *et al.*, 2020) (where arguably most of the rapid developments have been made). Similar to earlier efforts in expert-system, most of the current methods in the space look at creating tools meant to address explanatory requirements for specific AI approaches. Attempts at taking stock of the field have generally focused on categorizing works in terms of methodological similarity and generally tend to silo the methods for single-shot decisions from sequential decision-making problems.

However, this increased interest in developing explainable AI systems has brought attention to the problem of generating explanations in the context of other AI problems, including planning. Explainable automated planning, popularly referred to by the moniker **XAIP**, has been in particular getting a lot of attention. The interest in explainable planning has been fueled by several factors, including the effectiveness of modern-day planners. Many of the state-of-the-art planners are capable of solving quite complex planning problems, defined over extremely large state and action spaces and now increasingly getting used in many real-world applications (cf. (Biundo *et al.*, 2021)). There is the inherent complexity of the solution concepts themselves, particularly for more complex problems, the question of what following a plan or policy may constitute may not be clear. Finally, the wide use of human-interpretable symbolic models and the use of sound and formal methods to derive decisions over them opens up the possibility that, in fact, one could derive meaningful explanations generated using these methods. Chapter 22, will be devoted specifically to providing an exhaustive characterization of various works done within this space.

The specific contributions that will be described in the thesis were also developed while keeping in mind, some of the more widely accepted properties of effective everyday explanations (cf. (Lombrozo, 2006; Miller, 2017a)). In this thesis, we will almost exclusively look at generating explanations that are *contrastive, selective, and social*. A *contrastive explanation* takes the form of answers to questions of the form "Why P and not Q?", where P is the fact being explained and Q is the foil that the explainee was expecting. Such explanations would need to help the user contrast the fact with the foil. On the other hand, a *selective explanation* is one where the explainer presents only the information relevant to the current query, and as such selectivity is closely related to the minimality of explanations. Finally, an explanation would be considered *social* if the contents are tailored to the background and beliefs of each explainee. In particular, this property speaks against the use of one-size-fits-all explanations.

1.2 Human-Aware Explanation

The basic interaction setting we will consider throughout most of this thesis is the one illustrated in Figure 1.1. Here we have an automated agent (henceforth referred to just as the robot) that is using a model $\mathcal{M}^R$ to derive its decisions. For now, we won't make any specific commitments on the representation or on the form the model takes but just that the model can be specified using a set of factors $F_{\mathcal{M}^R}$. Here the factors could include information like the state variables used to define the state space over which the model $\mathcal{M}^R$ is described. Now the robot uses a decision-making algorithm along with the model to derive a decision (henceforth referred to as plans). If Π is the space of robot plans possible for the task, then we can characterize

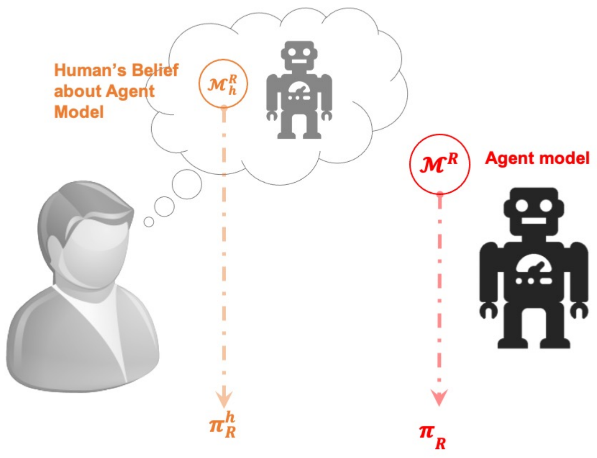


Figure 1.1: A Diagrammatic Representation of the Overall Interaction Setting We Will Be Focusing on Through the Dissertation. We Have a Robot Using Its Model $\mathcal{M}^r$ to Derive Its Decision π_r. In the Same Environment, We Might Have a Human Observer Trying to Make Sense of the Robot Behavior in Terms of Their Expectation of the Robot Model $\mathcal{M}_h^r$. The Human May Require Help Understanding the Robot Plans If Their Model and Their Inferential Process Leads to Them Expecting a Completely Different Plan. Thus Any Attempt to Explain the Current Decision Should Help Reconcile These Differences Between the Robot and the Human.

the robot's decision-making process by the total ordering ($\preceq_{\mathcal{M}^R}$) it induces over this space of plans. That is, given two plans $\pi_1, \pi_2 \in \Pi$, $\pi_1 \prec_{\mathcal{M}^R} \pi_2$ means given their current model and decision-making process, the robot would choose π_1 over π_2. In this thesis, we will mainly focus on cases where the robot makes use of a sound, complete and optimal decision-making process. Commonly, we will use π_R to denote the actual plan chosen by the robot and in general expect $\forall \pi \in \Pi,\ \pi_R \preceq_{\mathcal{M}^R} \pi$. For most planning formalisms, the ordering would be corresponding to the cost of the plan, thus $\pi \preceq_{\mathcal{M}^R} \pi'$, if $C(\pi) \leq C(\pi')$, where C is the cost function associated with the model $\mathcal{M}^R$. Note that in this thesis, we generally take an expansive view of what constitutes a model. In addition to transition dynamics, a model here could include the reward/cost function, any explicit goals, and even specific algorithmic choices that may be made by the robot.

On the other hand, we have a human who is observing the robot, and uses a model $\mathcal{M}_h^R$ to make sense of the decision being proposed by the robot. Where $\mathcal{M}_h^R$ represents the human observer's belief about the robot's model. Such mental models could also be understood in terms of the theory of mind (Premack and Woodruff, 1978) formed by the human in regards to the robot. Moreover, we expect this model to be represented by a set of factors $F_{\mathcal{M}_h^R}$. Now the human uses their own decision-making algorithm (which may be their estimation of what robot's reason process might be) that induces a total ordering $\prec_{\mathcal{M}_h^R}$ over the plan space Π (for now we assume that both the robot and the human share the same possible decision space). It may be more accurate to represent $\prec_{\mathcal{M}_h^R}$ as a partial ordering, since the humans are at best bounded rational agents (Simon, 1957). As such, given their limitations (both in terms of computational resources and deliberation time) there may be plans that they cannot realistically evaluate and thus compare. However, to simplify the discussion, we will assume a total ordering and will assume that in cases where the human can't correctly evaluate some plans, they will assume that the plans are preferred over the current one being proposed by the robot.

Now for a given plan $\pi_R \in \Pi$, the human observer would expect an explanation from the robot, if the human believes there are other possible plans that could potentially be better than the plan proposed ,i.e, as per the human $\exists \pi^{'} \in \Pi$, such that $\pi^{'} \prec_{\mathcal{M}_h^R} \pi_R$. The goal of explanations thus becomes to establish why $\pi_R \prec_{\hat{\mathcal{M}}_h^R} \pi^{'}$ holds, where $\hat{\mathcal{M}}_h^R$ could potentially be an updated human mental model. In terms of the concepts established in the philosophical works related to explanation (cf. (Hempel and Oppenheim, 1948)), this would thus correspond to the explanandum related to our explanation. Effectively, the above description boils down the goal of explanation generation to that of resolving a mismatch in expectations between the decision-maker and the one observing or receiving the decisions.

1.2.1 Other Notions of Explainability

Before we delve further into using this framework, it would be helpful to discuss how this proposed notion of explanation compares and contrasts against some of the other existing notions of explainability studied in the context of planning problems.

Explicability: To start with, let us discuss how the framework relates to explicable planning and explicability as studied in the context of interpretable behavior generation (Sreedharan *et al.*, 2022a; Chakraborti *et al.*, 2019b; Zhang *et al.*, 2017). At a high-level, one could view the problem of explicable behavior generation as that of generating plans that align with human expectations. An explicable planner would in effect try to identify plans with high explicability scores, i.e., a measure of how close the current behavior aligns with the expected behavior. In our framework, the human expectations about the robot behavior is encoded using the ordering $\prec_{\mathcal{M}_h^R}$ and it is easy to see that within our framework, a plan is said to require no explanation only if the current plan is the most preferred plan per human's expectation and as such is explicable. This relation between explicability and the framework sketched above is further illustrated by works like Sreedharan *et al.* (2021c) that have shown that existing schemes for explicability scores can be understood by considering different decision-making algorithms for the human. The explicability score of a plan can thus become a reflection of the potential relationship between the current plan and the set of minimal plans as identified by the ordering $\prec_{\mathcal{M}_h^R}$. In other words, the framework sketched above can be equivalently thought to be asserting that a plan requires explanation, if the human doesn't assign a perfect explicability score and the goal of the explanation is to ensure that the human would find the given plan perfectly explicable. In almost all of the methods, we focus solely on the fact that a plan doesn't have perfect explicability and not on the degree by which it deviates from the perfect score. This is because the deviation just tells us about how confused the human is at the robot's choice and isn't necessarily an indicator of how easy or hard it is to explain the current plan.

Contrastive Explanations: In recent years, contrastive explanations have come to represent the vast majority of works that have been done within the context of explainable planning. As discussed earlier, contrastive explanations tend to focus on explanations that contrast the fact being explained over alternate foils. Explanations, as outlined in Section 1.2, are inherently contrastive. In this setting, π^R forms the fact being explained and the foil consists of the plans humans may consider to be potentially better than the robot plan. In this thesis, we will consider both explicit contrastive explanations (where the human would explicitly raise the alternate plans they were expecting) and implicit ones, which can be best understood as answering the question 'Why did you choose plan π^R?'. In the latter category, the foil set essentially consists of all plans possible in the given domain.

Process Explanation vs Preference Explanation: Another popular categorization for explanations was the one introduced by Langley (2019), which divides the explanatory techniques into those that provide a process account and ones that provide a preferential account. Process accounts are those that effectively try to explain how the current system came up with the current decision and preference accounts are the ones that try to explain why the current decision may be preferred over alternatives. Within the latter category, the preference of the current decision may be established without necessarily discussing how the current approach reached the current decision. In the thesis, all the specific works we will discuss are going to be preference accounts. One of the central hypothesis laid out by Langley (2019) is that lay users would prefer preference accounts, while process accounts will be primarily useful to system developers. Thus the choice to focus on preference accounts, allows us to propose methods that may be of use to a larger number of users. However, it is still worth noting that the framework outlined in Section 1.2, is equally applicable to process explanation as it is for preference explanation. After all, in both cases the need for explanation arises due to a mismatch between the plan proposed by the system and what was expected by the system.

1.3 Three Dimensions of Explanations

As established earlier, the essential objective of explanation is to reconcile the asymmetry between the human's expectations about the agent with the agent's true decision-making process. Therefore, any effective explanations in this scenario should address the core cause for the divergence of human expectations from the agent's preference for the current plan. Additionally, any information we provide to the user as part of the explanation, should be presented in terms that would be easy for the human to understand.

In this framework, we will see that explanation involves acknowledging if not reconciling the inherent asymmetry between the human and the robot along three distinct dimensions. Two of these dimensions correspond to the two possible sources of mismatch in expectation between the human and the robot, namely the difference in their knowledge about the task and the differences in inferential/decision-process and capability. The third dimension focuses on the agent's ability to effectively communicate with the human the required information in terms they can understand. As we will see, I have looked at explanation generation methods that touch on all three dimensions of this framework. Specifically, the dimensions are as follows:

- Asymmetry in Task Knowledge: One of the possible reasons why the human may be confused about the agent's choice of plans may be a mismatch between the agent's knowledge of the task from the human's belief regarding it, i.e., the contents of $\mathcal{M}^R$ may not match $\mathcal{M}_h^R$. This means that even if the human were an optimal reasoner, they may find the system's decisions confusing, as the plan π^R may not be optimal in the model $\mathcal{M}_h^R$ (which means there may be a plan $\pi^{'} \prec_{\mathcal{M}_h^R} \pi^R$). To address this issue, the explanation would have to help update the human beliefs about the task, so they can correctly evaluate the plan in question. We will specifically investigate the generation of such explanations using the framework of model reconciliation.

- Asymmetry in Inferential Capabilities: Another potential source of confusion would be the difference in the actual inferential techniques and capabilities available to the human observer as compared to the agent. Specifically, even if the human's mental model about the robot was the same as the robot (i.e., $\mathcal{M}_h^R = \mathcal{M}^R$), human may not be capable of establishing the fact that $\forall \pi^{'}, \pi^R \prec_{\mathcal{M}_h^R} \pi^{'}$. My approaches to address this asymmetry, will be built around three strategies, namely,

    1. Allowing the user to raise more specific explanatory queries: One of the specific examples of this strategy would be to allow the user to specify some specific alternate plans they may be expecting. In general, an explanation as to why a given plan may be preferred over a small set of alternatives may be easier for the user to understand than an explanation for the optimality of a given plan. Additionally, more specific explanatory queries also allow us to better exploit the next two strategies.

    2. Perform model simplification: Rather than exposing the human to the full

complexity of the true system model, one could provide simpler representations of the model that suffices to respond to the specific user query. Here, one could leverage techniques such as state abstraction, local approximation, problem decomposition, etc. to form such simplified representations.

3. Provide additional explanatory information: Even with the simplified representation, the human may have a hard time verifying that the property being explained (i.e. plan π^R is preferred over some set of alternatives). One could provide additional information that demonstrates why the property holds in the model provided to the user. Examples of such *explanatory witnesses*, include information like sample execution traces of alternate plans, with information about potential points of failures and explicitly contrasting them against the execution of π^R.

- Asymmetry in Vocabulary: A prerequisite for generating explanations that align with either of the previously mentioned dimensions is the ability to convey the required information in terms the users can understand. Thus a core requirement for effective explanation is access to a shared vocabulary in which the models are or can be expressed. Barring this, we need methods that can build such shared vocabulary and be able to map the system's models in terms of concepts from this shared vocabulary. Such methods are particularly important when the agent may be using learned models or deriving its decisions from simulators that rely on complex inscrutable state representations. In particular, we will look at my efforts at using a set of concepts collected from the user and use them to identify model fragments of equivalent symbolic models that can represent the original inscrutable model used by the agent.

As sketched above, the explanation in this case involves helping the human build or improve their mental model of the system, so they can correctly evaluate the plan in question. This means a central component of the explanation, or precisely the explanan (Hempel and Oppenheim, 1948), provided involves information about the model used by the agent to derive its decisions. However this information could be simplified or translated into terms that the human understands before being presented to them. In addition to the model information, the user may also provide additional information demonstrating the fact that the underlying plan property being explained holds in the updated human mental model. We can allow this framework to also support process explanation, by expanding our notion of task knowledge to also include information about the algorithm used by the system and therefore the explanation may involve correcting any misconceptions the human may have had about the decision-making algorithm used by the system. Additionally, the explanatory witness may help walk the human through the actual inference steps used by the system.

1.4 Impact of Work

In terms of the impact that my work has had on the larger XAIP and XAI community, the explanatory frameworks I have helped develop have already been utilized by various applications and extended by many researchers in the community. In this section, to summarize this impact, I will list some of the prominent planning-based systems that directly use the explanation methods I have helped develop or other explanation generation methods that I haven't helped develop but directly build on one of my earlier works.

1. Decision-Support: The work I have done, particularly on model-reconciliation explanation has been used in multiple decision-support systems, starting with RADAR (Grover *et al.*, 2020a). Since then, the explanation generation methods I have helped developed have been utilized in multiple extensions of RADAR like

MA-RADAR (Sengupta *et al.*, 2018), iPOS (Grover *et al.*, 2020b) and RADAR-X (Valmeekam *et al.*, 2020), some of whose development I wasn't directly involved in.

2. Plan visualization: The FRESCO system (Chakraborti *et al.*, 2019a) uses a variation of model-reconciliation explanation to generate a minimal visualization for a given plan. This system also won the runner-up for the best demo award at ICAPS-2018.

3. Explanation Visualization: The VizXP (Kumar *et al.*, 2021) was a system that was developed to visualize model-reconciliation explanations.

4. Domain Authoring/Model Debugging Tool- The D3WA+(Sreedharan *et al.*, 2020b), which is also described in the thesis, uses the explanation generation methods described in Part II of this thesis. The system won the best demo award at ICAPS-2020.

5. Intelligent Tutoring System: The system MOO (Grover *et al.*, 2018) was an intelligent tutoring system that utilizes model-reconciliation explanation methods to help teach students various topics.

6. Interpretable Interface for Robots: JEDAI (Shah *et al.*, 2022) is an interpretable interface for robot control that leverages the HELM explanation generation framework.

7. Explanation for Logical Knowledge Bases: The methods presented by Vasileiou *et al.* (2021, 2022) directly extend model reconciliation methods for different logical knowledge bases.

8. Using theory-of-mind-based methods in general XAI problems: Shvo *et al.*

(2020) extends model-reconciliation and its relation to epistemic reasoning, to derive a general theory-of-mind-based framework to reason about explanations.

1.5 Thesis Outline

Figure 1.2 presents a graphical overview of the overall structure of the thesis. The bulk of the technical contents in this thesis will be divided across six parts. The first three parts will be focused on various explanation generation methods I have developed. I will be classifying the work along the primary explanatory dimension it was designed to address and each part will be focused on a specific dimension. Specifically, Part I will cover the works focused on addressing knowledge asymmetry, Part II on those focused on addressing inferential capability asymmetry and Part III will focus on the problem of addressing vocabulary mismatch. Part IV will cover my works focused on generating interpretable behaviors. In particular, the planning methods designed to take into account the overhead of explanation when generating the plan. I will also present a unified Bayesian formulation that can be used as a basis to understand all the various interpretable metrics covered within the literature. Part V will look at methods developed to effectively communicate the proposed robot decision to the human. In particular, we will look at methods to generate summaries for policies computed for stochastic shortest path problems. Finally Part VI will conclude the thesis. In this part, we will also use the human-aware explanation framework and the three dimensions as a means of analyzing the current landscape of XAIP works. Each part (except Part VI) will start with an overview chapter that provides a brief summary of the technical contributions covered in that section and any takeaways from the work, including the relationship between the specific works and wider literature on XAI. After the overview section, the chapters corresponding to specific works are organized such that every chapter will build on the previous chapter or relax some assumptions made in the earlier chapters. Outside of the chapters that are covered in these six parts, Chapter 2 will cover some of the mathematical background that will be used throughout the thesis and I will conclude the thesis in Chapter 23.

1.6 Previous Publications Covered in the Thesis

The thesis contains contents that have been previously presented in two AIJ journal articles, seven IJCAI papers, three ICAPS papers, one ICLR paper and one AAAI paper. The specific papers covered in various parts are as follows

- Part I: This part covers methods presented by Sreedharan *et al.* (2021a); Chakraborti *et al.* (2017); Sreedharan *et al.* (2018a, 2019a)
- Part II: This part covers methods presented by Sreedharan *et al.* (2021d, 2019b, 2020b)
- Part III: This part covers methods presented by Sreedharan *et al.* (2022c); Sreedharan and Kambhampati (2021) and some discussions from Kambhampati *et al.* (2022).
- Part IV: This part covers methods presented in Chakraborti *et al.* (2019f); Sreedharan *et al.* (2020a, 2021c).
- Part V: This part mainly focuses on the technique presented by Sreedharan *et al.* (2020c) and also refers to some discussion presented by Sreedharan *et al.* (2022b).
- Part VI: The landscape presented is partly based on the survey presented by Chakraborti *et al.* (2020).

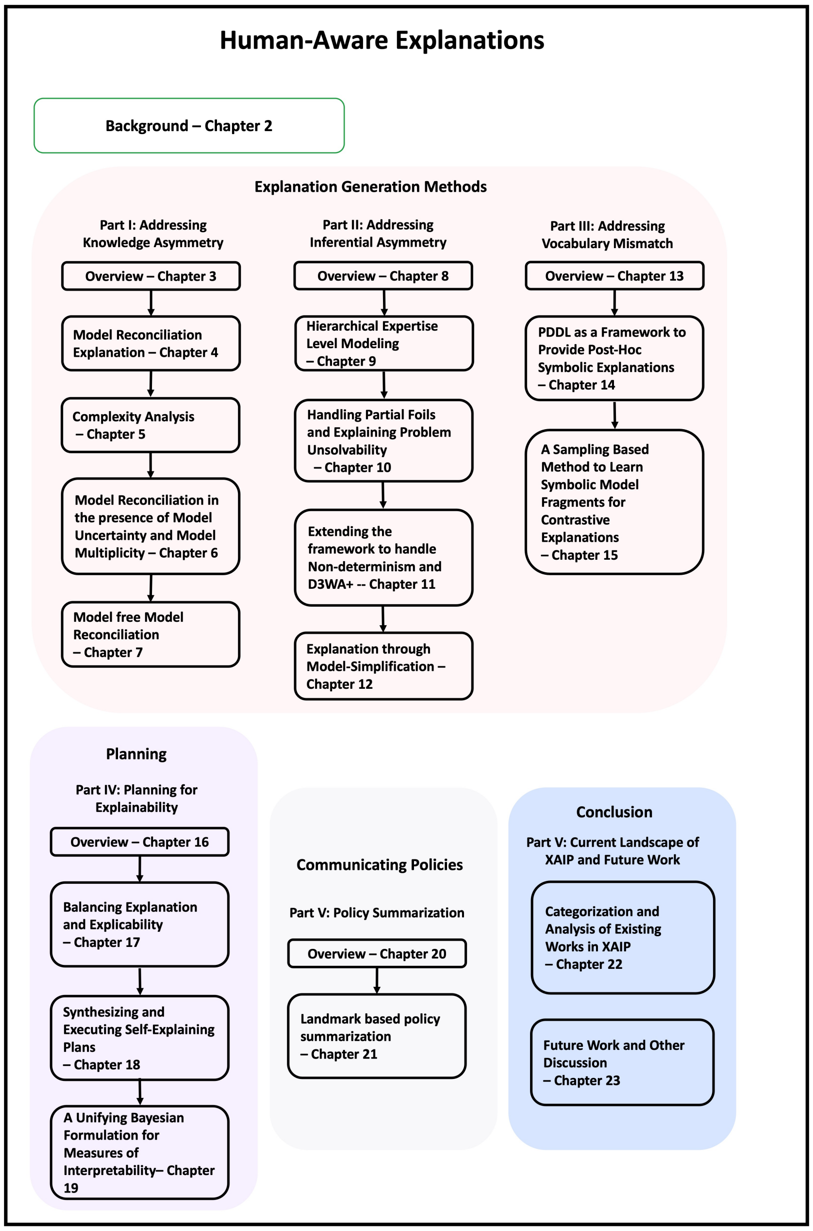

Human-Aware Explanations
Background – Chapter 2
Explanation Generation Methods
Part I: Addressing Knowledge Asymmetry
Overview – Chapter 3
Model Reconciliation Explanation – Chapter 4
Complexity Analysis – Chapter 5
Model Reconciliation in the presence of Model Uncertainty and Model Multiplicity – Chapter 6
Model free Model Reconciliation – Chapter 7
Part II: Addressing Inferential Asymmetry
Overview – Chapter 8
Hierarchical Expertise Level Modeling – Chapter 9
Handling Partial Foils and Explaining Problem Unsolvability – Chapter 10
Extending the framework to handle Non-determinism and D3WA+ -- Chapter 11
Explanation through Model-Simplification – Chapter 12
Part III: Addressing Vocabulary Mismatch
Overview – Chapter 13
PDDL as a Framework to Provide Post-Hoc Symbolic Explanations – Chapter 14
A Sampling Based Method to Learn Symbolic Model Fragments for Contrastive Explanations – Chapter 15
Planning
Part IV: Planning for Explainability
Overview – Chapter 16
Balancing Explanation and Explicability – Chapter 17
Synthesizing and Executing Self-Explaining Plans – Chapter 18
A Unifying Bayesian Formulation for Measures of Interpretability– Chapter 19
Communicating Policies
Part V: Policy Summarization
Overview – Chapter 20
Landmark based policy summarization – Chapter 21
Conclusion
Part V: Current Landscape of XAIP and Future Work
Categorization and Analysis of Existing Works in XAIP – Chapter 22
Future Work and Other Discussion – Chapter 23

Figure 1.2: A Graphical Overview of the Thesis. The Primary Contributions of the Thesis Are Distributed Across Six Parts. It Covers the Works I Have Done on Explanation Generation, Interpretable Behavior Generation, and Policy Summarization. Finally, the Human-aware Explanation Framework Is Used as a Tool to Analyze the Existing Works in the Space of XAIP.

Chapter 2

BACKGROUND

The primary goal of this chapter is to establish some of the basic notations that will be used throughout the thesis. In particular, we will focus on defining three classes of planning formalisms that the approaches discussed in this thesis rely on. This includes, (a) the fundamental goal-directed deterministic planning problems defined using STRIPS-like model definitions (Geffner and Bonet, 2013a), which will be used in the majority of works covered in the thesis, (b) fully observable non-deterministic or FOND planning problems (Bryce and Buffet, 2008), which deal with models that support qualitative non-determinism, and (c) Stochastic planning models, particularly ones specified using PPDDL-style descriptions (Younes and Littman, 2004).

2.1 Deterministic Planning Models

When we are referring to deterministic planning models, we are usually referring to models that can be mathematically represented by a tuple of the form $\mathcal{M} = \langle F, A, I, G, C \rangle$, where the elements correspond to

- F - The set of propositional fluents that defines the state space corresponding to the planning problem. We will represent the state space using the notation S, such that $S = 2^F$ and each state $s \in S$ is uniquely represented by the set of propositions that are true in that state, i.e, $s \subseteq F$.
- A - The set of actions and each action $a_i \in A$ is described by a tuple of the form $a_i = \langle pre_+(a_i), pre_-(a_i), \text{adds}(a_i), \text{dels}(a_i) \rangle$, where

  - $pre_+(a_i) \subseteq F$ are the positive preconditions for the action a_i. In general, an action is only executable if the state satisfies the positive preconditions.
  - $pre_-(a_i) \subseteq F$ are the negative preconditions for the action a_i. An action is executable in a state only if the fluents that are part of negative preconditions are false in the state.
  - $add(a_i)/del(a_i)$ correspond to the add and delete effects of the action a_i. Add effects correspond to the set of fluents set true by successful execution of the action and the deletes specify the set of fluent sets false by the action.

- I - Specifies the initial state for the planning problem.
- G - The specification of the goal, such that $G \subseteq F$. Any state $s \in S^{\mathcal{M}}$, such that $G \subseteq s$ is considered a valid goal state.
- C - The cost function for the problem. In most cases, the cost function depends only on the action and is specified as $C : A \rightarrow \mathbb{R}_{>0}$).

Note that each action $a \in A$ is also associated with a label that uniquely identifies the action. Overloading the notations a bit, we will use a to denote the specific action label for a and A to represent the set of all labels. We will use a transition function $\delta_{\mathcal{M}}$ to capture the effect of executing the plan under a given model. Specifically, $\delta_{\mathcal{M}}(s, a)$, will denote the resultant state obtained by executing the a in state s in accordance with the model $\mathcal{M}$, such that

$$\delta_{\mathcal{M}}(s,a) = \begin{cases} (s \setminus del(a)) \cup add(a), & \text{if } exe(a,s,\mathcal{M}) = true \\ undefined & \text{otherwise} \end{cases}$$

Where $exe(a,s,\mathcal{M}) = pre_+(a) \subseteq s$ and $s \cap pre_-(a) = \emptyset$

It's worth noting that the model components F, A and C are usually grouped together under the term the domain.

A (possibly empty) sequence of actions $\pi = \langle a_1, ..., a_k \rangle$ is called a *solution* (also: *plan*) if it is executable in the initial state and results into a goal state, i.e., $\delta(\pi, I, \mathcal{M}) = \delta(a_k, \delta(...(\delta(a_1, I, \mathcal{M}))...) \supseteq G$. Additionally, each action and by extension the plan can be associated with a cost. In this case, the cost of the plan $\pi = \langle a_1, ..., a_k \rangle$ will be given as $C(\pi) = \Sigma_{i=1}^{k} C(a_i)$. A plan π^* is said to be a optimal for a model $\mathcal{M}$, if

$$\nexists\ \pi' \text{ with } \delta(\pi', I, \mathcal{M}) \supseteq G, \text{ such that } C(\pi') < C(\pi^*).$$

We will use the notation $C^*_{\mathcal{M}}$ to capture the cost of an optimal plan for $\mathcal{M}$, i.e., the length of any shortest solution for $\mathcal{M}$.

Each model specification $\mathcal{M}$ corresponds to a transition system $\mathcal{T}$, where a transition system can be represented by a tuple of the form $\mathcal{T} = \langle S, L, T, s_o, S_g \rangle$, where S is the set of possible states in $\mathcal{M}$, L is the set of transition labels (corresponding to the action that induce that transition), T is the set of possible labeled transitions, s_0 is the initial state and S_g is the set of states that satisfies the goal specified by $\mathcal{M}$. We will refer to $\mathcal{T}$ to be the *safe transition* system induced by a model $\mathcal{M}$, if and only if, for any labeled transition $\langle s, a, s' \rangle \in T$, we have $exe(a, s, \mathcal{M}) = true$.

To simplify notations, in some of the chapters we will also use the logical entailment symbol ($\models$) instead of the subset relation symbol to capture the fact that a precondition may be satisfied in a given state. This will allow us to use more general preconditions than conjunctive preconditions. In fact in scenarios, where we don't want to explicitly differentiate between positive and negative preconditions, we will simply use the symbol *pre*. In such a case, *pre* could stand for some logical formula over the propositions that are specified by the model fluents.

In this thesis, there may be many works where we will need to consider multiple models and perform comparison between these models. In such scenarios, to help distinguish between the model components of different models, we will add the specific model label as a superscript over each model component. While the above description provides a basic overview of the basic deterministic planning model, many of the specific works covered in this thesis may consider more specialized or more general versions of this planning model. In such cases we will add a small discussion in the corresponding chapter that covers the specific model variant.

2.2 FOND Planning Models

We will be focusing on cases where the planning problem may be represented as a fully observable non-deterministic planning problem (FOND). Such models may be represented in declarative form using PDDL variants that use 'oneof' effects (Bryce and Buffet, 2008). Mathematically, we expect a FOND model to be represented by a tuple of the form $\mathcal{M} = \langle F, A, I, G \rangle$, where similar to Section 2.1, F is a set of propositional fluents that is used to define the state space for the planning problem ($S = 2^F$); A is the set of actions available to the agent; $I \subseteq F$ is the initial state from which the agent needs to try achieving the goal; and $G \subseteq F$ is the goal specification and any state that satisfy the goal specification (i.e., $G \subseteq s$) is considered to be a valid goal state. Each action $a \in A$ is further defined by a tuple $a = \langle pre_+(a), pre_-(a), \mathbb{E}(a) \rangle$. Where $pre_+(a)/pre_-(a)$ again corresponds to the preconditions and $\mathbb{E}(a)$ the set of possible effects. The effects are captured as $\mathbb{E}(a) = \{\langle add_1(a), del_1(a) \rangle, ..., \langle add_k(a), del_k(a) \rangle\}$ represents the set of mutually exclusive effects that could occur as the result of executing the action a and $add_i(a) \subseteq F$ and $del_i(a) \subseteq F$ correspond to the add and delete effect corresponding to the i^{th} effect. With the action definitions in place we can also define the set of transitions possible under this action definition, denoted as $\mathbb{T}$, such that we define a transition $\langle s_1, a, s_2 \rangle$ to be possible (denoted as $\langle s_1, a, s_2 \rangle \in \mathbb{T}$)

if

$$exe(a, s, \mathcal{M}) = \mathit{true} \text{ and } \exists j, \text{ such that}$$
$$\langle add_j(a), del_j(a) \rangle \in \mathbb{E}(a),\ s_2 = (s_1 \setminus del_j(a)) \cup add_j(a)$$

Where $exe(a, s, \mathcal{M})$ is defined as before. Throughout this thesis, we will focus on cases where non-determinism is considered to be fair, i.e., for every non-deterministic action every possible effect is guaranteed to occur if the action is executed infinitely often (Cimatti *et al.*, 2003). A solution to a FOND problem takes the form of a policy that maps a state to action, usually denoted by a function $\pi : S \rightarrow A \cup \{a^{\emptyset}\}$), where $a^{\emptyset}$ is an artificial empty action assigned to states that are either not supported by the policy or are goal states. A concept that will be central to the main of the techniques are traces supported by a given policy. We will refer to a state action state sequence of the form $\tau = \langle s_1, a_1, ..., s_k \rangle$ as a trace supported by a policy π if for every s_i, where $i \neq k$, we have $\pi(s_i) = a_i$, $\langle s_i, a_i, s_{i+1} \rangle \in \mathbb{T}$. A trace is said to be a goal achieving trace if $G \subseteq s_k$ and a state state s_j is said to be reachable from s_i if there exists a trace of the form $\tau = \langle s_i, a_i, ..., s_j \rangle$.

In terms of a valid policy for a FOND problem, the literature generally differentiates between weak solutions, strong and strong-cyclic solutions. Weak solutions are policies such that there exists at least one goal-achieving trace from the initial state. A policy is said to be strong-cyclic if the goal is reachable from all states reachable from the initial state. Finally, a policy is said to be a strong solution if we can again guarantee that goal is reachable from all states reachable from the initial state, but additionally, now we require that a state can never be repeated in any given goal-reaching trace.

2.3 Stochastic Planning Models

The most general stochastic planning model we will consider in this thesis would be the undiscounted MDP with absorbing goal states that tries to generate optimal solutions under total expected cost criteria. Such models can be described by a tuple of the form $\mathcal{M} = \langle S, A, P, C, I, G \rangle$, where S is the state space, A the set of possible actions, $P : S \times A \times S \rightarrow [0, 1]$ the transition probabilities, C captures the cost of executing a given action in the state, $I \in S$ the initial state and $G \subseteq S$, is the set of absorbing goal states. We will specifically limit ourselves to cases where for all state-action pairs $C(s, a) \geq 0$, that is the MDP as a whole satisfies P-assumption. MDPs satisfying the P-assumption are an extremely general formulation and covers several formalisms studied frequently in AI literature including non-negative cost infinite horizon discounted MDPs (Bertsekas, 2005), non-negative cost SSPs (Bertsekas, 2005), MAXPROB MDPs (Kolobov *et al.*, 2011), NEG Mdps (Puterman, 2014), non-negative cost GSSP's (Kolobov *et al.*, 2011), SSPUDE's (Kolobov *et al.*, 2012) etc.

We will use the functions $J : S \rightarrow \mathbb{R}$ and $Q : S \times A \rightarrow \mathbb{R}$ to capture the expected total cost and Q function and use J^* and Q^* to represent the optimal cost and Q function respectively. A policy is said to be optimal if $J^\pi(s) = J^*(s)$, where J^π is the cost function obtained by following the given policy. In the most general case, a policy takes the form $\pi = \langle \mu_1, \mu_2, ... \rangle$, where each μ_i is a mapping from state to action for a timestep i. A policy is said to be stationary if the mapping does not depend on the time step. Under P condition value iteration converges if the cost function is initialized with a bounded cost function less than the optimal cost function (say zero cost function) provided there exists a fixed point to the function. In addition to the optimal cost, a factor that we will be considering through the thesis is the probability that the execution of policy from a given state would lead it to a goal state

$P^{\pi}(s) = \sum_{g \in G} P(g|s,\pi)$ and we will $P^{*}(s)$ to denote the highest possible probability of achieving the goal under any policy for the given model (which we will refer to as the MAXPROB policy) and use $P^{\pi}(s)$ to capture probability under a given policy. In most cases, we will focus on the cost and probability of achieving the goal from the initial state and states reachable from the initial state.

Effectively in this scenario planning for MDPs becomes a multi-objective optimization problem, where the objectives become the cost of the solution and the reachability of goals. This may come across as surprising to readers who are most familiar with the more restricted classes of MDPs where any potential goals are usually compiled into the cost function. Thus one may anticipate all goal reachability queries being resolved through cost differences. More general variants of MDPs like iSSPUDE (Kolobov *et al.*, 2012) use the probability of getting to the goal as a separate optimization criterion for choosing the policy. So instead of choosing a plan that directly optimizes the cost, they use the probability of getting to the goal as primary criteria and the cost as the secondary one. This also means that in every case there exists a strict ordering between the objectives and we don't need to rely on additional considerations like establishing Pareto optimality or considering the two objectives simultaneously.

A given MDP could be represented in multiple ways, a particularly popular way one could represent such models is to describe them using problem description languages like PPDDL (Younes and Littman, 2004). Mathematically a model described in PPDDL is given by a mathematical object of the form $\mathcal{M}^{\mathcal{D}} = \langle F^{\mathcal{D}}, A^{\mathcal{D}}, I^{\mathcal{D}}, G^{\mathcal{D}}, C^{\mathcal{D}} \rangle$, where similar to previous models, $F^{\mathcal{D}}$ is the set of proposition fluents that are used to define the state space; $A^{\mathcal{D}}$ is the set of actions available in the model; $I^{\mathcal{D}}$ is the initial state and $G^{\mathcal{D}} \subseteq F^{\mathcal{D}}$ is the goal specification. Similar to Section 2.2, each $a \in A$ is further defined as $a = \langle pre_{+}(a), pre_{-}(a), \mathbb{E}(a) \rangle$. Where

$pre_{+}(a)/pre_{-}(a)$ again corresponds to the preconditions and $\mathbb{E}(a)$ the set of possible effects. $\mathbb{E}(a)$ is further defined as a set of the form $\mathbb{E}(a) = \{e_1(a), ..., e_k(a)\}$, where $e_j(a) = \langle add_j(a), del_j(a), p_j(a), c_j(a)\rangle$, where

- $add_j(a) \subseteq F$, called the add effects, is the set of facts that will be turned true by that effect in the resultant state.

- $del_j(a) \subseteq F$, called the delete effects, is the set of facts that will be turned false by that effect in the resultant state.

- $p_j(a)$, of the probability of occurrence of that particular effect (the distribution over the individual effects is expected to form a well formed probability distribution).

- $c_j(a)$ is the cost associated with the transition

Each valid description of the above form is expected to translate into an MDP of the form $\mathcal{M} = \langle S, A, I, P, C, G\rangle$, where

- S is the set of states of the model and corresponds to the state space defined by F (i.e $|S| = 2^{|F|}$). For each $i \in S$, we will use s_i to denote the symbolic state (described by the set of propositional facts that are true in the state).

- A is the action/control space of the underlying MDP and is isomorphic to the set A and will use $a^{mdp}_{a_j}$ to represent the corresponding action for the symbolic action a_j, moreover $A(i) = \{a^{mdp}_{a_j} | exe(a_j, s_i, \mathcal{M}) = true\}$ (i.e. the actions available at a state i).

- I is underlying atomic state corresponding to $I^{\mathcal{D}}$

- P is transition probability and is defined as follows

$$P(i, a^{mdp}_{a_k}, j) = p$$

if there exists an effect $e_m(a_k) \in E(a_k)$ such that $s_j = (s_i \setminus del_m(a_k)) \cup add_m(a_k)$ and $p_m(a_k) = p$, else it is 0. Note that given the assumption the effects are mutually exclusive, there exist at most one effect that can cause this transition provided there are no redundant effects (i.e., ones that leave the state unchanged because of adding existing facts or trying to remove missing facts).

- G is the set of destination or goal nodes that correspond to the ones specified by G, i.e., for $i \in G$, you have $G^{\mathcal{D}} \subseteq s_i$
- $C : S \times A \times S$ is the cost function.

For a given description $\mathcal{M}^{\mathcal{D}}$, we will use the notation $\mathcal{M}$ to refer to the MDP induced by it, especially if we don't explicitly mention the underlying MDP. Two facts that might be worth keeping in mind is (1) for any MDP with a finite set of actions and states can be captured by a description and (2) for a fixed fluent and action set, the model description for a given MDP is unique.

2.4 Variations of the Model

In this thesis, there may be many works where we will need to consider multiple models and perform comparison between these models. In such scenarios, to help distinguish between the model components of different models, we will add the specific model label as a superscript over each model component. While the above descriptions provides a basic overview of the different modeling formalism, many of the specific works covered in this thesis may consider more specialized or more general versions of these planning models. In such cases we will add a small discussion in the corresponding chapter that covers the specific model variant.

Part I

ADDRESSING KNOWLEDGE ASYMMETRY

Chapter 3

PART I OVERVIEW

In this first part of the thesis, we will focus on explanation generation techniques that were designed primarily to address the first dimension of human-aware explanation generation, namely *Asymmetry in Knowledge*. Specifically, we will introduce the framework of model-reconciliation explanation that will allow us to design explanation generation methods that will help resolve the knowledge gap between the human and the robot, so that the human can correctly evaluate the robot decision. In particular, we will consider a specific version of the model reconciliation explanation, namely minimally complete explanation and through the remaining chapters we will see how such explanations can be generated while relaxing many of the assumptions made by the original model-reconciliation framework. Most of the works in this part (except Chapter 7), will make the explicit assumption that the human is an optimal and complete planner capable of identifying the optimal plan corresponding to their current mental model about the task.

3.1 Structure for Part I and Technical Contributions

Part I will be divided into four chapters

1. Chapter 4: In this chapter we will introduce the basic model-reconciliation framework and define various types of model-reconciliation explanations. In particular, we will focus on two types, namely, Minimally Complete Explanations and Minimally Monotonic Explanations and will propose model-space search based algorithms to generate such explanations. This chapter will also present

the basic user study we ran to verify the effectiveness of the model-reconciliation explanations. Additionally, we will also briefly discuss some approximations of the minimally complete explanations.

2. Chapter 5: In this chapter, we will consider the computational complexity of generating model-reconciliation explanations. In particular, we will establish that the complexity of solving the decision version of the minimally complete explanation generation is Σ_2^P-Complete. As part of establishing this complexity, we will also present a quantified boolean formula compilation that can generate minimally complete explanations.

3. Chapter 6 This chapter will see us revising the problem of generating minimally complete explanations and trying to relax one of the central assumptions made by the original formulation, namely the access to the human's mental model $\mathcal{M}_h^R$. We will consider a case, where the robot has access to an incomplete representation of the human mental model, which effectively corresponds to a set of possible models. In this scenario, we will mainly focus on the problem of generating two forms of explanations that can deal effectively with such model uncertainty, namely conformant and contingent explanations.

4. Chapter 7: In this chapter, we will go one step further and assume the robot has no knowledge of the human mental model. Instead the method will look at the possibility of learning proxy functions, like labeling functions that would suffice to generate explanations. This chapter will also see us extending model-reconciliation to support MDPs.

3.2 Important Takeaways

One of the facts the reader will notice as they go through the various methods described in this thesis (and not just this part) is the fact that information about the robot model forms the core of every type of explanation discussed. The information may be transformed (part II) or translated (part III), but the explanation still contains model information. This part focuses purely on the question of identifying such information, while making simplifying assumptions. As such, this part of the thesis acts as the foundation for the rest of the explanatory methods discussed in the other parts. One of the connections we will briefly touch on in this part and develop much further in Part IV is the relationship between reasoning about model-reconciliation explanations and epistemic reasoning. In fact, methods like those presented by Shvo *et al.* (2020) have effectively extended the idea of model-reconciliation far beyond the context of planning and have considered such methods as a way to do model explanation generation in general. Additionally, looking at many of the popular methods of explanation generation in general XAI, one can see parallels to model-reconciliation explanations discussed in this part. For example, consider the popular feature attribution explanation techniques (Lakkaraju *et al.*, 2020). In this work, the goal is to convey a simplified model of how the various models influence the final decision as captured by the decision-making agent. However, one must note that most works in XAI don't just limit themselves to addressing knowledge asymmetry but they also implicitly try to address other dimensions, such as vocabulary mismatch and inferential capability asymmetry.

Chapter 4

MODEL RECONCILIATION EXPLANATION

In this chapter, we will introduce the foundational framework we will use to generate explanations that address knowledge asymmetry between the human and the robot. Throughout this chapter, we will assume that the human is a perfect reasoner and capable of identifying the optimal plan for a given plan. We will define the framework over deterministic planning models described using PDDL (Section 2.1). In this case, the human plan preferences ($\prec_{\mathcal{M}_h^R}$) are defined based on the plan cost, thus human would consider $\pi \preceq_{\mathcal{M}_h^R} \pi'$, if $C^{\mathcal{M}_h^R}(\pi) \leq C^{\mathcal{M}_h^R}(\pi')$, where $C^{\mathcal{M}_h^R}$ is the cost function that is part of $\mathcal{M}_h^R$. Model reconciliation framework, will allow us to update the human's expectation about the plan π_R^* by providing information about the robot model $\mathcal{M}^R$ that was previously missing from $\mathcal{M}_h^R$. Under this framework, we will consider various types of explanation and in particular look at Minimally Complete Explanations or MCE, which as we discussed we will further investigate through the other chapters in this part of the thesis.

We will start the chapter by discussing a motivating example that will not only allow us to provide an intuitive example for model reconciliation explanations, but will also act as a running example we will use to define the various components of the framework. Then we will delve into the formal definition of the various model-reconciliation explanations, the algorithms for generating the explanations, followed by an evaluation. The evaluation will include both a computational evaluation of the explanation generation algorithms and a user study to establish the effectiveness of the explanation types.

4.1 Running Example: The Fetch Domain

Consider the Fetch robot whose design requires it to `tuck` its arms and lower its torso or `crouch` before moving. This is not obvious to a human navigating it and it may lead to an unbalanced base and toppling of the robot if the human deems such actions as unnecessary. The move action for the robot is described in PDDL (Aeronautiques *et al.*, 1998) in the following model snippet –

```
(:action move
:parameters     (?from ?to - location)
:precondition   (and (robot-at ?from) (hand-tucked) (crouched))
:effect         (and (robot-at ?to) (not (robot-at ?from))))

(:action tuck
:parameters     ()
:precondition   ()
:effect         (and (hand-tucked) (crouched)))

(:action crouch
:parameters     ()
:precondition   ()
:effect         (and (crouched)))
```

Notice that the `tuck` action also involves a lowering of torso so that the arm can rest on the base once it is tucked in.[1] Now, consider a planning problem where the the robot needs to transport a block from one location to another, with the following initial and goal states –

[1]Fetch User Manual: `https://docs.fetchrobotics.com/`

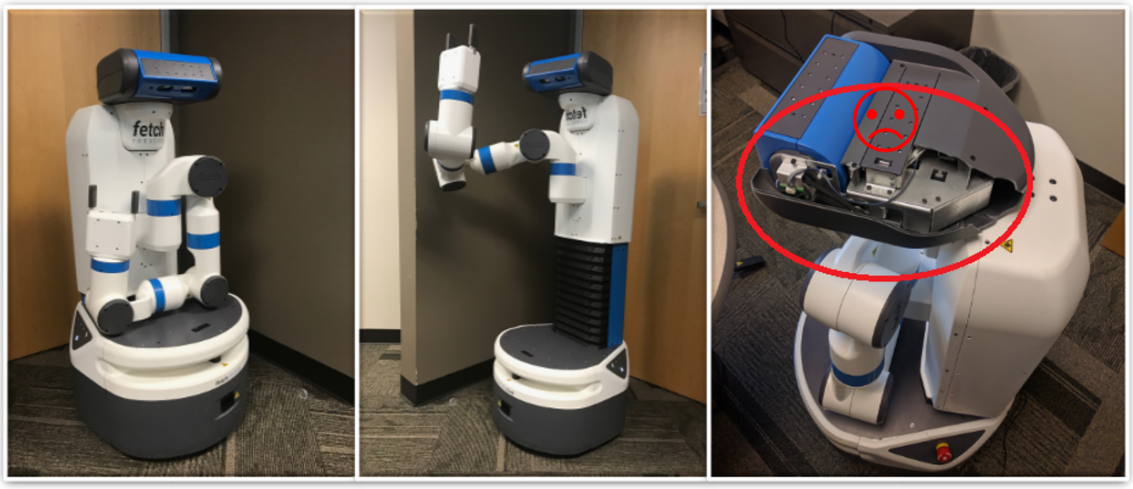


Figure 4.1: The Fetch in the Crouched Position with Arm Tucked (Left), Torso Raised and Arm Outstretched (Middle) and the Rather Tragic Consequences of a Mistaken Action Model (Right Showing a Fractured Head from an Accident).

```
(:init (block-at b1 loc1) (robot-at loc1) (hand-empty))
(:goal (and (block-at b1 loc2)))
```

An optimal plan for the robot involves a `tuck` action followed by a `move`:

```
pick-up b1 -> tuck -> move loc1 loc2 -> put-down b1
```

The human, on the other hand, expects a much simpler model, as shown below.[2] In the human's model of the robot, `move` action does not have the preconditions for tucking the arm and lowering the torso, and `tuck` does not automatically lower the torso either. This means the behavior expected by the human may not match what is generated by the robot.

[2]This is actually a common problem with deploying any software to end users: generic user models are used to model the average user and these lack details and nuances of the system at hand that only experts would be aware of.

```
(:action move
:parameters     (?from ?to - location)
:precondition   (and (robot-at ?from)
:effect         (and (robot-at ?to) (not (robot-at ?from))))

(:action tuck
:parameters     ()
:precondition   ()
:effect         (and (hand-tucked))

(:action crouch
:parameters     ()
:precondition   ()
:effect         (and (crouched)))
```

The original plan is no longer optimal to the human who can envisage better alternatives (a shorter plan without the extra `tuck` action) in their mental model. An explanation here is a model update that can address this disagreement.

```
Explanation >> MOVE_LOC1_LOC2-has-precondition-HAND-TUCKED
```

This correction brings the mental model (i.e. the model the human believes is being used by the robot) closer to the robot's ground truth and is necessary and sufficient to make the robot's plan optimal in the resultant domain so that the human cannot envisage any better alternatives. This process of selective update of human mental model to clarify the status of the current plan forms the essence of the model reconciliation process.

4.2 Explanation as Model Reconciliation

The model reconciliation framework introduces the *mental model* of the human in the loop into a planner's deliberative process, in addition to the planner's own model in the classical sense. As described previously, even if the robot is doing the best it can, a plan π^*_R that is optimal in the robot's model may not be optimal in the human mental model and thus *inexplicable* from the point of view of the human – this means that the human can come up with "better solutions" (in their mental model) to the planning problem at hand. The explanation process thus begins with the following question:

Q_1: Why plan π^*_R?

As discussed in Chapter 1, the goal of explanation here is to ensure that plan π^*_R that both the explainer and the explainee agree that this is the best decision that could have been made in the given problem. In this framework, we will achieve by providing model artifacts to the explainee so that π^*_R is now also optimal in the updated mental model (we will refer to this as the completeness property later).

Before we can formally define a model reconciliation explanation problem, we need to define a model parameterization function.

Definition 1. *Given a set of propositions F and a set of action names A, let $\mathbb{M}^{(F,A)}$ be the **space of models** that can be defined over F and A, i.e., $\forall \mathcal{M} \in \mathbb{M}^{(F,A)}$ there exist C, I, and G, such that $\mathcal{M} = \langle F, A, I, G, C \rangle$ is a planning model.*

Now each model from a given model space can be uniquely identified by a so-called model parameterization function.

Definition 2. *The **model parameterization** function $\Gamma : \mathbb{M}^{(F,A)} \rightarrow 2^{\mathcal{F}^{(F,A)}}$ for a given space of models $\mathbb{M}^{(F,A)}$, maps a model from $\mathbb{M}^{(F,A)}$ to a subset of propositions*

$\mathcal{F}^{(F,A)}$ *(henceforth referred to as model parameters), where*

$$\begin{aligned}\mathcal{F}^{(F,A)} = \; & \{\textit{init-has-f} \mid f \in F\} \cup \{\textit{goal-has-f} \mid f \in F\} \cup \\ & \bigcup_{a \in A} \{\textit{a-has-pos-prec-f}, \textit{a-has-neg-prec-f}, \\ & \qquad \textit{a-has-add-f}, \textit{a-has-del-f} \mid f \in F\}.\end{aligned}$$

For a model $\mathcal{M} = \langle F^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^{\mathcal{M}} \rangle$, the parameterization function $\Gamma(\mathcal{M})$ is defined by

$$\begin{aligned}\tau_I^{\mathcal{M}} &= \{\textit{init-has-f} \mid f \in I^{\mathcal{M}}\} \\ \tau_G^{\mathcal{M}} &= \{\textit{goal-has-g} \mid g \in G^{\mathcal{M}}\} \\ \tau_{\textit{pre}_+(a)}^{\mathcal{M}} &= \{\textit{a-has-pos-prec-f} \mid f \in \textit{pre}_+^{\mathcal{M}}(a)\} \\ \tau_{\textit{pre}_-(a)}^{\mathcal{M}} &= \{\textit{a-has-neg-prec-f} \mid f \in \textit{pre}_-^{\mathcal{M}}(a)\} \\ \tau_{\textit{add}(a)}^{\mathcal{M}} &= \{\textit{a-has-add-f} \mid f \in \textit{add}^{\mathcal{M}}(a)\} \\ \tau_{\textit{del}(a)}^{\mathcal{M}} &= \{\textit{a-has-del-f} \mid f \in \textit{del}^{\mathcal{M}}(a)\} \\ \tau_a^{\mathcal{M}} &= \tau_{\textit{pre}_+(a)}^{\mathcal{M}} \cup \tau_{\textit{pre}_-(a)}^{\mathcal{M}} \cup \tau_{\textit{add}(a)}^{\mathcal{M}} \cup \tau_{\textit{del}(a)}^{\mathcal{M}} \\ \tau_A^{\mathcal{M}} &= \bigcup_{a \in A^{\mathcal{M}}} \tau_a^{\mathcal{M}} \\ \Gamma(\mathcal{M}) &= \tau_I^{\mathcal{M}} \cup \tau_G^{\mathcal{M}} \cup \tau_A^{\mathcal{M}}\end{aligned}$$

Note that $\Gamma : \mathbb{M}^{(F,A)} \to 2^{\mathcal{F}^{(F,A)}}$ is a bijective mapping and we will use the function Γ^{-1} to identify the model in $\mathbb{M}^{(F,A)}$, corresponding to a specific subset of $\mathcal{F}^{(F,A)}$.

Definition 3. *A **model reconciliation explanation problem** is defined by the tuple $\mathcal{P}^{MRE} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \pi_R^* \rangle$, where $\mathcal{M}^R = \langle F^{\mathcal{M}^R}, A^{\mathcal{M}^R}, I^{\mathcal{M}^R}, G^{\mathcal{M}^R}, C^{\mathcal{M}^R} \rangle$ is a model that the robot is using in its decision-making; $\mathcal{M}_h^R = \langle F^{\mathcal{M}_h^R}, A^{\mathcal{M}_h^R}, I^{\mathcal{M}_h^R}, G^{\mathcal{M}_h^R}, C^{\mathcal{M}_h^R} \rangle$ is the model the human observer is associating with the robot; and π_R^* is the robot's plan to be explained. We will require that the models share the same fluents and action names $F^{\mathcal{M}^R} = F^{\mathcal{M}_h^R}$, $A^{\mathcal{M}^R} = A^{\mathcal{M}_h^R}$, and that π_R^* is optimal in the model $\mathcal{M}^R$.*

Note that the requirement of using identical action names and fluents is not a restriction. Using the same action name set is a canonical requirement as we assume that the only confusion that might exist is due to misaligned action definitions, i.e., the human observer might not have a perfect understanding of an action's preconditions and effects, but does know which action is being observed. Requiring identical fluent sets is just for convenience, but not a restriction either, since we can always define this shared/identical fluent set as the union of the individual ones in case they are different. To simplify the discussion, we will focus on models with positive precondition and unit cost in this chapter, however all the methods discussed in this chapter is easily extensible to more general model formalisms.

It is important to note here that $\mathcal{M}_h^R$ is the robot's approximation of the *information content* of the mental model – there is, of course, no PDDL inside the human's head (c.f. previous example in Section 4.1). This mental model here is just a copy of the robot's own decision making problem (here, a planning problem but it can be any other model of decision making or even a graph) that the robot believes is held by the human. This model is thus a generative model of user expectations of the robot. Also note, these two models could pretty much differ along any aspects, including the initial state, goal, action definitions (including cost) and even fluents used. For notational convenience, we will assume there is a one-to-one correspondence between actions in the models $\mathcal{M}^R$ and $\mathcal{M}_h^R$. Though, we can easily use this to capture cases where one model have actions absent from the other, by assuming the other model has a dummy version the same action with unachievable preconditions.

Having defined the problem definition formally, we still need to say what a solution to it is. Solutions to model reconciliation explanation problems are called *explanations*, which in turn are defined based on *model updates*, which we define next. A model update updates the *human's* model $\mathcal{M}_h^R$ to make it align with the actual model, i.e., the one of the robot, $\mathcal{M}^R$. Formally, such updates are defined as follows:

Definition 4. *For a given model reconciliation problem* $\mathcal{P}^{MRE} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \pi_R^* \rangle$, *a* ***model update*** *is given by a tuple* $\mathcal{E} = \langle \epsilon^+, \epsilon^- \rangle$, *such that* $\epsilon^+ \subseteq \Gamma(\mathcal{M}^R) \setminus \Gamma(\mathcal{M}_h^R)$ *and* $\epsilon^- \subseteq \Gamma(\mathcal{M}_h^R) \setminus \Gamma(\mathcal{M}^R)$. *We will refer to the model* $\mathcal{M}_h^R + \mathcal{E} = \Gamma^{-1}((\Gamma(\mathcal{M}_h^R) \setminus \epsilon^-) \cup \epsilon^+)$ *as the updated human model that results from applying* $\mathcal{E}$.

We can now define solutions for model reconciliation explanation problems. Note that we don't require a "complete" set of changes to the human model making it *identical* to the actual one. It suffices to "explain" the robot's plan, i.e., so that this plan becomes optimal in the human's model.

Definition 5. *For a given model reconciliation explanation problem* $\mathcal{P}^{MRE}$, *Where* $\mathcal{P}^{MRE}$ *is defined by the tuple*$\langle \mathcal{M}^R$,$\mathcal{M}_h^R, \pi_R^* \rangle$, *a model update* $\mathcal{E} = \langle \epsilon^+, \epsilon^- \rangle$ *is considered to be* ***a valid explanation*** *if the plan* π_R^* *is an optimal plan in* $\mathcal{M}_h^R + \mathcal{E}$.

The above question can also be posed in the following, more explicit contrastive form:

Q_2: *Why not a different plan* $\hat{\pi}$?

Here $\hat{\pi}$ are the possible foils. As before, the purpose of an explanation is to negate the foil so that both the human and the robot can come to the same page with regards to the decision that it has made, i.e. the explainee agrees that π is better than $\hat{\pi}$.

(4) $\delta(\widehat{I}_h^R, \pi, \widehat{\mathcal{M}}_h^R) \models \widehat{G}_h^R \ \wedge \ C^{\widehat{\mathcal{M}}_h^R}(\pi_R^*) < C^{\widehat{\mathcal{M}}_h^R}(\hat{\pi})$.[3]

[3]Note that the "closeness" or distance to an expected plan is modeled here in terms of cost optimality, but in general this can be any preference metric like plan similarity as investigated in existing literature on explicable planning (Zhang *et al.*, 2017, 2016; Kulkarni *et al.*, 2019a) and plan similarity. (Srivastava *et al.*, 2007; Nguyen *et al.*, 2012) This does not effect the algorithms described here, since the computation of similarity is only invoked during the evaluation process of a particular node and the stopping criterion of the search, rather than the search process itself.

Note that Q_1, in essence, involves an implicit quantifier over all possible foils, as handled by Condition (3). This is thus a more conservative target and subsumes the case of the explicit foil.

Note that we only consider cases where the robot is explaining a decision it has made with respect to its model – the robot model need not be the ground truth. However, the robot can only explain with respect to what it believes to be true. The grander scope of explanatory dialogue may involve cases where it is wrong in its understanding of the model of the world (or is suboptimal) and thus needs to update its own model (or plan, respectively) iteratively in the course of further explanatory dialogue with the human in the loop, for example, in a decision support setting. (Smith, 2012; Grover *et al.*, 2020a)

4.2.1 Model Space Search

We can now define a *model-space search problem* $\langle\langle\mathcal{F},\Lambda\rangle,\Gamma(\mathcal{M}_1),\Gamma(\mathcal{M}_2)\rangle$ (we will use $\mathcal{F}$ as a stand-in for $\mathcal{F}^{(F,A)}$ where F and A are shared between $\mathcal{M}_1$ and $\mathcal{M}_2$). Λ corresponds to the action set containing unit model change actions $\lambda : \mathcal{F} \rightarrow \mathcal{F}$. The new transition or edit function is given by $\delta_{\mathcal{M}_1,\mathcal{M}_2}(s_1,\lambda) = s_2$ such that `condition a` : $s_2 \setminus s_1 \subseteq \Gamma(\mathcal{M}_2)$, `condition b` : $s_1 \setminus s_2 \not\subseteq \Gamma(\mathcal{M}_2)$ and `condition c` $|s_1 \Delta s_2| = 1$ (Δ being the symmetric difference) are satisfied. This means that model change actions can only make a single change to a model at a time (starting from $\mathcal{M}_1$), and *all these changes are consistent with the target model* $\mathcal{M}_2$. The solution to a model-space search problem is given by a *set* of edit functions $\{\lambda_i\}$ that transforms the model $\mathcal{M}_1$ to $\mathcal{M}_2$, i.e. $\delta_{\mathcal{M}_1,\mathcal{M}_2}(\Gamma(\mathcal{M}_1),\{\lambda_i\}) = \Gamma(\mathcal{M}_2)$. An explanation can thus be cast as a solution to the model-space search problem $\langle\langle\mathcal{F},\Lambda\rangle,\Gamma(\mathcal{M}_h^R),\Gamma(\widehat{\mathcal{M}})\rangle$ with the transition function $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}$ such that Condition (3) above is preserved.[4]

4.2.2 Types of Explanations

Before we go on to different types of explanations, we consider the following requirements that characterize explanations in this multi-model setting.

R1. **Completeness -** Explanations of a plan should allow them to be compared and contrasted against other alternatives, so that no better solution exists. We enforce this property by requiring that in the updated human mental model the plan being explained is now optimal.

– *An explanation is complete iff* $C^{\widehat{\mathcal{M}}_h^R}(\pi) = C^*_{\widehat{\mathcal{M}}_h^R}$.

R2. **Conciseness -** Explanation should be concise so that they are easily understandable to the explainee. The larger the explanation, the harder it is for the human to process that information. Thus the length of the explanation, i.e. the number of edits made as part of the explanation ($|\Gamma(\mathcal{M}_h^R)\Delta\Gamma(\widehat{\mathcal{M}}_h^R)|$), serves as a useful proxy or first approximation for the complexity of an explanation.

R3. **Monotonicity -** This ensures that remaining model differences cannot change the completeness of an explanation, i.e. all aspects of the model that engendered the plan have been reconciled. Thus, monotonicity of an explanation subsumes completeness and requires more detail.

[4]Notice that we insisted that explanations must be compatible with the planner's model ($\mathcal{M}_2$ in the above definition). If this requirement is relaxed, it allows the planner to generate "explanations" that it knows are not true, and thus deceive the human. (Chakraborti and Kambhampati, 2019a) While endowing the planner with such abilities may warrant significant ethical concerns, we note that the notion of white lies, and especially the relationship between explanations, excuses and lies has received very little attention and affords a rich set of exciting research problems. (Isaac and Bridewell, 2017; Chakraborti and Kambhampati, 2019b)

Explanation Type	**R1**	**R2**	**R3**	**R4**
Plan Patch Explanation / VAL	✗	✓	✗	✓
Model Patch Explanation	✓	✗	✓	✓
Minimally Complete Explanation	✓	✓	✗	?
Minimally Monotonic Explanation	✓	✓	✓	?
(Approximate) Minimally Complete Explanation	✗	✓	✗	✓

Table 4.1: Requirements for Different Types of Explanations.

– *An explanation is monotonic iff*

$$C^{\hat{\mathcal{M}}}(\pi^*_R) = C^*_{\hat{\mathcal{M}}} \ \forall \hat{\mathcal{M}} : \Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R_h) \subset \Gamma(\hat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R_h).$$

That is no additional information revealed about the model should cause the human to question the validity of previous explanations. This is a very useful property to have. Doctors, for example, reveal different amount of detail to their patients as opposed to their peers, and often need to maintain monotonicity and resolve conflict of information (Palmieri and Stern, 2009) during the course of treatment. Further, the idea of completeness, i.e. withholding information on other model changes as long as they explain the observed plan, is also quite prevalent in how we deal with similar scenarios ourselves - e.g. progressing from Newtonian physics in high school to Einsteins Laws of Relativity in college.

R4. **Computability -** While conciseness deals with how easy it is for the explainee to understand an explanation, computability measures the ease of computing the explanation from the point of view of the planner.

We will now introduce different kinds of multi-model explanations that can participate in the model reconciliation process, propose algorithms to compute them, and compare and contrast their respective properties. We note that the requirements outlined above are in fact often at odds with each other - an explanation that is very easy to compute may be very hard to comprehend. This (as seen in Table 4.1) will become clearer in course of this discussion.

A simple way to explain would be to provide the model differences pertaining to only the actions that are present in the plan being explained –

Definition 6. ***A Plan Patch Explanation (PPE)*** *is given by –*

$$\mathcal{E}^{PPE} = \Delta_{\{\mathcal{M}^R, \mathcal{M}_h^R\}} \bigcup_{f \in pre_+^{\mathcal{M}}(a) \cup add^{\mathcal{M}}(a) \cup del^{\mathcal{M}}(a) : a \in \pi_R^*} \tau(f)$$

Clearly, such an explanation is easy to compute and concise by focusing only on plan being explained. However, it may also contain information that need not have been revealed, while at the same time ignoring model differences elsewhere in $\mathcal{M}_h^R$ that could have contributed to the plan being suboptimal in it. Thus, it is incomplete.

One could adapt VAL (Fox *et al.*, 2005; Howey *et al.*, 2004), to the multi-model setting to generate a version of PPE. VAL is plan validation tool which can simulate the execution of a plan in a given model. A multi-model VAL would need to extend this simulation to multiple models and compare and contrast the differing results of execution in the different models. Unfortunately, this would still suffer from the same limitations mentioned above. On the other hand, an easy way to compute a complete explanation would be to provide the entire model difference to the human –

Definition 7. ***A Model Patch Explanation (MPE)*** *is given by –*

$$\mathcal{E}^{MPE} = \Gamma(\mathcal{M}^R) \Delta \Gamma(\mathcal{M}_h^R)$$

This is also easy to compute but can be quite large and is hence far from being concise. Thus, in the following, we will try to minimize the size (and hence increase the comprehensibility) of explanations by searching in the space of models and thereby not exposing information that is not relevant to the plan being explained while still trying to satisfy as many requirements as we can.

Definition 8. ***A Minimally Complete Explanation (MCE)*** *is the shortest possible explanation that is complete –*

$$\mathcal{E}^{MCE} = \arg\min_{\mathcal{E}} |\Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}_h^R)| \textit{ with R1}$$

The explanation provided before in the Fetch domain, are indeed the smallest domain changes that may be made to make the given plan optimal in the updated action model, and is thus an example of a minimally complete explanation.

The optimality criterion happens to be relevant to both the cases where the human expectation is better, or worse, than the plan computed by the planner. This might be counter to intuition, since in the latter case one might expect that just establishing feasibility of a better plan would be enough. Unfortunately, this is not the case, as can be easily seen by creating counter-examples where other faulty parts of the human model might disprove the optimality of the plan.

Proposition 1. *If $C^{\mathcal{M}_h^R}(\pi_R^*) < \min_\pi C^{\mathcal{M}_h^R}(\pi)$, then ensuring feasibility of the plan in the modified planning problem, i.e. $\delta_{\widehat{\mathcal{M}}}(\widehat{I}, \pi_R^*) \models \widehat{G}$, is a necessary but* not *a sufficient condition for $\widehat{\mathcal{M}} = \langle \widehat{F}, \widehat{A}, \widehat{I}, \widehat{G} \rangle$ to yield a valid explanation.*

Note that a minimally complete explanation can be rendered invalid given further updates to the model. This can be easily demonstrated in our running example in the Fetch domain. Imagine that if, at some point, the human were to find out that the action move also has a precondition `(crouched)`, then the previous robot plan will no longer make sense to the human since now, according to the human's faulty model (being unaware that the tucking action also lowers the robot's torso) the robot would need to do *both* `tuck` and `crouch` actions before moving. Consider the following explanation in the Fetch domain instead –

```
Explanation >> TUCK-has-add-effect-CROUCHED
Explanation >> MOVE_LOC2_LOC1-has-precondition-CROUCHED
```

This explanation does not reveal all model differences but at the same time ensures that the plan remains optimal for this problem, irrespective of any other changes to the model, by accounting for all the relevant parts of the model that engendered the plan. It is also the smallest possible among all such explanations.

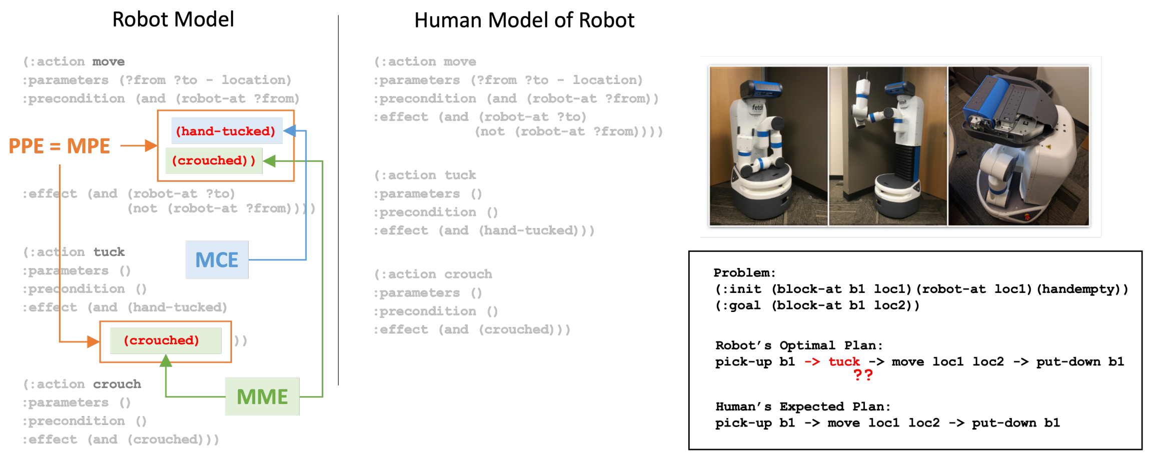
Definition 9. ***A Minimally Monotonic Explanation (MME)*** *is the shortest explanation that preserves both completeness and monotonicity –*

$$\mathcal{E}^{MME} = \arg\min_{\mathcal{E}} |\Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}_h^R)| \textit{ with R1 \& R3}$$

An MCE or MME solution may not be unique to an MRP problem. This can happen when there are multiple model differences supporting the same causal links in the plan - a minimal explanation can get by (i.e. guarantee optimality in the modified model) by only exposing one of them to the human. Interestingly, it was showed in (Zahedi *et al.*, 2019) how theoretically equivalent explanations are, in fact, sometimes interpreted differently by the explainee. The results from that study indicated a preference for explanations related to the effects of actions.

Proposition 2. *MCEs and MMEs are not unique, i.e. there might be multiple minimally complete and monotonic solutions to a given MRP.*

Even though MCEs are an abridged version of an MME, it is easy to see that an MCE may not necessarily be part of an actual MME. This is due to the non-uniqueness property of MCEs and MMEs. This is illustrated in Figure 4.2.



```
(:action move
:parameters (?from ?to - location)
:precondition (and (robot-at ?from)
                   (hand-tucked)
                   (crouched))
:effect (and (robot-at ?to)
             (not (robot-at ?from))))

(:action tuck
:parameters ()
:precondition ()
:effect (and (hand-tucked)
             (crouched)))

(:action crouch
:parameters ()
:precondition ()
:effect (and (crouched)))
```

```
(:action move
:parameters (?from ?to - location)
:precondition (and (robot-at ?from))
:effect (and (robot-at ?to)
             (not (robot-at ?from))))

(:action tuck
:parameters ()
:precondition ()
:effect (and (hand-tucked)))

(:action crouch
:parameters ()
:precondition ()
:effect (and (crouched)))
```

```
Problem:
(:init (block-at b1 loc1)(robot-at loc1)(handempty))
(:goal (block-at b1 loc2))

Robot's Optimal Plan:
pick-up b1 -> tuck -> move loc1 loc2 -> put-down b1
                  ??

Human's Expected Plan:
pick-up b1 -> move loc1 loc2 -> put-down b1
```

Figure 4.2: Illustration of the Different Kinds of Explanations in the Fetch Domain. Here the PPE and MPE Are Equivalent and Involves Notifying the Human about Both Missing Preconditions of the Move Action and the Missing Effect for the Tuck Action (Which Is the Worst Case for the Former) and Both Longer than the MCE or the MME. Also, the MCE (Which Involves Just Notifying the Human That There Hand Being Tucked Is a Precondition for Move Action) is Shorter than, and Interestingly Not a Subset of, the Mme.

Proposition 3. *An MCE may not be a subset of an MME, but it is always smaller or equal in size, i.e.* $|\mathcal{E}^{MCE}| \leq |\mathcal{E}^{MME}|$.

4.2.3 Model Space Search for Minimal Explanations

In the following, we will see how the state space designed in Section 4.2.1 can be used in model-space search for computing MCEs and MMEs (computation of PPE and MPE follows directly from $\mathcal{M}^R$, $\mathcal{M}^R_h$ and π^*_R).

Model Space Search for MCEs

To compute MCEs, we employ A* search, similar to (Wayllace *et al.*, 2016; Keren *et al.*, 2017), in the space of models, as shown in Algorithm 1. The algorithm is referred to as `MEGA` – **M**ulti-model **E**xplanation **G**eneration **A**lgorithm. Given an MRP, we start off with the initial state $\Gamma(\mathcal{M}_h^R)$ derived from the human's expectation of a given planning problem $\mathcal{M}^R$, and modify it incrementally until we arrive at a planning problem $\widehat{\mathcal{M}}$ with $C^{\widehat{\mathcal{M}}}(\pi_R^*) = C^*_{\widehat{\mathcal{M}}}$, i.e. the given plan is explained. Note that the model changes are represented as a set, i.e. there is no sequentiality in the search problem. Also, we assign equal importance to all model corrections. We can easily capture differential importance of model updates by attaching costs to the edit actions λ - the algorithm remains unchanged. We also employ a selection strategy for successor nodes to speed up search (by overloading the way the priority queue is popped) by first processing model changes that are relevant to actions in π_R^* and π_H before the rest.

Proposition 4. *The successor selection strategy outlined in Algorithm 1 yields an admissible heuristic for model space search for minimally complete explanations.*

Proof. Let $\mathcal{E}$ be the MCE for an MRP problem and let $\mathcal{E}'$ be any intermediate explanation found by our search such that $\mathcal{E}' \subset \mathcal{E}$, then the set $\mathcal{E} \setminus \mathcal{E}'$ must contain at least one λ related to actions in the set $\{a \mid a \in \pi_R \vee a \in \pi'\}$ (where π' is the optimal plan for the model $\hat{\mathcal{M}}$ where $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\Gamma(\mathcal{M}_h^R), \mathcal{E}') = \Gamma(\hat{\mathcal{M}})$. To see why this is true, consider an $\mathcal{E}'$ where $|\mathcal{E}'| = |\mathcal{E}| - 1$. If the action in $\mathcal{E} \setminus \mathcal{E}'$ does not belong to either π_R^* or π' then it can not improve the cost of π_R^* in comparison to π' and hence $\mathcal{E}$ can not be the MCE. Similarly we can show that this relation will hold for any size of $\mathcal{E}'$. We can leverage this knowledge about $\mathcal{E} \setminus \mathcal{E}'$ to create an admissible heuristic that considers only relevant changes. $\square$

Model Space Search for MMEs

As per the definition of MMEs, beyond the model obtained from the minimally monotonic explanation, there do not exist any models which are not explanations of the same MRP, while at the same time making as few changes to the original problem as possible. It follows that this is the largest set of changes that can be done on $\mathcal{M}^R$ and still find a model $\widehat{\mathcal{M}}$ where $C^{\widehat{\mathcal{M}}}(\pi^*) = C^*_{\widehat{\mathcal{M}}}$ - we are going to use this property in the search for MMEs.

Proposition 5. $\mathcal{E}^{MME} = \arg\max_{\mathcal{E}} |\Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R)|$ *such that* $\forall \hat{\mathcal{M}}\ \Gamma(\hat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R) \subseteq \Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R)$ *it is guarantee to have* $C^{\hat{\mathcal{M}}}(\pi^*) = C^*_{\hat{\mathcal{M}}}$.

This is similar to the model-space search for MCEs, but this time starting from the robot's model $\mathcal{M}^R$ instead. The goal here is to find the largest set of model changes for which the explicability criterion becomes invalid for the first time (due to either suboptimality or inexecutability). This requires a search over the entire model space (Algorithm 2). We can leverage Proposition 5 to reduce our search space. Starting from $\mathcal{M}^R$, given a set of model changes $\mathcal{E}$ where $\delta_{\mathcal{M}_R,\mathcal{M}_H}(\Gamma(\mathcal{M}^R), \mathcal{E}) = \Gamma(\widehat{\mathcal{M}})$ and $C^{\widehat{\mathcal{M}}}(\pi^*) > C^*_{\widehat{\mathcal{M}}}$, no superset of $\mathcal{E}$ can lead to an MME solution. In Algorithm 2, we keep track of such unhelpful model changes in the list h_list. The variable $\mathcal{E}^{MME}$ keeps track of the current best list of model changes. Whenever we find a new set of model changes where π^* is optimal and is larger than $\mathcal{E}^{MME}$, we update $\mathcal{E}^{MME}$ with $\mathcal{E}$. The resulting MME is all the possible model changes that did not appear in $\mathcal{E}^{MME}$.

Figure 4.3 contrasts MCE search with MME search. MCE search starts from $\mathcal{M}^R_h$, computes updates $\widehat{\mathcal{M}}$ towards $\mathcal{M}^R$ and returns the first node (indicated in orange) where $C^{\widehat{\mathcal{M}}}(\pi^*) = C^*_{\widehat{\mathcal{M}}}$. MME search starts from $\mathcal{M}^R$ and moves towards $\mathcal{M}^R_h$. It finds the longest path (indicated in blue) where $C^{\widehat{\mathcal{M}}}(\pi^*) = C^*_{\widehat{\mathcal{M}}}$ for all $\widehat{\mathcal{M}}$ in the path. The MME (green) is the rest of the path towards $\mathcal{M}^R_h$.

Approximate MCE-search

Both MCEs and MMEs may be hard to compute - in the worst case it involves a search over the entire space of model differences. Thus the biggest bottleneck here is the check for optimality of a plan given a new model. A check for necessary or sufficient conditions for optimality, without actually computing optimal plans can be used as a powerful tool to further prune the search tree.

In the following section, we investigate an approximation to an MCE by employing a few simple proxies to the optimality test. By doing this we lose the completeness guarantee but improve computability. Specifically, we replace the equality test in line 12 of Algorithm 1 by the following rules –

1. $\delta_{\widehat{\mathcal{M}}}(\widehat{I}, \pi^*_R) \models \widehat{G}$; **and**

2. $C^{\widehat{\mathcal{M}}}(\pi^*_R) < C^{\mathcal{M}^R_h}(\pi^*_R)$ **or** $\delta_{\widehat{\mathcal{M}}}(\widehat{I}, \pi^*_H) \not\models \widehat{G}$; **and**

3. Each action contributes at least one causal link to π^*_R.

(1) ensures that the plan π^*_R originally computed is actually valid in the new model. (2) requires that this plan has either become better in the new model or at least that the human's expected plan π^*_H has been disproved. Finally, in (3), we ensure that for each action $a_i \in \pi^*_R$ there exists an effect p that satisfies the precondition of at least one action a_k (where $a_i \prec a_k$) (where a_k can be a specialized goal action) and there exists no action a_j (where $a_i \prec a_j \prec a_k$) such that $p \in del^{\hat{\mathcal{M}}}(a_j)$. Such explanations are only able to preserve local properties of a plan and hence incomplete.

Proposition 6. *Criterion (3) is a necessary condition for optimality of π^* in $\widehat{\mathcal{M}}$.*

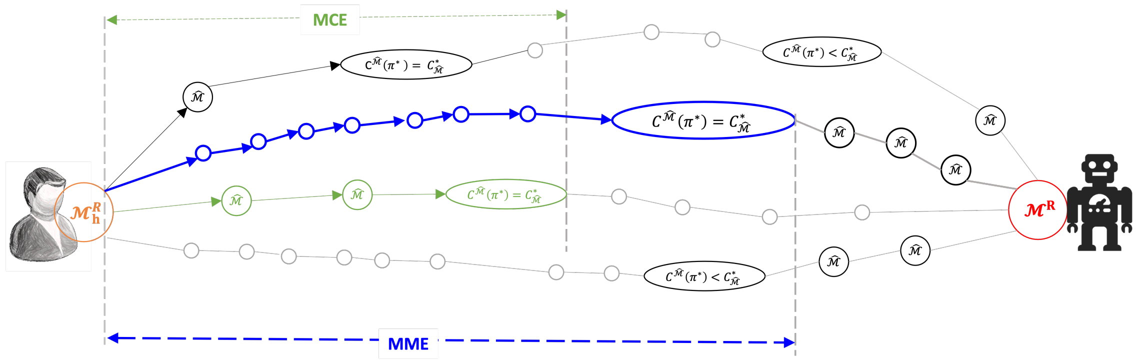

MCE
MME
$\mathcal{M}^R_h$
$\mathcal{M}^R$
$\widehat{\mathcal{M}}$
$C^{\widehat{\mathcal{M}}}(\pi^*) = C^*_{\widehat{\mathcal{M}}}$
$C^{\widehat{\mathcal{M}}}(\pi^*) < C^*_{\widehat{\mathcal{M}}}$

Figure 4.3: Illustration Contrasting MCE Search with MME Search.

Proof. Assume that for an optimal plan π^*_R, there exists an action a_i where criterion (3) is not met. Now we can rewrite π_R as

$$\pi'_R = \langle a_0, a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n, a_{n+1}\rangle$$

where in all models, $pre^{\cdot}_{+}(a_0) = \emptyset$ and $add^{\cdot}(a_0) = \{I\}$ and $pre^{\cdot}_{+}(a_{n+1}) = \{G\}$. It is easy to see that $\delta_{\widehat{\mathcal{M}}}(\emptyset, \pi'_R) \models G$. Now let us consider a cheaper plan $\hat{\pi'_R} = \langle a_0, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, a_{n+1}\rangle$. Since a_i does not contribute any causal links to the original plan π_R, we will also have $\delta_{\widehat{\mathcal{M}}}(\emptyset, \hat{\pi}'_R) \models G$. This contradicts our original assumption of π_R being optimal, hence proved. □

4.2.4 Minimal Contrastive Explanation

As discussed earlier, the explanations presented here could be seen as cases where the system is trying to address an implicit form of contrastive explanations. However, there may be cases where the user may present an explicit set of foils. In such cases, the goal of the explanations would become to identify a set of model updates that establishes why π^*_R is preferred over the foils. We will refer to such explanations as Minimally Contrastive Explanations. We will formally define the explanation as,

Definition 10. *Given the models $\mathcal{M}^R_h$, $\mathcal{M}^R$, and a set of plans expected by the human*

Algorithm 1 Search for Minimally Complete Explanations

```
1:  procedure MCE-SEARCH
2:      Input: MRP $\langle \pi^*_R, \langle \mathcal{M}^R, \mathcal{M}^R_h \rangle\rangle$
3:      Output: Explanation $\mathcal{E}^{MCE}$
4:      Procedure:
5:      fringe $\leftarrow$ Priority_Queue()
6:      c_list $\leftarrow \{\}$                                              ▷ Closed list
7:      $\pi_R \leftarrow \pi^*$                                               ▷ Optimal plan being explained
8:      $\pi_H \leftarrow \pi$ such that $C^{\mathcal{M}^R_h}(\pi) = C^*_{\mathcal{M}^R_h}$                    ▷ Plan expected by human
9:      fringe.push($\langle \mathcal{M}^R_h, \{\}\rangle$, priority $= 0$)
10:     while True do
11:         $\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c \leftarrow$ fringe.pop($\widehat{\mathcal{M}}$)
12:         if $C^{\widehat{\mathcal{M}}}(\pi_R) = C^*_{\widehat{\mathcal{M}}}$ then return $\mathcal{E}$                ▷ Return $\mathcal{E}$ if $\pi_R$ optimal in $\widehat{\mathcal{M}}$
13:         else
14:             c_list $\leftarrow$ c_list $\cup\ \widehat{\mathcal{M}}$
15:             for $f \in \Gamma(\widehat{\mathcal{M}}) \setminus \Gamma(\mathcal{M}^R)$ do                ▷ Models that satisfy Condition 1
16:                 $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{\}, \{f\}\rangle$                        ▷ Removes f from $\widehat{\mathcal{M}}$
17:                 if $\delta_{\mathcal{M}^R_h, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda) \not\in$ c_list then
18:                     fringe.push($\langle \delta_{\mathcal{M}^R_h, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda), \mathcal{E} \cup \lambda\rangle, c + 1$)
19:             for $f \in \Gamma(\mathcal{M}^R) \setminus \Gamma(\widehat{\mathcal{M}})$ do                ▷ Models that satisfy Condition 2
20:                 $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{f\}, \{\}\rangle$                        ▷ Adds f to $\widehat{\mathcal{M}}$
21:                 if $\delta_{\mathcal{M}^R_h, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda) \not\in$ c_list then
22:                     fringe.push($\langle \delta_{\mathcal{M}^R_h, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda), \mathcal{E} \cup \lambda\rangle, c + 1$)
23: procedure PRIORITY_QUEUE.POP($\hat{\mathcal{M}}$)
24:     candidates $\leftarrow \{\langle\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c^*\rangle \mid c^* = \arg\min_c \langle\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c\rangle\}$
25:     pruned_list $\leftarrow \{\}$
26:     $\pi_H \leftarrow \pi$ such that $C^{\hat{\mathcal{M}}}(\pi) = C^*_{\hat{\mathcal{M}}}$
27:     for $\langle\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c\rangle \in$ candidates do
28:         if $\exists a \in \pi_R \cup \pi_H$ such that $\tau^{-1}(\Gamma(\widehat{\mathcal{M}})$ is related to $a$ in $\mathcal{M}^R$ or $\mathcal{M}^R_h$ then
29:                                                                    ▷ Candidates relevant to $\pi_R$ or $\pi_H$
30:             pruned_list $\leftarrow$ pruned_list $\cup\ \langle\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c\rangle$
31:     if pruned_list $= \emptyset$ then $\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c \sim Unif$(candidate_list)
32:     else                              $\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c \sim Unif$(pruned_list)
```

Algorithm 2 Search for Minimally Monotonic Explanations

```
1: procedure MME-Search
2:     Input: MRP $\langle \pi^*, \langle \mathcal{M}^R, \mathcal{M}^R_h \rangle\rangle$
3:     Output: Explanation $\mathcal{E}^{MME}$
4:     Procedure:
5:     $\mathcal{E}^{MME} \leftarrow \{\}$
6:     fringe $\leftarrow$ Priority_Queue()
7:     c_list $\leftarrow \{\}$                                          ▷ Closed list
8:     h_list $\leftarrow \{\}$                        ▷ List of incorrect model changes
9:     fringe.push($\langle \mathcal{M}^R, \{\}\rangle$, priority = 0)
10:    while fringe is not empty do
11:        $\langle \widehat{\mathcal{M}}, \mathcal{E}\rangle, c \leftarrow$ fringe.pop($\widehat{\mathcal{M}}$)
12:        if $C^{\widehat{\mathcal{M}}}(\pi^*) > C^*_{\widehat{\mathcal{M}}}$ then
13:            h_list $\leftarrow$ h_list $\cup$ $(\Gamma(\widehat{\mathcal{M}}) \,\Delta\, \Gamma(\mathcal{M}^R))$      ▷ Updating h_list
14:        else
15:            c_list $\leftarrow$ c_list $\cup$ $\widehat{\mathcal{M}}$
16:            for $f \in \Gamma(\widehat{\mathcal{M}}) \setminus \Gamma(\mathcal{M}^R_h)$ do   ▷ Models that satisfy Condition 1
17:                $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{\}, \{f\}\rangle$              ▷ Removes $f$ from $\widehat{\mathcal{M}}$
18:                if $\delta_{\mathcal{M}^R,\mathcal{M}^R_h}(\Gamma(\widehat{\mathcal{M}}), \lambda) \not\in$ c_list
                     and $\nexists S$ s.t. $(\Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R)) \supseteq S \in$ h_list then   ▷ Proposition 5
19:                    fringe.push($\langle \delta_{\mathcal{M}^R,\mathcal{M}^R_h}(\Gamma(\widehat{\mathcal{M}}), \lambda), \ \mathcal{E} \cup \lambda\rangle, \ c+1$)
20:                    $\mathcal{E}^{MME} \leftarrow \max_{|\cdot|}\{\mathcal{E}^{MME}, \mathcal{E}\}$
21:            for $f \in \Gamma(\mathcal{M}^R_h) \setminus \Gamma(\widehat{\mathcal{M}})$ do   ▷ Models that satisfy Condition 2
22:                $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{f\}, \{\}\rangle$                 ▷ Adds $f$ from $\widehat{\mathcal{M}}$
23:                if $\delta_{\mathcal{M}^R,\mathcal{M}^R_h}(\Gamma(\widehat{\mathcal{M}}), \lambda) \not\in$ c_list
                     and $\nexists S$ s.t. $(\Gamma(\widehat{\mathcal{M}})\Delta\Gamma(\mathcal{M}^R)) \supseteq S \in$ h_list then   ▷ Proposition 5
24:                    fringe.push($\langle \delta_{\mathcal{M}^R,\mathcal{M}^R_h}(\Gamma(\widehat{\mathcal{M}}), \lambda), \ \mathcal{E} \cup \lambda\rangle, \ c+1$)
25:                    $\mathcal{E}^{MME} \leftarrow \max_{|\cdot|}\{\mathcal{E}^{MME}, \mathcal{E}\}$
26:    $\mathcal{E}^{MME} \leftarrow (\Gamma(\widehat{\mathcal{M}}) \,\Delta\, \Gamma(\mathcal{M}^R)) \setminus \mathcal{E}^{MME}$
27:    return $\mathcal{E}^{MME}$
```

$\hat{\Pi}^H$, *a set of model updates* $\mathcal{E}^{con}$ *is said to be a* Minimally Contrastive Explanations (MCrE), *if*

$$\mathcal{E}^{con} = \arg\min_{\mathcal{E}} |\mathcal{E}|$$

Such that

1. $\delta_{\widehat{\mathcal{M}}}(\widehat{I}, \pi^*_R) \models \widehat{G}$; ***and***

2. $C^{\widehat{\mathcal{M}}}(\pi^*_R) < C^{\mathcal{M}^R_h}(\pi')$ ***or*** $\delta_{\widehat{\mathcal{M}}}(\widehat{I}, \pi') \not\models \widehat{G}\ \forall \pi' \in \hat{\Pi}^H$.

The algorithm for generating such explanation was discussed by Valmeekam *et al.* (2020).

4.3 Evaluation

We performed a set of empirical evaluation to evaluate the computational characteristics of the explanation generation for some benchmark problems, including the time taken for generating the explanations and the size of generated explanations. Our explanation generation system integrates calls to Fast-Downward (Helmert, 2006) for planning, VAL (Howey *et al.*, 2004) for plan validation, and pyperplan (Alkhazraji *et al.*, 2016) for parsing. The results reported here are from experiments run on a 12 core Intel(R) Xeon(R) CPU with an E5-2643 v3@t3.40GHz processor and a 64G RAM. We use three popular planning domains (International Planning Competition, 2011) – BlocksWorld, Logistics and Rover – for our experiments. In order to generate explanations we created the human model by randomly removing parts (preconditions and effects) of the action model (the number of edits made per domain is equal to the model patch explanations, which is reported in Table 4.2). Though the following experiments are only pertinent to action model differences, it does not make any difference at all to the approaches, given the way the state was defined. Also note that these removals, as well as the corresponding model space search, were done in the lifted representation of the domain.

4.3.1 MCEs Versus MMEs

The first item of consideration is the size of explanations with respect to the total number of model differences, since we aimed for minimality as a desired feature for both MCEs and MMEs. Table 4.2 shows the number of explanations produced and the time taken (in secs) to produce them, against the ground truth. Heuristics seem to provide advantage in terms of the time spent on each problem, particularly for BlocksWorld domain. Further, note how close the approximate version of MCEs are to the exact solutions. As expected, MME search is significantly costlier to compute than MCE. However, note that both MCEs and MMEs are *significantly smaller* in size ($\sim$ 20%) than the total model difference (which can be arbitrarily large) in certain domains, further underlining the usefulness of generating minimally complete explanations as opposed to dumping the entire model difference on the human. A general rule of thumb is –

$$\mid \mathcal{E}^{\text{ approx.}MCE} \mid \leq \mid \mathcal{E}^{MCE} \mid < \mid \mathcal{E}^{MME} \mid << |\mathcal{E}^{MPE}|$$

Note that the time required to calculate an MME in the Logistics problems is lower than that for the corresponding MCE. This is because for most of these problems a single change in the planner's model made the plan be no longer optimal so that the search ended after checking all possible unit changes. In general, the closer an MCE is to the total number of changes shorter the MME search would be. Also note how PPE solutions, though much easier to compute, do not have completeness and monotonicity properties, and yet often spans the entire model difference, containing information that are not needed to support the optimality of the given plan.

Problem Instance		MPE (truth)		PPE		MME (exact)		MCE (exact w/o heuristic)		MCE (exact with heuristic)		MCE (approximate)	
		size	time	size	time	size	time	size	time	size	time	size	time
Blocks World	p1	10	n/a	5	n/a	3	1100.8	2	34.7	2	18.9	2	19.8
	p2			8		4	585.9	3	178.4	3	126.6	3	118.8
	p3			4		5	305.3	2	34.7	2	11.7	2	11.7
	p4			7		5	308.6	3	168.3	3	73.3	3	73.0
Rover	p1	10	n/a	10	n/a	2	2093.2	2	111.3	2	100.9	2	101.0
	p2			10		2	2018.4	2	108.6	2	101.7	2	102.7
	p3			10		2	2102.4	2	104.4	2	104.9	2	102.5
	p4			9		1	3801.3	1	13.5	1	12.8	1	12.5
Logistics	p1	5	n/a	5	n/a	4	13.7	4	73.2	4	73.5	4	63.6
	p2			5		4	13.5	4	73.5	4	71.4	4	63.3
	p3			5		5	8.6	5	97.9	5	100.4	3	36.4
	p4			5		5	8.7	5	99.2	5	95.4	3	36.4

Table 4.2: Comparison of MCEs and MMEs. The Size of the Explanation Corresponds to the Cardinality of the Explanations (I.E. $\mid \mathcal{E}^* \mid$)

We now increase the number of changes in the human model in BlocksWorld, and illustrate the relative time (in secs) taken to search for exact MCEs in Table 4.3. The human models are again generated by randomly removing model components (the generated models are not the same as the ones Table 4.2). As expected there is an exponential increase in the time taken, which can be problematic with even a modest number of model differences. This further highlights the importance of approximations in the model reconciliation process and motivates further research in heuristics for model space search.

Finally, Table 4.4 illustrates how Proposition 5 reduces the number of nodes searched to find MMEs in random problems from the BlocksWorld domain with 10 faults in the human model, as opposed to the total possible 2^{10} models that can be evaluated – equal to the cardinality of the power set of model changes between the robot model and the mental model (i.e., $|\mathcal{P}(\Gamma(\mathcal{M}^R)\Delta\Gamma(\mathcal{M}_h^R))|$).

$\lvert\mathcal{M}^R \Delta \mathcal{M}_h^R\rvert$	problem-1	problem-2	problem-3	problem-4
3	2.2	18.2	4.7	18.5
5	6.0	109.4	15.4	110.2
7	7.3	600.1	23.3	606.8
10	48.4	6849.9	264.2	6803.6

Table 4.3: MCE Search Time for Increasing Model Differences for Blocksworld.

BlocksWorld	problem-1	problem-2	problem-3	problem-4
Number of nodes expanded for MME (out of 1024)	128	64	32	32

Table 4.4: Usefulness of Proposition 5 in Pruning MME Search.

4.4 Human-Factors Study of the Model Reconciliation Process

The design of explainable AI algorithms is, of course, incomplete without evaluations with actual humans in the loop. Thus, in the following discussion, we will report on the the salient findings from a series of controlled user studies we undertook in order to evaluate the usefulness of the model reconciliation approach. Through these studies, we aim to validate whether explanations in the form of model reconciliation (in its various forms) suffice to explain the optimality and correctness of plans to the human in the loop. We also study participants who were asked to generate explanations in the form of model changes, to see if explanations generated by the humans align with any of the multi-model explanations identified in the discussion so far. The studies suggest that humans do indeed understand explanations of this form and believe that such explanations are necessary to explain plans.

We stick to the USAR domain for our study (Figure 4.4). In the study, we only

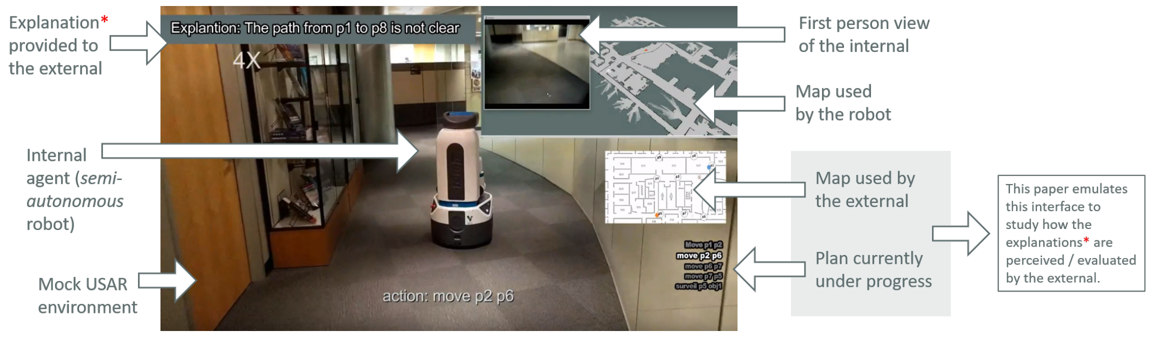

Explanation* provided to the external
Explantion: The path from p1 to p8 is not clear
4X
First person view of the internal
Map used by the robot
Internal agent (*semi-autonomous* robot)
Map used by the external
Move p1 p2
move p2 p6
move p6 p7
move p7 p5
surveil p5 obj1
Plan currently under progress
This paper emulates this interface to study how the explanations* are perceived / evaluated by the external.
Mock USAR environment
action: move p2 p6

Figure 4.4: Illustration of the Simulated USAR Setting. We Expose a Mock Interface to the External Agent (Right Inset) on the Browser to Study the Properties of Different Explanations Afforded by the Model Reconciliation Framework.

simulate the interface to the external. As we discussed before, in general, differences in the models of the human and the robot can manifest in any form (e.g. the robot may have lost some capability or its goals may have changed). In the current setup, however, we only deal with differences in the map of the environment as available to the two agents.

4.4.1 Study – 1: Participants Are Explainers

The first part of the study aims to develop an understanding of how humans respond to the task of generating explanations, i.e. if left to themselves, humans preferred to generate explanations similar to the ones developed here. To test this, we asked participants to assume the role of the internal agent in the explanation process and explain their plans with respect to the faulty map of their teammate. Specifically, we set out to test the following hypothesis –

H1. When asked to, participants would leverage model differences as a key ingredient for explanations.

H1a. Explanation generated by participants would demonstrate contrastiveness.

Thus, PPE type explanations would be overlooked in favor of complete solutions (MCEs and MPEs) when there are multiple competing hypothesis for the human.

H2. Participants would like to minimize the content of the explanation by removing details that are not relevant to the plan being explained.

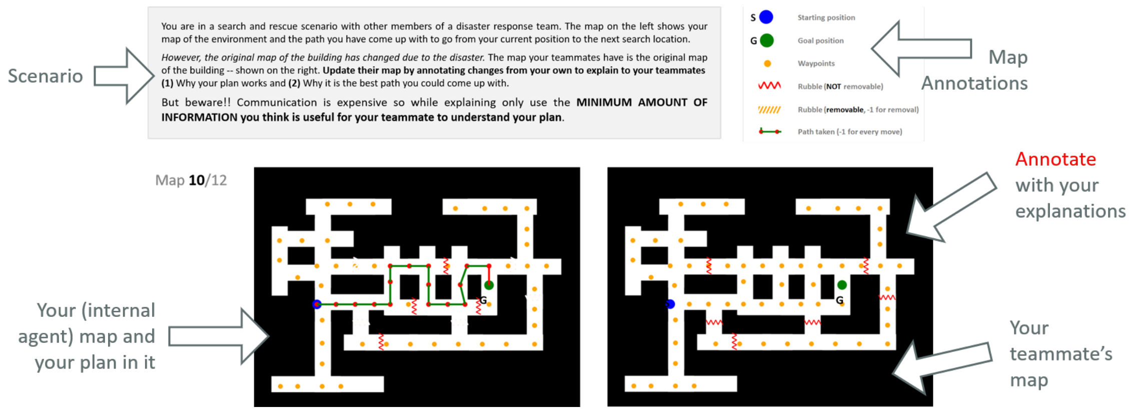
- H2a. Explanations from participants would be closer to MCEs than MPEs.
- H2b. This should be even more significant if restrictions are placed on communication.

As a result of this study, we intended to identify to what extent explanation types described in this chapter built upon principles of explanations in human-human interactions studied in social sciences (more on this in Section 4.5) truly reflect human intuition.

Note that we primed the subjects to annotate changes in the map, while giving them the opportunity to –

1. Provide more than annotations (and we did find other interesting kinds of explanations emerge as we discuss later in Section 4.4.3)
2. Comment on the sufficiency and necessity of such explanations (as we report in Section 4.4.1)

The reason for this choice is because in the work being evaluated here, communicating model differences has been considered to be the starting point of the explanation process. So we start from that assumption and evaluate to what extent the kinds of explanations introduced here – MCE / MPE / PPE / etc. – are actually useful. Additionally, this setup also helps to re-contextualize the real importance of


You are in a search and rescue scenario with other members of a disaster response team. The map on the left shows your map of the environment and the path you have come up with to go from your current position to the next search location.

However, the original map of the building has changed due to the disaster. The map your teammates have is the original map of the building -- shown on the right. **Update their map by annotating changes from your own to explain to your teammates (1)** Why your plan works and **(2)** Why it is the best path you could come up with.

But beware!! Communication is expensive so while explaining only use the **MINIMUM AMOUNT OF INFORMATION you think is useful for your teammate to understand your plan**.

Scenario
S
Starting position
G
Goal position
Waypoints
Rubble (NOT removable)
Rubble (removable, -1 for removal)
Path taken (-1 for every move)
Map Annotations
Map 10/12
Annotate with your explanations
Your (internal agent) map and your plan in it
Your teammate's map

Figure 4.5: Interface for Study-1: Participants Assumed the Role of the Internal Agent and Explained Their Plans to a Teammate with a Possibly Different Map of the World.

model difference in the explanation process in light of reasons explained in (1) and (2) above.

Experimental Setup

Figure 4.5 shows an example map and plan provided to a participant. On the left side, the participant is shown the actual map along with their plan, starting position and goal. The panel on the right shows the map available to the explainee. The maps have removable and non-removable rubble blocking access to certain paths (the maps may disagree as to the locations of the debris). The participants were asked to convince the explainee of the correctness and optimality of the given plan by updating the latter's maps with annotations they felt were relevant in achieving that goal. We ran the study with two conditions –

C1. Here the participants were asked to ensure, via explanations, that their plan was correct and optimal in the updated model of their teammate;

C2. Here, in addition to C1, they were also asked to use the minimal amount of information they felt was needed to achieve the condition in C1.

Each participant was shown how to annotate (not an explanation) a sample map and was then asked to explain 12 different plans using similar annotations. After each participant was finished with their assignment, they were asked the following subjective questions –

`Q1. Providing map updates were necessary to explain my plans.`

`Q2. Providing map updates were sufficient to explain my plans.`

`Q3. I found that my plans were easy to explain.`

The answers to these questions were measured using a five-point Likert scale. The answers to the first two questions will help to establish whether humans considered map updates (or in general updates on the model differences) at all necessary and/or sufficient to explain a given plan. The final question measures whether the participants found the explanation process using model differences tractable. It is important to note that in this setting we do not measure the efficacy of these explanations (this is the subject of Study-2 in Section 4.4.2). Rather we are trying to find whether a human explainer would have naturally participated in the model reconciliation approach during the explanation process.

In total, we had 12 participants for condition C1 and 10 participants for condition C2 including 7 female and 18 male participants between the age range of 18-29 (data corresponding to 5 participants who misinterpreted the instructions had to be removed, 2 participants did not reveal their demographics). Participants for the study were recruited by requesting the department secretary to send an email to the student body to ensure that they had no prior knowledge about the study or its relevance. Each participant was paid $10.

Results

Figure 4.6 – The first hypothesis we tested was whether the explanations generated by the participants matched any of the explanation types introduced in this chapter. We did this by going through all the individual explanations provided by the participants and then categorizing each explanation to one of the four types, namely MCE, PPE, MPE or Other (the "other" group contains explanations that do not correspond to any of the predefined explanation types – more on this later in Section 4.4.3). Each explanation type was identified by checking the explanation provided by the participants against what may have been generated by the algorithm. Figure 4.6a shows the number of explanations of each type that were provided by the participants of C1. The graph shows a clear preference for MPE, i.e. providing all model differences. A possible reason for this may be since the size of MPEs for the given maps were not too large (and participants did not have time constraints). Interestingly, in C2 we see a clear shift in preferences (Figure 4.6b) where most participants ended up generating MCE style explanations. This means at least for scenarios where there are constraints on communication, the humans would prefer generating MCEs as opposed to explaining all the model differences.

These findings are consistent with H1, with very few of the explanations in type "Other" (Figure 4.6). This is also backed up by answers to subjective questions Q1 and Q2 above. Further, the preference of MPE/MCE over PPE (H1a) is quite stark. Contrary to H2a, participants seemed to have preferred full model explanation (MPE) in C1 condition which is surprising. However, results of C2 condition are more aligned with H2b, even though we expected to see a similar trend (if not as strong) in C1 condition as well.

Figures 4.7 and 4.8 – These show the results of the subjective questions for C1 and C2 respectively. Interestingly, in C1, while most people agreed on the necessity of explanations in the form of model differences, they were less confident regarding the sufficiency of such explanations. In fact, we found that many participants left additional explanations in their worksheet in the form of free text (we discuss some of these findings in Section 4.4.3). In C2, we still see that more people are convinced about the necessity of these explanations than sufficiency. But we see a reduction in the confidence of the participants, which may have been caused by the additional minimization constraints.

4.4.2 Study – 2: Participants Are Explainees

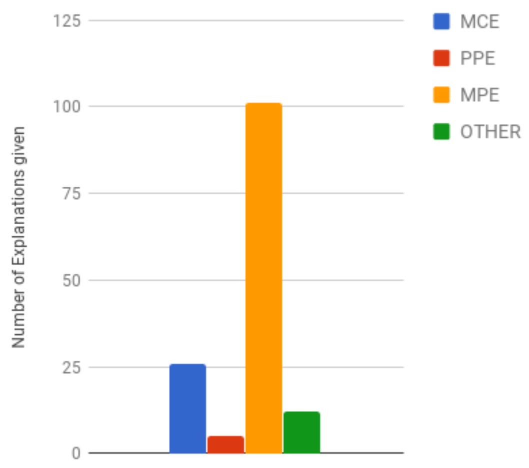
Now we study how different kinds of explanations outlined in Section 4.2 are perceived by the participants. This study was designed to provide clues to how humans comprehend explanations when provided to them in the form of model differences. Specifically, we intend to evaluate the following hypothesis, in line with the intended properties of each of the explanation types –

H1. Participants would be able to identify optimality given an MPE or MCE.

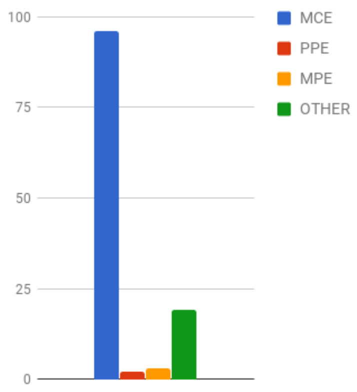
H2. Participants would be able to identify executability but possible suboptimality of a plan given a PPE.

H3. Participants would not ask for explanations when presented with explicable plans (optimal in mental model).

As a result of this study, we intend to validate whether desired properties of explanations for task planning designed by following norms and principles outlines in the social sciences in the context of human-human interactions (Miller, 2017a) do actually carry over for human-robot interactions.



(a) Study-1:C1



(b) Study-1:C2

Figure 4.6: Explanation Counts for Study-1:C:1-2.

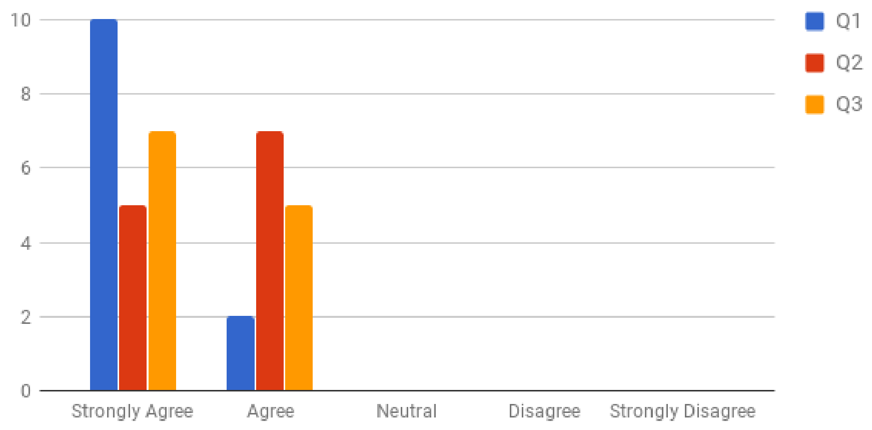


Figure 4.7: Subjective Responses of Participants in Study-1:C1.

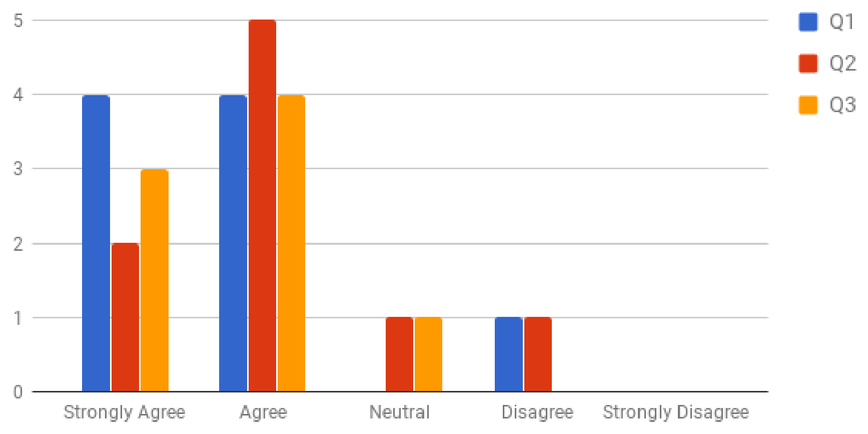


Figure 4.8: Subjective Responses of Participants in Study-1:C2.

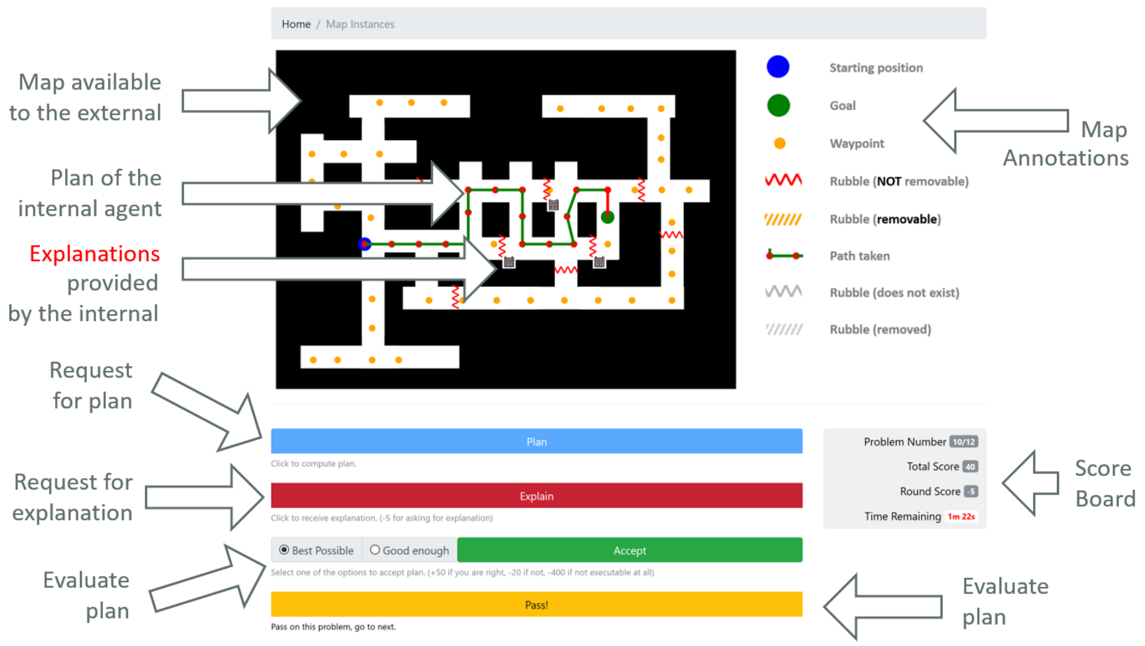


Figure 4.9: Interface for Study-2 Where Participants Assumed the Role of the External Commander and Evaluated Plans Provided by the Internal Robot. They Could Request for Plans and Explanations to Those Plans (If Not Satisfied) and Rate Them as Optimal or Suboptimal or (If Unsatisfied) Can Chose to Pass.

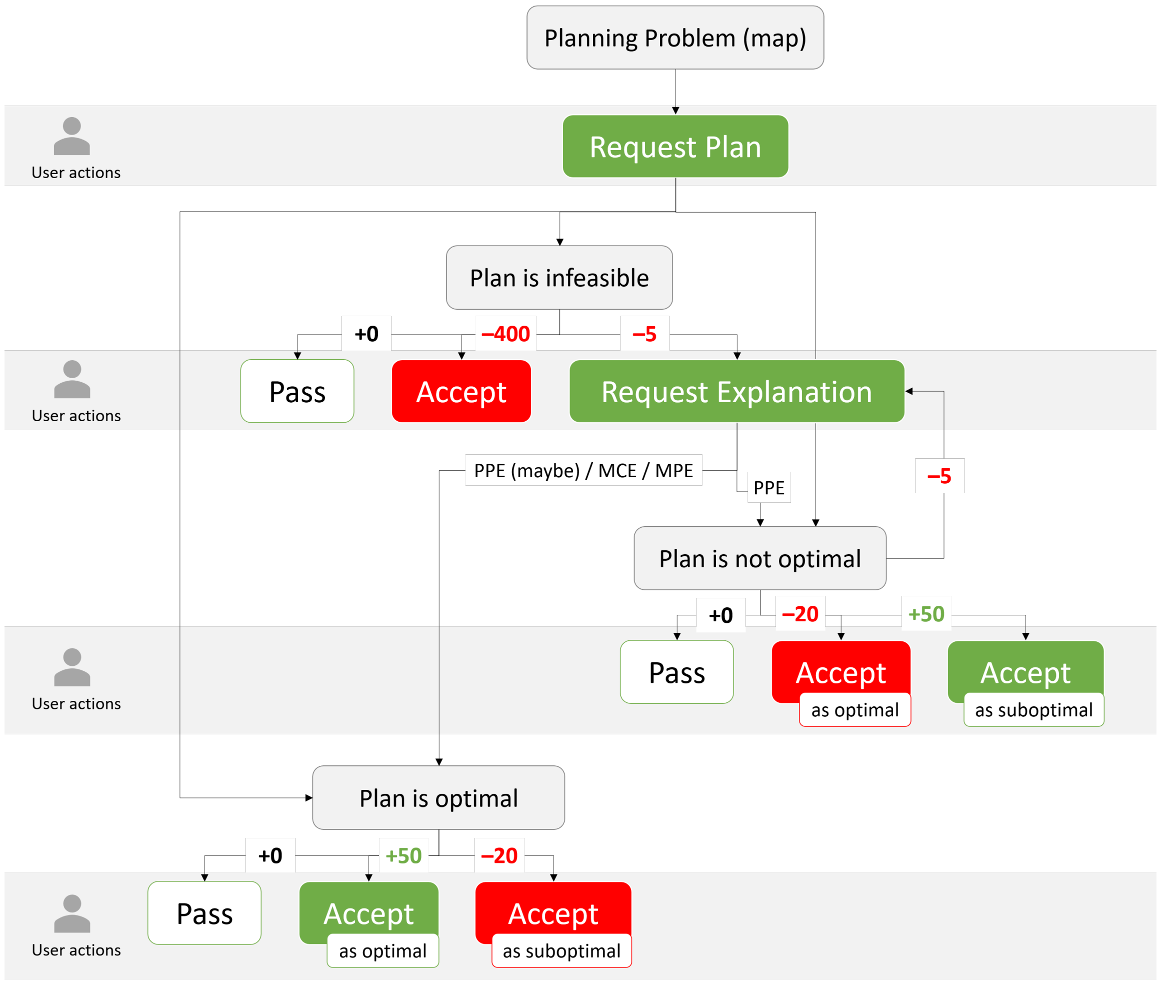


Figure 4.10: Illustration of the Flow of Logic in the Experimental Setup.

Experimental Setup

During this study, participants were incentivized to make sure that the explanation does indeed help them understand the optimality and correctness of the plans in question by formulating the interaction in the form of a game.

Figure 4.9 shows a screenshot of the interface.[5] The game displays to each partic-

[5]This domain has elements of both motion planning and task planning (e.g. removal of debris) in it. The approaches developed in this chapter are applicable to task plans in general, as done in the work on identifying preferences over logically equivalent explanations in (Zahedi *et al.*, 2019), where the study was conducted in a logistic domain with plans involving the transport of cargo.

ipant an initial map (which they are told may differ from the robot's actual map), the starting point and the goal. Once the player asks for a plan, the robot responds with a plan illustrated as a series of paths through waypoints highlighted on the map. The goal of the participant is to identify if the plan shown is optimal or just satisficing. If the player is unsure of the path, they can ask for an explanation from the robot. The explanation is provided to the participant in the form of a set of model changes in the player's map. If the player is still unsure, they can click on the pass button to move to the next map.

The scoring scheme for the game is as follows (Summarized in Figure 4.10). Each player is awarded 50 points for correctly identifying the plan as either optimal or satisficing. Incorrectly identifying an optimal plan as suboptimal or vice versa would cost them 20 points. Every request for explanation would further cost them 5 points, while skipping a map does not result in any penalty. The participants were additionally told that selecting an inexecutable plan as either feasible or optimal would result in a penalty of 400 points. Even though there were no actual incorrect plans in the dataset, this information was provided to deter participants from taking chances with plans they did not understand well.

Each participant was paid \$10 dollars and received additional bonuses based on the following payment scheme –

- Scores higher than or equal to 540 were paid \$10.
- Scores between 540 and 440 were paid \$7.
- Scores between 440 and 340 were paid \$5.
- Scores between 340 and 240 were paid \$3.

User studies have also been undertaken to test the validity of many variants of model reconciliation, including a warehouse scenario in (Sreedharan *et al.*, 2019a) as well as logistics and travel scenarios in (Sreedharan *et al.*, 2019b).

- Scores below 240 received no bonuses.

The scoring systems for the game was designed to ensure

- Participants should only ask for an explanation when they are unsure about the quality of the plan (due to small negative points on explanations).
- Participants are incentivized to identify the feasibility and optimality of the given plan correctly (large reward and penalty on doing this wrongly).

Each participant was shown a total of 12 maps (same maps as in Study–1). For 6 of the 12 maps, the player was assigned the optimal robot plan, and when they asked for an explanation, they were randomly shown either MCE, PPE or MPE explanation with regards to the robot model. For the rest of the maps, participants could potentially be assigned a plan that is optimal in the human model (i.e. an explicable plan) or somewhere in between as introduced in (Chakraborti *et al.*, 2019f) (referred to henceforth as the balanced plan) in place of the robot optimal plan[6]. The participants that were assigned the optimal robot plan were provided an MCE, PPE or MPE explanation, otherwise they were provided an empty explanation for the explicable plan. Also note that for 4 out of the 12 maps the PPE explanation cannot prove the optimality of the plan.

At the end of the study, each participant was presented with a series of subjective questions as follows. The responses to each question were measured on a five-point Likert scale.

[6]Note that of the 6 maps, only 3 had both balanced as well as explicable plans, the rest either had a balanced plan or the optimal human plan. Note that balanced plans are indistinguishable from the optimal plan from the point of view of the human. They are more useful to the robot for trading of explanation and explicability costs. Hence, we did not expand on further results on balanced plans here so as not to distract from the main focus of the chapter which is to evaluate explanations as model reconciliation. A detailed treatise is available in (Chakraborti *et al.*, 2019f).

`Q1. The explanations provided by the robot was helpful.`

`Q2. The explanations provided by the robot was easy to understand.`

`Q3. I was satisfied with the explanations.`

`Q4. I trust the robot to work on its own.`

`Q5. My trust in the robot increased during the study.`

In total, we had 27 participants for Study–2, including 4 female and 22 male between the ages of 19 to 31 (1 participant did not reveal their demographic).

Results

Figure 4.11 – As we mentioned before, the goal of this study is to identify if explanations in the form of model reconciliation can convey to humans the optimality and correctness of plans. Here, each participant was shown the 12 maps from Study-1 and each map was assigned a random explanation type (and in some cases different plans). We wanted to identify whether the participants that asked for explanations were able to come up with the correct conclusions. We chose to focus on participants who decided to ask for explanations, as people who didn't request for one may be operating off of a model different from the presented one. If this was indeed the case, results collected from these participants will not match the assumption required for model reconciliation explanations.. This means that the subjects who asked for MCE and MPE were able to correctly identify the plans as optimal, while the people who received PPE were able to correctly classify the plan to either optimal or satisficing (i.e. for all but 5 maps PPE is enough to prove optimality).

Figure 4.11 shows the statistics of the selections made by participants who had requested an explanation. The right side inset shows the percentage (for every map instance) of participants who selected the correct options (blue), the incorrect ones (red) or simply passed (orange), while the left side shows the average across all 12 maps. We notice that in general people were overwhelmingly able to identify the correct choice. Even in the case of PPEs, where the explanations only ensured correctness (map instances 1, 2, 3, 8 and 11) the participants were able to make the right choice. This is consistent with H1 and H2 and demonstrates that explanations in the form of model reconciliation are a viable means of conveying the correctness and optimality of plans – i.e. participants can differentiate between completeness and incompleteness of explanations.

Figure 4.12 – These conclusions are further supported by results from the subjective questionnaire (Figure 4.12). Most people seem to agree that the explanations were helpful and easy to understand. In fact, the majority of people strongly agreed that their trust of the robot increased during the study.

Figure 4.13 – We were also curious (H3) about the usefulness of explicable plans (that are optimal in human's model), i.e. if the subjects still asked for explanations when presented with explicable plans. Figure 4.13 shows the percentage of times subjects asked for explanations when presented with explicable versus robot optimal plans. The rate of explanations is considerably less in case of explicable plans as hypothesized. This matches the intuition behind the notion of plan explicability as a viable means (in addition to explanations) of dealing with model divergence in human-in-the-loop operation of robots.

It is interesting to see that in Figure 4.13 about a third of the time participants still asked for explanations even when the plan was explicable, and thus optimal in their map. We believe this is an artifact of the risk-averse behavior incentivized by the gamification of the explanation process. This is to make sure that participants were sufficiently invested in the outcome as well as mimic the high-stakes nature of USAR settings to accurately evaluate the explanations. It is also an indication of the cognitive burden on the humans who may not be (cost) optimal planners.

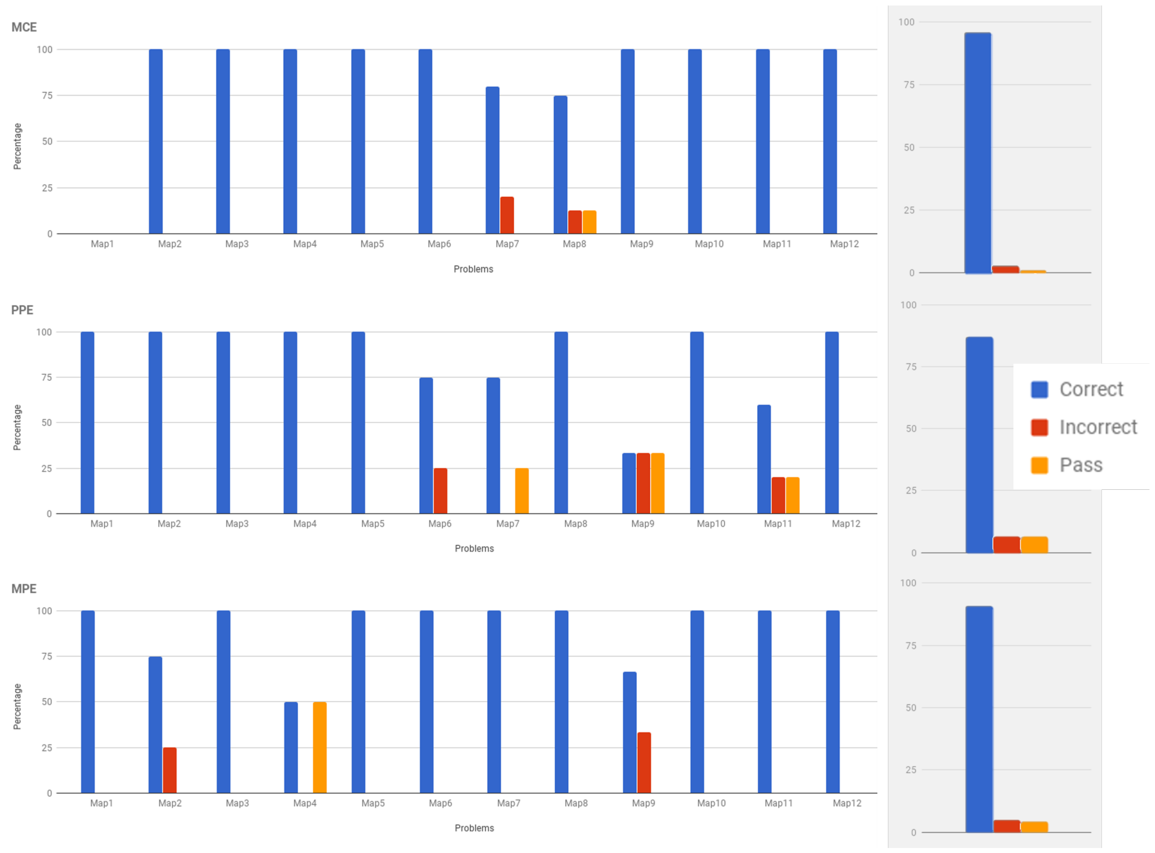


Figure 4.11: Percentage of times Different Explanations (i.e. MCE / MPE / PPE) Led to Correct Decision on the Human's Part in Each Problem (the Aggregated Result Is Shown on the Right). A "Correct Decision" Involves Recognizing Optimality of the Robot Plan on Being Presented an MCE or MPE, and Optimality or Executability (as the Case May Be) in Case of a PPE.

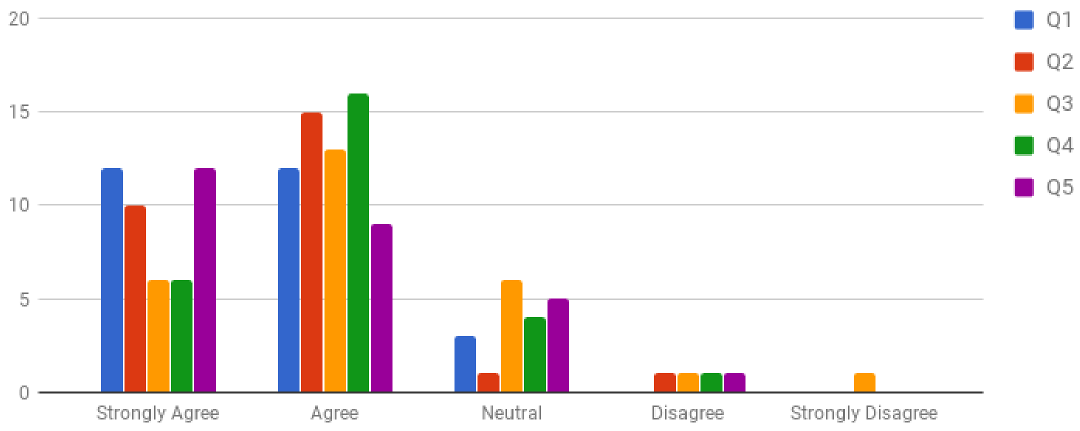


Figure 4.12: Subjective Responses of Participants in Study–2.

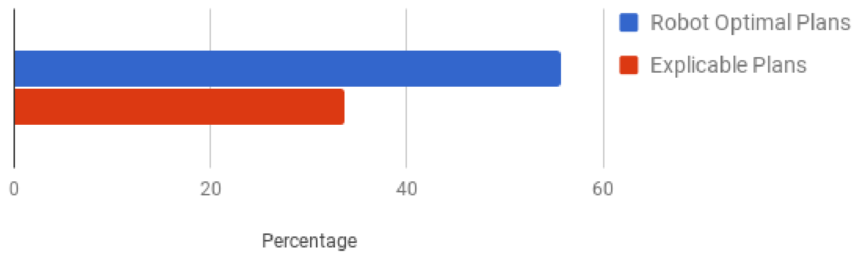


Figure 4.13: Percentage of Times Explanations Were Sought for in Study–2 When Participants Presented with Explicable Plans Versus Robot Optimal Plans with Explanations.

While this is consistent with the spirit of H3, the finding is also somewhat indicative of the limitations of plan explicability as it is defined in existing literature at the moment (Chakraborti *et al.*, 2019f). Thus, going forward, the objective function should incorporate the cost or difficulty of analyzing the plans and explanations from the point of view of the human in addition to the current costs of explicability and explanations modeled from the perspective of the robot.

Interestingly, the participants also did not ask for explanations around 40% of the time (c.f. Figure 4.13) when they "should have" (i.e. suboptimal plan in the human model) according to the theory of model reconciliation. We noticed no clear trend here (e.g. decreasing rate for explanations asked due to increasing trust). This was most likely due to limitations of inferential capability of humans and a limitation of the existing formulation of model reconciliation as well. The overall results from the study are also summarized in Table 4.5.

4.4.3 Discussion: Other Kinds of Explanations

As we mentioned before, there were some instances where the participants from Study 1 generated explanations that are outside the scope of any of the explanation types discussed in Section 4.2.2. These were marked as "Other" in Figure 4.6. In the following, we discuss three cases that we found interesting.

Post-hoc explanations Notice that parts of an MCE that contribute to the executability of a plan need not be explained in situations where the robot is explaining plans that have *already been done* as opposed to those that are being proposed for execution. The rationale behind this is that if the human sees an action, that would not have succeeded in his model, actually end up succeeding (e.g. the robot had managed to go through a corridor that was blocked by rubble) then he can rationalize that event by updating his own model (e.g. there must not have been a rubble there). This seems to be a viable approach to further reduce size (c.f. selective property of explanations in (Miller, 2017a)) of explanations in a post-hoc setting, and is out of scope of explanations developed here.

Identification of Explicit Foils Identification of explicit foils can help reduce the size of explanations as well. In the explanations introduced in Section 4.2 the foil was implicit – i.e. why this plan *as opposed to all other plans*. However, when the implicit foil can be estimated (e.g. top-K plans expected by the human or in estimation of the

		Outcome	Comments
Study-1	H1	✓	Participants largely agreed that model reconciliation was a necessary and sufficient part of the explanation process.
	H1a	✓	Participants preferred explanations that are complete, and preserve contrastive property across multiple hypothesis.
	H2	✗	Participants did not care to minimize size of explanations, i.e. exclude irrelevant details.
	H2a	✗	Explanations generated by participants in the free form condition were largely of the form of MPEs.
	H2b	✓	Participants did generate MCEs when their communication capability was explicitly restricted.
Study-2	H1	✓	Participants could identify the optimality of the given plan with complete explanations.
	H2	✓	Participants could identify suboptimality of the given plan for incomplete explanations.
	H3	✓ / ?	Some participants asked for explanations even for explicable plans, though the majority did not.

Table 4.5: Summary of Results from the User Studies.

mental model from the foil as done in (Sreedharan *et al.*, 2018c, 2021d; Valmeekam *et al.*, 2020)) then the explanations can only include information on why the plan in question is better than those other options (which are either not executable or costlier). Some participants provided explanations contrasting some of these foils in terms of (and in addition to just) the model differences.

Cost-based reasoning Finally, a kind of explanation that was attempted by some participants involved a cost analysis of the current plan with respect to foils (in addition to model differences, as mentioned above). Such explanations have been studied extensively in previous planning literature (Fox *et al.*, 2017; Smith, 2012) and is still relevant for plan explanations on top of the model reconciliation process.

4.5 Related Work

We started out in the introduction with the premise that plan explanations cannot be a soliloquy but is rather a means of reconciling differences between the AI model and the user expectations or the mental model of the user, thereby establishing common grounds with the human in the loop (Allan, 2013). Much of the work we cited there assume that the model of the planner and the end user are the same. This does not bear out in many applications and we saw some examples of this above. While we referred to relevant work on that topic in the course of our presentation wherever necessary, for a more detailed treaties of the evolution of the world of explainable AI planning, we refer the reader to Chapter 22.

One particular work we want to expand on a bit more here is a recent survey on lessons learned from social sciences on the dynamics of the explanation process in human-human interactions. Miller (2017a) outlines three key properties of explanations – *social* (in being able to model the explainee's expectations), *contrastiveness* (the ability to contrast potential foils), and *selectiveness* (to prioritize model details for explanations). Our approach is inherently social (by explicitly accounting for the mental model of the explainee). We also spent a fair bit of time expounding on the contrastive property in the chapter, while our method of selection is determined by the minimality and monotonicity criterion.[7] While the contrastive property has been the subject of much interest in the explainable AI planning community of late (Hoffmann and Magazzeni, 2019; Miller, 2018), to the best of our knowledge, the model reconciliation process remains the only existing plan explanation process that conforms to all three properties of social, contrastiveness, and selectiveness of explanations, as outlined by Miller (2017a).

Our view of explanation as a model reconciliation process is further supported by studies in the field of psychology which stipulate that –

> *"... explanations privilege a subset of beliefs, excluding possibilities inconsistent with those beliefs... can serve as a source of constraint in reasoning..." (Lombrozo, 2006)*

This is achieved in our case by the appropriate change in the expectation of the model that is believed to have engendered the plan in question. Furthermore, Lombrozo (2012) also underline that –

> *"... explanations are typically contrastive... the contrast provides a constraint on what should figure in a selected explanation..." (Lombrozo, 2012)*

[7] As we mentioned before, since minimal explanations in the model reconciliation framework are not unique, the selectiveness criterion can be further explored (Zahedi *et al.*, 2019) in the context of preferences over logically equivalent explanations. Recent work exploring the representation of plan properties for purposes of explanation (Eifler *et al.*, 2020b) can also help in this cause.

This is especially relevant in order for an explanation to be self-contained and unambiguous. Hence the requirement of optimality in our explanations, which not only ensures that the current plan is valid in the updated model, but is also better than other alternatives. This is consistent with the notion of optimal (single-model) explanations investigated by Sohrabi *et al.* (2011a) where less costly plans are referred to as *preferred explanations*. The optimality criterion, and argumentation over the human mental model, makes the problem fundamentally different from model change algorithms discussed by Göbelbecker *et al.* (2010); Herzig *et al.* (2014); Eiter *et al.* (2010); Bryce *et al.* (2016); Porteous *et al.* (2015) which focus more on the feasibility of plans or correctness of domains, or tackle model extensions in general without consideration of the human expectations i.e. the human mental model, and the constrastive property of potential foils in it.

The field of epistemic reasoning is also closely related to model reconciliation explanations as studied within this chapter. In fact, works like (Shvo *et al.*, 2020), have already used the epistemic reasoning framework to generalize model reconciliation beyond just classical planning problems. In (Sreedharan *et al.*, 2020a) (Covered in Chapter 18) we also leveraged tools from epistemic planning (Miller, 2017b; Muise *et al.*, 2015) to incorporate the reasoning about model reconciliation explanations into the planning process through the notion of "explanatory actions" or robot actions with purely epistemic effects that result in the update of the human's belief regarding the robot model.

4.6 Concluding Remarks

In this chapter we established the basic framework for model reconciliation and investigated a few specific types of model-reconciliation explanations. In the remainder of Part-I of the thesis, we will be revisiting the problem of generating minimally complete explanations. In particular, we will investigate how one could approximate the generation of such explanations while relaxing some of the core assumptions made by the framework described in this chapter.

Chapter 5

COMPLEXITY OF GENERATING MINIMALLY COMPLETE EXPLANATIONS

In the previous chapter we established the basic framework of model reconciliation explanation and we presented some basic algorithms for generating various types of model reconciliation explanations. However, in the previous chapter we made no attempt to establish the computational complexity of this problem. This chapter is aimed at addressing that specific shortcoming.

Specifically, we will formalize a decision-version of the minimal explanation problem studied there and show that this decision problem is in fact Σ_2^p-complete. The proof will focus on mapping the explanation problem to that of establishing the satisfiability of a particular subclass of quantified boolean formulas. Specifically, one where the quantification is restricted to a set of existentially quantified variables followed by a set of universally quantified variables. The reduction for the membership proof will leverage the universal quantification in the formula to capture the optimality test that is a central part of the explanation generation problem and the existentially quantified variables to capture the explanation. While in the case of the hardness proof, we will use the optimality test to capture the universal quantification and the explanation to capture the existential one.

5.1 Technical Background

We start by providing a brief introduction to some of the relevant complexity classes we will be considering in this chapter and compiling planning into satisfiability problem. Note that the planning models considered for the complexity proof will stay pretty much the same as the previous chapter, except for the fact that we will allow for explicit negative preconditions.

5.1.1 Relevant Complexity Classes

While many of the standard results related to classical planning tend to either fall into NP or PSPACE classes (Bylander, 1994), the problem studied here focuses on a class that is placed between these classes. Figure 5.1 presents a diagrammatic overview of the respective classes and their relationships.

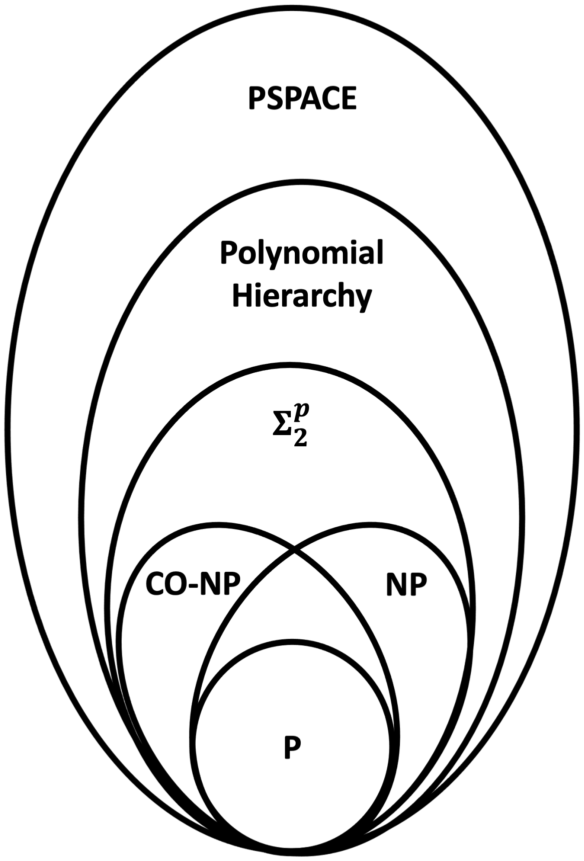


Figure 5.1: A Diagram Illustrating the Relationship of Relevant Complexity Classes. Note That Σ_2^p Is a Member of the Polynomial Hierarchy.

One way to view NP problems, is in terms of the existence of a witness or certificate that can be verified in polynomial time. Following Definition 2.1 by Arora and Barak

(2009), a language L is said to be in NP if

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$$
$$\text{such that } M(x,u) = 1,$$

where M is a polynomial time Turing machine, p a polynomial and u is the witness. On the other hand, a language L is said to be in Σ_2^p (Arora and Barak, 2009, Definition 5.1) if

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \; \forall v \in \{0,1\}^{p(|x|)}$$
$$\text{such that } M(x,u,v) = 1,$$

where M is again a polynomial time Turing machine. Another way to view Σ_2^p is in terms of oracle machines. In fact Σ_2^p can be represented equivalently as NP^{NP}, and can be understood as problems that could be solved by a non-deterministic polynomial time Turing machine with an NP oracle.

The canonical Σ_2^p-complete problem is the special subclass of quantified boolean formula called $QSAT_2$ (Stockmeyer, 1976), where $QSAT_2$ corresponds to the question of satisfiability of a formula of the form

$$\exists X \forall Y \phi,$$

where X and Y are vectors over boolean variables and ϕ is a propositional formula defined over X and Y. Note that per Theorem 4.1 (1) from Stockmeyer (1976), $QSAT_2$ is complete for Σ_2^p regardless of the form. This fact is exploited in our membership proof as we map the $\mathcal{P}^{MRE}$ to a quantified boolean formula which is not necessarily in either CNF or DNF form. In fact, in the formula used for the membership proof, while $\phi_1(X)$ and $\phi_3(X,Z)$ are in CNF, the subformula $\neg\phi_3(X,Y)$ is a negation. Additionally, Theorem 4.1 (2) by Stockmeyer (1976) also shows that the subset $QSAT_2 \cap$ 3-DNF is also complete for Σ_2^p. So in our hardness proof rather than reducing arbitrary quantified boolean formulas into a $\mathcal{P}^{MRE}$, we focus on reducing an instance from the set $QSAT_2 \cap$ 3-DNF.

The polynomial hierarchy (PH) is formed by taking the union over the various classes of the form Σ_i^P, where each class Σ_i^P is defined in the above form with i alternating existential and universal quantifiers (starting with an existential quantifier as in the case of Σ_2^P). Each level in this hierarchy can be characterized by a problem of satisfying a corresponding subclass of quantified boolean formula.

Finally, let us look at the PSPACE complexity class, which covers all the problems that can be solved by a Turing machine with polynomial space (Arora and Barak, 2009). A PSPACE-complete problem is the satisfiability of a TQBF or True Quantified Boolean Formula, where no restriction is placed on the quantification over the variables. While it is known that PH $\subseteq$ PSPACE holds, the problem of establishing PH $\neq$ PSPACE remains an important open problem (or question, as it's not known yet). In fact if, PH = PSPACE, it is known that the polynomial hierarchy collapses to some finite level (Papadimitriou, 1994).

5.1.2 Encoding Planning Problems as SAT

A popular way of solving planning problems, particularly when there exists a planning horizon (say T), is to encode it as propositional satisfiability problems (SAT) (Kautz *et al.*, 1996). The most common encoding for the problem uses propositional variables to capture whether a fluent is true at each possible time step and whether an action was executed at a time step. If $\mathcal{M}$ is the planning model, then we would have $T \times (|F^{\mathcal{M}}| + |A^{\mathcal{M}}|)$ variables. The encoding has three important classes for clauses (a) clauses that describe a component of a model, i.e, initial state, goal, or action definition, (b) explanatory frame axioms that enforce the requirement that any change in variable value should correspond to the execution of an action that could have caused the change and finally (c) clauses to enforce the fact that concurrent action execution is not possible. For example, let $a_i \in A^{\mathcal{M}}$ and $p \in add^{\mathcal{M}}(a)$, then as part of class (a) of clauses you would have clause of the form $a_i^t \Rightarrow p^{t+1}$, for all time steps t. Similarly, an example of an explanatory clause for p would be

$$\neg p^t \wedge p^{t+1} \Rightarrow \bigvee \{a_i^t | p \in add^{\mathcal{M}}(a)\}$$

Effectively the clause asserts that p could only have been turned true if one of these actions was executed. Finally, we assert that the actions cannot be executed concurrently using the clause

$$\bigwedge_{a \in A} (a^t \Rightarrow \bigwedge_{a_j \in A, a_j \neq a} \neg a_j^t)$$

5.2 Complexity Results for Model Reconciliation Explanation Problems

We are interested in the computational complexity of solving model reconciliation explanation problems. It is however rather obvious that checking whether *any* solution exists is a trivial problem:

Proposition 7. *Let $\mathcal{P}^{MRE} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \pi_R^* \rangle$ be a model reconciliation explanation problem. The question whether there exists a valid explanation can be decided in* constant time. *More precisely, the answer is always* yes.

The reason why the answer is always yes is because we could always simply compute the difference between the sets $\Gamma(\mathcal{M}_h^R)$ and $\Gamma(\mathcal{M}^R)$ and present these differences as explanation. (And we know that this is always possible, so we don't need to do so just to decide whether this explanation exists – it always does.) Thus, *computing* such an explanation is harder than deciding whether one exists; computing it is a linear problem. This corresponds to the class of explanation called *model patch explanation* presented in Chapter 4.

While model patch explanations are technically correct, they might in practice not be the best explanations as it would involve resolving differences that are irrelevant, in that they didn't cause confusion. Recall that the reason for the necessity of explaining something is that the robot's plan either isn't even a plan at all in the human's model, or just not optimal. So any explanation should restrict to finding reasons pointing to any of these facts.

Thus, what we are interested in is finding a *minimal* explanation, i.e., we want to present an explanation to the human user that involves the fewest possible number of model changes so that the observed plan is an optimal solution in his/her model – even if there are still some differences to the robot's model (of which the human would then still be unaware of). This corresponds to Minimally Complete Explanations discussed earlier.

To turn this *optimization problem* into a *decision problem*, we introduce (as it is usually done) an additional parameter representing the criterion that's being optimized – in our case the number of performed changes. Formally:

Definition 11. *For a given model reconciliation explanation problem*
$\mathcal{P}^{MRE} = \langle \mathcal{M}^R, \mathcal{M}^R_h, \pi^*_R \rangle$,
we define the ***optimal model reconciliation explanation decision problem*** *as:*

Given $\mathcal{P}^{MRE}$ *and a natural number* $k \in \mathbb{N} \cup \{0\}$*, does there exist a valid explanation* $\mathcal{E} = \langle \epsilon^+, \epsilon^- \rangle$ *for* $\mathcal{P}^{MRE}$*, such that* $|\epsilon^+| + |\epsilon^-| = k$*? (We call this* ***MRE****-k.)*

We are going to show that the problem is Σ_2^p-complete, which we show in the next two sections, one showing membership, the other showing hardness.

To prove the computational complexity, we will focus on the canonical Σ_2^p complete problem called $QSAT_2$ (Stockmeyer, 1976), where $QSAT_2$ corresponds to the question of satisfiability of a formula of the form $\exists X \forall Y \phi$, where X and Y are disjoint sets of boolean variables and ϕ is a propositional formula defined over X and Y. When we focus on specific forms propositional formula, say 3-DNF, we will denote it as $QSAT_2 \cap$ 3-DNF.

5.2.1 Membership Proof

Our first task would be to establish the fact that **MRE-k** is in a fact a member of the complexity class Σ_2^p. We will do so by reducing the problem into a $QSAT_2$ problem. In particular, by mapping a model reconciliation explanation problem $\mathcal{P}^{MRE} = \langle \mathcal{M}^R, \mathcal{M}_h^R, \pi_R^* \rangle$ into $QSAT_2$ of the form

$$\exists X, Z \phi_1(X) \wedge \neg(\exists Y \phi_2(X, Y)) \wedge \phi_3(X, Z),$$

such that $|X| = k \cdot |E^+ \cup E^-|$, where $E^+ = \Gamma(\mathcal{M}^R) \backslash \Gamma(\mathcal{M}_h^R)$ and $E^- = \Gamma(\mathcal{M}_h^R) \backslash \Gamma(\mathcal{M}^R)$ are the set of propositional variables that will capture the *possible* individual model updates, Y corresponds to the propositional variables required to encode a planning problem where the maximum plan length is limited to $|\pi_R^*| - 1$ that will be used to capture the possible shorter plans and Z includes the propositional variables required to encode a planning problem where the maximum plan length is limited to $|\pi_R^*|$ which will be used to encode the validity of π_R^*. Following the variables, $\phi_1(X)$ is a CNF formula that enforces the fact that only explanations of size k are possible, $\phi_2(X, Y)$ is a CNF formula that encodes whether given the explanation a plan of length $|\pi_R^*| - 1$ can achieve the goal in the updated model and finally, $\phi_3(X, Z)$ is a CNF formula that encodes whether given the explanation π_R^* is valid in the updated model. Applying the negation and moving the quantification upfront, we get the pre-nex $QSAT_2$ form

$$\exists X, Z \forall Y \phi_1(X) \wedge \neg \phi_2(X, Y) \wedge \phi_3(X, Z)$$

Now the important parts of this compilation are encoding the enforcement of explanation length (through $\phi_1(X)$) and encoding planning models in such a way that they can reflect the effects of model updates captured by a particular instantiation of X variables (used in $\phi_2(X, Y)$ and $\phi_3(X, Z)$.

Encoding Explanation Length

The variable set X consists of k propositional variables for each individual model update, i.e,

$$X = \bigcup_{\tau_i \in E^+ \cup E^-} \{\tau_i^1, ..., \tau_i^k\},$$

where each τ_i^m can be thought of as capturing the fact that the model update τ_i is applied at step m. Now our requirement is to enforce that only a single model update is applied at a given step. This is exactly done by $\phi_1(X)$, where $\phi_1(X)$ is a conjunction of clauses of the form

$$\tau_i^m \Rightarrow \bigwedge_{\tau_j \in E^+ \cup E^- \wedge \tau_j \neq \tau_1} \neg \tau_j^m.$$

$\phi_1(X)$ will contain a clause for every pair of model updates $\tau_i, \tau_j \in E^+ \cup E^-$ and every step $m \in \{1, ..., k\}$). Now $\phi_1(X)$ cannot be true if for any step more than one model update variable is true.

Encoding Planning Models Conditioned on Model Updates

Now the second part of the encoding requires that we have a way to capture the horizon-limited planning problem, that reflects the model updates captured by a given instantiation of X. This will be used in both $\phi_2(X, Y)$ and $\phi_3(X, Z)$. The encoding will be based on the planning-as-SAT encoding discussed by Kautz *et al.* (1996). Let $\phi_T^{\mathcal{M}}$ be the unmodified original SAT encoding (in CNF form) corresponding to a model $\mathcal{M} = \langle F, A, I, G \rangle$ for a planning horizon T. $\phi_T^{\mathcal{M}}$ consists of propositional variables for each fluent and action indexed by possible time steps, i.e., it contains $T \times (|F| + |A|)$ variables. Now we can broadly divide the clauses in $\phi^{\mathcal{M}}$ into three categories, (a) formulas capturing a model component, say action preconditions, effects, initial state and goal (specifically, there exists a model parameter $\tau \in \Gamma(\mathcal{M})$ for every clause), (b) explanatory frame axioms (that ensures that every state change is accounted for in terms of an action execution) and (c) formulas to enforce the fact that at any time step only one action is executable.

To allow for the model update, we will start with setting the encoding to be equal to the human model ($\phi_T^{\mathcal{P}^{MRE}} = \phi_T^{\mathcal{M}_h^R}$). We first augment the model component clauses in this model. Specifically, let ψ be a clause corresponding to a model component that is part of the human model but not part of the robot model, and let the corresponding model parameter be $\tau_j \in E^-$. Then we replace ψ in $\phi_T^{\mathcal{P}^{MRE}}$ with a clause

$$(\neg\tau_j^1 \wedge ... \wedge \neg\tau_j^k) \Rightarrow \psi,$$

This clause captures the fact that if the model component is not removed in the k explanation steps, then the model component should be considered when coming up with the plan. Note that this is still a clause as conjunction is on the left side of the implication.

Let ψ' be a clause corresponding to a model component that is part of the robot model but missing from the human model, with a corresponding model parameter be $\tau_j \in E^+$. We add a conjunction of the form given below to $\phi_T^{\mathcal{P}^{MRE}}$

$$(\tau_j^1 \Rightarrow \psi_i) \wedge ... \wedge (\tau_j^k \Rightarrow \psi')$$

That is, the model component needs to hold if the corresponding explanation is provided at any explanation step.

Now we will remove all the original explanatory frame axioms and add one that covers action definitions from both models, i.e., for ever fluent $f \in F^{\mathcal{M}^R}$ and each time step m up to $T-1$ we add a clause of the form $\neg f^m \wedge f^{m+1} \Rightarrow A_{add}^f$, where

$A^f_{add} = \{a \mid a \in A^{\mathcal{M}^R} \wedge f \in add^{\mathcal{M}^R} \cup add^{\mathcal{M}^R_h}\}$. We can similarly add an explanatory fluent for the deletes.

Now to account for the explanation, we will add new clauses that will ensure that to use any previously missing adds or deletes at a time step, the respective model update should be performed at some explanation step or not performed at all if it was an effect that was not part of the robot model. Finally we leave the action exclusion clauses unmodified in the new model $\phi_T^{\mathcal{P}^{MRE}}$. This is sufficient as the human, robot and updated model all share the same action names. We can now see that this encoding is equivalent to the updated model.

Proposition 8. *Let $\vec{x}$ be a specific instantiation of the variable X corresponding to a model update $\mathcal{E} = \langle \epsilon^+, \epsilon^- \rangle$, such that $\epsilon^+ = \{\tau_1, ..., \tau_r\}$, $\epsilon^- = \{\bar{\tau}_1, ..., \bar{\tau}_p\}$ and $|\epsilon^+| + |\epsilon^-| = k$. Now let $\phi_{\vec{x}}(X)$ be a logical conjunction of the form*

$$x^1_{\tau_1} \wedge ... \wedge x^r_{\tau_r} \wedge x^{r+1}_{\bar{\tau}_1} \wedge ... \wedge x^k_{\bar{\tau}_p}$$

Now for the combined logical formula

$$\phi_{\vec{x}}(X) \wedge \phi_1(X) \wedge \phi_T^{\mathcal{P}^{MRE}},$$

every instantiation of propositional variables that satisfies the formula corresponds to a plan for $\mathcal{M}^R_h + \mathcal{E}$.

This fact should be obvious from the validity of the original encoding. Any differences in the encoding are only those related to the explanations. For example, in the new encoding the enforcement of a positive precondition f for an action a_l at time step m that could be removed by a model update is going to be, $\neg\tau^1 \wedge ...\neg\tau^k \Rightarrow (a^m_l \Rightarrow f^{m-1})$, where $\tau = a_l$*-has-pos-prec-*f. So if τ^i is set true at any of the k sets then the precondition needs no longer to be satisfied.

We get $\phi_2(X, Y)$ by using the $\phi_T^{\mathcal{P}^{MRE}}$ encoding but for time horizon $|\pi_R^*|-1$ (the encoding also includes NOOP actions to allow for shorter plans) and we define $\phi_3(X, Z)$ to be equal to

$$\phi_{|\pi_R^*|}^{\mathcal{P}^{MRE}}(X, Z) \wedge \phi_{\pi_R^*},$$

where $\phi_{\pi_R^*} = a_1^1 \wedge ...a_{|\pi_R^*|}^{|\pi_R^*|}$, a_i^i is the variable in Z corresponding to the i^{th} action in plan π_R^* for time step i.

Lemma 1. *A given problem $\mathcal{P}^{MRE}$ has a k-sized explanation if and only if*

$$\exists X, Z \forall Y \phi_1(X) \wedge \neg\phi_2(X, Y) \wedge \phi_3(X, Z)$$

is satisfiable.

This follows directly from the construction and Proposition 8. The formula is only satisfied if there exists an instantiation of X corresponding to an explanation of length k (enforced by $\phi_1(X)$) that allows for the validity of the current plan (enforced by $\phi_3(X, Z)$) and doesn't allow for any shorter plans (enforced by $\neg\phi_2(X, Y)$). This leads us to:

Theorem 1. ***MRE**-k is in Σ_2^p*

5.2.2 Hardness Proof

To prove that the problem **MRE-**k is Σ_2^p-hard, we will provide a polynomial reduction of a $QSAT_2 \cap$ 3-DNF instance (which has been shown to be Σ_2^p-complete (Stockmeyer, 1976)) into a k-bounded Model Reconciliation Explanation problem $\mathcal{P}^{MRE}$ thus solving **MRE-**k. To present the reduction, consider an arbitrary $QSAT_2 \cap$ 3-DNF instance $\exists X \forall Y\ \phi$, where ϕ is a disjunction consisting of N disjcunts (i.e., conjunctions) denoted as $C_1, .., C_N$ (each of size 3 as ϕ is in 3-DNF) defined over the propositions in X and Y.

As mentioned earlier, we will map the problem of checking the satisfiability of a propositional formula over a universal quantification to that of an optimality check. In particular, we will construct a planning model where the plan space covers the space of all possible instantiations of the universally quantified variables. We then establish the satisfiability of the universally quantified formula by showing that no plan (i.e., a specific instantiation of the variables) can satisfy the negation of the formula. This follows from the fact that

$$\forall Y \phi \iff \neg(\exists Y \neg\phi)$$

Though in our case, there is not only a universally quantified variable set Y but an existentially quantified variable set X. We will map this existentially quantified variable set into the initial state of the problem and will allow the model reconciliation problem to select any possible instantiation of X as part of the explanation. Thus mapping the problem to

$$\exists X \forall Y \phi \iff \exists X \neg(\exists Y \neg\phi)$$

That is, a valid explanation will show that there exists an initial state for which there exists no shorter action sequence that can satisfy the goal $\neg\phi$. We will also add some additional constraints in the two models that will form our model reconciliation explanation problem $\mathcal{P}^{MRE}_{QSAT}$ to ensure that the plan being explained will be optimal after all the differences between the models have been resolved.

The exact construction of the $\mathcal{P}^{MRE}_{QSAT}$ problem is given below. Let us first start by defining the fluent set and the action names. Let the fluent set F be defined as

$$F = F_X \cup F_Y \cup F_N \cup F_G \cup F_{D_1} \cup F_{D_2},$$

where F_X and F_Y are the sets of fluents corresponding to X and Y, respectively (i.e., we could just set $F_X = X$ and $F_Y = Y$), F_N consists of a fluent per disjunct in ϕ

(i.e., we could just set $F_N = \{C_1, \ldots, C_N\}$), $F_G = \{g\}$ contains a single goal fluent to be used by the models, F_{D_1} is a set of dummy fluents such that $|F_{D_1}| = |X| + 1$ and F_{D_2} is a second set of dummy fluents such that $|F_{D_2}| = |Y| + N + 1$. Now the action names A to be shared between the two models would be such that $|A| = |Y| + 3 \cdot |F_N| + |F_{D_2}| + |F_{D_1}| + 1$, where we would have an action for each of the fluents in the corresponding fluent subsets Y and F_{D_2}, and an action for each conjunction of the propositional formula ϕ. Finally, there are $|F_{D_1}| + 1$ "goal actions", where there are $|F_{D_1}|$ goal actions by which the goal can be established if $\neg\phi$ can be achieved and there is one goal action to be used as part of π. Specifically, we will represent the action names as follows

$$A = A_Y \cup A_{\neg\phi} \cup A_{(G,\neg\phi)} \cup A_{D_2} \cup A_{(G,D_2)}$$

Now we will define a model reconciliation explanation problem $\mathcal{P}^{MRE}_{QSAT} = \langle \mathcal{M}_1, \mathcal{M}_2, \pi \rangle$. The models are defined as $\mathcal{M}_1 = \langle F, A, I^{\mathcal{M}_1}, G \rangle$ and $\mathcal{M}_2 = \langle F, A, I^{\mathcal{M}_2}, G \rangle$, where $I^{\mathcal{M}_1} = X$ and $I^{\mathcal{M}_2} = F_{D_1}$. Note that the models only differ in their initial states.

We will now go on defining each of the actions. Starting with A_Y, an action $a^i_Y \in A_Y$ (for a variable $y_i \in Y$) is defined by $pre_+(a^i_Y) = pre_-(a^i_Y) = \emptyset$, $add(a^i_Y) = \{f^i_Y\}$, and $del(a^i_Y) = \emptyset$. That is, the action definition for a^i_Y is empty but for a single add effect that sets the fluent corresponding to the variable y_i true ($f^i_Y \in F_Y$).

Next come the actions in $A_{\neg\phi}$, which will help us test whether for a given instantiation of X and Y, the negation of the propositional formula $\neg\phi(x, y)$ is satisfiable. Note that when we negate the 3-DNF ϕ, we obtain a 3-CNF $\neg\phi$, where each (i^{th}) conjunction C_i gets turned into a disjunction (clause) C'_i given by $\{p^i_1, p^i_2, p^i_3\}$ (where the literals got inverted, i.e., switched from negated to positive and vice versa). Now for each literal in C'_i we will define an action $a^{i,j}_{\neg\phi}$, $1 \leq j \leq 3$, as follows.

- if p^i_j is positive we have:

$pre_+(a^{i,j}_{\neg\phi}) = \{f^j_{X\cup Y}\}$, $pre_-(a^{i,j}_{\neg\phi}) = \emptyset$,

$add(a^{i,j}_{\neg\phi}) = \{f^i_N\}$, and $del(a^{i,j}_{\neg\phi}) = \emptyset$

- and if p^i_j is negative then we define it as

  $pre_+(a^{i,j}_{\neg\phi}) = \emptyset$, $pre_-(a^{i,j}_{\neg\phi}) = \{f^j_{X\cup Y}\}$,

  $add(a^{i,j}_{\neg\phi}) = \{f^i_N\}$, and $del(a^{i,j}_{\neg\phi}) = \emptyset$,

where $f^i_N \in F_N$ is an indicator variable identifying whether the clause C'_i is satisfied and $f^j_{X\cup Y} \in F_X \cup F_Y$ is the fluent corresponding to the proposition. That is, the fluent becomes a positive precondition if it was a positive literal in the clause (resulting from negating the conjunction), otherwise it becomes a negative precondition. If at least one of the literals in the clause is satisfied the clause is satisfied (captured by the add effect).

Now one way for the goal g to be satisfied would be to satisfy all the negated clauses in ϕ, which is captured by the set of actions $A_{(G,\neg\phi)} = \{a^1_{(G,\neg\phi)}, ..., a^{|X|+1}_{(G,\neg\phi)}\}$. Here we have a possible goal action for each fluent in F_{D_1}. The actions here are defined such that $pre_+(a^i_{(G,\neg\phi)}) = F_N \cup \{f^i_{D_1}\}$, $pre_-(a^i_{(G,\neg\phi)}) = \emptyset$, $add(a^i_{(G,\neg\phi)}) = \{g\}$, and $del(a^i_{(G,\neg\phi)}) = \emptyset$, where $f^i_{D_1} \in F_{D_1}$. As we will see, once we complete the mapping of the satisfaction problem into the explanation problem, all the shorter action sequences that the explanation would need to invalidate would use these goal generating actions.

The optimal plan π that needs to be explained is given by $\pi = \langle a^1_{D_2}, ..., a^{|F_{D_2}|}_{D_2}, a_{(G,D_2)}\rangle$. This plan contains all the actions that are part of $A_{D_2} = \{a^1_{D_2}, ..., a^{|F_{D_2}|}_{D_2}\}$ and $A_{(G,D_2)} = \{a_{(G,D_2)}\}$, such that

- $pre_+(a^i_{D_2}) = pre_-(a^i_{D_2}) = \emptyset$, and for all $f^i_{D_2} \in F_{D_2}$:

  $add(a^i_{D_2}) = \{f^i_{D_2}\}$, $del(a^i_{D_2}) = \emptyset$, and

- $pre_+(a_{(G,D_2)}) = F_{D_2}$, $pre_-(a_{(G,D_2)}) = \emptyset$,

  $add(a_{(G,D_2)}) = \{g\}$, and $del(a_{(G,D_2)}) = \emptyset$.

This brings us to the important properties (captured by the following propositions and lemmata) that will let us establish the soundness of the reduction.

Proposition 9. *π is optimal in $\mathcal{M}_2$.*

In $\mathcal{M}_2$ none of the variables in F_{D_1} are true, neither can any action turn it true. Thus none of the goal actions in $A_{(G,\neg\phi)}$ can be used, leaving the model to use all actions in π to achieve the goal.

Proposition 10. *One can reach a state that captures (in terms of the truth values of F_Y) any possible instantiation of the variables Y by a plan of length less than $|\pi| - N - 1$.*

The proposition follows directly given the size of $|A_Y|$ and the fact that the actions in this set do not have preconditions.

Proposition 11. *Any possible instantiation of the variables X can be captured by the initial state of the updated model formed from a model update of size $|X|$.*

Note that the model update takes the form of $\langle \epsilon^+, \epsilon^- \rangle$, such that $\epsilon^+ \subseteq X$ and $\epsilon^- \subseteq F_{D_1}$. Thus one can create an explanation that sets some subset of $X' \subseteq X$ true, by making ϵ^+ equal to that subset (in terms of F_X) and selecting a subset of F_{D_1} as ϵ^-, such that $|\epsilon^-| = |X| - |\epsilon^+|$. Since $|F_{D_1}| = |X| + 1$ and $|\epsilon^-|$ is a non-negative number upper-bounded by $|X|$, we can always find a subset of F_{D_1} of size $|X| - |\epsilon^+|$.

Proposition 12. *For a model update $\mathcal{E} = \langle \epsilon^+, \epsilon^- \rangle$, where X^{ϵ^+} contains the values from X corresponding to the update, there exists an action sequence π' that is a valid plan in $\mathcal{M}_2 + \mathcal{E}$ such that $|\pi'| < |\pi|$, if and only if there exists some instantiation of variables Y (say $Y^{\pi'}$), such that for X^{ϵ^+} and $Y^{\pi'}$ it satisfies $\neg\phi$ (denoted as $(X^{\epsilon^+}, Y^{\pi'}) \models \neg\phi$).*

This follows directly from the fact that an action sequence of length less than $|\pi|$ can only satisfy the goal by using an action in $A_G^{\neg\phi}$, which requires satisfying all the clauses in $\neg\phi$. Similarly all instantiations of Y and testing validity of $\neg\phi$ can be done in less than $|\pi|$ steps, which brings us to the lemma:

Lemma 2. *For the model reconciliation explanation problem* $\mathcal{P}_{QSAT}^{MRE} = \langle \mathcal{M}_1, \mathcal{M}_2, \pi \rangle$, *there exists a valid explanation of size* $|X|$ *if and only if the corresponding* $QSAT_2$ $\exists X \forall Y \phi$ *is satisfiable.*

Proof. This lemma can be directly built from the previous three propositions. If there exists a valid explanation of size $|X|$ that means no plan of size less than $|\pi|$ is valid, that means there exists an instantiation of variables X (say $X^{\mathcal{E}}$), such that for all instantiations Y' of variables Y, we have

$$(X^{\mathcal{E}}, Y') \not\models \neg\phi$$

which means, we have for all Y' of Y

$$(X^{\mathcal{E}}, Y') \models \phi$$

Similarly if the $QSAT_2$ formula was not satisfiable, then for every $X^{\mathcal{E}}$ we should have at least one instantiation Y' for which $(X^{\mathcal{E}}, Y') \models \neg\phi$. In the updated model we can now construct a plan that corresponds to Y' and the evaluation of $\neg\phi$ for Y' which should satisfy an action in $A_G^{\neg\phi}$. □

Which bring us to the theorem.

Theorem 2. ***MRE**-k is Σ_2^p-hard*

This theorem follows directly from Lemma 2 and the fact that $QSAT_2 \cap$ 3-DNF is Σ_2^p-complete. Finally, Theorem 1 and Theorem 2 brings us to our central result.

Theorem 3. ***MRE**-k is Σ_2^p-complete*

5.3 Related Work

While the complexity of the original model-reconciliation explanation has gone unexplored, there are complexity results from related problems that give us some clues about the actual complexity. For one, Sreedharan *et al.* (2020a) (also discussed in Chapter 18), showed PSPACE-completeness for providing a plan and a set of model updates such that the plan is valid in both the robot and the updated human model. Of course, here there is the additional complexity of identifying the plan and the problem overlooks the complexity of establishing the optimality, a central concern in the original model reconciliation formulation. On the other hand, Lin and Bercher (2021) established that the complexity of updating the model (with any or a minimal number of changes) ensuring the validity of a given plan is an NP-complete problem (for most cases, only a few are in P). However, they do not constrain the model updates to those that align with the target robot model directly, as this is only implicitly provided via the plan that's supposed to be valid. Finally, Vasileiou *et al.* (2021), looked at a variant of model reconciliation that is framed as the problem of finding the shortest logical support of a given propositional formula in the context where the human and robot model are represented as propositional knowledge bases. They discuss a possible membership of this problem in the Σ_2^p class, but unfortunately, again the problem they define is different from the one studied in Chapter 5. Vasileiou *et al.* (2021) looked at a case where there exists a knowledge base associated with the system KB_a and one associated with the human KB_h and there is a logical formula ϕ that needs to be explained such that $KB_a \models \phi$ and $KB_h \not\models \phi$. The explanation here takes the form of a support $\epsilon \subseteq KB_a \wedge KB_h$ such that $KB_h \wedge \epsilon \models \phi$ (as discussed by Vasileiou *et al.* (2021) in Definition 7). Given the generality of this formulation, one could map explanations for a classical planning problem partially into this framework, however it is not the exact problem studied in Chapter 5.

The first point to note is the fact that the explanation here doesn't support removal of rules from the human's knowledge base, a key form of model update discussed in Chapter 5. While the definition does not allow for such a change, their algorithm (Algorithm 2) does include an ad-hoc test for satisfiability of $KB_a \wedge KB_h$ and the algorithm allows for the removal of formulas from KB_h if the conjunction is unsatisfiable. But this doesn't cover all the changes that are supported in Chapter 5. For example, consider a case when an action a has an additional add effect e in the human knowledge base, which manifests in the form of two rules, a rule of the form

$$a^i \Rightarrow e^i$$

and a rule in the explanatory frame axiom that says a is a possible action that can satisfy the transition from $\neg e^*$ to e^*. In this case, the problem encoding of length n (n being the length of the plan being explained), the formula $KB_a \wedge KB_h$ needs not be unsatisfiable, especially if the plan being explained is not using a or e. However this rule could prevent some ϕ, say there doesn't exist a plan of length shorter than n, from being entailed without its removal. For example, let there be an action a_2 that directly sets the goal true if e is satisfied and there are no actions in KB_a that could have satisfied e. Now without removing this additional add effect there will always be a shorter plan. Unfortunately, after the first satisfiability test Vasileiou *et al.* (2021) don't provide any other way of removing rules from the human's knowledge base.

Next the paper breaks down the problem of explaining optimality of the plan into two separate problems. First it explains the validity and then it explains optimality. For explaining validity they mention adding the plan and the goal as constraints into the planning model as additional clauses and then testing for satisfiability. If the encoding is unsatisfiable, the authors mention that they "add the missing actions as part of the explanation" (Vasileiou *et al.*, 2021, page 5). If this means they add all missing information in KB_h for actions in the plan, this will already result in more information that the one considered in the original model reconciliation work (cf. Chapter 5). However, if they have some procedure to find the minimal information needed to be added to ensure optimality, this could still result in longer explanations. This is because choices made to ensure validity has an impact on the information needed to be provided to ensure optimality. As such the problem of finding model updates to ensure validity and optimality cannot be separated. For example, consider a case where the plan is invalid because a precondition for an action is unsatisfied in the human model. Now assume there are two possible minimal updates to ensure validity. One could say that the unsatisfied precondition is not part of that action in the knowledge base KB_h or there exists a previously missing effect of an earlier action that can satisfy this precondition (though it's not needed in KB_a). However one could build the rest of the model in such a way that adding the add effect information would result in other shorter plans being feasible. Say there are actions which can satisfy the goal directly whose only precondition in KB_h is satisfied by this effect, while those actions have extra preconditions in KB_a. This means the choice to introduce the add effect could make the explanation longer. Similarly we can create domains where the choice to remove the precondition would result in longer explanations.

5.4 Concluding Remarks

One of the immediate points of interest is the comparison of the complexity of model-reconciliation with the complexity of classical planning. When no information about the problem to solve is given, worst case complexity is PSPACE-complete (Bylander, 1994). Unless the polynomial hierarchy collapses the problem of model reconciliation is thus easier than simple plan existence. This comparison is however not perfectly fair since in model reconciliation we have a plan given as an input thus bounding possible changes. While bounded plan existence is still PSPACE-complete since plan length can be bounded logarithmically, it turns NP-complete when encoded unarily (thus simulating the situation of model reconciliation where the bound is not given as number but explicitly via an input plan). Model reconciliation is thus harder than the respective bounded plan existence problem unless the polynomial hierarchy collapses.

Another consequence of the proof is the existence of an alternate problem formulation for generating a minimally complete explanation, namely by mapping it to $QSAT_2$ of increasing explanation length. This means one could use fast quantified boolean formula solvers to generate such explanations. One of the future directions for the work may be to investigate whether the use of this compilation provides an advantage over the A^* model space search proposed by Chapter 4.

Chapter 6

MODEL RECONCILIATION IN THE PRESENCE OF MODEL UNCERTAINTY

In the previous chapters, we formalized the framework of model reconciliation and looked at the problem of generating minimal explanations. However, the version of the framework and algorithms we studied made one central assumption, namely the access to the human's mental model $\mathcal{M}_h^R$. Although we made this assumption as a first step towards formalizing the model reconciliation process, this can be hard to achieve in practice. Instead, the agent may end up having to explain its decisions with respect to a *set of possible models* which is its best estimation of the human's knowledge state learned in the process of interactions. Sets of possible models can be concisely represented as planning models with *annotations* for possible preconditions and effects (Nguyen *et al.*, 2017; Bryce *et al.*, 2016). In theory, the robot could try to compute MCEs for each possible configuration, however this can result in situations where the explanations computed for individual models independently are not consistent across all the possible target domains. Thus, in the case of model uncertainty, such an approach cannot guarantee that the resulting explanation will be acceptable. Instead, in this chapter we will look at methods that take into account the entire space of possible human models while generating the explanations. We will also discuss how the method developed for generating explanation in the presence of model uncertainty could be used to generate explanations for multiple users.

6.1 Running Example: The Urban Search and Reconnaissance Domain

The second domain is a typical Urban Search and Reconnaissance (USAR) domain[1] (Bartlett, 2015; Murphy *et al.*, 2008) where a remote robot is put into disaster response operation often controlled partly or fully by an external human commander, as shown in Figure 6.1. The robot's job is to scout areas that may be otherwise harmful to humans and report on its surroundings as instructed by the external supervisor. The scenario can also have other internal agents (humans or robots) with whom the robot needs to coordinate. The USAR domain thus affords a rich set of characteristics, such as multiple agents distributed in space, partial observability, evolving domain models, and so on. The USAR domain is also ideal for visualizing to non-expert participants in comparison to, for example, logistics-type domains which should ideally be evaluated by experts. This became an important factor while designing the user studies. The USAR domain is thus at once close to motion planning as easily interpreted by non-experts but also incorporates typical aspects of task plans such as preconditions and effects in terms of rubble removal, collapsed halls, etc. and relevant abilities of the robot. As such, simulated USAR scenarios provide an ideal testbed (Bartlett, 2015; Zhang *et al.*, 2015; Talamadupula *et al.*, 2014) for developing algorithms for effective human-robot interaction.

Here, even though all agents start off with the same model – i.e. the blueprint of the building – their models diverge as the internal agent interacts with the scene. Due to the disaster new paths may have opened up due to collapsed walls or old paths may no longer be available due to rubble. This means that plans that are valid and optimal in the robot's model may not make sense to the external commander. In the

[1]Video demonstrations of all examples in this domain can be viewed at `https://ibm.box.com/v/aij-model-reconciliation`. (Duration 5:52)

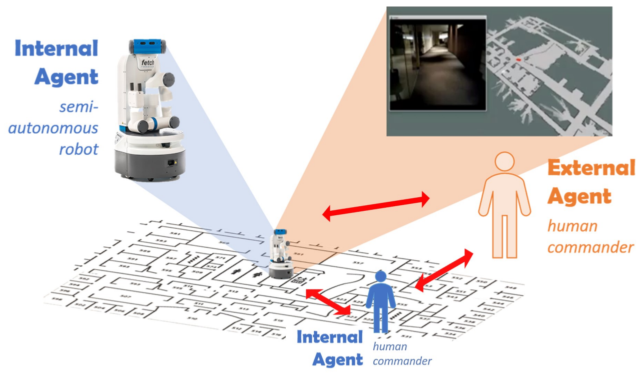


Figure 6.1: A Typical USAR Domain with an Internal Robot and an External Commander.

scenario in Figure 6.2, the robot is tasked to go from its current location marked blue to conduct reconnaissance in the location marked orange. The green path is most optimal in its current model but this is blocked in the externals mental model while the expected plan in the mental model is no longer possible due to rubble. Without removing rubble in the blocked paths, the robot can instead communicate that the path at the bottom is no longer blocked. This explanation preserves the validity and optimality of its plan in the updated model (even though further differences exist).

6.2 Supporting Mental Model Uncertainty

As we mentioned before, we have assumed thus far that $\mathcal{M}_h^R$ is known as a first step towards formalizing the model reconciliation process. This is hard to achieve in practice. Instead, the agent may end up having to explain its decisions with respect to a *set of possible models* which is its estimation of the human's knowledge state learned in the process of interactions. For example, consider the work in (Bryce *et al.*, 2016) where model drift is tracked via filters in the form of a set of models, or in (Nguyen *et al.*, 2017) where a set of models is computed to fit to observed plan traces. Such uncertainty or incompleteness over a model can be represented in the

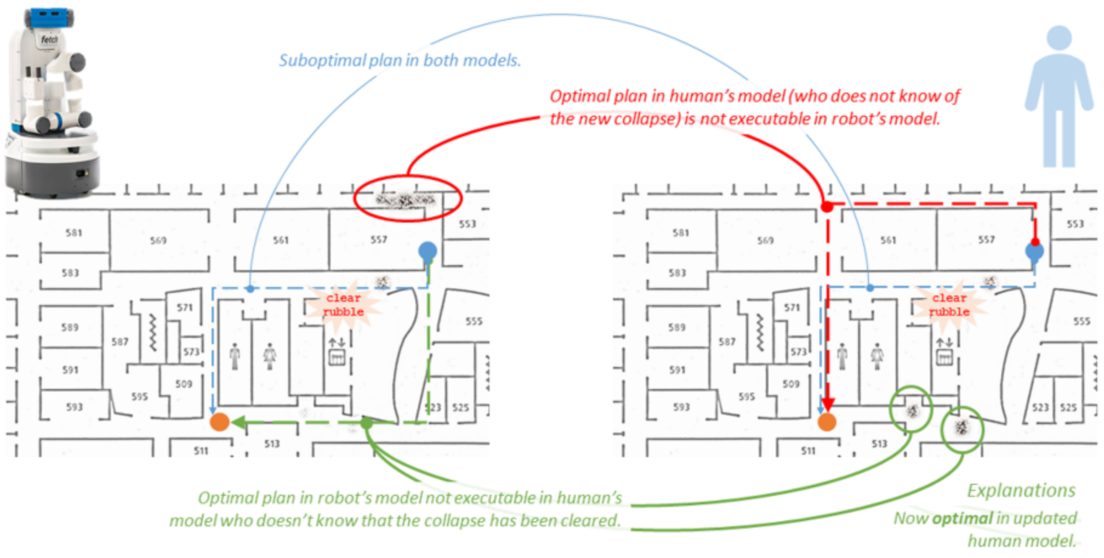

Suboptimal plan in both models.
Optimal plan in human's model (who does not know of the new collapse) is not executable in robot's model.
clear rubble
clear rubble
Optimal plan in robot's model not executable in human's model who doesn't know that the collapse has been cleared.
Explanations
Now optimal in updated human model.

Figure 6.2: Model Differences in the USAR Domain.

form of *annotated* models or APDDL. (Nguyen *et al.*, 2017) In addition to the standard preconditions and effects associated with actions, it introduces the notion of *possible* preconditions and effects which may or may not be realized in practice.

Definition: **An Annotated Model** is the tuple $\mathbb{M} = \langle F, \ \mathbb{A}, \mathbb{I}, \mathbb{G}, C \rangle$ – where F is a finite set of fluents that define a state $s \subseteq F$, $\mathbb{A}$ is a finite set of annotated actions – and annotated initial and goal states $\mathbb{I} = \langle \mathcal{I}^0, \mathcal{I}^+ \rangle$, $\mathbb{G} = \langle \mathcal{G}^0, \mathcal{G}^+ \rangle$; $\mathcal{I}^0, \mathcal{G}^0, \mathcal{I}^+, \mathcal{G}^+ \subseteq F$. Action $a \in \ \mathbb{A}$ is a tuple $\langle pre(a), \widetilde{pre}(a), add(a), del(a), \widetilde{add}(a), \widetilde{del}(a) \rangle$ where in addition to its *known* preconditions and add/delete effects $pre(a), add(a), del(a), \subseteq F$ each action also contains *possible preconditions* $\widetilde{pre}(a) \subseteq F$ containing propositions that it *might* need as preconditions, and *possible add (delete) effects* $\widetilde{add}(a), \widetilde{del}(a) \subseteq F$) containing propositions that it *might* add (delete, respectively) after execution. $\mathcal{I}^0, \mathcal{G}^0$ (and $\mathcal{I}^+, \mathcal{G}^+$) are the known (and possible) parts of the initial and goal states. Finally, C is the cost associated with each action. We will generally assume the models have unit cost function to simplify the discussion and not explicitly include it in the model definitions.

Each possible condition $f \in \widetilde{pre}(a) \cup \widetilde{add}(a) \cup \widetilde{del}(a)$ has an associated probability $p(f)$ denoting how likely it is to be a known condition in the ground truth model – i.e. $p(f)$ measures the confidence with which that condition has been learned. The sets of known and possible conditions of a model $\mathcal{M}$ are denoted by $\mathbb{S}_k(\mathcal{M})$ and $\mathbb{S}_p(\mathcal{M})$ respectively. An *instantiation* of an annotated model $\mathbb{M}$ is a classical planning model where a subset of the possible conditions have been realized, and is thus given by the tuple $inst(\mathbb{M}) = \langle F, A, \mathcal{I}, \mathcal{G} \rangle$, initial and goal states $\mathcal{I} = \mathcal{I}^0 \cup \chi;\ \chi \subseteq \mathcal{I}^+$ and $\mathcal{G} = \mathcal{G}^0 \cup \chi;\ \chi \subseteq \mathcal{G}^+$ respectively, and action $A \ni a = \langle pre(a) \leftarrow pre(a) \cup \chi;\ \chi \subseteq \widetilde{pre}(a), (add(a)/del(a)) \leftarrow (add(a)/del(a)) \cup \chi;\ \chi \subseteq (\widetilde{add}(a)/\widetilde{del}(a)) \rangle$. Clearly, given an annotated model with k possible conditions, there may be 2^k such instantiations, which forms its *completion set.* (Nguyen *et al.*, 2017)

Definition: **Likelihood** $\mathcal{L}$ of instantiation $inst(\mathbb{M})$ of an annotated model $\mathbb{M}$ is:

$$\mathcal{L}(inst(\mathbb{M})) = \prod_{f \in \mathbb{S}_p(\mathbb{M}) \wedge \mathbb{S}_k(inst(\mathbb{M}))} p(f) \quad \times \prod_{f \in \mathbb{S}_p(\mathbb{M}) \setminus \mathbb{S}_k(inst(\mathbb{M}))} (1 - p(f))$$

As discussed before, such models turn out to be especially useful for the representation of human (mental) models learned from observations, where uncertainty after the learning process can be represented in terms of model annotations (Nguyen *et al.*, 2017; Bryce *et al.*, 2016). Let $\mathbb{M}_H^R$ be the culmination of a model learning process and $\{\mathcal{M}_{h_i}^R\}_i$ be the completion set of $\mathbb{M}_H^R$. One of these models is the actual ground truth (i.e. the human's real mental model). We refer to this as $g(\mathbb{M}_H^R)$. We will explore now how this representation will allow us to compute *conformant explanations* that can explain with respect to all possible mental models and *conditional explanations* that engage the explainee in dialogue to minimize the size of the completion set to compute shorter explanations.[2]

Conformant Explanations

In this situation, the robot can try to compute MCEs for each possible configuration. However, this can result in situations where the explanations computed for individual models independently are not consistent across all possible target domains. Thus, in the case of model uncertainty, such an approach cannot guarantee that the resulting explanation will be acceptable.

Instead, we want to find an explanation such that $\forall i \; \pi^*_{\widehat{\mathcal{M}}^R_{h_i}} \equiv \pi^*_{\mathcal{M}^R}$ (as shown in Figure 6.3). This is a single model update that makes the given plan optimal (and hence explained) in all the updated domains (or in all possible domains). At first glance, it appears that such an approach, even though desirable, might turn out to be prohibitively expensive especially since solving for a *single* MCE involves search in the model space where each search node is an optimal planning problem. However, it turns out that the same search strategy can be employed here as well by representing the human mental model as an *annotated* model. Condition (3) for an MCE (c.f. Section 4.2) now becomes –

(3) $C^{g(\mathbb{M}^R_h)}(\pi) = C^*_{g(\mathbb{M}^R_h)}$

This is hard to achieve since it is not known which is the actual mental model of the human. So we want to preserve the optimality criterion for all (or as many)

[2]The purpose of this section is to demonstrate how existing notions of conditional and conformant solutions in planning can be adopted for the explanation process equally well in the presence of uncertainty over the human mental model. While there are significant differences between how conditional or conformant explanations work with respect to their planning counterparts, it may be worth exploring the state-of-the-art (Albore *et al.*, 2009; Bonet and Geffner, 2005) in those fields to further develop on the concepts introduced in the chapter.

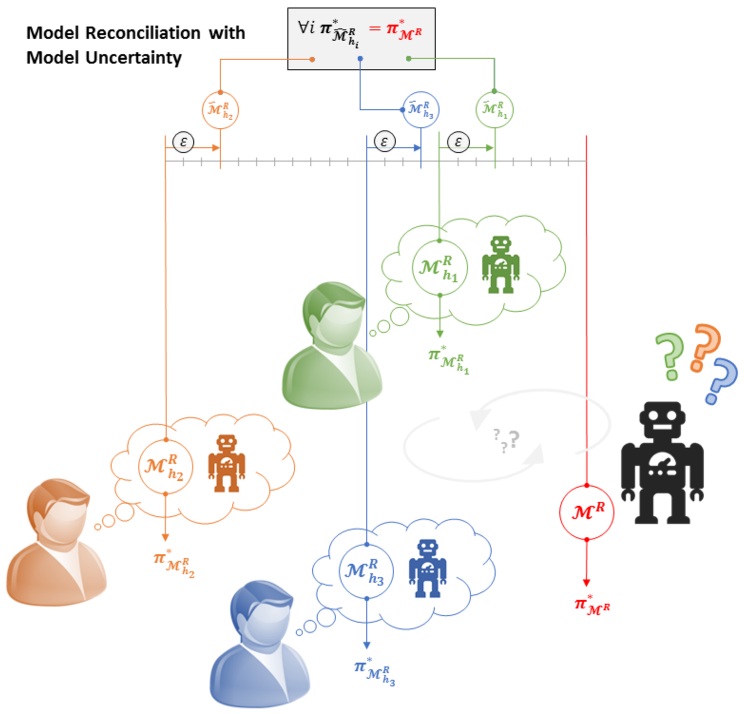

Model Reconciliation with Model Uncertainty
$\forall i\ \pi^*_{\widehat{\mathcal{M}}^R_{h_i}} = \pi^*_{\mathcal{M}^R}$
$\widehat{\mathcal{M}}^R_{h_2}$
$\widehat{\mathcal{M}}^R_{h_3}$
$\widehat{\mathcal{M}}^R_{h_1}$
$\mathcal{E}$
$\mathcal{E}$
$\mathcal{E}$
$\mathcal{M}^R_{h_1}$
$\pi^*_{\mathcal{M}^R_{h_1}}$
$\mathcal{M}^R_{h_2}$
$\pi^*_{\mathcal{M}^R_{h_2}}$
$\mathcal{M}^R_{h_3}$
$\pi^*_{\mathcal{M}^R_{h_3}}$
$\mathcal{M}^R$
$\pi^*_{\mathcal{M}^R}$

Figure 6.3: Model Reconciliation in the Presence of Model Uncertainty or Multiple Explainees.

instantiations of the incomplete estimation of the mental model. Keeping this in mind, we define *robustness* of an explanation for an incomplete mental models as the probability mass of models where it is a valid explanation.

Definition: **Robustness** of an explanation $\mathcal{E}$ is given by –

$$R(\mathcal{E}) = \sum_{inst(\widehat{\mathcal{M}}^R_h) \text{ s.t. } C^{inst(\widehat{\mathcal{M}}^R_h)}(\pi) = C^*_{inst(\widehat{\mathcal{M}}^R_h)}} \mathcal{L}(inst(\widehat{\mathcal{M}}^R_h))$$

Definition: **A Conformant Explanation** is such that $R(\mathcal{E}) = 1$.

A conformant explanation thus ensures that the given plan is explained in all the models in the completion set of the human model. Let's look at an example. Consider again the USAR domain (Figure 6.4), the robot is now at P1 (blue) and

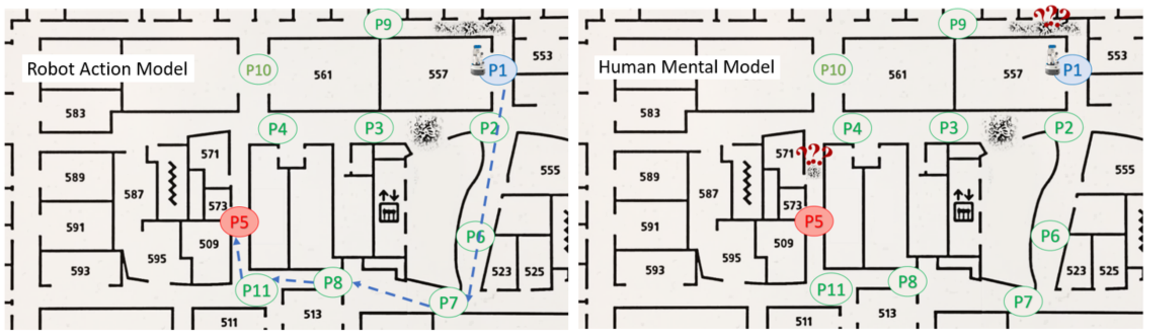


Figure 6.4: Back to Our USAR Scenario: The Robot Plan Is Marked in Blue and Uncertain Parts of the Human Model Is Marked with Red Question Marks.

needs to collect data from P5. While the commander understands the goal, she is under the false impression that the paths from P1 to P9 and P4 to P5 are unusable (red question marks). She is also unaware of the robot's inability to use its hands. On the other hand, while the robot does not have a complete picture of her mental model, it understands that any differences between the models are related to (1) the path from P1 to P9; (2) the path from P4 to P5; (3) its ability to use its hands; and (4) whether the it needs its arm to clear rubble. Thus, from the robot's perspective, the mental model can be one of sixteen possible models (one of which is the actual one). Here, a conformant explanation for the optimal robot plan (blue) is as follows –

```
Explanation >> remove-known-INIT-has-add-effect-hand_capable
Explanation >> add-annot-clear_passage-has-precondition-hand_capable
Explanation >> remove-annot-INIT-has-add-effect-clear_path P1 P9
```

Model-Space Search for Conformant Explanations

As we discussed before, we cannot launch an MCE-search for each possible mental model separately, both for issues of complexity and consistency of the solutions.

However, in the following discussion, we will show how we can reuse the model space search from the previous section with a compilation trick.

We begin by defining two models – the most relaxed model possible $\mathcal{M}_{max}$ and the least relaxed one $\mathcal{M}_{min}$. The former is the model where all the possible add effects and none of the possible preconditions and deletes hold, the state has all the possible conditions set to true, and the goal is the smallest one possible; while in the latter all the possible preconditions and deletes and none of the possible adds are realized and with the minimal start state and the maximal goal. This means that, if a plan is executable in $\mathcal{M}_{min}$ it will be executable in all the possible models. Also, if this plan is optimal in $\mathcal{M}_{max}$, then it must be optimal throughout the set. Of course, such a plan may not exist, but we are not trying to find one either. Instead, we are trying to find a set of model updates which when applied to the annotated model, produce a new set of models where a *given* plan is optimal. In providing these model updates, we are in effect reducing the set of possible models to a smaller set. The new set need not be a subset of the original set of models but will be equal or smaller in size to the original set. For any given annotated model, such an explanation always exists (entire model difference in the worst case), and we intend to find the smallest one. $\mathbb{M}_h^R$ thus affords the following two models –

$\mathcal{M}_{max} = \langle F, A, \mathcal{I}, \mathcal{G} \rangle$

- initial state $\mathcal{I} \leftarrow \mathcal{I}^0 \cup \mathcal{I}^+$; given $\mathbb{I}$
- goal state $\mathcal{G} \leftarrow \mathcal{G}^0$; given $\mathbb{G}$
- $\forall a \in A$
  - $pre(a) \leftarrow pre(a);\ a \in \mathbb{A}$
  - $add(a) \leftarrow add(a) \cup \widetilde{add}(a);\ a \in \mathbb{A}$

- $del(a) \leftarrow del(a);\ a \in\ \mathbb{A}$

$\mathcal{M}_{min} = \langle F, A, \mathcal{I}, \mathcal{G} \rangle$

- initial state $\mathcal{I} \leftarrow \mathcal{I}^0$; given $\mathbb{I}$

- goal state $\mathcal{G} \leftarrow \mathcal{G}^0 \cup \mathcal{G}^+$; given $\mathbb{G}$

- $\forall a \in A$

  - $pre(a) \leftarrow pre(a) \cup \widetilde{pre}(a);\ a \in\ \mathbb{A}$

  - $add(a) \leftarrow add(a);\ a \in\ \mathbb{A}$

  - $del(a) \leftarrow del(a) \cup \widetilde{del}(a);\ a \in\ \mathbb{A}$

As explained before, $\mathcal{M}_{max}$ is a model where all the add effects hold and it is easiest to achieve the goal, and similarly $\mathcal{M}_{min}$ is the model where it is the hardest to achieve the goal. These definitions might end up creating inconsistencies (e.g. in an annotated `BlocksWorld` domain, the `unstack` action may have add effects to make the block both `holding` and `ontable` at the same time), but the model reconciliation process will take care of these.

Proposition 7 – For a given MRP $\Psi = \langle \pi, \langle \mathcal{M}^R, \mathbb{M}_h^R \rangle \rangle$, if the plan π is optimal in $\mathcal{M}_{max}$ and executable in $\mathcal{M}_{min}$,then conditions (1) and (3) from the definition of valid model reconciliation explanation (as defined in Section 4.2) hold for all i.

This now becomes the new criterion to satisfy in the course of search for an MCE for a set of models. We again reuse the state representation in Section 4.2.1. We start the `MEGA*-Conformant` search (Algorithm 3) by first creating the corresponding $\mathcal{M}_{max}$ and $\mathcal{M}_{min}$ model for the given annotated model $\mathbb{M}_H^R$. While the goal test for the original MCE only included an optimality test, here we need to both check the optimality of the plan in $\mathcal{M}_{max}$ and verify the correctness of the plan in $\mathcal{M}_{min}$. As stated in Proposition 7, the plan is only optimal in the entire set of possible models if it satisfies both tests. Since the correctness of a given plan can be verified in polynomial time with respect to the plan size, this is a relatively easy test to perform.

The other important point of difference between the algorithm mentioned above and the original MCE is how we calculate the applicable model updates. Here we consider the superset of model differences between the robot model and $\mathcal{M}_{min}$ and the differences between the robot model and $\mathcal{M}_{max}$. This could potentially mean that the search might end up applying a model update that is already satisfied in one of the models but not in the other. Since all the model update actions are formulated as set operations, the original MRP formulation can handle this without any further changes. The models obtained by applying the model update to $\mathcal{M}_{min}$ and $\mathcal{M}_{max}$ are then pushed to the open queue.

Proposition 8 – $\mathcal{M}_{max}$ and $\mathcal{M}_{min}$ only need to be computed once – i.e. with a model update $\mathcal{E}$ to $\mathbb{M}$: $\mathcal{M}_{max} \leftarrow \mathcal{M}_{max} + \mathcal{E}$ and $\mathcal{M}_{min} \leftarrow \mathcal{M}_{min} + \mathcal{E}$.

These models form the new $\mathcal{M}_{min}$ and $\mathcal{M}_{max}$ models for the set of models obtained by applying the current set of model updates to the original annotated model. This proposition ensures that we no longer have to keep track of the current list of models or recalculate $\mathcal{M}_{min}$ and $\mathcal{M}_{max}$ for the new set.

Contingent Explanations

Conformant explanations can contain superfluous information – i.e. asking the human to remove non-existent conditions or add existing ones. In the previous example, the second explanation (regarding the need of the hand to clear rubble) was already known to the human and was thus superfluous information. Such redundant information can be annoying and may end up reducing the human's trust in the robot. This can be

Algorithm 3 `MEGA*-Conformant`

```
1: procedure MCE-Search
2: Input: MRP $\langle \pi^*, \langle \mathcal{M}^R, \mathbb{M}_h^R \rangle \rangle$
3: Output: Explanation $\mathcal{E}^{MCE}$
4: Procedure:
5:     fringe $\leftarrow$ Priority_Queue()
6:     c_list $\leftarrow$ {}                                                    ▷ Closed list
7:     $\pi_R^* \leftarrow \pi^*$                                                 ▷ Optimal plan being explained
8:     $\mathcal{M}_{max},\ \mathcal{M}_{min} \leftarrow (\mathbb{M}_h^R)$        ▷ Proposition 8
9:     fringe.push($\langle \mathcal{M}_{min}, \mathcal{M}_{max}, \{\} \rangle$, priority = 0)
10:    while True do
11:        $\langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max}, \mathcal{E} \rangle, c \leftarrow$ fringe.pop()
12:        if $C^{\widehat{\mathcal{M}}_{max}}(\pi_R^*) = C^*_{\widehat{\mathcal{M}}_{max}} \wedge \delta_{\widehat{\mathcal{M}}_{min}}(\mathcal{I}_{\widehat{\mathcal{M}}_{min}}, \pi_R^*) \models \mathcal{G}_{\widehat{\mathcal{M}}_{min}}$ then
13:            return $\mathcal{E}$                                            ▷ Proposition 7
14:        else
15:            c_list $\leftarrow$ c_list $\cup \langle \widehat{\mathcal{M}}_{max}, \widehat{\mathcal{M}}_{min} \rangle$
16:            for $f \in \{\Gamma(\widehat{\mathcal{M}}_{min}) \cup \Gamma(\widehat{\mathcal{M}}_{max})\} \setminus \Gamma(\mathcal{M}^R)$ do
17:                $\lambda \leftarrow \langle 1, \langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max} \rangle, \{\}, \{f\} \rangle$      ▷ Removes f from $\widehat{\mathcal{M}}$
18:                if $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max}) \rangle, \lambda) \notin$ c_list then
19:                    fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max}) \rangle, \lambda),$
                                $\mathcal{E} \cup \lambda \rangle,\ c+1$)
20:            for $f \in \Gamma(\mathcal{M}^R) \setminus \{\Gamma(\widehat{\mathcal{M}}_{min}) \cup \Gamma(\widehat{\mathcal{M}}_{max})\}$ do
21:                $\lambda \leftarrow \langle 1, \{\langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max} \rangle, \{f\}, \{\} \rangle$      ▷ Adds f to $\widehat{\mathcal{M}}$
22:                if $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max}) \rangle, \lambda) \notin$ c_list then
23:                    fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max}) \rangle, \lambda),$
                                $\mathcal{E} \cup \lambda \rangle,\ c+C_\lambda$)
```

avoided by –

- Increasing the cost of model updates involving uncertain conditions relative to those involving known preconditions or effects. This ensures that the search prefers explanations that contain known conditions. By definition, such explanations will not have superfluous information.

- However, sometimes such explanations may not exist. Instead, we can convert conformant explanations into *conditional* ones. This can be achieved by turning each model update for an annotated condition into a question and only provide an explanation if the human's response warrants it – e.g. instead of asking the human to update the precondition of `clear_passage`, the robot can first ask if the human thinks that action has a precondition `hand_usable`. Thus, one way of removing superfluous explanations is to reduce the size of the completion set by gathering information from the human. Consider the following exchange in the USAR scenario –

```
R : Are you aware that the path from P1 to P4 has collapsed?
H : Yes.
> R realizes the plan is optimal in all possible models.
> It does not need to explain further.
```

If the robot knew that the human thought that the path from P1 to P4 was collapsed, it would know that the robot's plan is already optimal in the human mental model and hence be required to provide no further explanation. This form of explanations can thus clearly be used to cut down on the size of conformant explanations by reducing the size of the completion set.

Definition: **A Conditional Explanation** is represented by a policy that maps the annotated model (represented by a $\mathcal{M}_{min}$ and $\mathcal{M}_{max}$ model pair) to either a question regarding the existence of a condition in the human ground model or a model update request. The resultant annotated model is produced, by either applying the model update directly into the current model or by updating the model to conform to human's answer regarding the existence of the condition.

In asking questions such as these, the robot is trying to exploit the human's (lack of) knowledge of the problem in order to provide more concise explanations. This can be construed as a case of lying by omission and can raise interesting ethical considerations (Chakraborti and Kambhampati, 2019b). Humans, during an explanation process, tend to undergo this same "selection" process (Miller, 2017a) as well in determining which of the many reasons that could explain an event is worth highlighting.

Modified AO^*-search to find Conditional Explanations We can generate conditional explanations by either performing post-processing on conformant explanations or by performing an AND-OR graph search with AO^*(Nilsson, 1980). AO^* is a heuristic search procedure for acyclic AND-OR graph search, that is guaranteed to identify the optimal solution, provided the heuristics are admissible and monotonic.

Here each model update related to a known condition forms an OR successor node while each *possible* condition can be applied on the current state to produce a pair of AND successors, where the first node reflects a node where the annotated condition holds while the second one represents the state where it does not. So the number of possible conditions reduces by one in each one of these AND successor nodes. This AND successor relates to the answers the human could potentially provide when asked about the existence of that particular possible condition. Note that this AND-OR graph will not contain any cycles as we only provide model updates that are consistent with the robot model and hence we can directly use the AO^* search here.

Throughout this section, we will use h=0 as our heuristic and unit cost for explanations and queries. AO^* search can be understood to be operating in two distinct phases, in the first phase the algorithm considers the current best partial solution and identifies a node to expand. This is a top-down operation and the next phase is a bottom-up cost and label revision stage. Here both the label and the cost of the newly expanded node are propagated back up the graph. The acyclicity of our setting ensures that this backward propagation can be easily carried out. This is done by adding any parents of an updated node into the set S and the procedure continues until the set is empty. All goal nodes (i.e. explanations holds in all remaining models) are marked with SUCCESS label. Each parent node receives the label of the successor node with the minimum value. In the case of an AND successor, the parent only receives the SUCCESS label if all the possible children has the SUCCESS label. The search ends when the root node is assigned the SUCCESS label.

Unfortunately, if we used the standard AO^* search, it will not produce a conditional explanation that contains this "less robust" explanation as one of the potential branches in the conditional explanation. This is because, in the above example, if the human had said that the path was free, the robot would need to revert to the original conformant explanation. Thus the cost of the subtree containing this solution will be higher than the one that only includes the original conformant explanation.

To overcome this shortcoming, we introduce a discounted version of the AO^* search where the cost contributed by a pair of AND successors is calculated as –

$$min(node1.h\_val,\ node2.h\_val) + \gamma * max(node1.h\_val,\ node2.h\_val)$$

where node1 and node2 are the successor nodes and node1.h_val, node2.h_val are their respective h-values. Here γ represents the discount fact and controls how much the search values short paths in its solution subtree. When $\gamma = 1$, the search becomes standard AO^* search and when $\gamma = 0$, the search myopically optimizes for short branches (at the cost of the depth of the solution subtree). The rest of the algorithm stays the same as the standard AO^* search. Though with this modification, AO^* is no longer guarateed to generate optimal solutions when $\gamma \neq 1$. The pseudocode is provided in Algorithm 4.

Anytime Explanations

As we will see later in the evaluations, both the algorithms discussed above can be computationally expensive, in spite of the compilation trick to reduce the set of possible models to two representative models. However, as we did previously with MCEs, we can also shoot for an approximate solution by relaxing the minimality requirement of explanation to achieve much shorter explanation generation time when required. For this we introduce an anytime depth first explanation generation algorithm. Here, for each state, the successor states include all the nodes that can be generated by applying the model edit actions on all the known predicates and two possible successors for each possible condition – one where the condition holds and one where it does not. Once the search reaches a goal state (a new model where the target plan is optimal throughout its completion set), it queries the human to see if the assumptions it has made regarding possible conditions hold in the human mental model (the list of model updates made related to possible conditions). If all the assumptions hold in the human model, then we return the current solution as the final explanation (or use the answers to look for smaller explanations), else continue the search after pruning the search space using the answers provided by the human. Such approaches may also be able to facilitate iterative presentation of model reconciliation explanations to the human. (Zakershahrak *et al.*, 2019)

The pruning can be performed efficiently by keeping track of all the human an-

Algorithm 4 `MEGA*-Conditional`

```
1: procedure AO*-Search
2: Input:  MRP $\langle \pi^*, \langle \mathcal{M}^R, \mathcal{M}_h^R \rangle\rangle, \gamma$
3: Output: Explanation $\mathcal{E}^{MCE}$
4: Procedure:
5:     c_list ← {}                                                        ▷ Closed list
6:     $\pi_R^* \leftarrow \pi^*$                                          ▷ Optimal plan being explained
7:     $M_{min}^H$ ←MIN_MODEL($M^H$)                                       ▷ Generates $M_{min}$ as per definition
8:     $M_{max}^H$ ←MAX_MODEL($M^H$)                                       ▷ Generates $M_{max}$ as per definition
9:     G← Node($\langle \mathcal{M}_{min}^H, \mathcal{M}_{max}^H, \{\}\rangle$)
10:    while G.label != Success do
11:        current_node ← Unexpanded_node(G)     ▷ Return an unexpanded node in the current best path
12:        S ← {curent_node}
13:        curent_node.successors ← GetSuccessors(current_node)
14:        while S not empty() do
15:            node ← S.pop()     ▷ Refer to (Nilsson, 1980) to see how to prioritize which nodes to remove
16:            min_val ← 0
17:            label ← None
18:            for succ in node.successors do
19:                if succ is a OR_Succ then
20:                    node1 ← succ
21:                    if min_val > node1.h_val then
22:                        min_val = node1.h_val
23:                        label = node1.label
24:                if succ is a AND_Succ then
25:                    node1, node2 ← succ
26:                    tmp_val = min(node1.h_val, node2.h_val) + $\gamma$ * max(node1.h_val, node2.h_val)
27:                    if min_val > tmp_val then
28:                        min_val = tmp_val
29:                        if node1.label == node2.label then
30:                            label = node1.label
31:                node.label = label
32:                nodel.h_val = 1 + min_val
33:            Add all parents of node to S
```

Algorithm 5 Procedures used by `MEGA*-Conditional`

```
1: procedure GetSuccessors(node, $\mathcal{M}^R$)
2:     min_state, max_state ← node.state
3:     Known_predicates ← $\Gamma(min\_state) \cap \Gamma(max\_state)$
4:     Possible_predicates ← $\Gamma(min\_state) \Delta \Gamma(max\_state)$
5:     OR_actions_deletes ← {Known_predicates $\setminus \Gamma(\mathcal{M}^R)$}
6:     OR_actions_adds{$\Gamma(\mathcal{M}^R) \setminus$Known_predicates}
7:     AND_actions ← Possible_predicates
8:     Succ_nodes ← Set()
9:     for a ∈ OR_actions_adds do
10:        tmp_node = Node($\langle \Gamma^{-1}(min\_state \cup a), \Gamma^{-1}(max\_state \cup a) \rangle$)
11:        tmp_node ← Evaluate_Node(tmp_node)
12:        Succ_nodes.push(OR_succ(tmp_node)))
13:    for a ∈ OR_actions_deletes do
14:        tmp_node = Node($\langle \Gamma^{-1}(min\_state \setminus a), \Gamma^{-1}(max\_state \setminus a) \rangle$)
15:        tmp_node ← Evaluate_Node(node)
16:        Succ_nodes.push(OR_succ(tmp_node))
17:    for a ∈ AND_actions do
18:        tmp_node1 = Node($\langle \Gamma^{-1}(min\_state \cup a), \Gamma^{-1}(max\_state \cup a) \rangle$)
19:        tmp_node2 = Node($\langle \Gamma^{-1}(min\_state \setminus a), \Gamma^{-1}(max\_state \setminus a) \rangle$)
20:        tmp_node1 ← Evaluate_Node(tmp_node1)
21:        tmp_node2 ← Evaluate_Node(tmp_node2)
22:        Succ_nodes.push(AND_succ(tmp_node1, tmp_node2))
       return Succ_nodes

23: procedure Evaluate_Node(node)
24:    if Check_For_Goal(node) then                ▷ Refer to MEGA*-Conformant for goal test
25:        node.h_val ← 0
26:        node.label ← SUCCESS
27:    else
28:        node.h_val ← heuristic(node)
       return node
```

swers and enforcing these specifications only at the time of expansion of new nodes. Algorithm 6 presents a depth-first search approach for an anytime solution. Here we add an additional variable $\mathcal{A}$ to the search node to keep track of the possible assumptions that we have made for any given search path. The `TEST_ASSUMPTION` denotes the function responsible for testing the set of assumptions during the goal test. `TEST_ASSUMPTION` returns the set of assumptions that were invalidated by the human $\mathcal{A}_{invalid}$ and we can return the current search path as a solution if the invalid set is empty. We will use the validated and invalidated assumption to update our current search stack (via the `UPDATE_STACK` function).

6.2.1 Supporting Multiple Humans in the Loop

While generating explanations for a *set of models*, the robot is essentially trying to cater to multiple human models at the same time. We posit then that the same approaches can be adopted to situations when there are *multiple humans* in the loop instead of a single human whose model is not known with certainty. As before, computing separate explanations for each agent can result in situations where the explanations computed for individual models independently are not consistent across all the possible target domains. In the case of multiple teammates being explained to, this may cause confusion and loss of trust, especially in teaming with humans who are known (Cooke *et al.*, 2013) to rely on shared mental models. Thus *conformant explanations* can be useful in dealing with not only model uncertainty but also model multiplicity.

In order to do this, from the set of target human mental models we construct an annotated model so that *the preconditions and effects that appear in all target models become necessary ones, and those that appear in just a subset are possible ones.* As before, we find a single explanation that is a satisfactory explanation for the entire

Algorithm 6 `MEGA*-Anytime`

```
1: procedure ANYTIME-EXPLANATION
2: Input:   MRP $\langle \pi^*, \langle \mathcal{M}^R, \mathbb{M}_h^R \rangle\rangle$
3: Output: Explanation $\mathcal{E}$
4: Procedure:
5:     fringe $\leftarrow$ Stack()
6:     $\pi_R^* \leftarrow \pi^*$                                    ▷ Optimal plan being explained
7:     $\mathcal{M}_{max},\ \mathcal{M}_{min}\ \leftarrow (\mathbb{M}_h^R)$                          ▷ Proposition 8
8:     $\mathcal{A} \leftarrow \{\}$                                    ▷ Current assumptions
9:     fringe.push($\langle \mathcal{M}_{min}, \mathcal{M}_{max}, \mathcal{A}, \{\}\rangle$)
10:    while True do
11:        $\langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max}, \mathcal{A}, \mathcal{E}\rangle \leftarrow$ fringe.pop()
12:        if $C^{\widehat{\mathcal{M}}_{max}}(\pi_R^*) = C^*_{\widehat{\mathcal{M}}_{max}} \wedge \delta_{\widehat{\mathcal{M}}_{min}}(\mathcal{I}_{\widehat{\mathcal{M}}_{min}}, \pi_R^*) \models \mathcal{G}_{\widehat{\mathcal{M}}_{min}}$ then
13:            $\mathcal{A}_{valid}, \mathcal{A}_{invalid} \leftarrow$ TEST_ASSUMPTION($\mathcal{A}$)
14:            $\mathcal{A}_{valid} \leftarrow \mathcal{A} \setminus \mathcal{A}_{invalid}$
15:            if $|\mathcal{A}_{invalid}| = 0$ then
16:                return $\mathcal{E}$                              ▷ Proposition 7
17:            else
18:                UPDATE_STACK(fringe, $\mathcal{A}_{valid}, \mathcal{A}_{invalid}$)
19:        else
20:            for $f \in \{\Gamma(\widehat{\mathcal{M}}_{min})\ \cup\ \Gamma(\widehat{\mathcal{M}}_{max})\}\ \setminus\ \Gamma(\mathcal{M}^R)$ do
21:                $\lambda \leftarrow \langle 1, \langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max}\rangle, \{\}, \{f\}\rangle$          ▷ Removes f from $\widehat{\mathcal{M}}$
22:                if $f \notin \{\Gamma(\widehat{\mathcal{M}}_{min})\ \cap\ \Gamma(\widehat{\mathcal{M}}_{max})\}\ \setminus\ \Gamma(\mathcal{M}^R)$ then
23:                    $\mathcal{A} \leftarrow \mathcal{A} \cup f$             ▷ Add to assumptions if possible condition
24:                fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max})\rangle, \lambda)$,
                                  $\mathcal{E}\ \cup\ \lambda\rangle, \mathcal{A}$)
25:            for $f \in \Gamma(\mathcal{M}^R)\ \setminus\ \{\Gamma(\widehat{\mathcal{M}}_{min})\ \cup\ \Gamma(\widehat{\mathcal{M}}_{max})\}$ do
26:                $\lambda \leftarrow \langle 1, \{\langle \widehat{\mathcal{M}}_{min}, \widehat{\mathcal{M}}_{max}\rangle, \{f\}, \{\}\rangle$          ▷ Adds f to $\widehat{\mathcal{M}}$
27:                fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\langle \Gamma(\widehat{\mathcal{M}}_{min}), \Gamma(\widehat{\mathcal{M}}_{max})\rangle, \lambda)$,
                                  $\mathcal{E}\ \cup\ \lambda\rangle,\ \mathcal{A}$)
```

set of models, without having to repeat the standard MRP process over all possible models while coming up with an explanation that can satisfy all of them and thus establish common ground.

While the explanation generation technique may be equivalent, the *explanation process* may be different depending on the setup. For example, while in the case of model uncertainty, the safest approach might be to generate explanations that work for the largest set of possible models, in scenarios with multiple explainees, the robot may have to decide whether it needs to save computational and communication time by generating one explanation to fit all models, or if it needs to tailor the explanation to each human. This choice may depend on the particular domain and the nature of the teaming relationship with the human.

In order to understand this better with the use of an example, we go back to our USAR domain, now with *two* human teammates, one external and one internal. The robot is now positioned at `P1` and is expected to collect data from location `P5`. Before the robot can perform its `surveil` action, it needs to obtain a set of tools from the internal human agent. The human agent is initially located at `P10` and is capable of traveling to reachable locations to meet the robot for the handover. Here the external commander incorrectly believes that the path from `P1` to `P9` is clear, while the one from `P2` to `P3` is closed. The internal human agent, on the other hand, not only believes in the errors mentioned above but is also under the assumption that the path from `P4` to `P5` is not traversable. Due to these different initial states, each of these agents ends up generating a different optimal plan. The plan expected by the external commander requires the robot to move to location `P10` (via `P9`) to meet the human. After collecting the package from the internal agent, the commander expects it to set off to `P5` via `P4`. The internal agent, on the other hand, believes that he needs to travel to `P9` to hand over the package. As he believes that the corridor from

P4 to P5 is blocked, he expects the robot to take the longer route to P5 through P6, P7, and P8 (orange). Finally, the optimal plan for the robot (blue) involves the robot meeting the human at P4 on its way to P5. MEGA*-Conformant finds the smallest explanation which explains this plan to both humans.

In this particular case, since the models differ from each other with respect to their initial states, the initial state of the corresponding annotated model is –

$\mathcal{I}^0 = \{$(at_P1), (at_human P10), ..., (clear_path P10 P9), (clear_path P9 P1)$\}$

$\mathcal{I}^+ = \{$(clear_path P4 P5), (collapsed_path P4 P5)$\}$

where I^+ represents the state fluents that may or may not hold in human's model. The corresponding initial states for M_{min} and M_{max} will be as follows –

$\mathcal{I}_{min} = \{$(at_P1), (at_human P10), ..., (clear_path P10 P9), (clear_path P9 P1)$\}$

$\mathcal{I}_{max} = \mathcal{I}_{min} \cup \{(clear\_pathP4P5), (collapsed\_pathP4P5)\}$

MEGA*-Conformant thus generates the following explanation –

```
Explanation >> add-INIT-has-clear_path P4 P5
Explanation >> remove-INIT-has-clear_path P1 P9
Explanation >> add-INIT-has-clear_path P2 P3
```

The first update specifically helps the internal to understand that the robot can indeed reach the goal through P4, while the next two are relevant for both the explainees to explain why they should meet at P4 rather instead.

6.3 Empirical Evaluations

We performed a set of empirical evaluation to evaluate the computational characteristics of the explanation generation for some benchmark problems, including the time taken for generating the explanations and the size of generated explanations. Our explanation generation system integrates calls to Fast-Downward (Helmert, 2006) for planning, VAL (Howey *et al.*, 2004) for plan validation, and pyperplan (Alkhazraji *et al.*, 2016) for parsing. The results reported here are from experiments run on a 12 core Intel(R) Xeon(R) CPU with an E5-2643 v3@t3.40GHz processor and a 64G RAM. We use three popular planning domains (International Planning Competition, 2011) – BlocksWorld, Logistics and Rover – for our experiments. In order to generate explanations we created the human model by randomly removing parts (preconditions and effects) of the action model (the number of edits made per domain is equal to the model patch explanations, which is reported in Table 4.2). Though the following experiments are only pertinent to action model differences, it does not make any difference at all to the approaches, given the way the state was defined. Also note that these removals, as well as the corresponding model space search, were done in the lifted representation of the domain.

In the experiments, we will look at the empirical properties of generating conformant, conditional, and anytime explanations.

6.3.1 Conformant, Conditional, and Anytime Explanations

To evaluate explanations against a set of mental models, for each domain, we chose ten problems (generated from the IPC problem generators), and created a new domain and problem pair by removing five random predicates. This new domain and problem represent the ground truth human model. Next, we generate the uncertain estimate of this model by moving three random predicates into the annotated list. By doing this, we ensure that the ground truth model remains in the completion list of this incomplete model. For these tests, we assume all the possible conditions are equally likely.

As before, Table 6.1 documents the runtime and the size of explanations generated by each of the algorithms. Note that the `MEGA`*`-Conditional` was run with γ set to 0.4 and the results for the anytime algorithm only presents the time and size of the *first* solution found. Also, both `MEGA`*`-Conditional` and `MEGA`*`-Anytime` expect that it can query the human about the ground truth (each question that the algorithm comes up with is tested against that ground model). The "Question Size" column reports the number of questions that were produced by the search, where each question is related to a single annotated condition, while the "Explanation Size" is the size of the actual explanation presented to the human. For `MEGA`*`-Conditional` and `MEGA`*`-Anytime`, we also report the sum of 'Question Size" and "Explanation Size" in parantheses in the explanation column reflecting the total interaction overhead on the human's end.

Unlike `MEGA`*`-Conditional` and `MEGA`*`-Anytime`, `MEGA`*`-Conformant` generates no questions but may produce superfluous explanations. Thus, in the "Explanation Size" column for `MEGA`*`-Conformant`, we present both the size of the non-superfluous component of the explanation (model updates involving only the known conditions) and the total size of the explanation generated (within parenthesis). The results closely follow intuition. `MEGA`*`-Anytime` generally takes less time. However, since the `MEGA`*`-Anytime` algorithm uses a depth first search we cannot guarantee the quality of the solution. In fact, for more than ten problems the solution generated by `MEGA`*`-Anytime` is strictly worse (in terms Question size + Explanation size) than `MEGA`*`-Conditional` and for the majority of problems the Question size + Explanation size produced by `MEGA`*`-Anytime` is strictly larger than the total size of explanations generated by `MEGA`*`-Conformant`.

Depending on the order in which the successors are visited `MEGA`*`-Anytime` can end up with smaller sequences. While `MEGA`*`-Conformant` tend to terminate faster than `MEGA`*`-Conditional`, the latter produces shorter explanations whenever possible.

Finally, the purpose of compiling the set of possible models into $\mathcal{M}_{max}$ and $\mathcal{M}_{min}$

Problem Instance		Conformant explanations		Conditional Explanations			Anytime Explanations		
		Explanation Size	Time (secs)	Question Size	Explanation Size	Time (secs)	Question Size	Explanation Size	Time (secs)
Blocksworld	p1	3 (6)	134.84	3	5 (8)	140.75	3	2 (5)	23.7
	p2	1 (1)	1.64	0	1 (1)	9.2	0	2 (2)	7.32
	p3	2 (3)	20.56	1	3 (4)	55.91	3	3 (6)	20.51
	p4	1 (2)	11.23	1	2 (3)	128.5	0	9 (9)	21.51
	p5	3 (6)	130.64	3	5 (8)	150.61	3	3 (6)	29.43
	p6	2 (4)	279.71	2	4 (6)	539.2	3	2 (5)	25.78
	p7	2 (5)	343.04	3	1 (4)	495.2	3	3 (6)	26.79
	p8	3 (3)	60.35	0	3 (3)	204.72	0	3 (3)	9.7
	p9	2 (4)	234.7	2	4 (6)	379.21	2	2 (4)	18.57
	p10	1 (3)	218.38	3	2 (5)	444.61	2	2 (4)	19.92
Logistics	p1	2 (4)	62.3	2	4 (6)	99.78	2	2 (4)	21.96
	p2	2 (5)	61.45	3	5 (8)	80.73	3	3 (6)	26.68
	p3	3 (5)	246.23	2	4 (6)	297.71	2	2 (4)	21.6
	p4	2 (5)	54.79	3	5 (8)	72.69	3	3 (6)	21.63
	p5	2 (5)	59.87	3	5 (8)	86.72	3	3 (6)	26.04
	p6	2 (4)	489.36	2	3 (5)	729.42	3	2 (5)	24.54
	p7	2 (5)	402.66	1	2 (3)	544.23	3	3 (6)	28.98
	p8	3 (5)	522.47	2	4 (6)	731.1	3	3 (6)	17.78
	p9	3 (6)	1719.26	3	4 (7)	1535.02	3	1 (4)	22.48
	p10	4 (6)	1747.62	2	5 (7)	1783.33	2	4 (6)	21.2
Rover	p1	2 (2)	3.83	0	1 (1)	8.63	0	3 (3)	9.37
	p2	2 (3)	26.93	1	2 (3)	141.2	2	3 (5)	12.91
	p3	2 (4)	99.02	2	3 (5)	165.82	3	2 (5)	25.62
	p4	3 (4)	102.57	1	3 (4)	253.41	1	4 (5)	13.57
	p5	1 (2)	14.87	0	1 (1)	10.58	3	2 (5)	25.65
	p6	1 (2)	146.21	1	1 (2)	835.16	1	4 (5)	14.15
	p7	2 (3)	182.81	1	2 (3)	599.48	1	3 (4)	15.31
	p8	1 (1)	12.07	0	1 (1)	32.92	0	1 (1)	5.14
	p9	1 (2)	125.49	1	2 (3)	523.48	1	1 (2)	15.74
	p10	1 (2)	89.89	1	2 (3)	525.24	2	1 (3)	19.57

Table 6.1: Runtime and Solution Size for Explanations with Uncertain Mental Models.

is that we no longer need to compute explanations over each individual model in the set of possible models separately (baseline). Table 6.2 illustrates the significant scale-ups we can achieve as a result of this.

# of models →	2	4	8	16
Baseline	10.95	41.71	195.81	936.30
`MEGA`*`-Conformant`	11.11	37.01	117.26	291.88

Table 6.2: Comparison of the Runtime for `MEGA`*`-Conformant` Versus the Time Needed to Run MCE for Every Member of the Completion Set.

6.4 Concluding Remarks

This chapter looked at methods that allow one to generate model-reconciliation explanations in the absence of the exact human model. We also looked at how one could also use methods designed to generate explanations in the presence of model uncertainty to generate explanations for multiple users. Such methods are particularly well suited for scenarios where one would want to help build a common ground between the various users receiving the explanations. In fact, the conformant explanation generation method was used by the decision-support system called MA-RADAR (Sengupta *et al.*, 2018), for exactly this purpose. That system also makes sure that the explanations provided to the user will respect any privacy requirements the various users may have. In the next chapter, we will further relax the requirement for the human mental model and consider cases where even an incomplete version of the human mental model may not be available.

Chapter 7

MODEL-FREE MODEL RECONCILIATION

In the previous chapter, we relaxed the requirement that one would need the exact human mental model, however the method still expects access to an incomplete model representation. Additionally, the degree of effectiveness of the method would be directly dependent on the degree of completeness of the model. In this chapter, we will look at an alternate way to generate model-reconciliation explanations.Specifically, we will investigate the use of learned models that can predict how human expectations could be affected by possible explanations (derived completely from information about the agent model) and in fact show how this method could be viewed as a variation of previous approaches that have been put forth to identify explicable behavior.

We will start by extending model reconciliation to the more general setting of planning with Markov Decision Processes (Section 7.2). The rest of the chapter will investigate how these ideas could be used when the human mental model of the task is unavailable, and will formulate a learning problem that allows us to learn simple models that could be used to identify minimal explanations (Section 7.3). Finally, we will evaluate our method on a set of standard MDP benchmarks and perform user studies to validate its viability (Section 7.4).

7.1 Illustrative Example

Consider a warehouse scenario, where a robot is tasked with moving packages from racks and dropping them off at the dispatch chute. The robot is powered by a battery pack that can be recharged by visiting its docking station. The docking station also doubles as a quality assurance station that the robot needs to visit whenever it picks up a box labeled #013 (which means the box is fragile). The robot's operations are mostly deterministic, apart from a small probability of slipping (0.25) in some cells, that could leave the robot in the same position.

Now suppose the warehouse has just hired a new part-time employee to oversee the operations. The employee is just getting used to this new setting and is puzzled by the robot's decision to once in a while take a detour from the drop-off activity and visit a specific position of the factory floor (which is, in fact, the docking location). If we wished the robot to be explainable, then it would need to be capable of helping the employee better understand the underlying model used by the robot (i.e achieve some form of model reconciliation). Given the fact that the robot may not have an exact model of the user, one way to achieve this could be by providing robot's entire model to the user. Unfortunately, this could easily overwhelm the user.

Another possibility could be to allow the user to specify which robot actions appear inexplicable, and focus on providing facts relevant to those actions. This explanation may still prove to be quite verbose and may in fact not help resolve their confusion. For example, imagine a case where the robot is visiting the station to recharge its batteries and the human says that the visit action is inexplicable. Now even if the robot mentions that visiting the station recharges it, the employee may still be confused if they are under the incorrect assumption that the robot is operating on full battery. Similarly, if the human had expected the robot to go to the docking station due to some confusion regarding the box codes, the human may mark the robot decision to not go to the dropoff as being inexplicable and the explanations that could resolve the confusion may have little to do with that specific action marked as inexplicable.

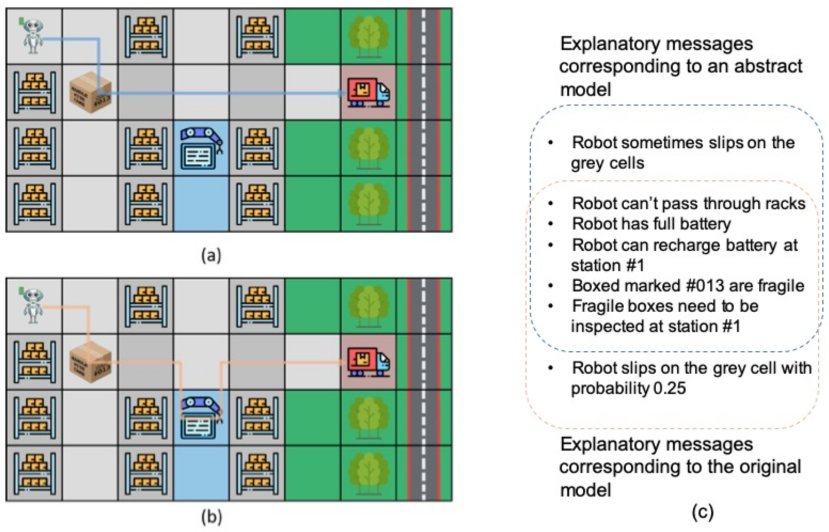

(a)
(b)
Explanatory messages corresponding to an abstract model
Robot sometimes slips on the grey cells
Robot can't pass through racks
Robot has full battery
Robot can recharge battery at station #1
Boxed marked #013 are fragile
Fragile boxes need to be inspected at station #1
Robot slips on the grey cell with probability 0.25
Explanatory messages corresponding to the original model
(c)

Figure 7.1: Subfigure (a) Shows a Visualization of a Trajectory Expected by the User Described in the Illustrative Example, and (b) Shows the Visualization of a Trajectory the User May Observe. Subfigure (c), Shows the Various Explanatory Messages That Could Be Used in This Scenario, Note That the Messages Span Information from Multiple Abstractions of the given Task

7.2 Explanation as Model Reconciliation for MDPs

In this setting the human and robot models are captured as MDPs defined over the same set of states. In particular, we will look at representing each MDP $\mathcal{M}$ by a tuple of the form $\langle S, A, T, R, \gamma, \mu \rangle$, where the S provides the set of possible atomic states, A defines the set of actions, T is the transition function, R the reward, γ the discounting factor (where $0 \leq \gamma < 1$) and μ corresponds to the distribution of possible initial states.

For model paramterization, we will characterize each model by the tuple $\theta = \langle \theta_T, \theta_R, \theta_\gamma, \theta_\mu \rangle$, where the θ_T provides the set of parameters that defines the transition probabilities $P(.|s, a)$, while θ_R the parameters corresponding to the reward function, θ_γ the parameters corresponding to the discount factor and θ_μ the parameters for the initial state distribution. For simple MDP models with atomic states, θ_T contains parameters of the categorical distribution for each transition (θ_μ will contain similar parameters for the initial state distribution), θ_R contains the reward associated with each transition (an $\langle s, a, s' \rangle$ tuple) and θ_γ just contains the value of the discount factor. The specific instantiations of the parameters for each model $\mathcal{M}$ is captured as $\theta(\mathcal{M})$. For simplicity, we will denote each of the unique parameters in the tuple θ using indexes. For example, $\theta_T^{s,a}(\mathcal{M}^R)$, will correspond to the parameters for the distribution $P(.|s, a)$ for the model $\mathcal{M}^R$.

If we use $\mathbb{M}$ to capture the set of all possible models and $\Theta = \theta_T \times \theta_R \times \theta_\gamma \times \theta_\mu$, then model reconciliation operation can be captured as a function $\mathcal{E}_{\langle \mathcal{M}_h^R, \mathcal{M}^R \rangle} : 2^\Theta \rightarrow \mathbb{M}$ that takes in a set of model parameters and generates a new version of the model $\mathcal{M}_h^R$ where the set of specified parameters will be set to values from $\mathcal{M}^R$. For example, $\hat{\mathcal{M}} = \mathcal{E}_{\langle \mathcal{M}_h^R, \mathcal{M}^R \rangle}(\theta_T^{s_1,a})$ will be a new model such that $\theta(\hat{\mathcal{M}})$ will be identical to $\theta(\mathcal{M}_h^R)$, except that $\theta_{T_{\hat{\mathcal{M}}}}^{s_1,a}$, will be equal to $\theta_{T^R}^{s_1,a}$.

Practically, the model reconciliation operation corresponds to the robot informing the human about some part of its model. This communication could incur cost and we can capture this by using the cost function $\mathcal{C} : 2^\Theta \rightarrow \mathbb{R}$ that maps a given set of a threshold to a cost.

Now the question we need to ask is whether the agent is trying to explain its policy or if it is trying to explain some behavior (i.e an execution trace). Most of the earlier work that looks at model reconciliation explanation (cf. (Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2018a,c)) has looked at sequential plans and has generally ignored this differentiation and treated the problem of explaining plans to be same as that of explaining behavior. In general, a given plan or policy compactly represents a set of possible behaviors and the choice of explaining behavior *vs* explaining the plans/policies could affect the content of the explanation being given. For example, when explaining policies there is the additional challenge of presenting the entire policy to the user and the explainer may need to justify action choices for extremely unlikely states or contingencies. On the other hand, when explaining a given set of behaviors the explainer needs to only justify their action choices for cases they actually witnessed. For example, when explaining traces from the warehouse scenario, given the small probability of slipping, the robot may never have to mention what to do when it slips, but on the other hand if we are dealing with full policies, the agent may need to talk about the states where the robot is in the slipped positions and they need to get up from that position and move on.

Explaining policies or plans becomes more relevant when we consider explanatory dialogues where the agent and the user are trying to jointly come to agreement on what policy/plans to follow (eg: decision support systems), while the latter may be more useful when the user is observing some agent operating in an environment.

With respect to policies, we assume that the user is presented with the entire policy. A given policy is said to be explicable to the human, if the policy is optimal for the human model. Therefore the goal of the explainer becomes that of ensuring the optimality of the given policy

Definition 12. A set of parameters $\theta_{\mathcal{E}}$ corresponds to a **complete policy explanation** for the given robot policy $\pi^*_{\mathcal{M}^R}$, if the policy is also optimal for $\mathcal{E}_{\langle \mathcal{M}^R_h, \mathcal{M}^R \rangle}(\theta_E)$ and is said to be the minimally complete policy explanation if there exists no other complete explanation $\theta_{\mathcal{E}'}$, such that, $\mathcal{C}(\theta_{\mathcal{E}'}) < \mathcal{C}(\theta_{\mathcal{E}})$

Finding a complete policy explanation is relatively straightforward (the set of all parameters automatically meets this requirement). The more challenging case becomes that of finding the minimal or the cheapest explanations i.e. the minimally complete explanations. Such minimally complete explanations can be calculated by adopting a search strategy similar to the one presented by Chakraborti *et al.* (2017). The search can start at the human model and try to find the minimal number of parameters that needs to be updated in the human model for the current policy to become optimal. Similar to generating minimally complete explanations, i.e, we can also generate monotonic explanations (i.e explanations where no further information about parameters in the robot model can affect the optimality of the current plan).

In the case of policies, we can also describe explicable planning and balancing cost of explanations with that of choosing policies that are inherently explicable, where inexplicablity score ($\mathcal{I}_{\mathcal{E}}$) of a policy π is defined as

$$\mathcal{I}_{\mathcal{E}}(\pi, \mathcal{M}_h^R) = |E[V_{\pi*}^{\mathcal{M}_h^R}(s)|s \sim \mu_h^R] - E[V_{\pi}^{\mathcal{M}_h^R}(s)|s \sim \mu_h^R]|$$

Where $\pi*$ is the optimal policy in the human model. Explicable planning thus becomes the problem of choosing policies that minimize inexplicability score (Kulkarni *et al.*, 2019a), while minimizing the potential loss in optimality due to the policy choice (since the most explicable plan may not be an optimal policy). Balanced planning, as studied by Chakraborti *et al.* (2019f), proposes going one step further and also takes into account possible savings in inexplicability score that can be achieved by providing explanation (while incurring additional cost of communicating the required explanations).

For explaining behavior, we will look at the simplest case, namely the agent needs to explain a set of behaviors that the user has just observed. We will assume that the observer has full observability of the state and is seeing the robot behavior for the first time. In such a setting, a given trace τ would appear explicable to the user if it could be sampled from their expected MDP policy (i.e a policy optimal in their model) or more generally, i.e $P_{\mathcal{M}_h^R}(\tau|\pi) > \delta$, where δ is some small threshold. [1]

[1]We use δ in the general case to allow for the possibility that people can be surprised by unlikely

Definition 13. A set of parameters $\theta_{\mathcal{E}}$ corresponds to a **complete behavior explanation** for a set of traces $\mathbb{T} = \{\tau_1, ...\tau_n\}$, if $\forall \tau \in \mathbb{T}$, $\exists \pi$ such that $P_{\mathcal{E}_{\langle \mathcal{M}_h^R, \mathcal{M}^R \rangle}(\theta_E)}(\tau|\pi) > \delta$ and π is an optimal policy for the model $P_{\mathcal{E}_{\langle \mathcal{M}_h^R, \mathcal{M}^R \rangle}(\theta_E)}$. The explanation is said to be the minimally complete behavior explanation if there exists no other complete explanation $\theta_{\mathcal{E}'}$, such that, $\mathcal{C}(\theta_{\mathcal{E}'}) < \mathcal{C}(\theta_{\mathcal{E}})$

Note that given the above definition, if δ is set very high it may not be possible to find a complete explanation, as the trace may genuinely contain low probability transitions. In this work we will assume δ to be zero.

While model reconciliation could be an important component of either policy or behavior explanation, the applicability of the model reconciliation explanations on their own for policies is limited by the fact that in all but problems with the smallest state spaces, the user would have trouble going over the entire policy. Thus in these settings, explanatory systems would need to also utilize policy approximation or summarization methods, then allow users the ability to drill down on policy details as required. Since our main goals was to focus on developing approaches that allow us to generate model reconciliation explanations without explicitly defined user models, the rest of the chapter will mostly focus on behavior explanation. In Section 7.5, we will have a brief discussion on how these methods could potentially be extended to policy explanation scenarios.

7.3 Explaining Without Explicit Human Mental Models

Now we will look at how we can identify cheap complete behavior explanations when the human model is unknown. We will go one step further from identifying not only the parameters that need to be explained, but also capturing the right modality/abstractions to present the information about the parameters. That is, we

events of non-zero probability

will no longer assume that the human is using a full MDP model to come up with their decisions. Instead, the robot starts with a set of explanatory messages $\Psi = \{m_1, m_2, ..., m_n\}$ that can be presented to the user. Where the messages correspond to a set of parameter values (the parameters corresponding to a set of messages $\{m_1, ..m_k\}$ is denoted as $\mathcal{E}(\{m_1, ..m_k\})$) of the model as captured in some abstraction of this model and has a corresponding cost ($\mathcal{C}$) associated with it. The abstractions to consider may depend on the specific scenario and the previous information about the intended users (laypeople vs. experts). Some simple possibilities may be to consider qualitative models (say non-deterministic ones instead of stochastic) and considering state abstractions the given task. Note that, technically $\mathcal{E}(\Psi)$, need not span the set of all possible model parameters, but could rather be limited to a subset of parameters identified to be relevant to the given problem. One possible way may be to consider variations of explanation techniques like MSE (Khan *et al.*, 2009) to identify set of possible factors that affect the optimality of each action. In Figure 7.1, the subfigure (c) shows a set of possible explanatory messages for the warehouse domain, that consists of each parameter mapped to some english statement. For models captured using factored representations that use relational or propositional fluents, such statements could be easily generated using templates (cf. (Hayes and Shah, 2017)).

Given this setting, we will now make some simplifying assumptions, namely, (1) the order in which the explanatory messages are presented does not matter (2) we have access to a set of observers with similar models and they share this model with the target user (3) the robot is viewing the task at the same level or at a more detailed level of granularity than the user and (4) the user and robot have some shared vocabulary in regards to the task. While assumption (1) is easily met since we are mostly dealing with model information and (4) is a prerequisite for most explanatory approaches, in section 7.5 we will discuss how we can possibly relax requirements (2) and (3).

Now our goal is to learn a predictive model that is able to predict whether a given user would find a given $\langle s, a, s \rangle$ tuple explicable and how the user's perception changes with the given explanatory messages.

For example, at the beginning of an episode the user may be presented with the following explanatory messages
$\hat{\Psi} = \{m_1 =$ "Robot slips with probability 0.25 at grey cells"$\}$,
which corresponds to the fact that $P(s_i|a, s_i) = 0.25$, for all states s_i where the feature grey cell is true and for all actions a. Now the user will be presented with a sequence of transitions, say $\langle (1,2), \text{right}, (2,2) \rangle$ and asked whether the transition was explicable or not. Then the tuple $\langle \langle (1,2), \text{right}, (2,2) \rangle, \{m_1\}, l_1 \rangle$, where l_1 is the label assigned by the user to the transition, becomes input to our learning method.

The exact function that we would want to learn would be

$$\mathcal{L}(\langle s, a, s' \rangle, \{m_1, ..., m_k\}) = \begin{cases} 1 & \text{if } \langle s, a, s' \rangle \sim \\ & \pi^*_{\mathcal{E}_{\langle \mathcal{M}_h^R, \mathcal{M}^R \rangle}(\theta(\{m_1,...,m_k\})}(s) \\ 0 & \text{otherwise} \end{cases}$$

Note that this is a modified version of the sequential model we introduced by Zhang *et al.* (2017) for identifying whether a given plan is explicable or not. Though our methods vary in some significant aspects, namely, (1) we allow for the possibility that the explicability of the actions/traces could be affected by explanations provided by the system; (2) we no longer use labels of high level tasks as a proxy for the explicability of the trace. Instead, we just use a simple binary label on whether the transition is explicable or not; (3) we no longer consider sequence models but rather a much simpler labeling model that maps a single transition to the explicability label.

We argue that in cases where the human is markovian on the same set of features as the agent, this rather simpler model suffices.

It is also important that our learning approach is more tractable than the ones studied by Zhang *et al.* (2017), since in their case to build a balanced dataset (of explicable and inexplicable plans), they would need to uniformly sample through the entire plan space (an extremely hard endeavour with no obvious known approaches), while we stick to traces generated from the optimal policy and only need to randomly generate possible sets of explanatory messages, which is clearly a smaller set.

Once we have learned an approximation of the above labeling function $\hat{\mathcal{L}}$, the problem of explanation generation for a trace $\tau = \langle s_0, a_0, s_1, ..., s_n, a_n, s_{n+1}\rangle$ becomes that of finding the subset of Ψ that balances the cost of communication with the reduction in the inexplicability of the given trace, i.e

$$\arg\min_{\hat{\Psi}}(\mathcal{C}^{\mathcal{M}}(\hat{\Psi}) + \alpha * \Sigma_{i=0}^{n}(1 - \hat{\mathcal{L}}(\langle s_i, a_i, s_{i+1}\rangle, \hat{\Psi})))$$

Where $\hat{\Psi}$ is a subset of Ψ and α is some scaling factor that balances the cost of explanation with the number of inexplicable transitions for a given trace.

7.4 Evaluation

The success of the approach described above would be directly dependent on whether we can learn high accuracy labeling models. Once we have access to such a model, we could be quite confident in our ability to generate useful explanation (provided the user's model is the same as the one the labeler was trained on) and identifying the best explanation becomes a matter of just searching for the required subset of messages that minimizes the objective defined in section 7.3. So to evaluate the method our focus was on identifying if it was possible to learn high accuracy models. We validated our approach on both simulations and on data collected from users.

7.4.1 Evaluation on Simulated Data

For simulations, we used a slightly modified versions of the Taxi domain (Dietterich, 1998) (of size 6*6), the Four rooms domain (Sutton *et al.*, 1999) (of size 9*9) and the warehouse scenario (of size 9*9) described before (implemented using the SimpleRL framework (Abel, 2019)). For each domain, we start with an MDP instance (henceforth referred to as the robot model) and then create a space of possible user models by identifying a set of possible values for each MDP parameter. For example, in the taxi domain the parameters include position of the passengers, their destination, the step cost, discounting etc., for the Four rooms this included the goal locations, locations with negative rewards, discounting, step cost, slip probability, etc., and finally for the warehouse, the position of the box, the position of station #1, the step cost, slipping probabilities and the discounting factors were selected as potential parameters that can be updated. In this setting, we assume that there exists a single explanatory message for each possible parameter.

For each individual test, we select a random subset of three parameters and then randomly choose a value for each of these. We then treat this new MDP model as a stand-in for the user model and use it to label traces generated from the original MDP. The traces were generated by choosing a random initial state and then following the optimal policy of the robot until either the terminal state is reached or the trace length reaches a predefined limit. For each trace, a random subset of the explanations was selected and presented to the human. This means updating the MDP parameters to their corresponding values in the robot model only for the parameters specified by the current subset of explanation. Each individual transition was then labeled using this updated MDP. A transition was labeled as inexplicable if the action is not the

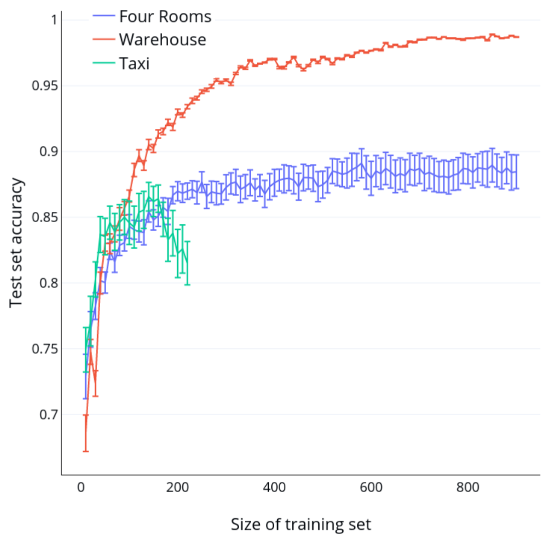


Figure 7.2: The Test Accuracy for Increasing Sizes of Training Set.

optimal one in the human model (i.e. Q value is lower) or the next state had a probability of occurring of $\delta = 0$.

We then used this set of labeled transitions to create a training set and test set for a decision tree learner. The input features to the decision tree consist of current state features, (just x and y for Four rooms and the position of the the taxi and passengers for the Taxi domain and for Warehouse it included the position of the robot and the fact whether the agent picked up the box or visited station #1), the index of the action and features capturing the current subset of explanations being considered. In each Warehouse and Four rooms test instance, we collected 900 unique data points as training set and 100 data points as the test set. Due to the complexity of the taxi domain, we generated less data points (since for each different explanation subset we need to solve a new planning problem) and used close to 220 unique points as training data and on average 28 data points as the test set.

We then tested on 20 such instances for each domain. Figure 7.2 plots the average test accuracy for models trained with training sets of varying sizes. As evident from the graph, a simple decision tree seems to be able to easily model the effect of explanations on labeling for these simulated scenarios. We chose a simple learning model to establish the viability of this method, but one could easily see that the use of more sophisticated learning methods and/or more informed features should lead to better results.

7.4.2 User Studies

Next, we wanted to establish if we can still learn such simple models when the labels are collected from naive users. Our goal here is not to consider scenarios with possible differences in the user's knowledge, but rather cases where, even in the presence of a set of users with similar backgrounds, their responses to explanations would be too varied to learn useful models. To test this, we used the Warehouse domain as a test bed and collected feedback on how users would view the explicability of traces generated from this domain when presented with explanatory messages detailed in Figure 7.1.

For the study, we recruited 45 master turkers from the Amazon Mechanical Turk. Each participant was provided with the URL to a website (`https://goo.gl/Hun3ce`) where they could view and label various robot behaviors. We considered a setting where the robot had a full battery, but was picking up a fragile box and thus still needs to visit the station #1. The robot could slip on some cells marked in dark grey with probability 0.25 (slipping here meant the robot picture is tilted to give an impression that it slipped on the cell and didn't prevent the robot from moving to the next cell). To make sure that all the users had similar mental models at the start, they were provided with the following facts, (a) that robot couldn't pass through racks, (b) whenever the robot runs low on battery it needs to get to Station 1, (c) whenever the robot has a green battery sign next to the robot, that means their battery is full and (d) the robot needs to take the shortest route to the goal. Also, they were presented with an example trace in this instructions section and were made to take a small pre-test that allowed them to revise the above facts in various scenarios. After the pre-test, they were shown eight traces from the robot policy sampled according to their probabilities. After the first trace, the user was given an explanation message before each trace, where the message was taken from the seven possible messages (the order of the messages was always randomized).

From the data collected from 45 turkers, we removed data from seven users, based on the fact they didn't find any of the transition in the first trace (i.e the case where no explanation was provided) inexplicable. We imagine this number would go down when we move to expert users or users who are invested in the success of the robot. The data generated for the remaining 38 users were then used to train a decision tree. Since the placement of other objects in the environment were fixed, we were able to use rather simple features for the model like the current position of the robot (x and y), previous position (again x and y), the action, whether they have slipped and finally the explanations given. We found the model to have an average 10-fold cross validation score of 0.935. For a randomly generated train and test split (where the test split was 10% and contained around 7% inexpicable labels) the precision score was 0.9637 and the recall score was 0.9568.

Furthermore, we could see that the model was able to correctly predict the usefulness of intuitive minimal explanations for the given scenario. For example, it predicted that while the robots decision to visit station #1 would be considered inexeplicable by the user in the absence of any explanation, the user would mark it as explicable when they are explained about the box being fragile and that fragile boxes need to be inspected at station #1. In fact the model predicted that only the message that "fragile boxes need to be inspected at station #1" is enough to convince the user about the need for that action (i.e the user could deduce that the box must have been fragile). This shows that such learned models may help us generate cheaper explanations (the above set of explanations is smaller than the corresponding minimal complete behavior explanation for the domain), by taking into account the users ability to correctly predict missing information in simple cases. Another point of interest was that the model predicted all slipping events as explainable even in the absence of any explanations. The cases where the user saw a slip before being told about the possibility of slipping was rare (since there are two explanatory messages related to slipping and the probability of slipping was 0.25). Furthermore when we went over the data, we found that in most such cases, the users did mark it as explainable. This may be because the effect of slipping may not have been that detrimental to the overall plan (it doesn't take you off the current path). It would be interesting to see if this result would be the same in cases where slipping was a more likely event and if it had a more apparent effect on the robot's plan.

7.5 Related Work

To the best of our knowledge, this work represents the first attempt at learning proxies for user mental models that allows an agent to predict the potential impact of providing explanations as model reconciliation to observers. With that said, there have been works that have looked at the problem of generating explanations in the presence of model uncertainty for human models. In Chapter 6, we looked at cases where the agent has access to a set of potential human models. One drawback of considering a set of possible models is either they would need to have explicit sensing to identify the user model (which could mean asking a large number of questions to the user) or providing a large amount of information to cover the space of all possible models. In our work, the problem of identifying the specifics of the user model is resolved through an offline training process.

Another method quite related to the discussion covered in this chapter is Reddy *et al.* (2018), wherein the authors tried to identify cases where they can learn a potential model for the human's expectation of the task transition dynamics when they do not align with the real world dynamics. Unlike their work, we do not assume that the user can provide traces for the given task, rather they may be able to provide some high-level feedback on the action (i.e. they may not be able to do or even know the right action but may be able to point out actions or transitions that surprise them). Moreover, their work requires that the user and the robot must have the same reward function, which is again an assumption we do not make. Even if we had followed their technique to learn a potential approximation of the human's transition model for the task, there is no guarantee that the learned representation would be one that makes sense to the human.

This chapter proposes a possible way in which model reconciliation explanation could be applied to cases where the user model is unknown. The method described here is a rather simple and general method to identify information that could potentially affect the user's mental model and produce effects that align with the agent's requirements. There is no requirement here that the messages have to align with actual facts about the world. This again points to the rather troubling similarities between the mechanisms needed to generate useful explanations and lies (Chakraborti and Kambhampati, 2019b).

Two important assumptions we made throughout the work is that the user only considers the current state (as defined by the robot) to make their decisions and we have access to a model that was learned from interactions to previous users who had similar knowledge level to the current user. Relaxing the first assumption would require us to go beyond learning models that map each transitions to labels. Instead we have to consider sequential labeling models (for example models based on LSTM or CRF) of the type considered by Zhang *et al.* (2017) to capture the human's expectations. For example, we considered a simple extension of the warehouse domain where the human believes the robot should visit two locations (i.e the human state contains variables that record whether the user has visited the locations). Even though here the user is considering a more detailed model, we were able to learn labeling models of 80% accuracy by using simple CRFs. As for the second, instead of assuming that all users are of the same type, a more reasonable assumption may be that the users could be clustered into N groups and we could learn a different labeling model for each user type. Now we still have a challenge of identifying the user type of a new user and one way to overcome this would be by adopting a decision-theoretic approach to this problem and modeling it as a POMDP (where user labels become observations and previously learned user models the observation models).

The work discussed in this chapter only covers explanations that allow the user and the system to reconcile any model difference. This only covers a part of the entire explanatory dialogue. Even if there is no difference in models, the user may still have questions about parts of the policy or may raise alternative policies they think should be followed. This may arise from a difference in inferential abilities and may require providing information that is already part of their deductive closure eg: help them understand the long term consequences of taking some actions. Once you have access to a set of such messages one could use a method similar to the one described in the chapter to find the set of helpful ones. Unlike the model reconciliation setting where the messages stand for information about the model, it is not quite clear how one could automatically generate such messages.

7.6 Concluding Remarks

This chapter proposes a possible way in which model reconciliation explanation could be applied to cases where the user model is unknown. The method described here is a rather simple and general method to identify information that could potentially affect the user's mental model and produce effects that align with the agent's requirements. There is no requirement here that the messages have to align with actual facts about the world. This again points to the rather troubling similarities between the mechanisms needed to generate useful explanations and lies (Chakraborti and Kambhampati, 2019a).

Two important assumptions we made throughout the work is that the user only considers the current state (as defined by the robot) to make their decisions and we have access to a model that was learned from interactions to previous users who had similar knowledge level to the current user. Relaxing the first assumption would require us to go beyond learning models that map each transitions to labels. Instead we have to consider sequential labeling models (for example models based on LSTM or CRF) of the type considered by Zhang *et al.* (2017) to capture the human's expectations. For example, we considered a simple extension of the warehouse domain where the human believes the robot should visit two locations (i.e the human state contains variables that record whether the user has visited the locations). Even though here the user is considering a more detailed model, we were able to learn labeling models of 80% accuracy by using simple CRFs. As for the second, instead of assuming that all users are of the same type, a more reasonable assumption may be that the users could be clustered into N groups and we could learn a different labeling model for each user type. Now we still have a challenge of identifying the user type of a new user and one way to overcome this would be by adopting a decision-theoretic approach to this problem and modeling it as a POMDP. In fact this method was investigated by Soni *et al.* (2021).

The work discussed in this chapter only covers explanations that allow the user and the system to reconcile any model difference. This only covers a part of the entire explanatory dialogue. Even if there is no difference in models, the user may still have questions about parts of the policy or may raise alternative policies they think should be followed. This may arise from a difference in inferential abilities and may require providing information that is already part of their deductive closure eg: help them understand the long term consequences of taking some actions. Once you have access to a set of such messages one could use a method similar to the one described in the chapter to find the set of helpful ones. Unlike the model reconciliation setting where the messages stand for information about the model, it is not quite clear how one could automatically generate such messages.

Part II

ADDRESSING INFERENTIAL ASYMMETRY

Chapter 8

PART II OVERVIEW

This part of the thesis will focus on techniques I have helped develop that have tried to address the second dimension of human-aware explanation generation, namely *Asymmetry in Inferential Capabilities*. The majority of Part II will focus on a series of techniques for explicit contrastive explanations that will primarily leverage state abstractions to simplify the model information provided to the user. However, in the closing chapter of Part II, we will see how these individual works could potentially be recontextualized and understood as an instance of the larger framework of model-simplification.

8.1 Structure for Part II and Technical Contributions

Part II will be divided into four chapters

1. Chapter 9 In this chapter we will introduce the basic explanation generation framework, which we will refer to as Hierarchical Expertise-Level Modeling or HELM. The rest of the chapters will cover the progressive generalization of this basic framework. In this foundational chapter we will look at the simplest formulation of the framework designed for deterministic planning problems. This version will take as input a set of fully specified but invalid plans as the foils expected by the user and try to identify the most abstract version of the robot model, where the invalidity of the foils can be established. In addition, to looking at the basic version, we will also discuss a variant of HELM that support explanation of suboptimality, one that supports cases where the human could

have incorrect beliefs about the robot model and also perform analysis on how the system designer could enforce their preference on types of abstract models to be used for explanation. We will also establish the computational complexity of generating the basic explanation (which is identified to be NP-Complete), and also run user studies that evaluate the effectiveness of state-abstraction in explanations.

2. Chapter 10: In this chapter, we present the first generalization of HELM, one where we relax the assumption that foils have to be fully specified plans. Instead we look at a framework that can support partial foils, namely, cases where the user only provides a partial specification of plans they were expecting. This brings up an entirely new problem, namely that of explaining the unsolvability of planning problems. As we will see, explaining the infeasibility of partial foils, corresponds to establishing the unsolvability of a constrained planning problem. Thus the primary focus of this chapter will be to extend HELM to support explaining the unsolvability of planning problems. In addition to the state abstraction, this version will also support another model simplification, namely identifying an unachievable subgoal for the problem. This will help further simplify the model, by decomposing the problem horizon the user would have to reason about. We will again present results from user studies that verify the effectiveness of this version of the framework.

3. Chapter 11 This chapter will see us further generalizing the framework, by allowing it to support FOND problems. As part of the generalization, we will map over the tools we had developed in Chapter 10 to support partial foils in the context of deterministic domains to domains where actions could potentially have non-deterministic effects. We will then use this updated framework to

develop a tool for debugging non-deterministic domains that are designed to model conversational agents. We will see how this tool allows domain authors to identify potential bugs in the domain description they may have designed.

4. Chapter 12: In this chapter, we will present a generalized framework for creating explanations that address human queries that arise from asymmetry in inferential capabilities. This framework will be built around the concept of an explanatory property, whose establishment in the human mental model can help resolve user queries and model-simplification transformations that will generate simplified versions of the robot model where it is easier to establish a specific explanatory property. We will ground this framework within the context of stochastic planning problems and will look at various model-simplification transformations, including versions of state abstraction and problem decomposition. This chapter will also look further into the idea of using an explanatory witnesses as another way of alleviating the inferential burden.

8.2 Important Takeaways

One of the main takeaways from this part of the thesis is the effectiveness of the three strategies we had outlined in Chapter 1, namely, allowing for explicit user queries, model-simplification and the use of explanatory witnesses. We have also seen that these aren't mutually independent techniques and if we use the right combination, the effectiveness of the sum may be greater than the parts. In terms of the next step the obvious one would be to build on the framework laid out in Chapter 12. The specific transformations and explanatory witnesses presented in that chapter only represents a small portion of the possibilities. Additionally, much more work needs to be done in better understanding the human computation model and how different transformations may influence the final inferential burden placed on the human. It is also worth mentioning that the versions of ideas presented in this section have also been used outside explaining plans and sequential decisions. For example, local approximation of a model transformation we look at in Chapter 12 was first introduced for classification problems (Ribeiro *et al.*, 2016). Additionally, a lot of work in post-hoc explanations for XAI try to focus on creating simpler and easy to under proxy models (Lakkaraju *et al.*, 2020), which can be seen as model-simplification. However, we should note that in addition to model-simplification these methods also generally use a different user-understandable feature set to specify these proxy models. We will be covering such model translations in the next part of the thesis. Finally, generating counterfactual cases, a type of explanatory witness we noted in Chapter 12, is also popular in explainable machine learning.

Chapter 9

HIERARCHICAL EXPERTISE LEVEL MODELING

In part-I of the thesis, we looked at explanatory methods that are designed to address human confusion (i.e., the mismatch in the human's expectations and robot's decisions) that arise from the difference in knowledge between the human and the robot. As we discussed in Chapter 1, this could be one of many reasons why humans may be confused. In this chapter, we will start by introducing a framework that is designed to address the second dimension.

In this chapter, we propose a new approach to this problem where the agent explains its ongoing or planned behavior in a way that is both tailored to the user's background and is designed to reduce cognitive burden on the user's end. This is done by modeling a user's expertise, or the level of detail at which a user understands the task using abstracted models. We can estimate this level based on questions that the user asks and provide explanations that are close to this estimated level of expertise.

We consider explanations in the framework of counterfactual reasoning, where a user who is confused by the agent's activity (or proposed activity) presents alternative behavior that they would have expected the agent to execute. This aligns with the widely held belief that humans expect explanations to be *contrastive* (Miller, 2017a). In keeping with the terminology used in social sciences literature, we will denote the set of alternative behaviors as *foils* to the proposed robot behavior.

Specifically, we present the **Hierarchical Expertise-Level Modeling** or the **HELM** approach for facilitating such context and user-specific explanations. We assume that the user's understanding of the task is an abstraction of the model used by the robot; which captures both the limited information and computational capabilities of the user. HELM generates appropriate explanations by searching through a *model lattice* of possible abstractions of the agent's model. The model lattice provides a concise way for the system designer to encode their prior knowledge about potential users. Each model within this lattice represents a different level of understanding of the task, with the highest fidelity representation (corresponding to the most detailed understanding of the domain used by the robot) forming the base of the lattice and the model representing the most naive understanding of the task (for example one held by a lay person) forming the highest nodes. Since the user's level of expertise is unknown to the agent, it has to estimate the human model before searching for an explanation.

We focus on contrastive explanations, where an explanation that is an answer to a question of the form "Why P and not Q?", in our case, P and Q are stand-ins for the current robot plan and the foil respectively.

Specifically, our explanations will consist of model information that may be absent in the user's abstract model and possible proofs for foil failure. Thus, in addition to helping convince the user of the incorrectness of the foils in question, the explanations should also shift the user's model to a more accurate model in the lattice. This approach could be understood as a variant of the model refinement methods discussed in the counter-example guided model checking (CEGAR) literature (Clarke *et al.*, 2000). Our methods extend these principles to settings with uncertainty regarding the current level of abstraction of the model (a non-issue in the model-checking settings where CEGAR methods are typically used).

In addition to introducing the basic framework, some of the other contributions of this chapter will include;

- We consider the use of non-standard lattices as a way to allow designers to incorporate more information about the user's model into the explanation generation process and discuss potential computational tradeoffs introduced by the use of such lattice types over the ones considered in the rest of the chapter.

- We investigate the use of such methods for domains that contain state dependent costs (hence affected by the abstraction) and discuss the potential explanatory dialogue that could occur in such settings.
- We also show how our method could be used in cases where the user model may not just be abstract but the user may also hold erroneous beliefs about the task.
- We perform a user study to verify the utility of abstraction in generating explanations that are easier for users to work with.

The rest of this chapter is structured as follows. Section 9.1 a brief overview of the background and in Section 9.2 we present our formal framework. Section 9.3 covers different approaches for generating explanations and sections 9.3.1, 9.3.2 and 9.3.3 extend these methods to more general settings. Section 9.4 presents evaluations of the method. In section 9.5 we will discuss some related work and Section 9.6 will delve into the relationship between the framework introduced in this chapter and the ones discussed in Part I of this thesis.

9.1 Background

In this Chapter, we focus on abstractions that form models by projecting out state fluents. While the presentation in the following sections is equally valid for both predicate and propositional abstractions, we will focus on propositional abstractions to keep our formulation clear and concise and later discuss potential changes required to meet the requirements of predicate abstractions. We will look at deterministic

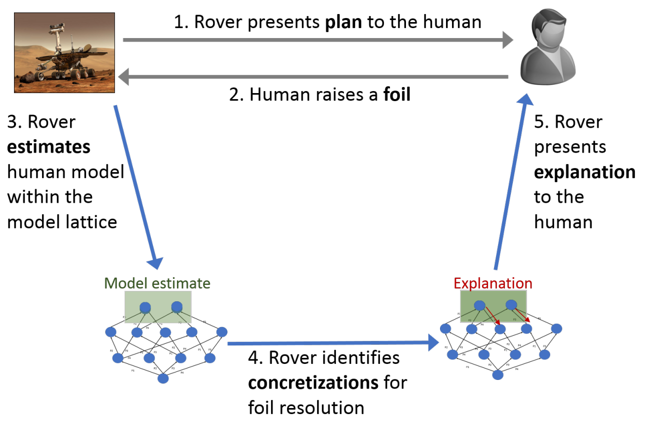

1. Rover presents **plan** to the human
2. Human raises a **foil**
3. Rover **estimates** human model within the model lattice
Model estimate
4. Rover identifies **concretizations** for foil resolution
Explanation
5. Rover presents **explanation** to the human

Figure 9.1: An Illustration of the Hierarchical Explanation Process. The Human Observer Who Views the Task at a Higher Level of Abstraction Expects the Rover to Execute a Different Plan from the One Chosen by the Rover. The Rover Presents the Human with an Explanation It Believes Will Help Resolve the Foils in the Human's Updated Model.

planning models of the type discussed in Chapter 2 (Section 2.1). To simplify the discussions, we will follow a slightly different notational scheme from what was used in Chapter 2. We will start by focusing on models with unit action costs and the model will be given by the tuple $\mathcal{M} = \langle F, S, A, I, G \rangle$, where S is the state space defined using the fluent set F.

Similar to previous part, each action $a \in A$ is associated with a set of positive preconditions $pre_+(a)$ (specified as a conjunction of propositions) and negative preconditions $pre_-(a)$ that need to hold for the effects (e_a) of that action to be applied to a particular state. Each effect set e_a can be further separated into a set of add effects $add(a)$ and a set of delete effects $del(a)$. We will eschew from using a separate transition function and denote the result of executing an action a on a state s by using the notation $a(s)$, which is defined as $a(s) = (s \cup add(a)) \setminus del(a)$, if $pre_-(a) \subseteq s \wedge pre_-(a) \cap s = \emptyset$. A plan π is again give as a sequence of actions $(\langle a_1, .., a_n \rangle$, n being the size of the plan), and a plan is said to solve $\mathcal{M}$ (i.e, $\pi(I) \models_{\mathcal{M}} G$) if $\pi(I) \supseteq G$. Note that unlike the previous part, where the differences in action definitions where reflected in the transition function, we will use the parameterized $\models_{.}$ to capture these differences.

Automated planning has a long tradition of employing abstraction both for plan generation (cf. (Sacerdoti, 1974)) and for generating heuristics (cf. (Seipp and Helmert, 2018; Keyder *et al.*, 2012)) and a number of different abstraction schemes have been proposed in these works. In fact, state abstractions as presented in this work have been widely used in pattern databases and are referred to as projections in that literature (cf. (Culberson and Schaeffer, 1998; Edelkamp, 2000)). Following discussion presented by Seipp and Helmert (2018) and Bäckström and Jonsson (2013), we will also use the concept of a transition system induced by the planning model to define state abstractions. Intuitively, a transition system constitutes a graph where the nodes represent possible states, and the edges capture the transitions between the states that are valid in the corresponding planning model.

Definition 14. A propositional abstraction function f_Λ for a set of propositions Λ and state space S, defines a surjective mapping of the form $f_\Lambda : S \rightarrow X$, where X is a projection of S, such that for every state $s \in S$, there exists a state $f_\Lambda(s) \in X$ where $f_\Lambda(s) = s \setminus \Lambda$.

Definition 15. For a planning model $\mathcal{M} = \langle F, S, A, I, G \rangle$ with a corresponding transition system $\mathcal{T}$, a model $\mathcal{M}' = \langle F', S', A', I', G' \rangle$ with a transition system $\mathcal{T}'$ is considered an **abstraction of $\mathcal{M}$** for a set of propositions Λ, if for every transition $s_1 \xrightarrow{a} s_2$ in $\mathcal{T}$ corresponding to an action a, there exists an equivalent transition $f_\Lambda(s_1) \xrightarrow{a'} f_\Lambda(s_2)$ in $\mathcal{T}'$, where a' is part of the new action set A'.

We will slightly abuse notation and extend the abstraction functions to models and actions, i.e in the above case, we will have $\mathcal{M}' \in f_\Lambda(\mathcal{M})$ (where $f_\Lambda(\mathcal{M})$ is the set of all models that satisfy the above definition for the set of fluents Λ) and similarly we will have $a' \in f_\Lambda(a)$. As per Definition 15, the abstract model is *complete* in the sense that all plans that were valid in the original model will have an equivalent plan in this new model. We will use the operator $\sqsubset$ to capture the fact that the model $\mathcal{M}'$ is an abstraction of $\mathcal{M}$, i.e if $\mathcal{M} \sqsubset \mathcal{M}'$ then there exist a set of propositions Λ such that $\mathcal{M}' \in f_\Lambda(\mathcal{M})$.

9.1.1 Designing Complete Abstractions

While there exists a number of works that have looked at the problem of designing abstractions (cf. (Srivastava *et al.*, 2016; Sacerdoti, 1974; Bäckström and Jonsson, 2013)), unfortunately many of these works have considered directly updating transition system or using specialized or more expressive problem formulation to capture abstract models. Thankfully, the fact that we are interested in complete abstractions (as opposed to sound abstractions) means we can employ simpler model transformation schemes to generate abstract models. In particular, we will consider transformations that simply drops the set of literals to be abstracted from all the action definitions, i.e,

Theorem 4. *For a given model $\mathcal{M} = \langle F, S, A, I, G \rangle$ and a set of propositions Λ, a model $\mathcal{M}' = \langle F', S', A', I', G' \rangle$ is a complete abstraction under safe execution semantics for Λ, if $F' = F - \Lambda$, $S' = [S]_{f_\Lambda}$, $I' = f_\Lambda(I)$, $G' = G \setminus \Lambda$ and for every $a \in A$ (where $a = \langle pre_+(a), pre_-(a), eff_a^+, eff_a^- \rangle$) there exists $a' \in A'$, such that $a' = \langle pre_+(a) \setminus \Lambda, pre_-(a) \setminus \Lambda, eff_a^+ \setminus \Lambda, eff_a^- \setminus \Lambda \rangle$.*

Proof Sketch. To see why the new model would be an complete abstraction, consider a transition $\langle s, a, s' \rangle$ induced by $\mathcal{M}$. Now as per the definitions of safe transition systems, we know that $s \subseteq pre_+(a)$ and $s \cap pre_-(a) = \emptyset$ and $s' = s \setminus \text{eff}_a^- \cup \text{eff}_a^+$. Its easy to see that given this setting, $(s \backslash \Lambda) \subseteq (pre_+(a) \backslash \Lambda)$ and $(s \backslash \Lambda) \cap (pre_-(a) \backslash \Lambda) = \emptyset$, which means there must be an action $a' \in A'$ that is executable in $f_\Lambda(s)$. Similarly we can show the result of executing a' must be $f_\Lambda(s)$, this shows that $\mathcal{M}'$ is a complete abstraction of $\mathcal{M}$ as every transition induced by it is present in the transition system induced by $\mathcal{M}'$. $\square$

An important point to note here is that this transformation scheme generates a unique abstract model for each model and proposition set, and we will denote this unique model as $f_\Lambda(\mathcal{M})$. For the rest of the chapter, we will mainly focus on this method to induce the abstractions, but general framework of explanation generation discussed in this chapter can be adapted to other methods of generating abstract models. In cases, where we prove specific results or present optimization that rely on this abstraction procedure we will denote the abstraction function by f_Λ^{safe} to differentiate it from other methods. With the definition of abstraction and related notations in place, we will look at our explanatory setting and a way to capture the space of possible user models that would allow for efficient estimation of unknown user model given user queries. While the above operation is defined for propositional fluents, we can perform similar operations on the lifted domain, where projecting out a predicate would correspond to projecting out a set of propositional fluents from the grounded domain.

9.2 Hierarchical Expertise-level Modeling

As mentioned earlier, we are investigating explanatory settings where the user's understanding of the task can be represented as an abstraction of the robot's model. While the exact level of abstraction may be unknown, given a set of candidate state fluents that may be missing from the human model, we can capture the potential models and their relationship through a **model lattice**

Definition 16. For a model $\mathcal{M}^{\#}$, the *model lattice* $\boldsymbol{\mathcal{L}}$ is a tuple of the form $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, where $\mathbb{M}$ is the set of lattice nodes such that $\mathcal{M}^{\#} \in \mathbb{M}$ and $\forall\ \mathcal{M}' \in \mathbb{M}, \mathcal{M}^{\#} \sqsubseteq \mathcal{M}'$, $\mathbb{E}$ is the lattice edges, $\mathbb{P}$ is the superset of propositions considered for abstraction within this lattice and ℓ is a function mapping edges to labels. Additionally, for each edge $e_i = (\mathcal{M}_i, \mathcal{M}_j)$ there exists a proposition $p \in \mathbb{P}$ such that $f_p(\mathcal{M}_i) = \mathcal{M}_j$ and $\ell(\mathcal{M}_i, \mathcal{M}_j) = p$.

Thus each edge in this lattice corresponds to an abstraction formed by projecting out a single proposition (represented by the label of the edge). We can also define a concretization function γ_p that retrieves the model that was used to generate the given abstract model by projecting out the proposition p, i.e, $\gamma_p(\mathcal{M}) = \mathcal{M}'$ if $(\mathcal{M}', \mathcal{M}) \in \mathbb{E}$ and $\ell(\mathcal{M}', \mathcal{M}) = p$ else $\gamma_p(\mathcal{M}) = \mathcal{M}$.

For a given lattice, if each node in $\mathbb{M}$ has an incoming edge for every proposition missing from its corresponding model then we will refer to such lattices as being *Proposition Conserving* lattices.

Definition 17. *A lattice $\mathcal{L}$ is proposition conserving-iff for any model $\mathcal{M} \in \mathbb{M}$ ($\mathcal{M} = \langle P_{\mathcal{M}}, S_{\mathcal{M}}, A_{\mathcal{M}}, I_{\mathcal{M}}, G_{\mathcal{M}} \rangle$) and $\forall p \in \mathbb{P}$, if p is not in $P_{\mathcal{M}}$ then there exists a model $\mathcal{M}' \in \mathbb{M}$, such that $(\mathcal{M}', \mathcal{M}) \in \mathbb{E}$ and $\ell(\mathcal{M}', \mathcal{M}) = p$).*

Notice that enforcing conservation of propositions doesn't require any further assumptions about the human model and can be easily ensured while generating the lattice. Additionally, we will call a proposition conserving lattice that contains an abstract node corresponding to each possible subset of $\mathbb{P}$ as the *Complete Abstraction Lattice* for $\mathcal{M}$ given $\mathbb{P}$. The earlier parts of this chapter will assume a proposition conserving lattices as they will allow us to simplify discussions and provide efficient solutions. In later sections, we will relax these assumptions and will look at potential tradeoffs for using non-proposition conserving lattices.

We also assume that all abstraction functions used in generating the models in the lattice are commutative and idempotent, i.e., $f_{p_2}(f_{p_1}(\mathcal{M})) = f_{p_1}(f_{p_2}(\mathcal{M}))$ and $f_{p_1}(f_{p_1}(\mathcal{M})) = f_{p_1}(\mathcal{M})$. In the wider literature, a lattice is generally defined to have a unique maximal element and a unique minimal element. While the abstraction lattices we consider in this work will have a unique minimal element (i.e the most concrete nodes), we do not assume that the lattices have a single maximal node (Figure 9.2 presents an example lattice that does not have a unique maximal node), in that sense the abstraction lattice may be better understood as meet-semilattices, but we will use the term model lattice or abstraction lattice for convenience.

As mentioned earlier, we consider an explanation generation setting where the human observer (H) uses a task model (this model will be denoted as $\mathcal{M}_h^R = \langle F_H, S_H, A_H, I_H, G_H \rangle$), that is a more abstract version of the robot's model ($\mathcal{M}_R = \langle F_R, S_R, A_R, I_R, G_R \rangle$). While the robot may not know $\mathcal{M}_h^R$, it knows that $\mathcal{M}_h^R$ is a member of the set $\mathbb{M}$ for the lattice $\mathcal{L}$. The human comes up with a **foil set** $\Pi_{\mathcal{F}} = \{\pi_1, \pi_2, ..., \pi_m\}$ that the robot needs to refute by providing an explanation regarding the task. The explanation should contain information about specific domain properties (i.e., state fluents) that are missing from the human's model, how these properties affect different actions (For example, which actions use these propositions as preconditions and which ones generate/delete them) and how the inclusion of these fluents result in the invalidity of the given foils. To illustrate the utility of such explanations consider an example involving a simplified version of the rover domain mentioned earlier.

Example 1. *Let us suppose that the rover uses a modified version of the IPC rover domain (International Planning Competition, 2011) that also takes into account the battery level of the rover. Each rover operation has a different energy requirement,* *and the battery level needs to be above a predefined threshold for it to execute them, e.g., it can perform rock sampling only if the battery level is above 75%. Furthermore, the rover needs to visit the base station (i.e., the lander) and perform a reset action to recharge its batteries.*

The rover knows that the human observer is at most ignorant of its energy requirements, ability to use solar cells and/or storage capabilities. So the model lattice $\mathcal{L}$ *needs to consider abstractions corresponding to the following propositions*
$\mathbb{P}$=$\{$battery_level_above_25_perc, battery_level_above_50_perc, battery_level_above_75_perc, full_store1, solar_panels_activated$\}$.

Figure 9.2 shows the lattice that the robot would use in this setting. Here we will create each abstract model by following the process discussed in section 9.1.1. For example, consider the action sample_rock_store0_w1*, it has the following definition*

$$\langle\{\textsf{battery\_level\_above\_75\_perc, at\_w1, empty\_store1, has\_store\_store1}\}, \{\},$$
$$\{\textsf{full\_store1, has\_rock\_sample}\}, \{\textsf{empty\_store1, battery\_level\_above\_75\_perc}\}\rangle$$

Now in an abstract version of this model, if the propositions full_store1 *and* battery_level_above_75_perc *are dropped the definition becomes*

$$\langle\{\textsf{at\_w1, has\_store\_store1}\}, \{\},$$
$$\{\textsf{has\_rock\_sample}\}, \{\textsf{empty\_store1}\}\rangle$$

Here the robot presents the plan

$$\pi_R = \langle\ \textsf{navigate\_w0\_lander, reset\_at\_lander,}$$
$$\textsf{navigate\_lander\_w1, sample\_rock\_store0\_w1}\rangle$$

and a naive observer may respond by proposing the foil set with a single plan

$$F = \{\langle \textsf{\textit{navigate\_w0\_w1, sample\_rock\_store0\_w1}} \rangle\}$$

If the observer was an engineer, they might instead raise a foil that already takes into account the energy requirements

$$\Pi'_{\mathcal{F}} = \{\langle \textsf{\textit{navigate\_w0\_w1,}}$$
$$\textsf{\textit{receive\_energy\_from\_solar\_cells, sample\_rock\_store0\_w1}} \rangle\}$$

If the robot knew that the human was ignorant about all the battery level predicates and nothing else, the robot could help resolve the naive foil by informing them about the fact that action sample rock requires the battery to be above 75% (i.e describing the proposition battery_level_above_75_perc). In terms of the human model, this would involve setting the value of the proposition battery_level_above_75_perc false in the initial state, updating the precondition of sample_rock_store0_w1 to include the fact (among other actions) and adding it as an add effect to the action reset_at_lander. In this updated model the human foil can no longer achieve the goal. In the case, of expert foil, the robot would need to inform the user about the proposition solar_panels_activated and that the action receive_energy_from_solar_cells require the solar panels to be activated which is not true for the rover. Thus in each case explanations to be provided to user can be generated once we know the set of propositions whose concretization is required to refute the given foils (henceforth referred to as *explanatory fluent set*).

Definition 18. *Let $E = \{p_1, ..., p_n\}$ be a set of fluents, then E is said to be an* explanatory set *for the human model $\mathcal{M}_h^R$ and a foil set $\Pi_{\mathcal{F}}$ if*

$$\forall \pi \in \Pi_{\mathcal{F}}, \pi(I_{\gamma_E(\mathcal{M}_h^R)}) \not\models_{\gamma_E(\mathcal{M}_h^R)} G_{\gamma_E(\mathcal{M}_h^R)}$$

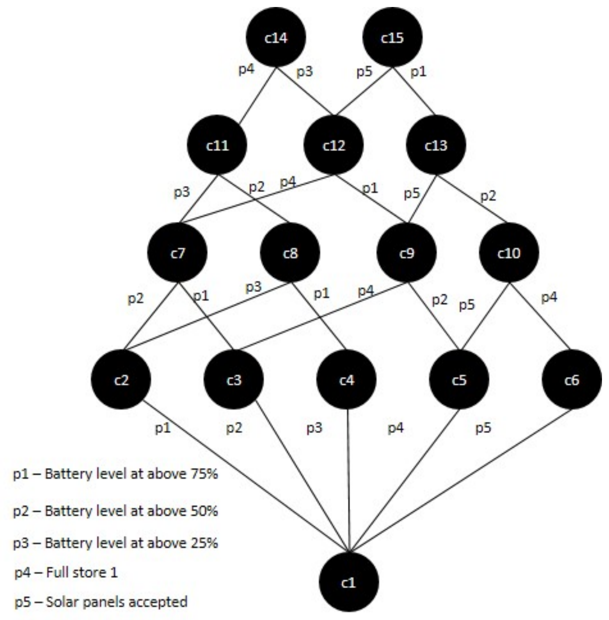

c14
c15
p4
p3
p5
p1
c11
c12
c13
p3
p2
p4
p1
p5
p2
c7
c8
c9
c10
p2
p1
p3
p1
p4
p2
p5
p4
c2
c3
c4
c5
c6
p1
p2
p3
p4
p5
p1 – Battery level at above 75%
p2 – Battery level at above 50%
p3 – Battery level at above 25%
p4 – Full store 1
p5 – Solar panels accepted
c1

Figure 9.2: A Possible Abstraction Lattice for the Rover Domain.

Where $\gamma_E(\mathcal{M}_h^R)$ is the model obtained by applying the concretizations corresponding to E on the model $\mathcal{M}_h^R$.

In the case of projection based abstractions of the form defined in Section 9.1.1, we can directly provide the model components covered by the explanatory fluent set as part of the final explanatory message provided to the user. For other abstraction techniques, we may need to employ more specialized methods to generate explanatory messages from the fluents. In Example 1 if we are to focus on the naive foil, the rover would have difficulty coming up with a single explanation as it does not know $\mathcal{M}_h^R$. However, it can restrict its attention to just the models that are consistent with the foils. In this scenario, it would correspond to $\{c2, c7, c8, c11, c12, c14, c15\}$.

Now we need to find a way of generating sets of explanatory fluents given this reduced set of models.

Proposition 13. *Let $\mathcal{M}_i$ be some model in $\mathcal{L}$ such that $\mathcal{M}_h^R \sqsubseteq \mathcal{M}_i$. If E is explanatory for $\mathcal{M}_i$ and some foil set $\Pi_{\mathcal{F}}$, then E must also explain $\Pi_{\mathcal{F}}$ for $\mathcal{M}_h^R$.*

This proposition directly follows from the fact that for a proposition conserving lattice $\gamma_E(\mathcal{M}_i)$ will be a logical weaker model than $\gamma_E(\mathcal{M}_h^R)$. Next, we will define the concept of a minimal abstracting set for a given lattice $\mathcal{L}$ and foils $\Pi_{\mathcal{F}}$

Definition 19. Given an the abstraction lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ and a foil set $\Pi_{\mathcal{F}}$, the *minimal abstracting set* $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ is the maximal elements of the subset of all the models that are consistent with $\Pi_{\mathcal{F}}$.
$\mathbb{M}_{min}^{\Pi_{\mathcal{F}}} = \{\mathcal{M}_i | \mathcal{M}_i$ is a maximal element of $\mathbb{M}_{sat}\}$ where $\mathbb{M}_{sat} = \{\mathcal{M} \mid \mathcal{M} \in \mathbb{M}, \forall \pi \in \Pi_{\mathcal{F}}(\pi(I_{\mathcal{M}}) \models_{\mathcal{M}} G_{\mathcal{M}})\}$

Proposition 14. *For a given model lattice $\mathcal{L}$, the minimal abstracting set $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ is a subset of the maximal elements of the entire abstraction lattice.*

The above property ensures that when searching for the minimal abstracting set, we do not need to test the entire set of nodes or even need to know the entire lattice. In Example 1, the minimal abstracting set for the naive foil will be $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}} = \{c14, c15\}$.

If we can find an explanation that is valid for all the models in $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ then by Proposition 13 it must work for $\mathbb{M}_H$ as well.

Proposition 15. *For a given model lattice $\mathcal{L}$ and a set of foils $\Pi_{\mathcal{F}}$ and the minimal abstraction set $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$, there exists an explanatory fluent set E such that $\forall\ \mathcal{M}' \in \mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ and $\forall \pi \in \Pi_{\mathcal{F}}$,*

$$\pi(I_{\gamma_E(\mathcal{M}')}) \not\models_{\gamma_E(\mathcal{M}')} G_{\gamma_E(\mathcal{M}')}$$

It is easy to see why this property holds, as any explanation that involves concretizing all possible propositions in $\mathbb{P}$ satisfies this property.

In most cases, we would prefer to compute the minimal or cheapest explanation to communicate. If all concretizations are equally expensive to communicate to the explainee, then this would correspond to finding the explanatory fluents set with the smallest size. For the naive foil in the rover example, even if the human is unaware of multiple task details, the robot can easily resolve the explainee's doubts by just explaining the concretizations related to the proposition battery_level_above_75_perc without getting into other details. Describing the details of remaining propositions is unnecessary and in the worst case might leave the human feeling overwhelmed and confused. In this case, the explanation would just include information regarding battery levels and how to identify when the battery level is or above 75% and model updates like

sample_rock-has-precondition-battery_level_above_75_perc

sample_soil-has-precondition-battery_level_above_75_perc

...

Before delving into the optimization version of the problem, let us look at the complexity of the corresponding decision problem

Theorem 5. *Given a the set of foils $\Pi_{\mathcal{F}}$ and the corresponding minimal abstraction set $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ for a model $\mathcal{M}$, the problem of identifying whether an explanatory fluent set of size k exists for the complete lattice (which is not given) defined over an abstraction function f is* ***NP-complete****, provided the abstract function generates planning models that belong to the class described in Section 9.1 in polynomial time.*

Proof (Sketch). The fact that we can test the validity of the given explanation in polynomial time (size of the explanation is guaranteed to be smaller than $|\mathbb{P}|$) shows that the problem is in **NP**. We can show **NP-completeness** by reducing the set covering problem (Bernhard and Vygen, 2008) to an instance of the explanation generation problem. Let's consider a set covering problem with U as the universe set and S as the set of sub-collections. Now let us create an explanation generation problem where the set of foils $\Pi_{\mathcal{F}}$ is equal to U and the propositions in the set $\mathbb{P}$ contain a proposition for each member of S. Additionally concretizing with respect to a proposition will resolve only the foils covered by its corresponding subset in S. For this setting, the $\mathbb{M}^{F}_{min}$ consists of a single node that contains none of the propositions (and hence all the foils hold) and the concrete model contains all of them. Now if we can come up with a set of explanatory fluents of size k in this setting, then this explanation corresponds to a set cover of size k. □

The above result considers a case where the lattice needs to be generated on the fly from the minimal abstraction set. Though there may be cases where the designer may be able to provide an explicit and smaller non-proposition conserving lattice upfront. As we will see in Section 9.3.1, such lattices can be used to capture the designer's knowledge about the end-users.

9.3 Generating Optimal Explanations

As mentioned earlier, we are interested in producing the minimal explanation. Additionally, in most domains, the cost of communicating the concretization details could vary among propositions. An explanation that involves a proposition that appears in every action definition might be harder to communicate than one that only uses a proposition that is part of the definition of a single action.

In addition to the actual size, the comprehensibility of the explanations may also depend on factors like human's mental load, the familiarity with the concepts captured by the propositions, etc.. To keep our discussions simple, we will restrict the cost of communicating an explanation to just the number of unique model updates this explanation would bring about in the human model.We will use the symbol $C^{\mathcal{E}}_{p}$ to represent the cost of communicating the changes related to the proposition p and also overload it to be applicable over sets of propositions.

Now our problem is to find the optimal explanation (represented as E_{min}) for a given set of foils $\Pi_{\mathcal{F}}$ or more formally

Definition 20. *A set of fluents E is said to be the optimal explanatory fluent set for the human model $\mathcal{M}_h^R$ and a foil set $\Pi_{\mathcal{F}}$, if*

1. *if E is an explanatory set and*

2. *there exists no other set $\widehat{E}$, such that $\widehat{E}$ is also an explanatory set and $C_E^{\mathcal{E}} > C_{\widehat{E}}^{\mathcal{E}}$.*

Given the fact that the human model is not known to start with, it may appear that there is no way to generate optimal explanations for the human model directly. A possible alternative might be to try identifying the set of fluents that is optimal for the set of models that could be $\mathcal{M}_h^R$. Calculating such an explanation naively could be extremely expensive as identifying all possible candidates for the human model would involve testing each node in the lattice for whether its a potential candidate for the human model and then searching over the space of all explanatory fluent set to find one that is optimal for the entire set of candidate models (where the optimality for a set of models is defined to be the cheapest set of fluents that is explanatory for all the models in the set). Thankfully, the properties of the lattice allow us to compute optimal solutions without keeping track of the entire set. Moreover, for lattices containing abstract models generated using procedures discussed in Section 9.1.1, we will see how fluent sets that are optimal for minimal abstracting set are still optimal for the original human model. *That is uncertainty over human models results in no loss of optimality.* But before proving that property, we will define the idea of the *resolution set*, that captures the specific plans resolved by concretizing the given propositions (i.e the proposition appears as an unsatisfied precondition or goal in the plan).

Definition 21. For a set of models $\mathbb{M}'$, a foil set $\Pi_{\mathcal{F}}$ and a proposition p, the **resolution set** $\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', p)$ gives the subset of foils that no longer holds in the concretized models generated through $f_{\Lambda}^{\text{safe}}$, i.e

$$\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', p) = \{\pi | \pi \in \Pi_{\mathcal{F}} \wedge (\forall \mathcal{M}' \in \mathbb{M}'(\pi(I_{\gamma_p(\mathcal{M}')}) \not\models_{\gamma_p(\mathcal{M}')} \ G_{\gamma_p(\mathcal{M}')} \wedge$$
$$\pi(I_{\mathcal{M}'}) \models_{\mathcal{M}'} \ G_{\mathcal{M}'}))\}$$

The idea of generating resolution sets are again closely related to the idea of resolving counter-examples used in CEGAR based method. We will also use $\mathcal{R}_{\Pi_{\mathcal{F}}}$ to also represent the set of foils resolved by a set of propositions. For notational convenience, we will use $\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', \{\})$ to capture the subset of foils that do not hold in the current model set $\mathbb{M}'$.

Proposition 16. *For a set of model* $\mathbb{M}'$ *and a foil set* $\Pi_{\mathcal{F}}$

$$\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', \{p_1, p_2\}) = \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', \{p_1\}) \ \cup \ \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', \{p_2\})$$

The above property implies that concretizing any n propositional fluents cannot resolve foils that weren't resolved by the individual fluents. The above property follows from the fact that adding a proposition into the model only resolves a foil if it adds a precondition not supported by previous actions in the plan. Since this is independent of other fluents already part of the abstraction, we can see that a set of fluents will only resolve the foils that are resolved by the individual elements of that set.

Proposition 17. *For two models* $\mathcal{M}_1$*,* $\mathcal{M}_2$ *and a set of foils* $\Pi_{\mathcal{F}}$*,*
if $\mathcal{M}_1 = f_{\Lambda}^{safe}(\mathcal{M}_2, \{p_1, .., p_k\})$ *then for any proposition* p*,*

$$\mathcal{R}_{\Pi_{\mathcal{F}}}(\{\mathcal{M}_1\}, \{p\} \supseteq \mathcal{R}_{\Pi_{\mathcal{F}}}(\{\mathcal{M}_2\}, \{p\}) \setminus \mathcal{R}_{\Pi_{\mathcal{F}}}(\{\mathcal{M}_2\}, \{\})$$

The proposition can be established by following the definition of resolution set and rewriting the lefthand side of the equation as

$$\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, p) = \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, \{p_1, .., p_k\} \cup \{p\})$$

From Proposition 16 we know

$$\begin{aligned}\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, p) &= \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, \langle p_1, .., p_k\rangle \cdot \langle p\rangle) \\ &= \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, p) \cup \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, \{p_1, .., p_k\})\end{aligned}$$

$$\begin{aligned}\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, p) &= \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, \langle p_1, .., p_k\rangle \cdot \langle p\rangle) \\ &= \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, p) \cup \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, \{\})\end{aligned}$$

Now removing elements $\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, \{\})$ from both LHS and RHS we get

$$\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, p) \setminus \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, \{\}) = \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_1\}, p) \setminus \mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}_2\}, \{\})$$

Which proves our original assertion.

This proposition directly leads to the following observation.

Proposition 18. *Let $\mathbb{M}^{\Pi_\mathcal{F}}_{min}$ be the minimal abstracting set for a foil set $\Pi_\mathcal{F}$ and $\mathcal{M}^R_h$ be the human model. if every model in $\mathbb{M}^{\Pi_\mathcal{F}}_{min}$ if formed from $\mathcal{M}^R_h$ through f^{safe}_Λ, then for any fluent set E_{min} that is optimal for $\mathbb{M}^{\Pi_\mathcal{F}}_{min}$ then E_{min} must be optimal for $\mathcal{M}^R_h$.*

We can show the validity of the above proposition through contradiction. To start with from the definition of foils we know, $\mathcal{R}_{\Pi_\mathcal{F}}(\{\mathcal{M}^R_h\}, \{\}) = \emptyset$ and thus $\mathcal{R}_{\Pi_\mathcal{F}}(\mathbb{M}^{\Pi_\mathcal{F}}_{min}, \{\}) = \emptyset$. Let us assume there exists an explanatory set E_1 that is optimal for human model but not optimal for $\mathbb{M}^{\Pi_\mathcal{F}}_{min}$. This could only be due to two possible reasons, i.e., E_1 is not an explanatory set for $\mathbb{M}^{\Pi_\mathcal{F}}_{min}$ or there exists another set E_2

that is optimal for $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$ but not applicable for $\mathcal{M}_h^R$. Through, Proposition 17 we have already established that any explanatory fluent set for human model must be an explanatory set for $\mathbb{M}_{min}^{\Pi_{\mathcal{F}}}$. Similarly, from Proposition 13, we know any explanatory set applicable for an abstract model set must be applicable for the concrete model as well.

Now the question is how to exactly identify E_{min}, one possibility is to perform an A* search (Hart *et al.*, 1968) over the space of possible fluent sets to identify E_{min}. Each search state consists of the minimal set of abstract models for the human model given the current explanation prefix. We will stop the search as soon as we find a state where the foils no longer hold for the current minimal set. In addition to the systematic search, we can see that the specifics of the setting also allows us to leverage greedy search (described in Algorithm 7). In each iteration of this search, the algorithm greedily chooses the proposition that minimizes $\frac{C_p}{|\Pi'_{\mathcal{F}} \cap \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}',p)|}$, where $\Pi'_{\mathcal{F}}$ is the set of unresolved foils at that iteration and the search ends when all foils are resolved.

Theorem 6. *The explanatory fluent set* $\widehat{E}$ *generated by Algorithm 7 for a set of foils* $\Pi_{\mathcal{F}}$ *and a lattice* $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ *is less than or equal to* $(\ln k) * C_{E_{min}}^{\mathcal{E}}$*, where* $C_{E_{min}}^{\mathcal{E}}$ *is the cost of an optimal explanatory fluent set and* k *represents the maximum number of foils that can be resolved by concretizing a single proposition, i.e,* $k = \max_p |\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}_{min}, p)|$.

Proof (Sketch). We will prove the above theorem by showing that Algorithm 7 corresponds to the greedy search algorithm for a weighted set cover problem. Consider a weighted set cover problem $\langle U, S, W \rangle$ such that the universe set $U = \Pi_{\mathcal{F}}$, the sub-collections set S is defined as $S = \{s_p | p \in \mathbb{P}\}$ where $s_p = \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}_{min}, p)$ and the cost of each subset s_p is gives as $W(s_p) = C_p^{\mathcal{E}}$. Proposition 16 ensures that the size of

Algorithm 7 Greedy Algorithm for Generating $\widehat{E}$

```
1: procedure GREEDY-EXP-SEARCH
2:     Input:    $\langle \Pi_{\mathcal{F}}, \mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle \rangle$
3:     Output: Explanation $\widehat{E}$
4:     Procedure:
5:     curr_model = $\langle \mathbb{M}_{min}, F \rangle$
6:     $\widehat{E} = \{\}$
7:     $\mathbb{M}_{min}$        $\leftarrow$MinimalAbstractModels($\mathcal{L}, \Pi_{\mathcal{F}}$)
8:     Precompute the resolution sets $\mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}_{min}, p)$ for each $p \in \mathbb{P}$
9:     while True do
10:        $\mathbb{M}', \Pi'_{\mathcal{F}}$ = curr_model
11:        if $|\Pi'_{\mathcal{F}}| = 0$ then return $\widehat{E}$        ▷ Return $\widehat{E}$ if all the foils are resolved
12:        else
13:            $p_{next} = \underset{p}{\text{argmin}}(\frac{C_p}{|\Pi'_{\mathcal{F}} \cap \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', p)|})$
14:            $\mathbb{M}_{new} = \{\gamma_{p_{next}}(\mathcal{M}) | \mathcal{M} \in \mathbb{M}'\}$
15:            curr_model = $\langle \mathbb{M}_{new}, F \setminus \mathcal{R}_{\Pi_{\mathcal{F}}}(\mathbb{M}', p) \rangle$
16:            $\widehat{E} = \widehat{E} \cup p$
```

resolution set is a submodular and monotonic function. In this setting, the act of identifying a set of propositions that resolve the foil set is identical to coming up with a set cover for U in the new weighted set cover problem. Furthermore, we can show that the optimal set cover $\mathcal{C}_{opt}$ must correspond to the cheapest explanation E_{min} (We can prove this equivalence using Propositions 13,15 and 16, we are skipping the details of this proof due to space constraints). Algorithm 7 describes a greedy way of identifying the cheapest set cover for this weighted set cover problem and thus the minimal explanation for the original problem. For weighted set cover the above greedy algorithm is guaranteed to generate solutions that are at most $\ln k * W(\mathcal{C}_{opt})$ (Young, 2008), where $k = \max_{s \in S} |s|$ and this approximation guarantee will hold for E_{min} as well. $\square$

We can use this algorithm to either generate solutions and or to calculate an inadmissible heuristic for the previously mentioned A* search. For the heuristic generation, we will further simplify the calculations (specifically step 8 in Algorithm 7) by considering an over-approximation of $\mathcal{R}_{\Pi_{\mathcal{F}}}$. Instead of considering the set of all foils resolved by concretizing each proposition p, we will consider the set of foils where p appears in the precondition of one of the actions in it. This set should be a superset for $\mathcal{R}_{\Pi_{\mathcal{F}}}$ for any proposition.

Now that we have formulated the basic form of explanation for this setting, we will look at how we can relax some of the assumptions made in earlier sections and how it effects the explanation generation problem. In particular, we will look at cases where the lattices are no longer proposition conserving, the users may be raising foils that are sub-optimal as opposed to invalid and finally how to support models with noise.

9.3.1 Supporting Explanation Generation for Non-proposition Conserving Lattices

Proposition conserving lattices, in particular, complete lattices provide a concise way for the problem designer to specify their knowledge about the end users. In fact, with well-defined abstraction functions, they need only specify the most concrete model and the set of most abstract models to generate the rest of the lattice. Unfortunately, there may be cases where such lattices may no longer be enough to capture all information the system designer may be capable of providing about the end users. For example, consider a scenario where a robot needs to put away groceries. The goal of the robot here is to put away a set of items in prespecified storage locations. In this case, medicines need to be put in the medicine cabinet while condiments should be placed in kitchen shelves. In addition to these task-level constraints, the robot's operations are restricted by various motion level constraints that limit the possible physical movements that the robot can perform, including possible ways an object can be grasped and areas in the workspace it can reach. Clearly, these two types of constraints are quite different in terms of the background knowledge needed to understand them. While the task constraints correspond to some simple rules of the task that are easy to explain to a lay user, understanding the motion constraints require knowledge about robotics that is usually absent in most users. Thus there is a natural hierarchy in the concepts related to this task. One way to capture such information could be by controlling the order in which the various fluents are considered for abstraction, i.e., remove a particular set of fluents before moving to others (thereby making the lattice non-proposition conserving). This means, the easier to understand fluents would get introduced higher up in the lattice and the harder to understand fluent appear lower in the lattice closer to the concrete node. The task mentioned above is a particularly good fit for non-proposition conserving lattices because even the motion constraints could be captured at multiple conceptual levels. In general, non-proposition conserving lattices are a useful tool to use when you have settings where there are different propositions that capture the same phenomena but at varying levels of detail or focus on different aspects. For example, in the case of picking up an object, one could talk about the ability to pick up the object, picking up the object by grasping a particular region and even grasping using a particular grasp point on the object. We can organize the lattice in such a way that the propositions are visited in the order that reflects the preferences of the end-user. For example, for this scenario, we can arrange the concepts in such a way that simpler concepts (for example propositions related to simple reachability) are tested before moving onto more complex concepts.

While there are reasons to choose non-proposition conserving lattices and we could generate explanations using such lattices with some minor modifications on the solution method described before, the use of such lattices also have a few disadvantages. The obvious one being that the designer now have to fully specify such lattices, also the use of such lattices prevents the use of heuristics and greedy search described in earlier sections. It should also be noted that when the foils can only be resolved by introducing fluents from lower levels then the search would still need to search through all the nodes in the above before identifying the nodes that resolve the foil. Also once such a node is identified, it won't be easy to separate the set of fluent that actually contribute to resolution from those that are redundant (particularly when there are multiple foils).

To overcome these shortcomings, we will allow designers to specify a non-proposition conserving lattice while the explanation generation algorithm itself operates on a modified proposition conserving lattice that uses an updated cost function. To achieve this, we will start by defining the concept of a well-formed lattice

Definition 22. *An abstraction lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ is said to be **well formed**, if there exists a unique minimal node (i.e the most concrete model), thus for any model $\mathcal{M} \in \mathbb{M}$, $\mathcal{M}^{\#} \sqsubseteq \mathcal{M}$.*

Any lattice we describe hence forth, will be assumed to be well-formed unless specified otherwise. While the concept of minimum abstraction set remains the same for a non-proposition conserving lattice, analyzing the results of concretizing the human model with respect to explanatory fluents requires us to look at a new concept named a completion of a lattice.

Definition 23. *For a given well formed non-proposition conserving abstraction lattice*

$\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, a second lattice $\hat{\mathcal{L}} = \langle \hat{\mathbb{M}}, \hat{\mathbb{E}}, \mathbb{P}, \hat{\ell} \rangle$ is said to be a completion if $\hat{\mathcal{L}}$ is a proposition conserving lattice, such that, $\mathbb{M} \subseteq \hat{\mathbb{M}}$, $\mathbb{E} \subseteq \hat{\mathbb{E}}$ and $\ell \subseteq \hat{\ell}$

A completion is relevant in this setting, because if we allow the system to freely choose propositions for the explanatory set, the updated human model (i.e the model obtained after the explanation) may not be part of the original non-proposition conserving lattice but is guaranteed to be part of the completion. Note that completions for a non-proposition conserving lattices are not unique, but in most cases we will consider a minimal completion. We can create such a completion by starting with the given lattice and adding any missing incoming edges iteratively (introducing new models only if there exists no current nodes that correspond to the set of missing propositions expected at the source of the edge).

Definition 24. *Given a non-proposition conserving lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, it's completion $\hat{\mathcal{L}} = \langle \hat{\mathbb{M}}, \hat{\mathbb{E}}, \mathbb{P}, \hat{\ell} \rangle$, the human model $\mathcal{M}_h^R \in \mathbb{M}$ and the foil set $\Pi_{\mathcal{F}}$, a set of propositions $E = \{p_1, ..., p_n\}$ is said to be a set of explanatory fluents if*

$$\forall \pi \in \Pi_{\mathcal{F}}, \pi(I_{\gamma_E(\mathcal{M}_h^R)}) \not\models_{\gamma_E(\mathcal{M}_h^R)} G_{\gamma_E(\mathcal{M}_h^R)} \text{ and } \gamma_E(\mathcal{M}_h^R) \in \hat{\mathbb{M}}$$

As the original human model is assumed to be part of the given lattice, it must be part of the completion as well, moreover, the relation between the min abstraction set and the human model is conserved in the completion as well. This means that any set of explanatory fluents identified by using the minimum completion of the given lattice would also be valid for the human model as well. Such a minimal completion lattice, need not be created beforehand, but could in fact be generated online when searching for the explanation. Unfortunately, directly using such a completion lattice for explanation generation (once the min abstraction set is found), would result in finding sets of propositions that ignore the information captured by the given lattice.

To incorporate this information we need to not only use the completion we need to consider a new cost function $C^{\mathcal{E}}_{\mathcal{L}}$ for the explanation generation.

Proposition 19. *Given a min abstraction set $\mathbb{M}_{min}$ for a non-proposition conserving lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, we can use it's completion $\hat{\mathcal{L}} = \langle \hat{\mathbb{M}}, \hat{\mathbb{E}}, \mathbb{P}, \hat{\ell} \rangle$ to identify the explanatory fluents provided the cost of explaining a given proposition p is defined as $C^{\mathcal{E}}_{\mathcal{L}}(p) = C^{\mathcal{E}}_{p} + \max_{\mathcal{M} \in \mathbb{M}_{min}} L(p, \mathcal{M})$, where $L(p, \mathcal{M})$ is a penalty, such that $L(p, \mathcal{M}) \propto C^{\mathcal{E}}_{\hat{P}}$ where $\hat{P}$ is the least costly set of propositions such that $p \in \hat{P}$ and $\gamma_{\hat{P}}(\mathcal{M}) \in \mathbb{M}$.*

This new penalty term ensures that a proposition is considered for explanation only after the propositions from higher levels of the given lattice is considered. Now that we are dealing with explanations using a new proposition conserving lattice, all earlier results directly carry over including the heuristic, though the search is less efficient as calculating the cost for each node requires lookup of the given lattice. Since the proposition conserving lattice is assumed to be provided upfront, we may be able to precompute the costs.

9.3.2 Supporting Explanations for Sub-optimal Foils

We will now consider scenarios where the explainee raises foils that are valid but may, in fact, be costlier than the one chosen by the robot. In such scenarios, we would want the robot to explain why the current plan may be preferred, but such explanations could be complicated by the fact that the actions in the domain may have state-dependent costs, for example, the cost of picking up a light block may be lower than picking up a heavier block. Here we would again need to present the user with a set of fluents and associated action costs that allow the user to correctly evaluate their alternate plans.

To investigate this setting, we will restrict our attention to cases where each action could be associated a set of positive conditional costs. We will consider a slightly updated action definition, where each action a for a model $\mathcal{M}$ is now defined by a tuple of the form $\langle prec_a, e_a^+, e_a^-, \mathcal{C}_a^{\mathcal{M}} \rangle$, where $prec_a, e_a^+$ and e_a^- are same as before and $\mathcal{C}_a$ are the set of state dependent costs associated with the action a. $\mathcal{C}_a^{\mathcal{M}}$ is itself defined as a set of individual costs of the from $\langle \phi, c \rangle$, where ϕ is a conjunction of state literals, which when satisfied in a state causes the action a to induce a cost c (where $c \in \mathbb{R}_{\geq 0}$). Now the cost of executing the action a at state s is defined

$$\mathcal{C}_a^{\mathcal{M}}(s) = \Sigma_{\langle \phi_i, c_i \rangle \in \mathcal{C}_a^{\mathcal{M}}}(\kappa(\phi_i, s, c_i))$$

Where $\kappa(\phi_i, s, c_i) = c_i$ if $s \models \phi_i$ else $\kappa(\phi_i, s, c_i) = 0$.

We will use the function $\mathcal{C}^{\mathcal{M}}$ to return the total cost of a plan for a given initial state, i.e, for a plan $\pi = \langle a_1, ..., a_n \rangle$ and an initial state I, $\mathcal{C}^{\mathcal{M}}(\pi, I) = \mathcal{C}_{a_1}^{\mathcal{M}}(I) + ... + \mathcal{C}_{a_n}^{\mathcal{M}}(a_{n-1}(...(a_1(I))...))$.

Following the convention set by Geißer *et al.* (2016), we can assert that such a domain model induces a transition system of the form $\mathcal{T} = \langle S, s_0, S_g, L, T, \mathcal{C}^{\mathcal{T}} \rangle$, which is similar to the original transition system definition except that now each transition is associated with a cost determined by both source state and action. An abstract model $\mathcal{M}'$ with a transition system $\mathcal{T}'$ for a set of propositions Λ is defined in a similar way with the cost of each transition (s, a, s') given by $\mathcal{C}^{\mathcal{T}'}(s, a, s') = min(\{\mathcal{C}^{\mathcal{T}}(\hat{s}, a, \hat{s}') \mid \hat{s}, \hat{s}' \in S \wedge f_\Lambda(\hat{s}) = s \wedge f_\Lambda(\hat{s}') = s'\})$.

We will also update the explanatory setting a bit and assume that the robot presents the user with the plan and the anticipated cost of the plan in the most concrete model (denoted as $\mathcal{C}_{\pi_R}$). The user responds by providing a foil set which they believe is less costlier than the plan in question. Here we can define a set of explanatory fluents to be

Definition 25. *A set of proposition $E = \{p_1, ..., p_n\}$ is said to be **explanatory fluents** for the human model $\mathcal{M}_h^R$ and a foil set $\Pi_{\mathcal{F}}$ if*

$$\forall \pi \in \Pi_{\mathcal{F}}, \pi(I_{\gamma_E(\mathcal{M}_h^R)}) \not\models_{\gamma_E(\mathcal{M}_h^R)} G_{\gamma_E(\mathcal{M}_h^R)} \vee \mathcal{C}^{\mathcal{M}_h^R}(\pi, I_{\gamma_E(\mathcal{M}_h^R)}) > \mathcal{C}_{\pi_R}$$

Revisiting the abstraction lattice, given the fact that we are dealing with only positive costs, the first property we can assert is that

Proposition 20. Given two models $\mathcal{M}_1$ and $\mathcal{M}_2$, such that $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$, then for any plan π, we have $\mathcal{C}(\pi, I_{\gamma_E(\mathcal{M}_1)}) \geq \mathcal{C}(\pi, I_{\gamma_E(\mathcal{M}_2)})$

This means that once we establish that a given foil is costlier than robot plan in a model, then it holds in all models that are more concrete than that one. This insight allows us to reassert Proposition 13 for this new extended definition of explanation and by extension allows us to use the idea of the minimal abstraction set in this new setting (Proposition 14 holds here as well).

This means that we can more or less directly use the search method discussed for the in-validity case here directly. Unfortunately, in this setting *the size of resolution set is no longer sub-modular* and hence we can not leverage the greedy method discussed for the pure invalidity case.

9.3.3 Supporting Explanations in the Presence of Human Models with Incorrect Beliefs

An underlying assumption for most of the earlier discussion has been the fact that the user's model of the task can be represented as an abstraction of the robot model, i.e. the user model may be imprecise but not incorrect. Unfortunately, this is not an assumption that can be met in all scenarios. More often than not, the user may not only be unaware of certain facts pertaining to the task but may also hold incorrect beliefs about it. Throughout this section, we will discuss how approaches discussed in earlier sections can be used to handle such cases.

Formally, let the real (but unknown) user model be $\mathcal{M}_h^R$ and we assert that this model is an abstraction of some (again unknown) model $\widehat{\mathcal{M}}_R$ that is defined over the same set of fluents as $\mathcal{M}_R$, but may have errors in regards to action definitions, perceived initial and goal state. Let us assume both $\widehat{\mathcal{M}}_R$ and $\mathcal{M}_h^R$ belong to the same class of planning problems as defined in Section 9.1. Again let the set of alternate plans raised by the user be $\Pi_{\mathcal{F}}$. It is important to note that the reason the user thinks these foils are valid may no longer be just due to missing fluents, but could also be due to the user's incorrect understanding of the task. This means that foils are not an accurate way of identifying the user's level of understanding, but we can still use the foils to figure out the level of abstraction at which the foils can be refuted. Though in scenarios with such models, we have to consider a complete lattice that contains all possible fluents (i.e assuming user could be wrong about the use of any of the fluents), i.e., the lattice we will use would be $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, where $\mathbb{P} = P_R$ (defined using f_{Λ}^{safe}). We can now use the methods described in earlier sections to find a set of explanatory fluents $\mathcal{E}$ that can refute the given set of foils. Once the information regarding the explanatory fluents in provided to the user, irrespective of the other fluents, the user should have a correct understanding of each fluents listed in $\mathcal{E}$. Let $\mathcal{M}_h^R + \mathcal{E}$ be the updated human model that contains the correct information about $\mathcal{E}$. Note that even though $\mathcal{M}_h^R$ or $\mathcal{M}_h^R + \mathcal{E}$ may not be part of $\mathcal{L}$, the abstraction of this updated human model that projects out all propositions absent from $\mathcal{E}$ must be part of the lattice $\mathcal{L}$, i.e, $f_{P \setminus \mathcal{E}}((\mathcal{M}_h^R) + \mathcal{E}) \in \gamma_{\mathcal{E}}(\mathbb{M}_{min})$. In this scenario, $\gamma_{\mathcal{E}}(\mathbb{M}_{min})$ will be singleton set and we will represent the only element in this set as $\mathcal{M}_{min}$. As per the definition of valid explanation, we know that $\mathcal{R}_F(\mathbb{M}_{min}, \mathcal{E}) = \emptyset$ and since $\gamma_{\mathcal{E}}(\mathcal{M}_h^R) \sqsubseteq \gamma_{\mathcal{E}}(\mathcal{M}_{min})$ and therefore the resolution set for $\gamma_{\mathcal{E}}(\mathcal{M}_h^R)$ must also be empty.

9.4 Evaluations

9.4.1 Empirical Evaluations on Explanation Generation for Invalid Foils

For our empirical evaluation, we wanted to understand how effective our basic approaches were in terms of the conciseness of the explanations produced, the solution computation time and the usefulness of approximation. For the approximation, we were interested in identifying the trade-off between decrease in runtime vs. reduction in solution quality. Since both explanation for incorrect beliefs and non proposition-conserving gets compiled down to finding explanation on proposition-conserving lattices, we didn't perform separate evaluations for those methods. All three explanation methods discussed in this chapter (blind, heuristic and greedy) were evaluated on five IPC benchmark domains (International Planning Competition, 2011). All the experiments detailed in this section were run on an Ubuntu workstation with 64G RAM.

For each domain, we selected 30 problems from either available test sets or by using standard problem generators (the problems sizes were selected to reflect the size of previous IPC test problems). The lattice for each problem-domain pair was generated by randomly selecting 50% of domain predicates and then generating a fully connected proposition conserving lattice using that set of predicates. Since none of the models contained any conditional effects, we created the abstract models by dropping the propositions to be abstracted from the domain models (which are complete for these domains). The foils were generated by selecting random models from the lattice and creating plans from these models that do not hold in the concrete model. Each search evaluated here, generates the set of proposition whose concretizations can resolve the foils set $\Pi_{\mathcal{F}}$. In actual applications, this set of propositions needs to be converted into an explanan (the actual message) by considering how this proposition is used in the robot model. Figure 9.4 shows the explanation generated by our approach for a

Domain Name	$C_{\mathbb{P}}$	$\lvert\mathbb{P}\rvert$	$\lvert\Pi_{\mathcal{F}}\rvert$	Blind Search (Optimal)			Heuristic Search			Greedy Set Cover		
				Cost	Size	Time(S)	Cost	Size	Time(S)	Cost	Size	Time(S)
	84.07	7	1	6.87	1	2.43	6.87	1	2.08	6.87	1	3.61
Barman	84	7	2	8.94	1.22	6.35	8.94	1.22	5.71	9.90	1.39	6.05
	90.7	7	4	17.19	1.77	24.99	17.19	1.77	23.7	18.45	1.97	10.34
	168.66	12	1	3.58	1	7.86	3.58	1	5.22	3.58	1	19.18
Rover	188.83	12	2	6.13	1.48	51.36	6.12	1.48	34.04	6.26	1.52	30.5
	192.83	12	4	10.87	2	203.83	10.87	2	181.87	11.42	2.19	49.32
	53.01	4	1	18.73	1	2.23	18.73	1	1.92	18.73	1	1.49
Satellite	60.77	4	2	32	1.61	7.21	32	1.6	5.86	32.53	1.7	3.04
	62.73	4	4	43.27	2.29	18.67	43.27	2.29	16.42	43.88	2.39	5.85
	156.71	7	1	14.45	1	2.84	14.45	1	2.23	14.45	1	3.35
Woodworking	146.33	7	2	20.62	1.21	6.88	20.62	1.21	4.93	21.38	1.38	6.25
	154	7	4	28.62	1.69	24.70	28.62	1.69	19.49	30.41	2	12.13
	220.6	3	1	51.21	1	1.51	51.21	1	1.35	51.21	1	1.28
Sokoban	151.72	3	2	94.52	1.55	3.93	94.52	1.55	3.35	98.31	1.73	2.59
	220.69	3	4	136.41	2.22	8.75	136.41	2.22	8.3	141.93	2.37	5.23

Table 9.1: Table Showing Runtime/Cost for Explanations Generated for Standard IPC Domains. Column $\lvert\mathbb{P}\rvert$ Represents Number of Predicates That Were Used in Generating the Lattice, While $C^{\mathcal{E}}_{\mathbb{P}}$ Represents the Cost of an Explanation That Tries to Concretize All Propositions in $\mathbb{P}$ and Provides an Upper Bound on Explanation Cost.

problem in Rover domain.

Table 9.1 presents the results from our empirical evaluation on the IPC domains. The table shows the average cost/size of each explanation along with the time taken to generate them. Note that by size, we refer to the number of predicates that are part of the explanation while the cost reflects the total number of unique model updates induced by that explanation. We attempted explanation generation for foil set sizes of one, two and four per problem.

Our main conclusion is that heuristic search seems to outperform blind search in almost every problem and generates near-optimal solutions (Blind search always generates the minimal explanation). Further, we saw that greedy search outperformed heuristic search in most cases barring a few exceptions. The greedy search was able

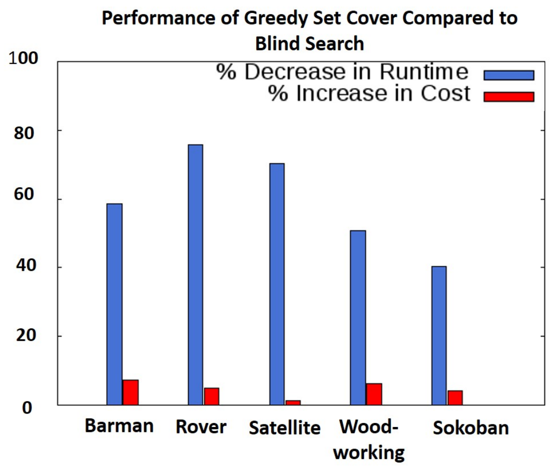
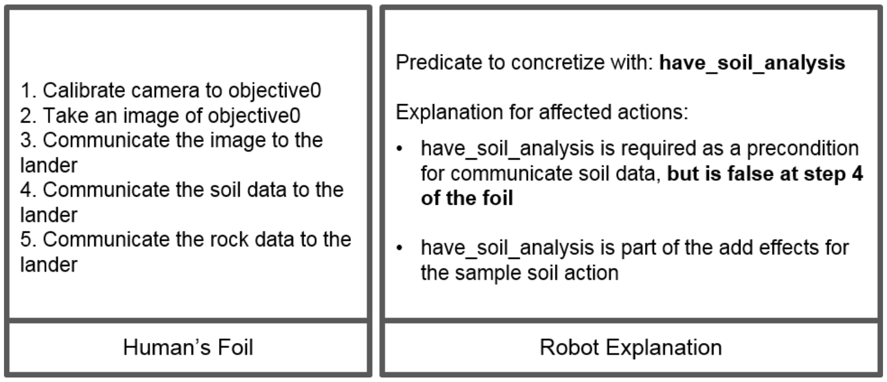


Figure 9.3: The Graph Compares the Performance of Greedy Set Cover Against the Optimal Blind Search for $|\Pi_{\mathcal{F}}| = 4$. It Plots the Average Time Saved by the Set Cover and the Average Increase in Cost of the Solution for Each Domain.

to make significant gains especially for higher foil set sizes. This is entirely expected due to the fact that step 8 in Algorithm 7 can be expensive for problems with long plans (but still polynomial). This expensive pre-computation pays off as we move to cases where E_{min} consists of multiple propositions. Additionally, we found out that greedy solutions were quite comparable to the optimal solutions with respect to their costs. For example in $|\Pi_{\mathcal{F}}| = 4$ for satellite domain, while the greedy solution cost took a penalty of $\sim 1.4\%$ the search time was reduced by $\sim 68\%$. Figure 9.3 plots the comparison between the time saved by the greedy search versus any loss in optimality incurred by the greedy search.

9.4.2 Empirical Evaluations on Explanation Generation for Sub-optimal Foils

Next, we wanted to evaluate the empirical performance of the approach for domains with state dependent cost. For this setting, since we don't have standard benchmark domains with this property, we chose standard IPC domains and modi-



1. Calibrate camera to objective0
2. Take an image of objective0
3. Communicate the image to the lander
4. Communicate the soil data to the lander
5. Communicate the rock data to the lander

Human's Foil

Predicate to concretize with: **have_soil_analysis**

Explanation for affected actions:

- have_soil_analysis is required as a precondition for communicate soil data, **but is false at step 4 of the foil**
- have_soil_analysis is part of the add effects for the sample soil action

Robot Explanation

Figure 9.4: An Example Explanation Generated by Our System for IPC Rover Domain. The Human Incorrectly Believes That the Rover Can Communicate Sample Information Without Explicitly Collecting Any Samples. While the Abstraction Lattice in This Example Was Generated by Projecting out Upto 12 Predicates, the Search Correctly Identifies Concretizations Related to *(Have_soil_analysis ?R - Rover ?W - Waypoint)* as the Cheapest Explanation ($C^{\mathcal{E}}_{E} = 2$ as Opposed to $C^{\mathcal{E}}_{\mathbb{P}} = 55$)

fied them to include conditional cost updates. In particular, we chose blocksworld, zenotravel, gripper and rover. For blocksworld, we introduced three new predicates, namely heavy, light and unsteady each of which takes a block as an argument. For each problem instance, we assigned each block to be either heavy or light and set some of the blocks as unsteady. We also updated the stack action so that stacking a heavy block on a light one or an already unsteady one cause the block to be unsteady. We also set a high cost penalty for stacking any block on an unsteady one. For zenotravel, we came up with three binary predicates near, farther and farthest that takes cities as arguments. We also assigned a higher cost for traveling between far away cities than nearby ones (so the optimal plan may involve the plane making a lot more stops). For gripper, we again mark a ball to be heavy or light and now each robot can also pick up two balls at the same time. We assign a high cost to picking up heavy balls and picking up the second ball in a gripper that is already holding

Domain	Average Explanation Size	Runtime
BLOCKS	4.4	8.319
Gripper	5.4	7.368
Rover	4	9.690
Zeno	6.6	8.905

Table 9.2: The Sample Runtime and Average Explanation Size for Five Problem Instances from the Modified Domains.

a ball. We also provide the robot with a push action, that allows for it to move heavy balls without accruing large cost.Finally in the case of rover domains, we set some of the waypoints as being hilly area and communicating from these waypoints are assigned higher costs. Table 9.2 presents the explanation generation time and average explanation sizes for the modified domains. For each domain, we generated five problems and the test was run using systems of the same configuration as Section 9.4.1. For Blocksworld we considered instances where the number of blocks spanned from four blocks to 20, for Gripper all problems had two rooms and up to 12 balls and for Rover domain problems had upto three objectives and four waypoints. Finally, in Zenotravel all problems considered traveling between 10 cities and the number of passengers ranged from 20 to 60. The fact plan being explained were generated using optimal planners when possible and the foils were generated either using a satisficing planner (Metric-FF (Hoffmann, 2003)) or hand written using knowledge about the domain. As expected, the search was able to find the minimal number of predicates to be included into the problem to resolve the foils, for example in Blocksworld, the approach was able to correctly identify the predicate unsteady as being enough to explain the foils in the example.

In addition to the empirical results discussed in this chapter on classical planning problems, the approaches discussed have also shown to be useful in modeling explanatory dialogue in the context of Task and Motion Planning (as shown in (Sreedharan *et al.*, 2018b)).

9.4.3 User Study to Evaluate Role of Abstractions in Explanation

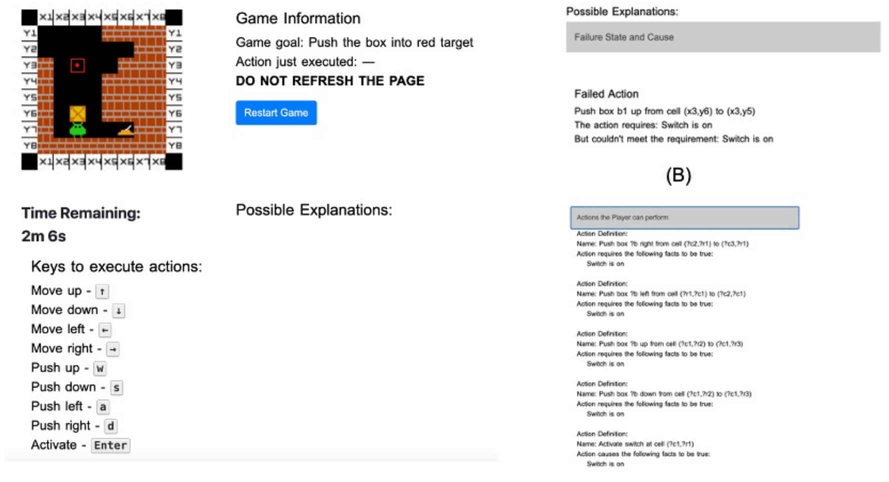
In this section, we will consider one of the assumptions that we made throughout the work, namely that providing the explanations at an abstract level would help reduce the cognitive burden on the user's end. Specifically, we will test the following hypothesis

Hypothesis 1. *Given two models $\mathcal{M}_1$ and $\mathcal{M}_2$, such that $\mathcal{M}_1 \sqsubset \mathcal{M}_2$ and $\mathcal{M}_2$ is formed using methods presented in Section 9.1.1, a user would find it easier to work with the more abstract model $\mathcal{M}_2$ when compared to $\mathcal{M}_1$*

We will evaluate this hypothesis over two different dimensions. One with respect to the subjective workload the user may experience when working with such a model to achieve some task, and then with respect to the actual ability of the user to successfully complete the task. For the former, we will employ NASA-Task Load Index (NASA-TLX) survey (Hart and Staveland, 1988), while for the latter we measure the time taken by the user to complete the task. NASA-TLX is a very influential and widely used method to gauge the subjective workload experienced by the user. NASA-TLX, divides the workload of a task over six different dimensions; namely, Mental Demand, Physical Demand, Temporal Demand, Effort, Performance, and Frustration. The users are first asked to rate the task, across these dimensions on a 20 point scale (with larger value denoting higher workload). They are then required to provide relative weights across these dimensions, by making pairwise choices between these different dimensions. A weighted average of the ratings provided across these dimensions is then used as a measure of the workload.

For the actual study, we relied on a between-subject study design, wherein the study participants are divided into two groups. All study participants were students from ASU. One received abstract explanations and the other group was given concrete domain model information as explanations. As the task in question, we used a variation of the Sokoban domain, that involves the agent pushing a box to a prespecified domain. Unlike the common versions of Sokoban, this variant involved the robot needing to first turn on a switch before pushing the boxes. Each participant in the study was allowed to play the game through a web-interface (which is shown in Figure 9.5, along with a sample explanation). While they were told the actions they can perform, they weren't told what each of the action achieves or their preconditions. Each player was allotted a total time of five minutes to complete the game. As the users play the game and if they perform an invalid action, they were provided with an explanation appropriate for their group.

For both groups, the current action sequence being executed was treated as the foil and the explanation consisted of the following information; the state at which the sequence failed, the specific action that failed, the expected set of preconditions, the failed precondition, and lifted model information about relevant actions. While one group of users were shown the information with respect to the concrete model, i.e., they were shown the full state, all the preconditions, all failed preconditions, and the entire model of the task, the second group was shown the abstracted version of each of the above-mentioned information. The level of abstraction for this group is identified based on the foil failure. To make sure the explanation generation time is symmetric between the groups, we avoided search to identify the best level of abstraction and rather we simulated the failing foil in the most concrete model and randomly selected



Game goal: Push the box into red target
Action just executed: —
DO NOT REFRESH THE PAGE

Time Remaining:
2m 6s

Keys to execute actions:
Move up - ↑
Move down - ↓
Move left - ←
Move right - →
Push up - w
Push down - s
Push left - a
Push right - d
Activate - Enter

Possible Explanations:

Possible Explanations:
Failure State and Cause

Failed Action
Push box b1 up from cell (x3,y6) to (x3,y5)
The action requires: Switch is on
But couldn't meet the requirement: Switch is on

Actions the Player can perform
Action Definition:
Name: Push box ?b right from cell (?c2,?r1) to (?c3,?r1)
Action requires the following facts to be true:
Switch is on

Action Definition:
Name: Push box ?b left from cell (?r1,?c1) to (?c2,?c1)
Action requires the following facts to be true:
Switch is on

Action Definition:
Name: Push box ?b up from cell (?c1,?r2) to (?c1,?r3)
Action requires the following facts to be true:
Switch is on

Action Definition:
Name: Push box ?b down from cell (?c1,?r2) to (?c1,?r3)
Action requires the following facts to be true:
Switch is on

Action Definition:
Name: Activate switch at cell (?c1,?r1)
Action causes the following facts to be true:
Switch is on

Figure 9.5: Screenshots from the User Interface Exposed to the End User. (A) The Participant Is Shown the Current State of the Game, They Are Allowed to Control the Agent via Their Keyboard and Whenever They Perform an Invalid Action, They Are Shown a Possible Explanation. (B) and (C) Presents a Sample Explanation Provided to Participants Who Were Exposed to Abstract Explanations. Here the Current State Shown to the Participant Is Empty as None of the Facts Are True in That State.

the predicate corresponding to one of the failing precondition to generate the abstract models. While this may result in a more detailed model than required, as we will see, even with this simple approach we did see a significant difference between the two groups. We also carried over predicates from consecutive failures, so the users of the second group saw increasingly more concrete models if they failed repeatedly (though still more abstract compared to the first group).

In total, we collected responses from 28 participants, 14 of whom had access to concrete explanation (henceforth referred to as Concrete-explanation group), and the remaining 14 were provided with abstract explanation (i.e the Abstract-explanation

Scale	Concrete-Explanation Group	Abstract-Explanation Group
Effort	1.595	1.252
Frustration	2.5	2.19
Mental Demand	2.9	1.414
Performance	1.186	1.705
Physical Demand	0.038	0.152
Temporal Demand	1.929	1.719

Table 9.3: The Weighted Average Workload Reported by the Participants of the User Study Across the Individual Scales Used in NASA-TLX.

group). While the Concrete-explanation group on average took 200.857 secs to finish the task, the abstraction group only took 163.5 seconds. In terms of the weighted average workload for the Concrete-explanation group, we saw 10.147 and for the Abstract-explanation group, we found it to be 8.433. The distribution across the six scales are presented in Table 9.3. As seen from the table, in all but Performance and Physical demand, people reported a higher workload for the concrete explanation group. We see a particularly significant difference across the mental demand dimension, which was the main focus of the assumptions made by our work. Thus the results from both the subjective workload study, and the performance of the user (measure in terms of the time taken by the user to finish the task), conform to our original hypothesis and we see that the use of abstractions provides a distinct advantage over providing complete details.

9.5 Related Work

There is increasing interest within the automated planning community to solve the problem of generating explanations for plans ((Fox *et al.*, 2017; Langley *et al.*, 2017)).

Earlier works (Seegebarth *et al.*, 2012; Bercher *et al.*, 2014; Kambhampati, 1990) have looked at explanations as a way of describing the effects of plans, while methods like those presented by Sohrabi *et al.* (2011b) and Meadows *et al.* (2013) have looked at plans itself as explanations for a set of observations. Another approach that has received a lot of interest recently is to view explanations as a way of achieving model reconciliation (Chakraborti *et al.*, 2017). Such explanations are seen as a solution to a *model reconciliation problem* (referred to as MRP) and this approach postulates that the goal of an explanation is to update the model of the observer so they can correctly evaluate the plans in question. The methods discussed in this Chapter can be seen as performing a type of model-reconciliation, but one could also leverage the methods discussed here to relax some of the assumptions made by model-reconciliation works for certain conditions We discuss the relationship between model-reconciliation and the methods studied in this Chapter in more detail in Section 9.6

As noted, our work is closely related to the well studied method of counter-example guided refinement or CEGAR that was originally developed for Model checking (Clarke *et al.*, 2000). Many planning works have successfully used CEGAR based methods to generate heuristics for plan generation (Seipp and Helmert, 2018, 2014). The idea of foil resolution set for a given concretization is also closely related to the process of identifying spurious counter examples employed by CEGAR based methods (cf. (Haslum *et al.*, 2012; Keyder *et al.*, 2012; Steinmetz and Hoffmann, 2016)). One major difference between our work and standard CEGAR based methods is the fact that in our setting the abstract model producing the foil (or counter-example) is unknown. Since we are exclusively dealing with spurious counter-examples we are also not bound to testing our foils (in other words identifying faults or pivot states) in the most concrete model (which could be quite expensive). Further, traditional CEGAR methods are generally not as focused on identifying the cheapest refinements.

Many abstraction schemes have been proposed for planning tasks (starting with Sacerdoti (1974)), but in this chapter, we mainly focused on state abstractions and based our formulation on previous methods like those discussed by Srivastava *et al.* (2016) and Bäckström and Jonsson (2013). It would be interesting to see how we can extend the approaches discussed in this chapter to handle temporal and procedural abstractions (e.g., HLAs (Marthi *et al.*, 2007)).

There exists a rich body of literature that has debated and discussed the role of abstraction in Social sciences (cf. (Garfinkel, 1982; Hitchcock and Woodward, 2003) for arguments towards abstraction, while Bechlivanidis *et al.* (2017) argues for adding more details provided the task constraints allow for it). Unlike these works that study explanation in everyday scenarios, explanation in the context of AI systems have a markedly different flavor, in so far that the explainer may be representing and reasoning about the task at levels of details that may be too hard for the users to understand. Thus abstraction can be a powerful tool in identifying just the required level of information to allow people to achieve their goals. This is an intuition being leveraged by more and more works to help generate explanations or even decisions that are easier to understand. For example, state abstractions have been leveraged by Ferrer-Mestres *et al.* (2020) to generate simpler models that generate easier to understand policies, and Topin and Veloso (2019) used abstraction to simplify policies. Even in the realm of machine learning explanations, abstractions have been considered as a way to generate multi-resolution explanations (Bayani and Mitsch, 2020). The importance of adjusting the level of details for different users have also been considered and argued by Martin *et al.* (2019), where they propose three levels of explanations, namely, high-level, low-level, and co-created level explanations. While high and low-level explanations focus on generating summaries and detailed descriptions respectively, co-created explanations use the user interaction to determine the contents of the explanation. Our specific methods could be considered closely related to the co-created explanation studied in the paper.

There have also been recent works that have looked at generating contrastive explanations for planning. Some significant examples for these include the ones presented by Eifler *et al.* (2020b) and Cashmore *et al.* (2019). Both these cases treat the cause of user's confusion to be their limited computational capabilities and the explanations tend to help them realize the consequences of following the foil without worrying about model reconciliation.

A closely related but distinct form of explanations is the one where the explanan (i.e. the information provided to the explainee) constitutes a counterfactual example (Keane and Smyth, 2020). Such explanations are particularly popular in classification settings, where when queried about an inexplicable classification, the system responds with a counterfactual example where the desired decision may have been made. Note that in such cases, the system needs to focus on generating counterfactual instances that the user would find acceptable. Many recent works have looked at identifying desirable properties for such counterfactual explanations (cf. (Keane and Smyth, 2020; Byrne, 2019)), and some of the prominent ones identified in the literature include, making sure the counterfactual example is close enough to the decision-point in question and the counterfactual is plausible, in terms of not only being a plausible datapoint but also that it is actionable. Actionability can be particularly important in domains like loan approval, wherein the counterfactual represents the changes the user needs to make to achieve the desired outcomes. Note that in our method, it is the user who is responsible for generating the counterfactual example and as such is guaranteed to come up with foil they believed to be most likely or most useful. Thus our focus has been on ensuring that the explanations generated in response meet the desired properties discussed in the literature. As discussed above, our explanations do meet many of the important requirements discussed in the literature including being selective and social.

9.6 HELM and Model Reconciliation Explanation

As mentioned earlier, the methods discussed in this case could be seen as a special case of model reconciliation. Here the model updates are limited to model concretization and the human's model is an abstraction of the original model. Rather than assuming that we are given an explicit human model, we assume that the human model belongs in the set of possible models that corresponds to the various abstractions of the robot model. In this sense, this method is also comparable to the work done on generating explanations for a set of possible models (Chapter 6), and in particular to the conformant explanations studied in that chapter. Though unlike the methods discussed in Chapter 6, in this setting we can guarantee that the conformant explanation is also minimal for the unknown human model (provided all model updates and hence explanations are restricted to model concretizations over fluents). Our use of minimal abstractions for explanations also allows the methods to handle cases where the user questions arise due to a mismatch in the inferential capabilities and not just a mismatch in the knowledge about the task. While the original model reconciliation work focused on explanations that address all possible foils, our work specifically tries to address foils raised by the user. This allows us to provide more concise explanations and allows us to scale to larger problems as compared to the original MRP approaches.

Another way to connect this work with model reconciliation is to leverage the insights from Section 9.3.3, to show that the method described in this chapter can also be used in the more general model-reconciliation setting. Section 9.3.3 shows that for the class of planning problems studied in this chapter, even when the human model may not meet the assumption that it is, in fact, an abstraction of the original robot model, we can still generate an explanation that refutes the given set of foil using abstractions formed from the robot model. Provided we use a complete abstraction lattice which contains a single maximal node formed by projecting out all the propositions. This means for explanatory queries related to just refuting alternate plans, abstraction lattices give us a way to circumvent one of the most restrictive assumptions made in model-reconciliation works, namely the need to know or learn the human model. As discussed in Section 9.3.3, the explanations generated over abstraction lattices will remain valid model-reconciliation explanations regardless of how the human model may be different from the robot model (provided it is still of the form described in Section 9.1 and doesn't contain any fluents absent from the robot model). Though compared to model-reconciliation techniques like those studied in Chapter 4, the methods discussed in this chapter could generate much larger explanations. For one, the explanations here involve providing information about all the uses of the explanatory fluents in the robot model, many of which the user may already know. This approach can also be extended to generate explanations of unsolvability and for partial foils of the type discussed in Chapter 10

9.6.1 Properties of Explanations

In Chapter 4, we used four properties to characterize the various types of explanations that were introduced in the chapter. These properties were Completeness, Monotonicity, Conciseness, and Computability. We too can use these properties to describe the explanations we have looked at (with small updates to meet our specific setting).

Any explanation generated by our methods will be complete and monotonic. While Chapter 4 defines a complete explanation as one that guarantees optimality of the plan under question. For our scenario, a complete explanation can be redefined as one that resolved all the given foils ($|\mathcal{R}_{\Pi_{\mathcal{F}}}(\Pi', E)| = |\Pi_{\mathcal{F}}|$). Chapter 4 considers an explanation monotonic if no future explanation can invalidate it. In our setting, this means that once a foil has been resolved by an explanation, no future explanation (or model concretizations) can reintroduce it. Which is satisfied by any explanation as concretization.

As for the remaining two properties (Conciseness and Computability), the definitions laid out in the original MRP chapter directly applies for our setting as well. Similar to MRP explanations, computability and conciseness remains incompatible properties for explanations in our case too. The explanations that are easier to compute end up being neither concise nor easy to understand. For example, one simple strategy to provide explaining a plan would be to provide enough details to the explainee that the human model completely converges to the robot model, but this strategy could be extremely expensive and even unnecessary.

In addition to these properties, we also saw how the explanations discussed in Chapter 4, also aligns with characteristics of useful explanations as identified by works from social sciences (Miller, 2017a). Chief among them is generating contrastive explanations, which remains the central thrust of the methods discussed in this work. The other two properties usually cited by such sources are *selectiveness* and being *social*. An explanation is considered selective if it chooses to focus only on the aspects relevant to the current explanatory query. As such this is directly related to the minimality of explanation and thus the methods discussed in this chapter can be considered to be selective. On the other hand, an explanation is considered social if it is tailored to the user's background. Our method supports this property in two distinct ways, one by explicitly trying to localize the user's model on the abstraction lattice, and by allowing the abstraction lattice itself to be tailored to reflect the preferences of the users.

9.7 Concluding Remarks

This chapter establishes the basic framework of HELM that will act as our primary explanation generation method through the majority of part of the thesis. In addition, to introducing the basic framework, our user study also establishes the primary result that we will be leveraging through most of the following chapter, namely the fact that use of state abstraction does reduce the cognitive burden placed on the human by the explanations.

The explanatory approaches discussed in this chapter have mostly focused on helping users resolve their confusion about foils, but it may also be possible that they have questions about the original robot plan. For the robot plan $\pi_R = \langle a_1, ..., a_n \rangle$, user could raise questions of the following types

1. Why perform action a_i? (where $a_i \in \pi_R$)
2. How can action a_i be performed when the precondition p is not met? (where $a_i \in \pi_R$ and $p \in pre_+(a_i)$ or $p \in pre_-(a_i)$ in the human model $\mathcal{M}_h^R$)

Question (1) captures the user's concerns regarding the use of any particular action in the plan, while question (2) captures their concerns regarding the validity of the plan. Other questions, such as achievement of goals and questions about the overall plan can be cast in terms of these more basic questions. For answering questions of the first type, we can easily adapt approaches discussed in works like Seegebarth *et al.* (2012) For a given action, these approaches try to find causal links that capture the specific action's contributions. We can leverage the hierarchy specified by the abstraction lattice to identify causal links consisting of higher-level concepts.

For Question (2), it is possible to view such questions as another type of foil.

While in earlier sections we tried to find abstract models where a particular foil can be refuted, here we just need to find the level at which the specified preconditions can be met. In the absence of disjunctive preconditions, we wouldn't need to perform a search to find such models, but rather choose the first abstract model where fluents corresponding to the preconditions in question are introduced.

Finally, while we did mention that one could couple the model information with additional plan failure information in the explanation, we didn't delve into too much detail. Such information corresponds to a larger class of information called explanatory witness, which we will cover in more detail in Chapter 12.

Chapter 10

HANDLING PARTIAL FOILS AND EXPLAINING PROBLEM UNSOLVABILITY

In this chapter, we will revisit the HELM framework and relax one of the central assumptions made in that framework, namely that the human would provide a set of fully specified plans as foils and also extend it to support explaining the unsolvability of a given planning problem.

We will start with explaining the unsolvability of the planning problem, and present an extension of HELM, where we will try to find the most abstract model (where abstraction models are generated through state abstractions) where the unsolvability of the planning problem can be established. In addition to state abstractions, in this version of the framework we will also leverage an additional technique to reduce the inferential burden placed on the user. Namely, help the user understand the infeasibility of a given planning problem by pointing out unreachable but necessary subgoals. Thus the user no longer has to reason about the unsolvability of the true original problem, but can focus on the unsolvability of a smaller subproblem.

In this chapter, we will see how we can use the same framework for explaining unsolvability to explain invalidity of partial foils. We can do this by leveraging the basic intuition that establishing invalidity of partial foils is equivalent to establishing unsolvability of planning problems with additional plan constraints. We will use a very generous definition of partial foils and take it to mean any temporal preferences on the type of solutions expected. In fact, we will go one step further and just consider explaining the unsolvability of planning problems in the presence of plan advice (Myers, 1996). By placing it in the context of planning problems with plan advice, we can capture additional scenarios where the constraints don't necessarily

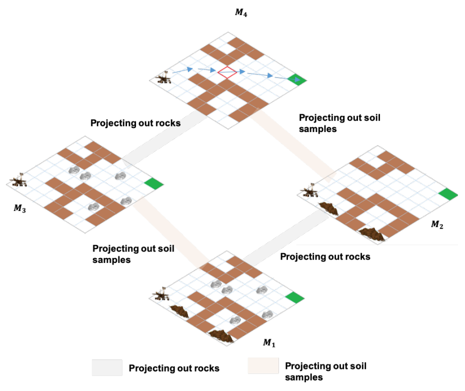

M_4
Projecting out rocks
Projecting out soil samples
M_3
M_2
Projecting out soil samples
Projecting out rocks
M_1
Projecting out rocks
Projecting out soil samples

Figure 10.1: A Sample Abstraction Lattice. The Lattice Consists of Models Generated by Projecting out Rocks or Soil Samples. Dark Blobs Are Locations for Soil Samples, Gray Objects Are Rocks, and the Goal Is Marked in Green. The Problem Is Unsolvable in the Most Concrete Model but Solvable in Models Where Rocks Are Projected Out.

correspond to foils raised by the user, but could be just solution preference the user had, whose inclusion rendered the problem unsolvable.

10.1 Our Approach

Before we start discussing the technical details of our approach, let us look at a possible explanatory scenario.

Example 2. *Consider the following scenario where a rover is tasked with collecting a rock sample and a soil sample from the region illustrated in Figure 10.2. The rover can only traverse the region via the waypoints marked on the map and its maneuverability is affected by the conditions of the terrain. The rover cannot easily traverse the region between P3 and P4 without special precautions as the region is quite rocky. Suppose a mission control operator is also keeping track of the rover's plan but may not have access to a map with the same level of fidelity or may have incomplete knowledge*

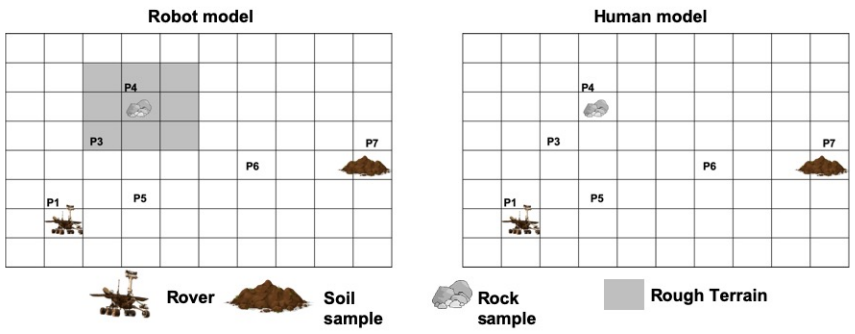


Figure 10.2: The Map for the Rover Mission Planning Problem. The Rover Is Required to Collect a Rock Sample and a Soil Sample and Then Return to the Original Position P1. One of the Rock Samples Is Located in Rough Terrain (Gray) That Can Not Be Traversed by the Rover. The Mission Control Operator Who Is Monitoring the Plan Is Currently Unaware of This Detail.

of the rover's capabilities. The rover reports to the mission controller that in fact the task can not be solved. The mission control operator is confused by the rover's response and could even ask
"Why don't you collect the rock sample from P4 and Soil sample from P7?"
Here if the rover wants to explain the reason as to why it couldn't achieve the goal a possible way would be to clarify that certain parts of the map are hard to traverse (particularly the region around the rock sample) and because of this issue it can never reach the location of the rock sample. Thus the explanation in this case consist of two distinct parts, information about the problem (i.e traversability of certain paths) and the required subgoal that can no longer be achieved in the light of this new information. In the proceeding sections, we will layout our framework and discuss how we could leverage it to generate such explanations.

The input to our approach thus includes an unsolvable problem represented by the model $\mathcal{M}_R = \langle F_R, A_R, I_R, G_R \rangle$ (in the above example this would correspond to the complete rover model) and an abstraction lattice $\mathbb{L}_{\mathcal{M}_R,P} = \langle \mathbb{M}, \mathbb{E}, \ell \rangle$, where $\mathbb{M}$ represents the space of possible models that could be used to capture the human's understanding of the task (i.e by assuming the user may or may not be aware of the fluents in the set $P \subseteq F_R$). In Example 2, P could include fluents related traversability of various paths or fluents related to various rover capabilities. Given this setting, our method for identifying explanations, includes the following steps

- Identify the level of abstraction at which the explanation should be provided (Section 10.1.1)
- Identify a sequence of necessary subgoals for the given problem that can be reasoned about at the identified level of abstraction (Section 10.1.2)
- Identify the first unachievable subgoal in that sequence (Section 10.1.3)

Intuitively, one could understand the three steps mentioned above as follows. First, identify the level of detail at which unsolvability of the problem needs to be discussed. The higher the level of abstraction, the easier the user would find it to understand and reason about the task, but the level of abstraction should be detailed enough that the problem is actually unsolvable there. In most cases, this would mean finding the highest level of abstraction where the problem is still unsolvable.

Now even if the system was to present the problem at this desired level of abstraction, the user may be unable to grasp the reason for unsolvability. Again, our method involves helping the human in this process by pointing out a necessary subgoal (i.e., any valid solution to that problem must achieve the subgoal) that can't be achieved at the current abstraction level. Thus the second point relates to the challenge of finding a sequence of subgoals (defined by state fluents present at the explanatory level) for a given problem. In the third step, we try to identify the first subgoal in the sequence that is actually unsolvable in the given level.

Given our approach, the final explanatory message provided to the user would include model information that brings their understanding of the task to the required level and information on the specific subgoals (and previous ones that need to be achieved first) that can no longer be achieved. In cases where the unachievable subgoals are hard to understand formulas or large disjunctions, we can also use these subgoals to produce exemplar plans in the more abstract models and illustrate their failures alongside the unachievable subgoals.

10.1.1 Identifying the Minimal Level of Abstraction Required for Explanation

Following Chapter 9, we will assume that the human's understanding of the task could be approximated by a model $\mathcal{M}_h^R = \langle F_H, A_H, I_H, G_H \rangle$, such that, the model is part of the abstraction lattice ($\mathcal{M}_R \sqsubset \mathcal{M}_h^R$ and $\mathcal{M}_h^R \in \mathbb{M}$). While the earlier work is able to use alternative plan provided by the user to identify the human model, we instead use the fact that the user expected the problem to be solvable to identify $\mathcal{M}_h^R$, i.e., $\exists \pi, \pi(I_H) \models_{\mathcal{M}_h^R} G_H$.

We now need to abstract this human model to a level where the problem is unsolvable (i.e the explanation level) by providing information about a certain subset of fluents previously missing from the human model (i.e information on their truth values in the initial and goal state, and how they affect various actions etc...). In the case of Example 2, this would include information on whether various paths are traversable and how the traversability of a path is a precondition for the robot to move across it. We will refer to the set of fluents that the human needs to be informed about as explanatory fluents ($\mathcal{E}$) and for Example 2, it will be $\mathcal{E} = \{can\_traverse(?x, ?y)\}$.

Definition 26. Given a human model $\mathcal{M}_h^R$, we define a set of propositions $\mathcal{E}$ to be **explanatory fluents** if $f_{\mathcal{E}}^{-1}(\mathcal{M}_h^R)$ is unsolvable, i.e, $|\Pi_{f_{\mathcal{E}}^{-1}(\mathcal{M}_h^R)}| = 0$.

Unfortunately, this is not an operational definition as we do not have access to $\mathcal{M}_h^R$. Instead, we know that $\mathcal{M}_h^R$ must be part of the lattice, and thus there exists a subset of the maximal elements of the lattice (denoted as $\mathbb{M}^{abs}$) that is more abstract than $\mathcal{M}_h^R$. In this section, we will show how the explanatory fluents for models in this subset of $\mathbb{M}^{abs}$ would satisfy $\mathcal{M}_h^R$ as well.

The first useful property to keep in mind is that if $\mathcal{M}_1$ is more concrete than $\mathcal{M}_2$ then the models obtained by concretizing each model with the same set of fluents would maintain this relation (although they may get concretized to the same model), i.e.,

Proposition 21. Given models $\mathcal{M}_1$, $\mathcal{M}_2$ and a set of fluents ϵ', if $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$, then $f_{\epsilon'}^{-1}(\mathcal{M}_1) \sqsubseteq f_{\epsilon'}^{-1}(\mathcal{M}_2)$.

Next, it can be shown that any given set of explanatory fluents for an abstract model will be a valid explanatory fluent set for a more concrete model

Proposition 22. Given models $\mathcal{M}_1$, $\mathcal{M}_2$, if $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$, then any explanation $\mathcal{E}$ for $\mathcal{M}_2$ must also be an explanation for $\mathcal{M}_1$.

To see why this proposition is true, let's start from the fact that $f_{\mathcal{E}}^{-1}(\mathcal{M}_1) \sqsubseteq f_{\mathcal{E}}^{-1}(\mathcal{M}_2)$ and therefore $\Pi_{f_{\mathcal{E}}^{-1}(\mathcal{M}_1)} \subseteq \Pi_{f_{\mathcal{E}}^{-1}(\mathcal{M}_2)}$. From the definition of explanation we know that the concretization with respect to explanatory fluents would render the problem unsolvable (i.e $|\Pi_{f_{\mathcal{E}}^{-1}(\mathcal{M}_2)}| = 0$) and thus $|\Pi_{f_{\mathcal{E}}^{-1}(\mathcal{M}_1))}|$ must also be empty and hence $\mathcal{E}$ is an explanation for $\mathcal{M}_1$.

Definition 27. Given an abstraction lattice $\mathbb{L}$, let $\mathbb{M}^{abs}$ be its maximal elements. Then the **minimum abstraction set** is defined as $\mathbb{M}_{min} = \{\mathcal{M} | \mathcal{M} \in \mathbb{M}^{abs} \wedge |\Pi_{\mathcal{M}}| > 0\}$.

Note that for any model $\mathcal{M}_1 \in \mathbb{M}_{min}$, $\mathcal{M}_h^R \sqsubseteq \mathcal{M}_1$, this means by Proposition 22, any explanation that is valid for models in $\mathbb{M}_{min}$, should lead $\mathcal{M}_h^R$ to a node where the problem is unsolvable. Now we can generate the explanation (even the optimal one) by searching for a set of fluents that when introduced to the models $\mathcal{M} \in \mathbb{M}_{min}$ will render the problem $f_{\mathcal{E}}^{-1}(\mathcal{M})$ unsolvable. In this work, we will mostly consider the use of uniform cost search to find the least costly set of explanatory fluent, where the cost of each fluent reflects the cost of communicating information about a particular fluent. In this case, the search state consists of sets of models (with $\mathbb{M}_{min}$ being the initial state), the actions consist of the various fluent concretizations, the edges of the lattice define the successor functions and the goal test involves verifying whether the problem is solvable in each possible model in the current state.

10.1.2 Generating Subgoals of a Given Problem

Note that it would be hard to identify non-trivial subgoals for the given problem in the node at which the problem was found to be unsolvable (i.e $f_{\mathcal{E}}^{-1}(\mathbb{M}_{min})$) since there are no valid plans in that model. Fortunately, we can use models more abstract than $f_{\mathcal{E}}^{-1}(\mathbb{M}_{min})$ to generate such subgoals. We will use planning landmarks (Hoffmann *et al.*, 2004) extracted from $\mathcal{M}$, where $|\Pi_{\mathcal{M}}| > 0$, as subgoals. Intuitively, state landmarks (denoted as $\Lambda = \langle \Phi, \prec \rangle$) for a model $\mathcal{M}$ can be thought of as a partially ordered set of formulas, where the formulas and the ordering need to be satisfied by every plan that is valid in $\mathcal{M}$. We will only be considering sound orderings (c.f (Richter *et al.*, 2008)) between landmarks, namely, (1) *natural orderings* ($\prec_{nat}$) - $\phi \prec_{nat} \phi'$, then ϕ must be true before ϕ' is made true in every plan, (2) *necessary orderings* ($\prec_{nec}$) - if $\phi \prec_{nec} \phi'$ then ϕ must be true in the step before ϕ' is made true every time and (3) *greedy necessary orderings* ($\prec_{gnec}$) - if $\phi \prec_{gnec} \phi'$ then ϕ must be true in the step before ϕ' is made true the first time. The landmark formulas may be disjunctive, conjunctive or atomic landmarks.

Our use of landmarks as the way to identify subgoals is further justified by the fact that logically complete abstractions conserve landmarks. Formally

Proposition 23. Given two models $\mathcal{M}_1$ and $\mathcal{M}_2$, such that $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$, let $\Lambda_1 = \langle \Phi_1, \prec_1 \rangle$ and $\Lambda_2 = \langle \Phi_2, \prec_2 \rangle$ be the landmarks of $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively. Then for all $\phi_i^1, \phi_j^1 \in \Phi_1$, such that $\phi_i^1 \preceq_1 \phi_j^1$ (where $\prec_1$ is some sound ordering), we have ϕ_i^2 and ϕ_j^2 in Φ_2, where $\phi_i^1 \preceq_2 \phi_j^1$, $\phi_i^1 \models \phi_i^2$ and $\phi_j^1 \models \phi_j^2$.

This is true because $\phi_i^2 \prec_1 \phi_j^2$ holds over all the plans that are valid in $\mathcal{M}_2$, and therefore must also hold over all plans in $\mathcal{M}_1$. Though in $\mathcal{M}_1$ these landmark instances may be captured by more constrained formulas, and additionally $\mathcal{M}_1$ may also contain landmarks that were absent from $\mathcal{M}_2$. Now if we can show that in a particular model, a landmark generated from a more abstract model is unachievable (or the ordering from the previous level is unachievable) then ϕ_*^1 becomes $\perp$ (thereby meeting the above requirement). Thereafter, for any model more concrete than $\mathcal{M}_2$, the formula corresponding to that landmark must be $\perp$. In other words, if for any model a landmark is unachievable, then that landmark can't be achieved in any models more concrete than the current one.

So given the explanatory level, we can move one level up in the lattice and make use of any of the well established landmark extraction methods developed for classical planning problem to generate a sequence of potential subgoals for $\mathcal{M}_R$.

10.1.3 Identifying Unachievable Subgoals

Now we need to find the first subgoal from the sequence that can no longer be achieved in the models obtained by applying the explanatory fluents ($f_{\mathcal{E}}^{-1}(\mathbb{M}_{min})$) which will then be presented to the user. For example, in the case of Figure 10.1, the unachievable subgoal would correspond to satisfying $at\_rover(5,4)$ (marked in red in M_4).

It is important to note that finding the first unachievable subgoal is not as simple as testing the achievability of each subgoal at the abstraction level identified by methods discussed in section 10.1.1. Instead, we need to make sure that each subgoal is achievable while preserving the order of all the previous subgoals. To test this we will introduce a new compilation that allows us to express the problem of testing achievement of a landmark formula as a planning problem. Consider a planning model $\mathcal{M}$ and the landmarks $\Lambda = \langle \Phi, \prec \rangle$ extracted from some model $\mathcal{M}'$, where $\mathcal{M} \sqsubset \mathcal{M}'$. We will assume that the formulas in Φ are propositional logic formulas over the state fluents and are expressed in DNF. Each $\phi \in \Phi$ can be represented as a set of sets of fluents (i.e, $\phi = \{c_1, ..., c_k\}$ and each c_i set takes the form $c_i = \{p_1, ..p_m\}$), where each set of fluents represent a conjunction over those fluents. For testing the achievability of any landmark $\phi \in \Phi$, we make an augmented model $\mathcal{M}_\phi = \langle F^\phi, A^\phi, I^\phi, G^\phi \rangle$, such that the landmark is achievable *iff* $|\Pi_{\mathcal{M}_\phi}| > 0$. The model $\mathcal{M}_\phi$ can be defined as follows: $F^\phi = F \cup F^{meta}$, where F^{meta} contains new meta fluents for each possible landmark $\phi' \in \Phi$ of the form

- $achieved(\phi')$ keeps track of a landmark being achieved and never gets removed
- $unset(\phi')$ Says that the landmark has not been achieved yet, usually set true in the initial state unless the landmark is true in the initial state
- $first\_time\_achieved(\phi')$ Says that the landmark has been achieved for the first time. This fluent is set true in the initial state if the landmark is already true there

The new action set A^ϕ, will contain a copy of each action in A. For each new action corresponding to $a \in A$, we add the following new effects to track the achievement of each landmark

- for each $\phi' \in \Phi$ if the action has existing add effects for a subset of predicates $\hat{c}_j$ for a $c_j \in \phi'$, then we add the conditional effects $cond_1(\phi') \rightarrow \{achieved(\phi')\}$ and $cond_2(\phi') \rightarrow \{first\_time\_achieved(\phi')\}$, where
  $cond_1(\phi') = c_j \setminus \hat{c}_j \cup \{\hat{\phi} | \hat{\phi} \in \Phi \wedge (\hat{\phi} \prec_{nec} \phi')\} \cup \{achieved(\hat{\phi}) | \hat{\phi} \prec_{nat} \phi'\}$ and
  $cond_2(\phi') = cond_1(\phi') \cup \{\hat{\phi} | \hat{\phi} \prec_{gnec} \phi'\} \cup \{unset(\phi)\}$

- We add a conditional delete effect to every action of the form

$$first\_time\_achieved(\phi') \rightarrow (not(first\_time\_achieved(\phi')))$$

The new goal would be defined as $G^{\phi} = \{first\_time\_achieved(\phi)\}$.

This formulation allows us to test each landmark in the given sequence and find the first one that can no longer be achieved. To ensure completeness, we will return the final goal if all the previously extracted landmarks are still achievable in $f_{\mathcal{E}}^{-1}(\mathbb{M}_{min})$. Now given an ordered set of landmarks, we can identify the first unsolvable landmark by testing the solvability of the F^{ϕ} for each landmark in the order they appear in the sequence.

Since the above formulation is designed for DNF, we can generate compilation for cases where the landmarks use either un-normalized formulas or CNF by converting them first into DNF formulas.

10.2 Planning Problems with Plan Advice

Let us now discuss how we could extend the methods presented in earlier sections to cases where the user provides plan advice. In such cases,the user imposes certain restrictions on the kind of solution they expect, either as an alternative to the solution the system may come up with on its own or as a guide to help the system come up with solutions when it claims unsolvability.

As pointed out in (Myers, 1996), such advice can be compiled into plan constraints in the original problem. A number of approaches have been proposed to capture and represent plan constraints (Bacchus and Kabanza, 2000; Nau *et al.*, 2001; Kambhampati *et al.*, 1995; Baier and McIlraith, 2006), and each of these representational choices has its unique strengths and weaknesses. In general, we can see that these plan constraints as specify a partitioning of the space of all valid plans to either acceptable (i.e it satisfies the constraints) or unacceptable. So we can define, constraints as follows

Definition 28. The partition specified by a **constraint** σ on a given set of plans that is specified by a membership function $\sigma : \Pi \rightarrow [0, 1]$, where Π is the set of all plans.

We will slightly abuse notation and for a given set of plans $\hat{\Pi}$ we will use $\sigma(\Pi')$ to denote $\{\pi | \pi \in \hat{\Pi} \wedge \sigma(\pi) = 1\}$ (i.e the subset of Π' that satisfies the constraint). If we can assume some upper bound on the possible length of plans in $\sigma(\Pi_{\mathcal{M}})$ (which is guaranteed when we restrict our attention to non-redundant plans for standard classical planning problems), then we can assert that there always exists a finite state machine that captures the space of acceptable plans

Proposition 24. Given a constraint σ and a model $\mathcal{M}$, there exists a finite state automaton $\mathcal{F}^{\sigma,\mathcal{M}} = \langle \Sigma, \mathbb{S}_{\mathcal{F}^{\sigma,\mathcal{M}}}, S_0, \delta, E \rangle$, where Σ is the input alphabet, $\mathbb{S}_{\mathcal{F}^{\sigma,\mathcal{M}}}$ defines the FSA states, S_0 is the initial state, δ is the transition function and E is the set of accepting states, such that $\sigma(\Pi_{\mathcal{M}}) = \mathcal{L}(\mathcal{F}^{\sigma,\mathcal{M}}) \cap \ \Pi_{\mathcal{M}}$, where $\mathcal{L}(\mathcal{F}^{\sigma,\mathcal{M}})$ is the set of strings accepted by $\mathcal{F}^{\sigma,\mathcal{M}}$.

The existence of $\mathcal{F}^{\sigma,\mathcal{M}}$ can be trivially shown by considering an FSA that has a path for each unique plan in $\mathcal{F}^{\sigma,\mathcal{M}}$. We believe that this formulation is general enough to capture almost all of the plan constraint specifications discussed in the planning literature, including LTL based specifications, since for classical planning problems these formulas are better understood in terms of LTL_f (De Giacomo and Vardi, 2015) which can be compiled into a finite state automaton.

We can use $\mathcal{F}^{\sigma,\mathcal{M}}$ to build a new model $\sigma(\mathcal{M})$ such that a plan is valid in $\sigma(\mathcal{M})$ if and only if the plan is valid for $\mathcal{M}$ and satisfies the given specification σ, i.e., $\forall\pi, \pi \in \Pi_{\sigma(\mathcal{M})}$ *iff* $\pi \in \sigma(\Pi_{\mathcal{M}})$

For $\mathcal{M} = \langle F, A, I, G \rangle$, we can define the new model $\sigma(\mathcal{M}) = \langle F_\sigma, A_\sigma, I_\sigma, G_\sigma \rangle$ as follows

- $F_\sigma = F \cup \{\text{in-state-}\{S\} | S \in \mathbb{S}_{\mathcal{F}^{\sigma,\mathcal{M}}}\}$
- $A_\sigma = A \cup A_\delta$
- $I_\sigma = I \cup \{\text{in-state-}\{S_0\}\}$
- $G_\sigma = G \cup \{\text{in-state-}\{S\} | S \in E\}$

A_δ are the new meta actions responsible for simulating the transitions defined by $\delta : \mathbb{S}_{\mathcal{F}^{\sigma,\mathcal{M}}} \times \Sigma \rightarrow pow(\mathbb{S}_{\mathcal{F}^{\sigma,\mathcal{M}}})$. For example, if $\delta(S_1, a) = \{S_1, S_2\}$, where a corresponds to an action in A, then we will have two new actions $a^1_{S_1,a} = \langle prec^a \cup \{\text{in-state-}\{S_1\}\}, adds^a \cup \{\text{in-state-}\{S_2\}\}, dels^a \cup \{\text{in-state-}\{S_1\}\}\rangle$ and $a^2_{S_1,a} = \langle prec^a \cup \{\text{in-state-}\{S_1\}\}, adds^a, dels^a\}\rangle$. In cases like LTL, the FSA state transitions may be induced by the satisfaction of some formula, so the new meta action may have preconditions corresponding to that formula, with no other effects but changing the fluent corresponding to the state transition.

The above formulation merely points out that there always exists a way of generating $\sigma(\mathcal{M})$ from the given specification σ and $\mathcal{M}$. For many constraint types, there may exist more efficient ways of generating models that satisfy the requirements of $\sigma(\mathcal{M})$.

Once we have access to $\sigma(\mathcal{M})$, we should be able to use the methods discussed in earlier sections to explain unsolvability of $\sigma(\mathcal{M})$ and hence why the constraint isn't

Domain Name	$\|P\|$	Average Runtime (secs)	Explanation Cost	Cost of explaining M_R
Blocksworld	4	8.141	11.6	30.2
Satellite	8	19.15	6	43.6
Depots	5	20.229	13	51
Rover	8	263.635	7.5	15.75
Storage	7	50.348	20	55.8
Over-Rover*	8	2047.360	29.8	92.6
Over-tpp*	8	1065.542	842.8	881.2
Bottleneck*	3	504.431	60.8	66.2

Table 10.1: Table Showing Runtime for Explanations Generated for Standard IPC Domains. The Explanation Costs Capture the Number of Unique Model Updates (Changes in Effects/Precondition Etc..) Corresponding to Each Explanation

feasible. To facilitate such explanation, we will build an abstraction lattice for the constrained problems $\mathbb{L}_{\sigma\mathcal{M},P}$ such that $P \cap (F^\sigma \setminus F) = \phi$, i.e the abstraction lattice only affects the fluents from the original problem and not the new ones introduced as part of the compilation. In fact, we can induce such a lattice by considering the lattice generated for the original problem and then replacing each node in the lattice with the corresponding compiled problem, to see why this would induce a valid abstraction lattice, consider the following property

Proposition 25. Given models $\mathcal{M}_1$, $\mathcal{M}_2$ and a constraint specification σ, if $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$, then $\sigma(\mathcal{M}_1) \sqsubseteq \sigma(\mathcal{M}_2)$.

To see why this is true, just assume that the reverse was true, that $\sigma(\mathcal{M}_2)$ is not a logically complete abstraction of $\sigma(\mathcal{M}_1)$. This means that there are plans in $\Pi_{\sigma(\mathcal{M}_1)}$ that are not part of $\Pi_{\sigma(\mathcal{M}_2)}$. From the definition of $\sigma(\mathcal{M}_2)$, we know that $\Pi_{\sigma(\mathcal{M}_2)} = \Pi_{\mathcal{M}_2} \cap \mathcal{L}(\mathcal{F}^\sigma)$. If there exist a $\pi \in \Pi_{\sigma(\mathcal{M}_1)}$, such that $\pi \notin \Pi_{\sigma(\mathcal{M}_2)}$, then $\pi \notin \Pi_{\mathcal{M}_2}$. Which means $\mathcal{M}_1 \not\sqsubseteq \mathcal{M}_2$, hence contradicting our assumptions.

Revisiting Example 2, the question asked by the user could be seen as an advice,

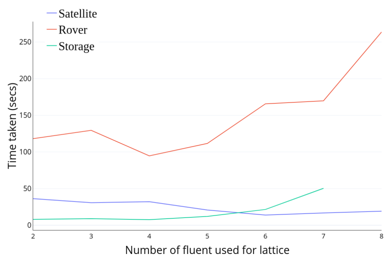


Figure 10.3: The Graph Compares the Time Taken to Generate the Explanation for Three of the Domains for Increasing Size of Lattices.

where the corresponding constraint covers all plans where the rover performs the actions collect_rock_sample P4, collect_soil_sample P7, irrespective of the exact position and order in which the actions appears in the plan. More generally, we could think of this plan advice as being a special case of advice where the user wants to ensure presence of certain actions in the plan with some partial ordering among them (eg: "Why don't you pickup block B and then C?", "Make sure that you have cleared Room1 before you move on to Room2 and Room3" etc..). Such advice could be represented as partial plans (Kambhampati *et al.*, 1995), which in general can be captured as a partially ordered multiset of the form[1] $\hat{\pi} = \langle \hat{A}, \leqslant \rangle$, where $\hat{A}$ is a multiset over grounded actions and $\leqslant$ defines ordering constraints over these grounded actions. A plan $\pi = \langle a_1, ..., a_n \rangle$ is said to be a candidate plan for the given partial plan $\hat{\pi}$, if there exists a mapping function $\mu : \hat{A} \rightarrow [1, |\pi|]$ that maps each action in $a \in \hat{A}$ to a position in the plan such that $a = a_{\mu(a)}$ and if $a < b$ for $a, b \in \hat{A}$, then $\mu(a) < \mu(b)$. Such partial plans can be fairly easily compiled into a classical planning model (such that it satisfies $\sigma(\mathcal{M})$) by extending the compilation methods discussed by Ramırez and Geffner (2010), without relying on an intermediate finite state machine.

[1]We are presenting a simplified definition of a partial plan. The full definition allows for the representation of more complex constraints than mere ordering constraints, such as contiguity constraints and interval protection constraints.

The corresponding partial plan for the question specified above would be
$\hat{\pi} = \langle \{\text{collect\_rock\_sample P4}, \text{collect\_soil\_sample P7}\}, \rangle$
Let us assume that in this case the observer could be unaware of certain domain constraints such as the rover's inability to traverse certain regions on the map the fact that not all rovers are capable of collecting rock and soil samples or that they are not always equipped to store these samples. In this case, possible user models can be captured by a lattice build using the following fluents $P = \{$(can_traverse ?x ?y), (equipped_for_soil_sample ?r), (equipped_for_rock_sample ?r), (store_of ?r)$\}$. Now our approach would identify the user need to be made aware of the fact that not all regions of the map are equally traversable (i.e inform the user about can_traverse ?x ?y) and how its a precondition for move action), furthermore given this property the robot can no longer reach the waypoint 4 which is required to complete this task (i.e the unreachable landmark is (at rover0 waypoint4)).

10.3 Evaluations

10.3.1 User Studies

Our first concern with evaluating explanations based on landmarks was to establish that they constitute meaningful explanations for naive users. As a simple alternative to our explanations, we consider providing to the user a set of potential solutions (generated from a higher level of abstraction) and their individual failures. For the study, we recruited around 120 master turkers from Amazon's Mechanical Turk and tested the following hypotheses

- **H1** - Users prefer concise explanations over ones that enumerate a set of possible candidates for a given piece of plan advice
- **H2** - Users prefer concise explanations that contain information about unachiev-

able landmarks over ones that only show the failure of a single exemplary plan

For the hypotheses, we presented the study participants with a sample dialogue between two people over a logistics plan to move a package from one location to another. The dialogue included a person (named Bob) presenting a plan to another (named Alice), and Alice asks for an alternative possibility (i.e specifies a constraint on the solution). Now the challenge for Bob is to explain why the constrained problem is unsolvable. For example, in one example Bob presents a rather convoluted plan that involves the package being transferred through multiple trucks to a train and then to the final destination. This leads to Alice asking the package to be delivered via an airplane.

For H1, in addition to some model information that Bob was unaware of, the potential explanations included either (a) the information on the unachievable landmark, (b) landmark information with the failure details of a specific exemplary plan or (c) a set of plans that satisfy the constraints and their corresponding failures. For the earlier example this meant Bob explains to Alice the limited availability of Truck fuel and (a) the impossibility of getting the package to the airport or (b) the the impossibility of getting the package to the airport and a specific plan (eg: *truck1 picks up package moves to location two then to three ...*) along with its point of failure (eg: *truck1 runs out of fuel when it reaches location three*) or (c) three example plans involving various trucks trying to get the package to the airport and their specific points of failures (each of which fails at different steps but before reaching the airport).

For this study, we used 45 participants and each participant was assigned one of three possible maps for each hypothesis and was paid \$1.25 for 10 mins. We used a control question to filter participant responses, so as to ensure their quality. Out of the 39 remaining responses, we found 94.8% of users chose to select the more concise explanation (i.e (a) or (b)), and 51.28% of the users chose explanations that involved just landmarks.

For H2, we used 75 participants and presented each participant with explanations that include (a) just landmark information, (b) landmark information with failure details of an exemplary plan and (c) just the exemplary plan failure. Here participants were paid \$1 for 10 mins for H2 as the explanatory options were much simpler. After filtering using the control question, we found that out of 60 valid entries 75.4% of participants preferred explanations that included landmark information ((a) or (b)) and 44.2% wanted both landmarks and exemplary plan (i.e (b)). The supplementary file at `http://bit.ly/2HQ5sTv` contains more details on the study setup.

10.3.2 Empirical Studies

In this section, we will present the results of an empirical evaluation of the computational characteristics of our approach. One big concern with the methods discussed in this work is the fact that they involve solving multiple planning problems. Thus we were interested in identifying the runtime for generating explanations on a set of standard planning benchmarks.

To evaluate our methods, we considered eight planning domains and chose five problem instances for each of the domains. For each domain, we used a subset of the domain predicates to generate the abstraction lattice (i.e we set the subset as the set of fluents P used to define the lattice). The first five domains and their problem instances consisted of standard IPC domains and problem instances used in previous IPC competitions (International Planning Competition, 2011). Each problem instance was made unsolvable by including plan constraints that avoid a specific landmark of the original problem. The constraints were coded using domain control programs (Baier *et al.*, 2007) of the form

```
while ¬$\phi$ ∧ ¬(goal_completed)
```

```
do    any
done
```

Where ϕ is the landmark formula and `(goal_completed)` is the goal fluent (generated by a new `goal_action` whose preconditions are the original goals of the problem). The constraints ensure that any valid plan must avoid the landmark ϕ and thereby rendering it unsolvable. The next three domains were selected from the set used for the 2016 unsolvability competition (Unsolvability International Planning Competition, 2016) (these domains are marked with an asterix in the results table). All instances were run with a timeout of 100 minutes (all problems were solvable under this time limit) and all landmarks were generated using the fast-downward implementation of the method discussed by Keyder *et al.* (2010) (where we set the subset sizes to one for the first five domains and to two for the rest).

Table 10.1 presents the results of our tests on these domains. It shows the number of fluents used to generate the lattice ($|P|$), the average runtime, the cost of the generated explanations and the cost of presenting the most concrete model to the user. For each scenario, we created a complete lattice for all the fluents considered for abstraction (i.e $|\mathbb{M}| = 2^{|P|}$). The cost of the explanation captures the amount of information to be provided to the user as part of the explanation. This could include information regarding the various explanatory fluents and is here captured roughly by the number of places within the domain definition where these fluents appear. The cost also reflects the inferential overhead demanded from the user (since providing more information translates to the user needing to understand the domain at a much more concrete level).

For a sample explanation, consider the overconstrained rover domain, where the rovers' actions are limited by their energy levels and the energy of the rover isn't enough to finish the task. In one of the instances where the rover energy level is at 33 and the original problem had a goal consisting of eight propositions (each referring to the need for communicating a particular soil sample, rock sample or sending an image for different objectives), our approach was able to identify that the user needs to understand fluents related to energy ((energy ?x ?y) and (energycost ?x ?y ?z)) and identified two subgoals out of the eight that it could not achieve.

Figure 10.3 presents the variations in average runtime for three of the domains as the size of the lattice were increased (the X-axis represents the number of fluents that were used to build the lattice and Y the runtime in seconds). Note that, in general, the runtime increases as the lattice size increase due to the increase in the search space, but in all three domains there are points where the runtime decrease when the lattice size increases. This is expected since with an increase in the size of lattice, the planning problems whose unsolvability are being tested becomes simpler.

10.4 Related Work

As discussed earlier, our methods for identifying the level of explanations are based on the expertise level modeling approaches introduced in Chapter 9. These two works are quite closely connected and in fact, the contrastive explanations of the type studied in the earlier chapter, where the user presents alternative plans (i.e the foils for the explanations) that are then refuted by the system, is a special case of our approach for handling problems with plan advice. The problems studied in that earlier chapter can be thought of as capturing cases where the advice only allows for a single plan. Also, one could argue that people would be more comfortable giving advices as foils rather than full plans. Part of our explanations also try to reveal to the user information about the current task that was previously unknown to them. Thus our methods could also be understood as an example of explanation as model-reconciliation (Chakraborti *et al.*, 2017). Since our methods use abstractions, our approach doesn't make too many demands on the inferential capabilities of the user and hence can be applied to much larger and more complex domains.

Another closely related direction has been the work done on explaining unsynthesizability of hybrid controllers for a given set of high-level task specifications (Raman and Kress-Gazit, 2013). The work tries to identify the subformulas of the given specification that lead to the unsynthesizability. This particular approach is specific to the planning framework detailed by Finucane *et al.* (2010) and the objective of the work parallels the goals of work like those presented by Göbelbecker *et al.* (2010).

Outside of explanation generation, the work done in the model checking community is closely related to our current problem (Grumberg and Veith, 2008). In fact, the hierarchical approach to identifying a model that can invalidate the given foil specification, can be seen as a special case of the CEGAR based methods studied in the model-checking community (Clarke *et al.*, 2000). Most work in this field focuses on developing methods for identifying whether a given program meets some specifications and failures to meet specification are generally communicated via counterexamples.

Another related problem is that of identifying whether a given problem is unsolvable. In our setting, we assume that the system is capable of correctly identifying whether a given problem is unsolvable or not and in general this can be a time consuming process. Thankfully the problem of efficiently identifying whether a given planning problem is unsolvable is an active research area (cf. (Steinmetz and Hoffmann, 2017; Kolobov *et al.*, 2010)) and solutions to this problem can be easily leveraged by our approach to improve the overall efficiency of the system.

10.5 Concluding Remarks

In this chapter, we saw how the basic HELM framework laid out in Chapter 9 to support explaining the unsolvability of a planning problem and by extension support cases where the user may only provide a partial specification of the foil, i.e, the foil may be best represented as a loose constraint over the planning problem. Another important addition the chapter makes is the use of subgoals as a way to further simplify the model information being used in the explanation. The user studies presented in this chapter also show that the study participants identified this to be useful information. In the next chapter, we will take this version of the HELM framework as the starting point and see how it could be applied to more general planning formulations.

Chapter 11

HANDLING NON-DETERMINISM AND D3WA+ DEBUGGING SYSTEM

Through the previous chapters, we saw the introduction of an explanation generation framework called HELM, which uses state abstraction to provide explanations to the end-user. Furthermore, the previous chapter saw the extension of this framework to support scenarios where the foil set may only be implicitly or partially specified. We also saw how the framework can also be used to support explanations of problem unsolvability. However, one constant through both these chapters was our focus on deterministic planning problems. In this chapter, we will take our first step to demonstrate how this approach is applicable beyond that specific problem class and apply it in the context of non-deterministic problems. In particular, we will consider fully observable non-deterministic problems (i.e. FOND), where each action could have a set of mutually exclusive effects, and the execution of an action could result in one of those effects being applied non-deterministically.

Once we introduce the basic extension of our explanation, we will demonstrate the utility of this extension by applying it in the context of a tool for debugging domain model specifications. In particular, we will consider the tool `D3WA` (Muise *et al.*, 2019a) that was developed to allow dialogue designers for automated conversational agents to specify the agent behavior declaratively. Specifically, they encode the potential conversations the agent can have in the form of a FOND model. The debugging tool (called `D3WA+`) was introduced to allow users to debug the model specification when the agent behavior doesn't match what the domain designer had in mind. So another contribution of this chapter would be to formalize the XAIP problem for the model acquisition scenario and illustrate the salient challenges involved on a tool for the design of goal-directed conversational agents.

11.1 Non-deterministic Models and Explanations

In this section, we will discuss how we extend the abstraction-based explanation, in particular the one designed to support partial foils introduced Chapter 10, to support non-deterministic models.

11.1.1 Background

We will be relying on the model definition laid out in Section 2.2, where a model is captured by a tuple of the form $\mathcal{M} = \langle F, A, I, G \rangle$. For the purposes of the discussion, we will focus on model where preconditions are limited to positive preconditions. This means an action a in the model is defined as: $a = \langle pre_{+}(a), \mathbb{E}(a) \rangle$ where $pre_{+}(a) \subseteq F$ is the set of preconditions that needs to be true for the action a to be executable and $\mathbb{E}(a) = \{o^a_1, \ldots, o^a_k\}$ is the set of k possible outcomes for action a. Each outcome is further defined as the tuple $o_i(a) = \langle add_i(a), del_i(a) \rangle$, where $add_i(a)$ and $del_i(a)$ are the adds and deletes corresponding to the outcome. Through this discussion, we will focus primarily on weak solutions. We will represent the space of goal achieving behaviors by the notations $\Pi(\mathcal{M})$. In the case of non-deterministic models $\Pi(\mathcal{M})$, would correspond to all possible traces from initial state to a goal state and not necessarily weak policies. However, it is worth noting that when we apply the notation in the context of deterministic planning problems (as we will do later), this will correspond to all the possible plans.

Also to support partial foils, we will use the model transformation function $Obs : \mathcal{M} \times \{\pi\} \mapsto \mathcal{M}'$ that receives a model and a (partial) sequence of actions (equivalent to a set of possible plans) and produces a compiled model where this sequence must be preserved, i.e. $\{\pi\} \subseteq \Pi(\mathcal{M}')$. In the preservation technique used here for deterministic models, we allow for partial observation of non-determinism as well, in addition to the usual partial plans, in order to account for the specific needs of the tool under study. This means that an action may be specified in a plan but its outcome may be left unspecified.

11.1.2 Extending HELM to Support Non-Deterministic Models

To map the previous explanation framework to this setting, the first step would be to define an abstraction function. In a parallel to Theorem 4, we will show that *syntactic projection of FOND description will also result in logically complete abstractions*. In other words, the model derived from the transformation will result in model that will allow for more behaviors.

Definition 29. *A model abstraction $\mathcal{M}' = Abs(\mathcal{M}, \mathcal{P})$ is logically complete if there exists a surjective mapping from states in $\mathcal{M}$ to $\mathcal{M}'$ and $\Pi(\mathcal{M}) \subseteq \Pi(\mathcal{M}')$.*

Now let us consider the syntactic projection function $Abs_p : \mathcal{M} \times 2^F \mapsto \mathcal{M}'$.

Definition 30. *Given a non-deterministc model $\mathcal{M}$ and a set of fluents $\mathcal{P}$: $Abs_p(\mathcal{M}, \mathcal{P}) = \langle F \setminus \mathcal{P}, \hat{A}, \mathcal{I} \setminus \mathcal{P}, \mathcal{G} \setminus \mathcal{P}\rangle$ where $\forall a \in A, \exists \hat{a} \in \hat{A}$ such that: $pre_+(\hat{a}) = pre_+(\hat{a}) \setminus \mathcal{P}$ and every outcome becomes $o_i(\hat{a}) = \langle add_i(a) \setminus \mathcal{P}, del_i(a) \setminus \mathcal{P}\rangle$.*

Consider a transition $\langle S_i, a_i, S_{i+1}\rangle$ that is valid for t$\mathcal{M}$, we can see that, $S_i \subseteq pre_+(a_i)$ and there must exist an outcome $o_k^{a_i}$ for a_i such that $S_{i+1} = (S_i \setminus del_k(\hat{a}_i) \cup add_k(\hat{a}_i))$. Then there must exist a corresponding transition $\langle S_i \setminus \mathcal{P}, \hat{a}_i, S_{i+1} \setminus \mathcal{P}\rangle$ that is valid for $Abs_p(\mathcal{M}, \mathcal{P})$ with an outcome $o_k^{\hat{a}_i}$. Thus the model is guaranteed to be logically complete abstraction.

With these tools in place, we can generate explanations described in earlier sections by doing a search over the space of abstract non-deterministic models and find the most abstract model that is still unsolvable. Though it would be easier if we can just focus on settings where we are testing solvability of classical planning model and extracting subgoals from these simpler models. We will do this by focusing on determinizations of the original non-deterministic model.

Determinization is the process of turning a non-deterministic model $\mathcal{M}$ into a deterministic one $Det(\mathcal{M})$ – e.g. an "all-outcomes" determinization scheme (Yoon *et al.*, 2007) transforms an action $a : S \rightarrow \{S_i\}$ into a set of actions $\forall i \; a_i : S \rightarrow S_i$ so that all the outcomes of an action with non-deterministic effects can be realized in the determinised model. With the fairness assumption (Cimatti *et al.*, 2003), a solution to $Det(\mathcal{M})$ is also a valid behavior for $\mathcal{M}$, i.e. $\Pi(Det(\mathcal{M})) = \Pi(\mathcal{M})$.

All of our explanation generation procedures will be performed on the all outcome determinization of the original non-deterministic model. We will motivate this choice, by showing that the abstraction of a determinised model is the determinization of the abstracted non-deterministic model:

Proposition 26. *Given a non-deterministic model $\mathcal{M}$ and a set of fluents $\mathcal{P}$:*

$$Det(Abs_p(\mathcal{M}, \mathcal{P})) = Abs_p(Det(\mathcal{M}), \mathcal{P})$$

.

The determinised model will contain an action for each possible outcome, i.e. $\forall o^a_i, Det(\mathcal{M})$ contains an action $a_{o_i} = \langle \text{prec}^a, adds_{o^a_i}, dels_{o^a_i} \rangle$. So the projection of this determinised action will be $\langle \text{prec}^a \setminus \mathcal{P}, adds_{o^a_i} \setminus \mathcal{P}, dels_{o^a_i} \setminus \mathcal{P} \rangle$, and you would get the same action if you were determinising based on projected o^a_i.

Subgoals and Landmarks In addition to presenting the abstractions, we will be following the conventions set by Chapter 10 by also presenting the user with a subgoal that is necessary for achieving the goal in the current model but can not be achieved. We will again use landmarks (Hoffmann *et al.*, 2004) for identifying such subgoals.

The earlier chapter constructs landmarks from models that are more abstract than the minimally unsolvable model but this can lead to less informed subgoals as it may not be considering many of the state factors. We will instead extract landmarks directly from the the minimally unsolvable model using the following proposition:

Proposition 27. *If $\mathcal{M}$ is unsolvable, and $Abs(\mathcal{M}, \mathcal{P})$ is solvable while $Abs(\mathcal{M}, \mathcal{P} \cup \{f\})$ is not, then either the delete relaxation of $Abs(\mathcal{M}, \mathcal{P} \cup \{f\})$ is solvable or there must be an unmet landmark corresponding to f.*

This follows from the fact that in the abstraction scheme we follow, when we consider the delete relaxation of $Abs(\mathcal{M}, \mathcal{P} \cup \{f\})$ over that of $Abs(\mathcal{M}, \mathcal{P})$, the only possible differences can be that f may be part of preconditions and adds for an action. If this addition makes the delete relaxation unsolvable then that means that all relaxed plans for $Abs(\mathcal{M}, \mathcal{P})$ contained actions that required f as a precondition. The problem becomes a bit more involved when we allow for negative precondition as we need to test whether the possible landmark is f or $\neg f$.

Given this proposition, we first test if delete relaxation of $Abs(\mathcal{M}, \mathcal{P})$ is solvable so we can extract landmarks from it similar to Chapter 10. Otherwise we test if addition of f in initial state makes the delete relaxation of $Abs(\mathcal{M}, \mathcal{P} \cup \{f\})$ solvable. If it does, then the landmark is f, else it is $\neg f$. While this can lead to more informed landmarks, for cases where the delete relaxation is unsolvable, we would not be able to leverage the ordering of the landmarks to find subgoals appearing earlier in the sequence.

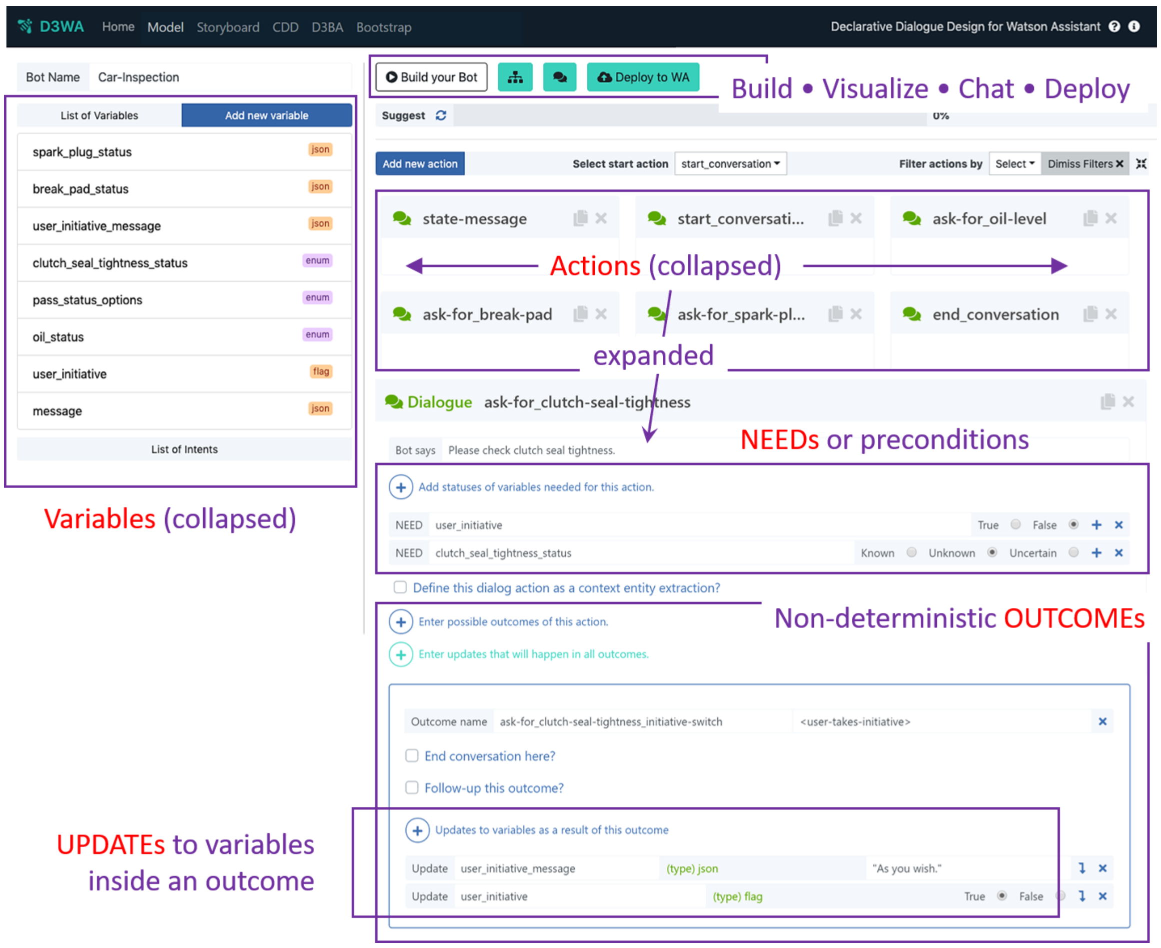
11.2 D3WA+ System and XAIP for Model Acquisition

With the basic framework for explanation generation in place, we will look at the application of this method within a tool for debugging domain models. Before delving deeper into the tool itself, we will take a quick look at the original domain specification tool `D3WA` which is extended by the debugging tool.

11.2.1 A Brief History of `D3WA`

At the core of the declarative specification proposed by Muise *et al.* (2019a) is an agent-centric view of the world – the dialogue designer specifies "capabilities" that are available to an agent and lets a non-deterministic planner generate (Muise *et al.*, 2012) and execute (Muise *et al.*, 2019b) the composed dialogue plan in the background. We demonstrated in ICAPS 2019 (Chakraborti *et al.*, 2019d) how this can lead to an exponential scale-up from the size of the specification to the complexity of the composed agent, as an illustration of the exciting fusion of planning based technologies (especially non-deterministic planning) and the design of dialogue agents. This is especially useful in the design of certain kinds of conversations, especially ones with an underlying process – e.g. a business process (Chakraborti and Khazaeni, 2020) – that drives the conversation. However, it turns out that while this provides a powerful tool for an experienced domain writer with expertise in planning, and declarative programming in general, for the uninitiated it presents too steep a learning curve. Since designers no longer explicitly compose the dialogue plan, they lose control over the composed agent if they do not grasp the *imperative consequences of their declarative specification.*

In this chapter, we thus build on this work *with the aim of making the core domain authoring engine more amenable to dialogue designers who are usually outside the planning community and do not readily subscribe to the declarative mental model.* In order to do so, we apply the explanation framework laid out in Section 11.1 to bridge the gap with the end user. Before we get to the specific contributions of the debugging tool, we start with a brief introduction to `D3WA` so as to make the rest of



Bot Name Car-Inspection

spark_plug_status	json
break_pad_status	json
user_initiative_message	json
clutch_seal_tightness_status	enum
pass_status_options	enum
oil_status	enum
user_initiative	flag
message	json

Select start action: start_conversation

state-message · start_conversati... · ask-for_oil-level · ask-for_break-pad · ask-for_spark-pl... · end_conversation

Dialogue ask-for_clutch-seal-tightness

Bot says Please check clutch seal tightness.

Add statuses of variables needed for this action.

NEED user_initiative True False [✓]

NEED clutch_seal_tightness_status Known Unknown [✓] Uncertain

Define this dialog action as a context entity extraction?

Enter possible outcomes of this action.

Enter updates that will happen in all outcomes.

Outcome name ask-for_clutch-seal-tightness_initiative-switch <user-takes-initiative>

End conversation here?

Follow-up this outcome?

Updates to variables as a result of this outcome

Update user_initiative_message (type) json "As you wish."

Update user_initiative (type) flag True [✓] False

Figure 11.1: Illustration of the Salient Aspects of Declarative Design of Conversational Agents on the D3WA Interface, Reproduced with Permission from Muise *et al.* (2019a).

the presentation self-contained to the extent possible.

Actions, Outcomes and Context Variables

The original system D3WA is illustrated in Figure 11.1. The interface surfaces two key elements to the dialogue designer: 1) **context variables** that model the agent's world; and 2) **actions** that are defined in terms of these variables. For a dialogue agent, these actions model the different capabilities available to it in terms of dialogue actions towards the end user, or internal system actions such as API calls or logical inferences.

Each action (Figure 11.1) has a set of NEEDs (or preconditions) and a set of OUTCOMEs which house a set of non-deterministic UPDATEs to a variable. The outcomes within an action are mutually exclusive and model how the user may respond or, in general, how the world may evolve in response to any action done by the agent. For example:

1. *Dialogue Action:* If the agent wants to ask the user their name, the corresponding dialogue action would have one outcome when the user responds with their name, and another one that models a digression in the conversation.

2. *System Action:* In order to make an API call, the agent would require as NEEDs, access to the link and the relevant payload. Two possible outcomes of the call may be a successful response (in which case the agent updates the values of the relevant variables it was looking for) or a 404 error (in which case the agent gets nothing).

A non-deterministic planner receives this specification, plans for all possible outcomes, and generates the resulting dialogue plans in Figure 11.1. This offline approach has two advantages: it allows the dialogue designer to inspect and sign off on the agent to be deployed, while also being able to support complex dialogues without having to plan and replan in real time. For more details on `D3WA`, and on how this specification is compiled to a planning problem in the background, we refer the reader to the discussion by Muise *et al.* (2019a).

11.2.2 `D3WA` + `XAIP` → `D3WA+`

The "explainable" version of `D3WA` – henceforth referred to as `D3WA+` – developed in this chapter provides a suite of debugging tools on top of the existing core model acquisition framework described above. This is aimed to make the dialogue designer more self-sufficient when they are faced with modeling errors. Specifically, based on difficulties observed during preliminary internal testing, we tackle two core issues faced frequently by dialogue designers grappling with the declarative paradigm:

- **Specification cannot be solved by the planner.** This is the case when the graph in Figure 11.3d does not appear at all, and the dialogue designer is left with an inscrutable "no solution found" message and nothing else to work with. Our goal here is to surface features from the current specification back to the designer so that they can fix identify the root cause of the unsolvability.

- **Solution does not match expectations.** Here, the problem is solvable but the solution does not match the designer's expectations – i.e. the graph in Figure 11.3d looks nothing like what they were aiming for. The goal here for us is to be able to respond to questions from the designer such as *Why is this a solution?* and *Why is this not a solution?*, so that they can modify the specification accordingly until they are satisfied with the outcome.

Some of these questions might look familiar with the line of investigation by Smith (2012); Fox *et al.* (2017); Cashmore *et al.* (2019) and the methods covered in the previous chapters. However, the setting here involves the domain designer and not the end user. Thus the suite of challenges not only include the unsolvability question, not addressed in those works, but also the explanatory dialogue here is geared towards the model acquisition task rather than the exploration of the decision making process.

11.2.3 The `XMAS` *Problem*

Explanations in the context of the model acquisition task provide a unique twist to the *model reconciliation* framework laid out in Chapter 4. There, the task is to compute, given the agent model, the mental model, and a plan to explain, a set of updates that when applied to the mental model would render the given plan optimal (and hence without any foil) in the updated mental model.

Here also, we have two models – the one currently specified by the domain writer or *the revealed model*, and the one that they wanted to specify or *the mental model.* Furthermore, similar to the case of model reconciliation process, here too we have model differences – the revealed model and the mental model do not match due to mistakes made by the domain writer. However, unlike in the case of explanations in the model reconciliation framework, the target here is for the explainable AI system to transform the revealed model to the mental model, rather than updating the mental model to agree with the revealed model as focused on traditionally in the model reconciliation framework.

This is an iterative process with the human (in this case, the domain writer) firmly in the loop. This is because, unlike in standard closed world model estimation tasks where the features of the model are known, and a transition function between domain models can be specified (Bryce *et al.*, 2016; Keren *et al.*, 2017), here the problem is open ended as the domain writer gradually builds their agent model. Thus, the model acquisition task is strictly *not one of estimation of the mental model.*[1]

An Explainable Model Acquisition Setting `XMAS` is defined by the tuple $\Psi =$

[1]It is for this reason that we, while pointing out problems with the current domain (and foils raised by the domain writer), do not suggest possible fixes. This is something that the domain expert is in charge of authoring. At the end of the chapter, we will point to some future work in this direction.

$\langle \mathcal{M}, \mathcal{M}^H \rangle$ where $\mathcal{M}$ is the revealed model and $\mathcal{M}^H$ is the mental model of the domain writer.

Since the eventual goal of writing a domain is to enable a desired set of behaviors, and there may be many ways to specify the same agent behavior, we condition the end goal of an `XMAS` in terms of the space of plans afforded by the agent model that is being specified. The solution to Ψ is a sequence of model updates (as described in Chapter 4) $\mathcal{D} = \langle \mathcal{E}_1, \mathcal{E}_2, \ldots \mathcal{E}_n \rangle$ such that:

$$\Pi(\mathcal{M} + \sum_i \mathcal{E}_i) = \Pi(\mathcal{M}^H)$$

This means that at the end of the model acquisition process the domain writer has successfully captured the space of behaviors of the agent they were trying to model. In contrast to the model reconciliation process, for the model acquisition task, these updates are, of course, generated by the domain writer. From the XAIP perspective, the task of the planner here is to empower the domain writer to come up with the most efficient $\mathcal{D}$ – e.g. $\min \sum_i |\mathcal{E}_i|$ to reduce the overall complexity of the model acquisition process or just $\min |\mathcal{D}|$ to reduce the number of steps to reach the final model. The evaluation of the entire process requires research on the UX of abstractions, and is out of scope of this chapter.

Instead, in the following discussion, we tackle key difficulties faced by domain writers for individual interactions during this process (as experienced in preliminary internal tests of `D3WA`). The focus of the proposed solutions here is to address the computational limitations of the domain writer using XAIP techniques like domain abstractions that have been shown to be useful in user studies (Chapter 10) as a vehicle for explanations in complex domains.

11.2.4 Q1. *Why Is There No Solution?*

This is the case when: $\Pi(\mathcal{M}^H) \neq \Pi(\mathcal{M}) = \emptyset$, i.e. the domain writer mistakenly thinks that they have a solvable model. As we mentioned before, for a model acquisition task, it is not enough to surface a cause for unsolvability but one must also make sure that the domain writer gets actionable information in order to remedy the situation. The directive from the planner thus has the following components:

1. **Minimal Unsolvability** We present to the domain writer the smallest possible abstraction $Abs(Det(\mathcal{M}), \mathcal{P})$ such that this model too does not have any solution.

$$\begin{aligned} \text{Find: } & \mathcal{P} \\ \text{s.t. } & \Pi(Abs(Det(\mathcal{M}), \mathcal{P} \cup G)) = \emptyset \\ \text{and } & \min |\mathcal{P}| \\ \text{Show: } & Abs(Det(\mathcal{M}), \mathcal{P} \cup G) \end{aligned}$$

   By the properties of the abstraction used, $\Pi(\mathcal{M}) \neq \emptyset$ only if $\Pi(Abs(Det(\mathcal{M}))) \neq \emptyset$. Thus the domain writer can fix the root cause of unsolvability in this simpler domain first. We will later show in the empirical evaluations how this approach can significantly reduce the size of the specification that the domain writer has to inspect in order to fix an unsolvable model. We include the goal in all abstractions – this fluent is not directly accessible to the designer.[2]

[2]This does not mean that the designer cannot specify a goal. They can specify any starting condition and point out which of the outcomes in one or more actions ends the conversation. The latter is then compiled internally to a single goal achieving condition. Thus the framework is quite generic for modeling any goal-directed process and not necessarily tied to any specific initial condition or goal state. For more details, please refer to the discussion by Muise *et al.* (2019a).

2. **Maximal Solvability & Exemplary Plan Failure** While the previous component of the explanation provides a simpler version of the model for the user to debug, it does not illustrate failures of any potential solutions to motivate fixes that the designer might attempt. The following complements it with an illustrative failure in the current model of a plan generated from the *maximally solvable* abstraction, while letting the domain writer continue fixing the issue on the minimally unsolvable one.[3] As we will see in Chapter 12 this is an example of an explanatory witness.

   Find: $\mathcal{P}$

   s.t. $\Pi(Abs(Det(\mathcal{M}), \mathcal{P} \cup G)) = \{\pi\} \neq \emptyset$

   and $\max |\mathcal{P}|$

   Show: $\Pi(Obs(\mathcal{M}, \{\pi\})) = \emptyset$.

   The point here is to find the most complete simplification of the model where a solution exists and illustrate – for example, using VAL (Howey *et al.*, 2004) – to the user why that solution does not apply to $\mathcal{M}$. The domain writer could also use this sample plan to explore where exactly a possible solution becomes invalid in the current specification, or even use it as a starting point to create more complex foils to investigate further.[4]

[3]Note that we cannot generate a plan from a minimal unsolvable abstractions since there are no solutions there. One possibility would be to choose a model more abstract than the minimally unsolvable one. Unfortunately, these model would ignore most model features and provide no useful plan for the user to debug.

[4]To improve the efficiency of the search, we will restrict the search for abstraction set over subsets of $(F \setminus P_{min}) \cup \mathcal{G}$ (where P_{min} is minimum abstracting set). Since the abstraction techniques followed here guarantees that abstractions formed from supersets of $P_{min} \cup \mathcal{G}$ will not result in solvable model.

3. In addition to presenting the abstractions and an exemplary plan failure, we can further help the domain writer debug the problem by presenting them with an unachievable subgoal that corresponds to a landmark. We will be using the approach discussed in Section 11.1 to extract the landmarks.

11.2.5 Q2. *Why Is This Not a Solution?*

This is the case when (for a set of plans $\{\pi\}$ presented by the domain writer): $\{\pi\} \subseteq \Pi(\mathcal{M}^H)$ but $\{\pi\} \nsubseteq \Pi(\mathcal{M})$. This is really a special case of **Q1** – we perform the following transformation to solve this:

Set: $\mathcal{M} \leftarrow \mathit{Obs}(\mathcal{M}, \{\pi\})$

If: $\Pi(\mathit{Obs}(\mathcal{M})) \neq \emptyset$ (the compilation is solvable) then demonstrate to the user that $\nexists \pi \in \Pi(\mathit{Obs}(\mathcal{M}, \{\pi\}))$ such that $\mathit{cost}(\pi) > \mathit{cost}(\pi_i)\ \forall \pi_i \in \{\pi\}$ – this means that the foil is not better than anything else already in the solution. At this point, the user can ask **Q3**.

Else: Follow **Q1** with $\mathcal{M}$.

Note that we cannot run VAL directly on the foil since: 1) It is very unlikely that the domain writer will provide full foils – this is largely due to the effort required in doing so but also uniquely infeasible for the current setting of dialogue design since internal system actions such as web calls and logical inferences are not part of logs that are used to stress test the design of the agent; and 2) in the case of a partial foil, $\mathit{Obs}(\mathcal{M})$ may not have a solution to run VAL with.

Moreover if P_{min} is the only subset of fluents leading to the unsolvability then $(F \setminus P_{min}) \cup \mathcal{G}$ should automatically be a solvable domain. So we will start our search from $(F \setminus P_{min}) \cup \mathcal{G}$ and will look at systematically relaxing the model until we get a solvable one. We evaluate this approximation against the exact approach in our evaluations.

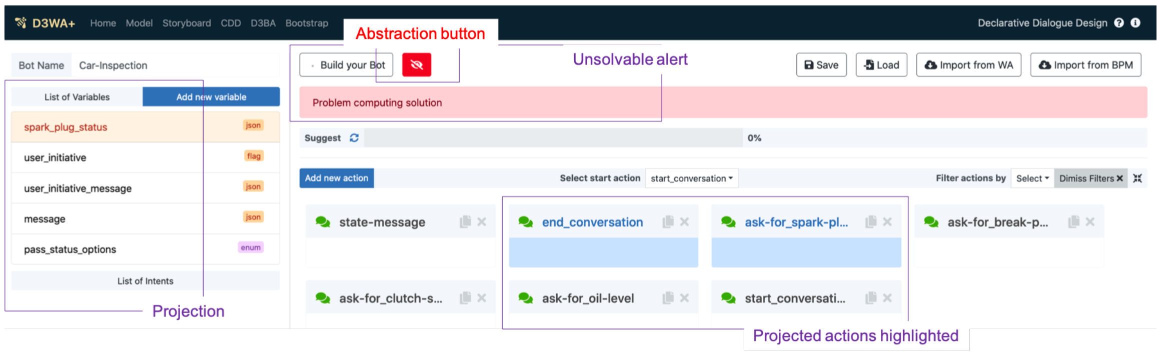


Figure 11.2: Snapshot of D3WA+ Illustrating Model Abstraction in Response to Unsolvability. Since the Model Has No Solution, There Is No Generated Dialogue Plan and the Canvas for Visualization Is Empty – Our User Story Starts with a White XMAS. The Minimal Unsolvable Abstraction Is Presented to the Designer so That They Can Devise a Fix in the Simpler Model First, Before Translating That to the Full Specification. Note How Some Variables in the Domain Have Disappeared in the Minimal Abstraction, in Comparison to Figure 11.1. The Designer Can Toggle Back and Forth Between the Full and Projected Models Using the Red Button, until They Are Able to Devise a Fix. The Highlight Feature Communicates to the User Which of the Projected Fluents (Orange Hover) Are Associated with Which of the Actions (Blue Highlight).

11.2.6 Q3. *Why Is This a Solution?*

This is the case when (for a set of plans $\{\pi\}$ presented by the domain writer): $\{\pi\} \nsubseteq \Pi(\mathcal{M}^H)$ but $\{\pi\} \subseteq \Pi(\mathcal{M})$. Here, the domain designer is surprised that a solution they did not expect is part of $\Pi(\mathcal{M})$. The provenance of actions along such a solution of $\mathcal{M}$ can be communicated to the designer through the visualization of the necessary causal links as done, for example, in (Seegebarth *et al.*, 2012; Chakraborti *et al.*, 2019a; Bercher *et al.*, 2014). We do not repeat this line of inquiry here.

11.3 Illustrations on D3WA+

We will now illustrate each of the use cases covered above on our tool D3WA+. In the context of the design of dialogue agents using planning, each "solution" is a potential conversation path allowed by the agent design. Hence, the model acquisition process is one of the dialogue designer ensuring which conversations are allowed and which are not.

Demonstration

While we attempt to illustrate as much of the use cases as possible in the limited space available here, please refer to the video here: http://ibm.biz/d3wa-xaip for a more detailed walkthrough of all the use cases discussed in the chapter. (Duration: 7min 55sec *excluding* explanation generation time reported in Table 11.1)

Car Inspection Bot

For purposes of illustration, we consider the design of a conversational agent tasked with helping in the inspection of a car. This domain is adapted from (Muise *et al.*, 2019a) as a typical demonstration of the design of conversational agents using automated planning techniques. The final dialogue plan, as seen in Figure 11.3d, has 63 nodes and 272 edges and is thus quite comfortably out of scope for the state of the art in dialogue design. The declarative specification as seen in Figure 11.1, on the other hand, has just 8 variables and 7 non-deterministic actions. Let us consider the following dialogue in this domain:

```
Bot: Ready to record.
User: Break pads pass.
Bot: Ok, break pads pass.
```

```
User: What's next? <-- initiative switch!

Bot: Check the spark plugs.

User: What are the options.

...

Bot: Inspection complete!
```

The interesting part is the potential for *initiative switch* during the conversation – either the user can go through all the parts by themselves or hand over control to the bot to guide them, or a combination of both.[5] The salient feature of the specification is thus: there is a catch-all dialogue action to respond to the user when they have initiative and a set of actions to ask the user for information when the bot has initiative. There is one outcome in all these actions that switch initiative based on what the user has said, while the other outcomes update the state of the inspection by logging the correct variables based on the user utterance. Next, we follow the designer's journey to get to this specification.

Q1. Why is there no solution?

In our user story, the designer has forgotten to add a few critical domain conditions – the OUTCOME for the initiative switch is missing in the catch-all action while the UPDATE for spark plugs is also missing in the corresponding OUTCOME (refer back to the introduction to `D3WA` for a refresher of this modeling artifacts). As a result the model has become unsolvable – there is no way for either the inspector or the bot to drive the initiative and visit all the parts. In Figure 11.2 the user is presented with the minimal abstraction where the model is unsolvable. They fix this simpler

[5]The `PDDL` models of the inspection domain are available online at the following link (along with snapshots of the specification on the `D3WA+` interface) for all the use cases discussed in the chapter: `http://ibm.biz/d3wa-inspection`.

problem instance		maximal abstraction (approx)						maximal abstraction (exact)						minimal abstraction						
		size				time		size				time		size				time		
		$\lvert\mathcal{P}\rvert$	abstraction	size in %	plan	abstraction	plan	$\lvert\mathcal{P}\rvert$	abstraction	size in %	plan	abstraction	plan	$\lvert\mathcal{P}\rvert$	abstraction	size in %	plan	abstraction	plan	landmark
	p_1	21	286	86.405	5	0.882	0.956	22	307	92.749	5	5.711	0.987	2	47	14.199	1	111.363	1.014	1.103
Car	p_2	21	293	88.52	5	0.911	1.011	22	314	94.864	5	10.816	1.057	2	40	12.085	1	175.995	0.826	1.158
Inspection	p_3	21	286	86.405	5	1.069	0.967	22	307	92.749	5	13.561	1.117	2	47	14.199	1	208.862	0.863	1.088
	p_4	21	282	85.196	5	0.926	0.981	22	303	91.541	5	16.290	0.987	2	51	15.408	2	154.084	0.891	1.140
	p_5	21	286	86.405	5	0.995	0.978	22	307	92.749	5	16.351	1.058	2	47	14.199	1	237.173	0.984	1.041
	p_1	25	338	94.15	1	0.999	0.993	25	338	94.15	1	11.606	17.452	1	23	6.407	1	9.216	0.936	1.056
Data	p_2	25	338	94.15	1	1.288	1.162	25	338	94.15	1	14.279	1.157	1	23	6.407	1	13.302	0.971	1.196
Doppelganger	p_3	25	338	94.15	1	2.405	2.914	25	338	94.15	1	25.561	1.050	1	23	6.407	1	23.444	0.973	1.211
	p_4	25	338	94.15	1	0.834	0.924	25	338	94.15	1	11.035	0.935	1	23	6.407	1	11.115	0.923	1.000
	p_5	25	338	94.15	1	2.661	2.845	25	338	94.15	1	13.157	4.792	1	23	6.407	1	12.895	0.792	1.136
	p_1	75	2104	97.227	8	1.139	1.209	76	2161	99.861	8	31.396	1.202	2	61	2.819	2	1106.939	0.950	1.137
Credit	p_2	76	2161	99.861	7	1.113	1.070	76	2161	99.861	7	46.256	1.051	1	5	0.231	1	40.325	0.934	1.093
Card	p_3	76	2153	99.492	4	1.155	1.069	76	2153	99.492	4	33.903	1.105	1	12	0.555	1	29.498	0.935	1.165
	p_4	76	2160	99.815	7	1.040	1.149	76	2160	99.815	7	54.096	1.159	1	3	0.139	1	47.422	0.878	1.016

Table 11.1: Empirical Properties of `XMAS` in Three Typical Conversational Domains Modeled in `D3WA+`. Notice the Massive Reduction in Size for the Minimal Abstraction, Intended to Make It Easier for the Domain Writer to Inspect Issues with the Domain. Also Notice the Large Difference in Size Between the Minimal and Maximal Abstractions, Thereby Indicating the Need for a Maximal Abstraction to Capture Enough Model Information in Order to Produce a Useful Foil for the Domain Writer to Fix.

specification to arrive at the solution in Figure 11.3a. The fix – adding the UPDATE for spark plugs – when applied to the original specification takes the designer to the current dialogue plan in Figure 11.3b. This interaction with `D3WA+` allowed the user to find a fix for an unsolvable model by inspecting a *much* simpler model.

Q2. Why is this not a solution?

However, the missing OUTCOME for the initiative switch in the catch-all action is still missing.[6] As a result, all solutions right now only involve the user driving the

[6]We want to impress on the reader at this point that this discussion of "mistakes" or missing components are in hindsight – the target model does not exist until the domain designer gets there.

conversation and the resulting solution in Figure 11.3b looks different from what the designer was expecting – i.e. it does not contain any conversation flow where the bot has initiative. The domain writer expects the sample conversation to be possible but do they know it's unsolvable, at all? In the spirit of XMAS, the domain writer gets to query D3WA+ with this foil – see Figure 11.4. The system again responds with a minimal abstraction to fix, along with a sample plan and an unachieved landmark in the maximally solvable abstraction illustrating why that foil fails in the current model. This not only explains the unsolvability but also provides powerful directive to fix the model.

11.4 Empirical Evaluations

We empirically investigate the properties of XMAS in terms of the size of the abstractions relative to the size of the original specifications, and the time taken to generate them. The size of the abstraction allows us to measure how much simplification of the model is achieved by our method, while the time taken is a measure of viability of our approach. We focus on **Q1** here since the properties of the solutions to **Q2** are derived from **Q1** while, as we mentioned before, **Q3** is already quite well understood in existing literature.

Test Domains

To test out the empirical properties of our approach, we use two new domains, in addition to the car inspection domain used in the illustrative examples. The first of them – Data Doppelganger – is an assistant chat-bot that that helps a user perform variety of data science tasks, such as plotting a graph, given a data set. The other new domain – Credit Card Recommendation – is again adopted from (Muise *et al.*, 2019a), and takes the user through choices of credit cards and their features until they make a selection.

To evaluate the effectiveness of `XMAS`, we needed to evaluate the systems on plausible mistakes on these test domain. For Table 11.1, we tried to create unsolvable problems for the first three domains by removing three model conditions at random. Specifically, we deleted adds and initial states from the model. Unfortunately, for the credit card domain randomly removing a subset of model components weren't yielding unsolvable problems. So instead, we went with a unique domain-specific mistake for each of the five cases. Each mistake was centered around the domain author missing adds for a specific outcome or initial state for a specific proposition. After identifying a reason for unsolvability, we further delete two additional adds per unsolvable instance. The solvability of the determinized problems was tested using FastDownward (Helmert, 2006) implementation of A^* and LM-Cut (Helmert and Domshlak, 2009). The landmarks were extracted using FastDownward implementation of method presed by Keyder *et al.* (2012) with $m = 1$. All experiments had a timeout of 60 mins.[7]

Model Compression

Table 11.1 shows the amount of compression offered by the abstraction, against the size of the full models, for five randomly generated unsolvable instances. Here the size of each possible model is reported in terms of the number of non-goal fluents that are part of it ($|\mathcal{P}|$), the number of model conditions that are part of the problem (denoted as size in the table) and the percentage of model conditions remaining as compared to the original domain model (denoted as 'size in %' in the table). The larger the compression, the easier we make it for the domain writer to understand the cause of unsolvability, as has been established in existing literature (Chapter 10). Clearly, we are able to significantly reduce the size of the specification for this purpose using the

[7] p_5 for credit card domain timed out (removed from Table 11.1).

# edits	1			2			3		
problem	$\|\mathcal{P}\|$	size	time	$\|\mathcal{P}\|$	size	time	$\|\mathcal{P}\|$	size	time
p_1	2	47	49.477	2	51	87.685	✓	✓	✓
p_2	2	40	152.989	2	51	193.358	-	-	-
p_3	2	40	184.747	✓	✓	✓	✓	✓	✓

Table 11.2: Size of Abstractions along the Entire Model Reconciliation Process During the Model Acquisition Task.

proposed abstraction approach. For the credit card domain, where the original model contains close to 2k components, being able to focus on a subset containing only 5 conditions is a massive reduction.

Maximal versus Minimal Abstractions

Table 11.1 also shows the difference in size of the plans in the minimal[8] and maximal abstractions. As we mentioned before, this was a specific design choice made so as to ensure that the domain writer has a reference point while inspecting an unsolvable model – they can use this either to debug the current model or to explore newer foils. The point of computing this reference point in the maximal as opposed to the minimal model (as evident from Table 11.1) is to provide more helpful debugging information to the domain writer – `XMAS` is not just about explaining unsolvability but also completing the model acquisition task. The reference point uses the maximal model in order to make sure maximal number of model features are considered so

[8]Note that since the minimal model is also unsolvable, the plans used here for comparison are from the models that are one abstraction simpler than the minimal unsolvable model.

that the generated foil is as close as possible to a plan the user might be looking for in $\mathcal{M}^H$. In Table 11.1, we refer to generating maximal model by starting from abstraction corresponding to $F \setminus P_{min}$ while the exact one refers to a systematic search starting from the most concrete domain to one where it is solvable. While search for maximal model was in general fast, we can see that the approximate method is much faster and finds models that are quite comparable to the exact minimal solution.[9]

Model Evolution

In Table 11.2, we illustrate the evolution of the size of the abstractions as the domain writer progressively fixes the car inspection model. To simulate this, we needed to create models with potentially multiple causes for unsolvability. We generated nine pairs of model conditions (adds or initial state), each of whose removal can lead to an unsolvable problem. We then created three problems by removing three of those pairs. This means the model could have from one to potentially six unique causes for unsolvability. We passed each of these problems through our system, looked at the abstractions, made fixes and then tested if there were further issues. For p_1 and p_3, we were able to successfully make the problem solvable. While for p_2, we were able to fix two issues, but the explanation system timed out on the third trial. This illustrates the journey of the domain writer towards converging the space of solutions in $\mathcal{M}$ and $\mathcal{M}^H$ and the massive simplification of the model complexity offered by the proposed approach along this journey.

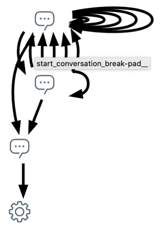
[9]An interesting course of investigation in the future would be adopting the considerable body of work in the determination of dead ends during planning (Steinmetz and Hoffmann, 2017; Muise, 2014; Kolobov *et al.*, 2010) for `XMAS`. While our motivation here is user facing and is thus quite different to those works – i.e. we want to use abstractions to facilitate explanations, particularly in the model acquisition task, rather than speed up planning – it would be interesting to explore whether those techniques can speed up the explanation generation step in the future.

Computation

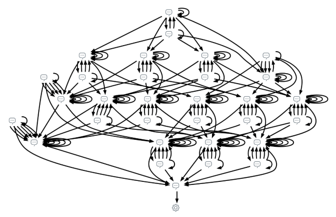
Finally, we report the time taken to compute the different components of the planner's directive to the domain writer, in Table 11.1. We do not explore optimizations in this chapter, but we are well within the bounds of real-time use even for the massive credit card domain.

11.5 Concluding Remarks

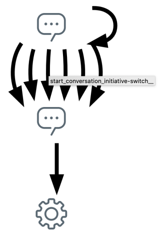
In this chapter, we revisited the HELM framework and extended it to support models with non-deterministic effects. As part of this generalization, we introduce an abstraction function for such models and also introduce a way to identify unsolvable landmarks. We then evaluate the utility of such explanation generation methods by utilizing them in the context of a model debugging tool and in the process also introduce the notion of model-acquisition task. The specific debugging tool was developed as an extension of an existing tool developed by IBM for allowing dialogue designers to specify the behavior of automated dialogue agents. The domain debugger tool was also presented as a demo in ICAPS-2020, where it was also one of the recipients of the best ICAPS demo award for that year. In the next chapter, we will take another step to further generalize this series of work, by showing how they can be understood as being instances of a more general explanatory framework called the Model-Simplification framework. Additionally, we will show how this general framework could be used to generate explanations for stochastic planning problems.



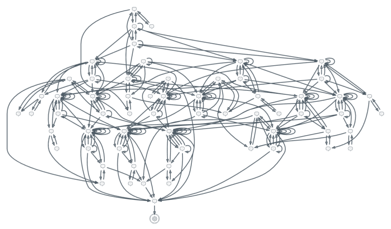
(a) XMAS Past: Solution of the Minimal Abstraction. Now That the Designer Was Able to Fix the Simplified Domain, They Can Try Applying the Fix to the Full Specification.



(b) XMAS Present: This Is the Solution When the Fix to the Minimal Abstraction Is Applied to the Full Specification. Now It Is at Least Solvable but Something Doesn't Look Quite Right: The Solution Does Not Match the Expectation of the Domain Designer. Now That They Have a Solution to Work with, They Now Direct Questions of the Form of **Q2** to D3WA+. Figure 11.4 Illustrates One Such Interaction.

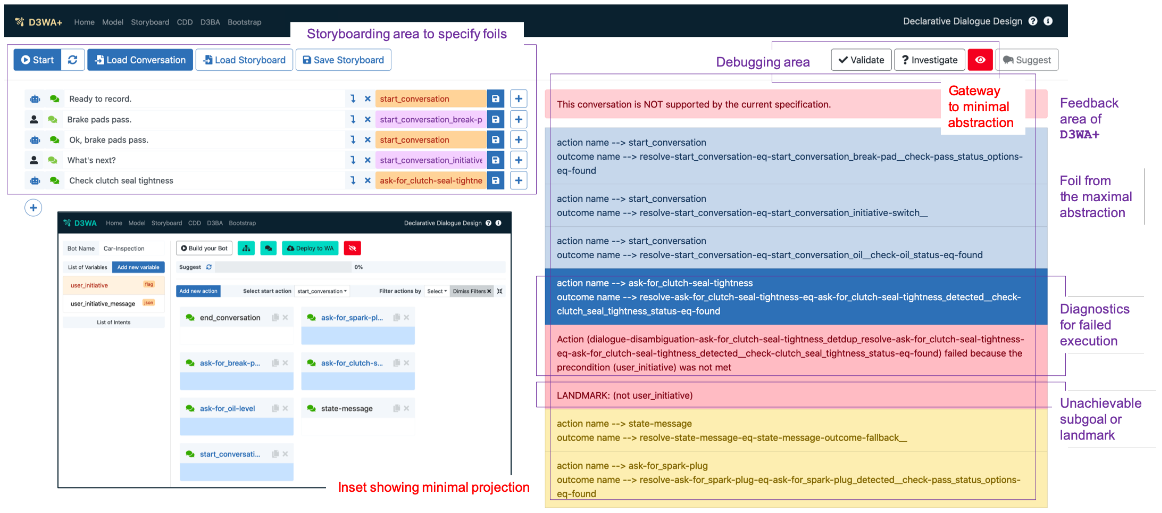


(c) XMAS Present: The Domain Writer Solves the Minimal Abstraction Again. Applying This Fix to the Original Specification Takes the Domain Writer from Figure 11.3b to 11.3d.



(d) XMAS Future: This Is the Final (Target) Solution: A Model That Is Equivalent to $\mathcal{M}^h$. The Domain Writer, of Course, Does Not Have Access to This until They Get There – and Neither Does D3WA+– but Instead They Continue Working on Their Current Specification and Its Solution (or the Lack Thereof) until They Are Satisfied. During the Course of XMAS, the System Was Able to Guide the Designer along This Journey by Exposing the Mistakes in Their past Models.

Figure 11.3: Examples of Generated Dialogue Graphs Illustrating the Reconciliation of $\mathcal{M}^h$ to $\mathcal{M}$ During XMAS. The Nodes in the Graph Stand for Agent Actions While the Edges Are the Non-deterministic Outcomes. The Graph Is Meant to Give the Reader a Sense of the Sizes of the Abstract Solutions. Though the Actual Labels on the Nodes and Edges Are Not Visible Here, the System Does Allow the Domain Designer to Drill down Further as Required.



Ready to record.	start_conversation
Brake pads pass.	start_conversation_break-p
Ok, brake pads pass.	start_conversation
What's next?	start_conversation_initiative
Check clutch seal tightness	ask-for_clutch-seal-tightne

This conversation is NOT supported by the current specification.

action name --> start_conversation
outcome name --> resolve-start_conversation-eq-start_conversation_break-pad__check-pass_status_options-eq-found

action name --> start_conversation
outcome name --> resolve-start_conversation-eq-start_conversation_initiative-switch__

action name --> start_conversation
outcome name --> resolve-start_conversation-eq-start_conversation_oil__check-oil_status-eq-found

action name --> ask-for_clutch-seal-tightness
outcome name --> resolve-ask-for_clutch-seal-tightness-eq-ask-for_clutch-seal-tightness_detected__check-clutch_seal_tightness_status-eq-found

Action (dialogue-disambiguation-ask-for_clutch-seal-tightness_detdup_resolve-ask-for_clutch-seal-tightness-eq-ask-for_clutch-seal-tightness_detected__check-clutch_seal_tightness_status-eq-found) failed because the precondition (user_initiative) was not met

LANDMARK: (not user_initiative)

action name --> state-message
outcome name --> resolve-state-message-eq-state-message-outcome-fallback__

action name --> ask-for_spark-plug
outcome name --> resolve-ask-for_spark-plug-eq-ask-for_spark-plug_detected__check-pass_status_options-eq-found

Figure 11.4: Snapshot of D3WA+ Illustrating Foil-based Interactions When a Model Is Solvable but Does Not Match Expectations. Note How Small This Projection Is (Inset), Even Smaller Compared to the First Unsolvable Model in Figure 11.2: the Corresponding Solution in Figure 11.3c is Similarly Smaller than the One in Figure 11.3a. This Showcases an Interesting Property of XMAS– the Size of Abstractions Is Non-monotonic in the Course of the Model Acquisition Task. On the Right, We Can See Here the Foil Generated by D3WA+ from the Maximal Abstraction with Diagnostic Information on How This Fails in the Current Specification.

Chapter 12

EXPLANATION THROUGH MODEL-SIMPLIFICATION

Through the previous chapter in this part of the thesis, we looked at the variations of an explanation generation framework called HELM, that are designed for scenarios where the human confusion about the robot decision comes from an inherent *inferential capability gap* between the robot and user. In this chapter, we will introduce a general framework for generating explanations in the presence of such inferential capability gaps. A framework that will be grounded in the generation of simplified representations of the agent model that is sound for a given explanatory query. This framework allows us to develop the most general algorithm we are aware of for generating explanations aimed at addressing inferential capability gaps. We will also see how many of the existing works in this direction can be understood as limited forms of this more general method. While the ideas we present in this chapter are general enough to be applied to any decision-making framework, we will focus on instantiating the framework in the context of stochastic planning problems. As part of the instantiation, we will also provide an exhaustive characterization of explanatory queries and an analysis of various classes of applicable transformation. We will evaluate the effectiveness of transformation-based explanation through both synthetic experiments and user studies.

The rest of the chapter will be structured as follows; In section 12.1 we will start with a running example that we will use throughout the chapter. We will then start the technical discussion by introducing the basic framework (Section 12.3), which will lay out the basic algorithm and the concept of the model simplifying transformations that will act as the basis for explanations. In Section 12.3.1, we will also introduce the concept of explanatory witness, which could be used to further help the users make sense of the explanation. With the basic framework in place, we will delve deeper into the grounding of the framework in the context of stochastic planning problems, and go over the characterization of each component of the framework in this setting. This would involve providing an exhaustive characterization of contrastive explanatory queries possible in stochastic planning settings (Section 12.4) and introducing a set of model transformations with some precedence in the explanation literature (Section 12.5). Each model transformation class presents a conceptually consistent formalization of a large class of specific transformations that could be applied to a given planning problem to simplify the problem. For each transformation class, we will also point to some previous works in the wider explainable AI (XAI) literature that have applied similar techniques thereby presenting a previous instance of an application of a limited form of our proposed general framework. For each transformation class, we introduce one specific technique which only takes polynomial time to perform and analyze the properties of the proposed technique for the class of stochastic planning problems we are interested in. In Section 12.6, we will look at the various possible stand-ins for the human computational model we could use in the framework. Section 12.7 will discuss a user study we ran to measure the effectiveness of the individual transformation. We will use the lessons from this user study to also introduce the stratified search algorithm (Section 12.5), which is a more constrained version of the algorithms introduced in Section 12.3 and is specifically designed for the transformation classes introduced in Section 12.5. Section 12.9 presents empirical results over a simulated study to evaluate the effectiveness of the explanation (using automated stand-ins to measure the complexity of the explanation). In section 12.10, we will further see how most current works aimed at addressing the inferential capability gap can be seen as a specific instance of our more general framework in that they look at the use of some limited set of transformations or focus primarily on the use of some explanatory witness. All the frameworks and analyses will be grounded in the context of MDPs with the P assumption (i.e., Positive action cost assumption) (Bertsekas, 2005), which generalizes over many commonly studied stochastic planning formalisms.

12.1 Running Example

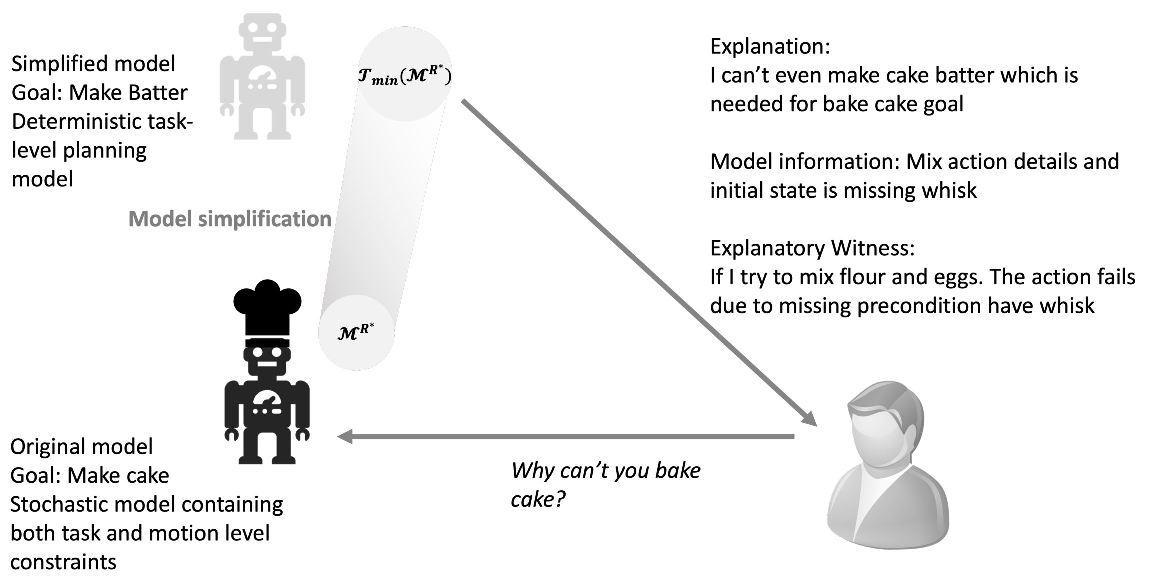

Simplified model
Goal: Make Batter
Deterministic task-level planning model
$\mathcal{T}_{min}(\mathcal{M}^{R^*})$
Model simplification
$\mathcal{M}^{R^*}$
Explanation:
I can't even make cake batter which is needed for bake cake goal
Model information: Mix action details and initial state is missing whisk
Explanatory Witness:
If I try to mix flour and eggs. The action fails due to missing precondition have whisk
Original model
Goal: Make cake
Stochastic model containing both task and motion level constraints
Why can't you bake cake?

Figure 12.1: The Model Simplification Process Used to Generate the Explanations for the Motivating Example, Where a Robotic Chef Needs to Explain Its Inability to Bake a Cake.

Figure 12.1 presents a diagrammatic representation of the overall explanatory process entailed by the methods proposed in the chapter. Specifically, the figure refers to an example scenario which considers a robotic chef working in a house. The user of the robot starts by asking the robot to bake a cake, to which it replies it can't bake a cake. Now the user demands an explanation for its failure to prepare a cake. The robot may have come to the conclusion that it cannot prepare a cake from the fact that its stochastic task and motion planner failed to produce a valid plan. Dumping its search tree or the underlying model to the human would hardly suffice as a satisfactory explanation. Once the robot further analyzes the query, it can realize that even if it were to ignore the stochasticity of the domain (related to various slippages the robot may incur in its operation) and all the motion constraints, the problem remains unsolvable. In fact, the robot can't even create cake batter which is required for the final goal because it doesn't have a whisk. Thus the robot could ignore all low-level details and just needs to inform the user about a simplified model that includes information about the mixing action that makes the cake batter and how the action has a precondition has a whisk and currently, there is no whisk. Additionally, the robot could even provide a hypothetical trace of actions it would perform to make cake batter and how it would fail due to a missing precondition. We will refer to such secondary explanatory information meant to help the user better understand the explanation being given as *explanatory witness.*

12.2 Background

Following Section 2.3, we will be assuming an undiscounted MDP that satisfies the P-assumption. Such a model will be represented by the tuple $\mathcal{M} = \langle S, A, P, C, I, G \rangle$, where S is the state space, A the set of possible actions, $P : S \times A \times S \rightarrow [0, 1]$ the transition probabilities, C the cost of executing a given action in the state (where $C(s, a) \geq 0$), $I \in S$ the initial state and $G \subseteq S$, is the set of absorbing goal states. We will use the functions $J : S \rightarrow \mathbb{R}$ and $Q : S \times A \rightarrow \mathbb{R}$ to capture the expected total cost and Q function and use J^* and Q^* to represent the optimal cost and Q function respectively. We will use J^π to denote the cost function obtained by following a given policy π. We will use $P^\pi(s) = \sum_{g \in G} P(g|s, \pi)$ to denote probability with which the execution of policy from a given state would lead it to a goal state. We will use $P^*(s)$ to denote the highest possible probability of achieving the goal under any policy for the given model (i.e as the MAXPROB policy) and use $P^\pi(s)$ to capture the probability under a given policy.

We will use the tuple $\mathcal{M}^\mathcal{D} = \langle F^\mathcal{D}, A^\mathcal{D}, I^\mathcal{D}, G^\mathcal{D}, C^\mathcal{D} \rangle$, to represent the PPDDL representation of an MDP. We will limit our attention to cases with positive preconditions, so, each $a \in A$ is further defined as $a = \langle pre_+(a), \mathbb{E}(a) \rangle$. For a given description $\mathcal{M}^\mathcal{D}$, we will use the notation $\mathcal{M}$ to refer to the MDP induced by it, especially if we don't explicitly mention the underlying MDP.

Such factored model descriptions are usually easier for people to understand and create, not just due to conciseness, but also because it is based on folk psychology principles related to actions and as such are intuitive descriptions (Boonzaier *et al.*, 2005). We assume that models are provided in such descriptions, though one could always start with an atomic or inscrutable representation of the model and derive such models through different learning methods (Sreedharan *et al.*, 2022c; Konidaris *et al.*, 2018)

12.3 Model Simplification Framework for Contrastive Explanation Generation

Our basic explanatory setting consists of a sound and optimal model-based decision-making system that uses model $\mathcal{M}^{R^*}$ (which may correspond to a description $\mathcal{M}^{D^*}$), to come up with its decisions and it needs to respond to possible explanatory queries a user of the system may raise either about its current decision or about alternative decisions the user may have expected.

The central focus in this chapter is to provide *contrastive explanations* (Miller, 2017a), where contrastive explanations are said to be explanations that are responses to questions of the form "Why P and not Q?", where 'P' is referred to as the fact being explained and 'Q' is referred to as the foil the fact is being contrasted against.

There is a lot of evidence from social science literature that contrastive explanations underpin most of our everyday explanatory dialogues and as such has been receiving a lot of attention in recent years (Miller, 2017a; Hoffmann and Magazzeni, 2019; Weld and Bansal, 2019). In our scenario, we are always interested in answering questions of the form.

"Why policy π and not any policy in the set Π'"

That is the explanation always comes down to establishing the choice of one policy π (possibly the current one being proposed by the system), over the set of alternate policies (possibly expected by the user). If the system is actually claiming that problem is unsolvable, i.e., the goal cannot be achieved (as in the case of the example covered in the introduction), the fact could also correspond to having no solution (more formally one could also map this to a special type of policy, but we will skip this formalization for simpler exposition). In cases where the system is a sound, complete and optimal decision-maker, the answer to this question always takes the form of establishing the preference of one policy over another (or at the very least establishing they are equivalent). In the case of the MDP classes studied in this chapter, this always takes the form of establishing that the fact policy either has a higher probability of achieving the goal or lower cost as compared to the policies in the foil set. One of the important challenges to generate explanations for such queries is that in most cases the foil set may not be exactly specified. For example, in the case of the introductory example, the question *"Why can't you bake cake?"* only implicitly specifies the set of possible foils. In fact, in this case, the foil set corresponds to the set of all possible policies, thus one could re-frame the question in an explicitly contrastive form as

"Why is the system claiming that no solution is possible as opposed to following any policy?"

In this chapter, we will argue that one could achieve a deeper insight into the explanatory process by separating the form of the question from the underlying model property that needs to be established as part of resolving the user query. In essence, one could convert every contrastive explanatory query into a problem of establishing that a planning model can only generate solutions that satisfy a particular threshold with respect to one of its optimization criteria.

Definition 31. *For any given explanation query, a tuple $\mathcal{P} = \langle \mathcal{O}, \mathcal{X} \rangle$, where $\mathcal{O}$ is an optimization criterion (cost $\boldsymbol{C}$ or probability $\boldsymbol{P}$) and $\mathcal{X}$ is a threshold (either an upper bound or a lower bound) over $\mathcal{O}$, is said* ***to be an explanatory property for the given query*** *if one could resolve the query by establishing that a model $\mathcal{M}$ cannot generate a solution that violates that threshold (we will denote this as $\mathcal{M} \models \mathcal{P}$).*

With respect to the example query from the introduction, one explanatory property could be $\mathcal{P} = \langle \mathbf{P}, = 0 \rangle$, i.e., if the property $\mathcal{P}$ is true for a model, then the model can't allow any solution whose probability of achieving the goal is not equal to 0.

While Definition 31 may seem like a very permissive definition where one could have arbitrary thresholds over each optimization criteria, the fact that these explanatory properties are meant to resolve meaningful explanatory queries does restrict the forms they can take. For example, one would never require the establishment of the fact that one can generate policies that are worse off than the current policy either in terms of cost or goal achievement likelihood. After all, any response to a contrastive query is trying to show that you can't do better than the current policy in terms of either optimization criteria.

In Section 12.4, we will look at an exhaustive characterization of all the possible types of explanatory properties and how in fact every possible contrastive query map into this particular form. It is also worth noting that, for any given explanatory query, there might be multiple possible explanatory properties, especially ones that use a different optimization criteria. However, given the fact that there exists a strict ordering between the optimization criteria, we will generally prefer explanatory properties related to the criterion with the higher preference. We will denote the fact that the explanatory property $\mathcal{P}_i$ is preferred over a property $\mathcal{P}_j$, by using the notation $\mathcal{P}_i \prec \mathcal{P}_i$.

This chapter will build on the basic fact that humans are agents capable of reasoning about sequential decision-making problems. Thus one way to respond to a given query may be to provide the underlying model used to generate the decision. As long as the model is in a human-readable form[1] one could always communicate this information and expect the human to try to reason about whether the relevant explanatory property holds on their own. This directly connects to earlier works in model reconciliation explanation (Chakraborti *et al.*, 2017), where the goal of explanation is to provide humans with enough model information that they can correctly evaluate the plan in question. In fact, the setting studied in this chapter corresponds to a model-reconciliation explanation scenario where we assume the human has no previous information about the system model. Model-reconciliation explanations focus on optimizing for additional criteria like minimizing the amount of information to be provided as part of the explanation. While still relevant in our case, we will leave operationalization of such criteria as future work and focus on the basic tenet that a person would find their explanatory queries resolved if their updated model can support a justification for the query (in our case this corresponds to whether the updated model support an explanatory property $\mathcal{P}$). However, the original model-

[1]We assume our models to be already in human-readable form, and even if they are not one could always convert them to such a form using methods like those discussed by Sreedharan *et al.* (2022c)

reconciliation explanation method assumes that humans have adequate inferential capability to correctly use the given model to perform the necessary verification. This is not an assumption that is necessarily met in practice as the model could be quite complex. Thus our explanation method needs to explicitly take into account how easy it is for a user to understand the explanation.

To capture the inferential burden placed on the human by any explanation, we will introduce the function $Comp_H(\cdot, \cdot)$ to denote the computational capability of the human. Specifically, $Comp_H(\cdot, \cdot)$ takes a model $\mathcal{M}$ and an explanatory property $\mathcal{P}$ and gives the expected time taken for the user to verify that an explanatory property holds in the given model. We will assume $Comp_H(\cdot, \cdot)$ returns ∞ if the human is incapable of establishing the truth of the property.

An inferential capability gap is said to exist between a human and a decision-making system *in the context of a model $\mathcal{M}$ and an explanatory property $\mathcal{P}$*, if the computational effort required by the human doesn't match the one required by the agent, i.e, let $Comp_{sys}$ is the corresponding function that stands-in for the computational effort required by the agent, then we assert that an inferential capability gap exists between the two in the context of $\mathcal{M}$ and $\mathcal{P}$, if

$$Comp_{sys}(\mathcal{M}, \mathcal{P}) \neq Comp_H(\mathcal{M}, \mathcal{P})$$

In theory, we could have $Comp_{sys}(\mathcal{M}, \mathcal{P}) > Comp_H(\mathcal{M}, \mathcal{P})$ and one could also have cases where for the same model the human has an easier time establishing some property, while in the system is better at establishing others. However, in this chapter we are mostly interested in cases where we have $Comp_{sys}(\mathcal{M}, \mathcal{P}) < Comp_H(\mathcal{M}, \mathcal{P})$

Now the central argument we will make throughout this chapter is that even if the complexity of establishing an explanatory property in the original model is high (measured in terms of $Comp_H(\mathcal{M}, \mathcal{P})$), one can use simplified representations of the original model where one could establish the same property with lower effort. We can now give the user information about this simplified model and even use this model as a basis for generating other explanatory information. In particular, we will look at the generation of explanatory witnesses that will help the user better understand why the property holds in the current model.

Towards formalizing this intuition of explanation generation, we will first start by introducing the notion of model transformation that will become the basis of our remaining framework. In this chapter, we will refer to a model transformation function as one that takes a valid MDP model and generates a new MDP specification, i.e., it takes the form $\mathcal{F}_{\cdot} : \mathcal{M}_i \mapsto \mathcal{M}_j$ where $\mathcal{M}_i$ was the original MDP and $\mathcal{M}_j$ is the newly generated MDP. However, this is an extremely weak notion and not one that in anyway helps us build model representations that could help resolve users' queries. To establish that, we need to introduce two new notions about the transformations, namely, soundness of transformation and whether the transformations are building simpler representations.

The soundness of transformation relates to whether any explanatory property that holds in the new model has a corresponding property that holds in the original model, or more formally:

Definition 32. *A model transformation $\mathcal{F}_i$ is said to be a* ***sound transformation*** *for a model $\mathcal{M}$ with respect to an explanatory property $\mathcal{P}_i$, if $\mathcal{P}_i$ holds in $\mathcal{F}_i(\mathcal{M})$ only if it holds in $\mathcal{M}$.*

Note that we are not requiring the transformations to always preserve the explanatory property, i.e., $\mathcal{P}_i$ holds in $\mathcal{M}$ if and only if $\mathcal{P}_i$ holds in $\mathcal{F}_i(\mathcal{M})$. As we will soon see, most of the effective transformations we will look at, result in optimistic approximations, where we can't always guarantee that the transformations will preserve the property but we can be sure that the transformations are sound as defined above, i.e., there could exist properties $\mathcal{P}$, such that $\mathcal{M} \not\models \mathcal{P}$ but $\mathcal{F}_i(\mathcal{M}) \models \mathcal{P}$. Each transformation described in the running example, which we will go in further details in Section 12.5, are examples of sound transformation for the explanatory properties $\langle \mathbf{P}, = 0 \rangle$

Now we will assert that a transformation results in a simpler representation or equivalently is a simplifying transformation if it is easier to establish the property in question in the resultant model, i.e.,

Definition 33. *A transformation $\mathcal{F}_i$ is said to be a* ***simplifying transformation*** *for a model $\mathcal{M}$ and property $\mathcal{P}$, if $Comp_H(\mathcal{M}, \mathcal{P}) > Comp_H(\mathcal{F}_i(\mathcal{M}), \mathcal{P})$, additionally $\mathcal{F}(\mathcal{M})$ is referred to as a simpler representation of $\mathcal{M}$.*

One could now see that what we hope to build as explanations are models that are built by repeated application of simplifying but sound transformations. However, there is a big question we have yet to answer, namely, how does one stop the method from generating extremely simple models that are nonetheless completely disconnected from the original problem at hand. Effectively this would turn the output of the methods into lies rather than satisfying explanations (comparable to discussions provided by papers like (Chakraborti and Kambhampati, 2019c)), as it would give the user no further insights into the original planning problem. One could try to avoid this by placing restrictions on the types of models generated by the transformation functions. For example, requirements like the need to share action/state-space or the need to preserve some transition function. However, most of these methods are too limiting and insufficient when we consider cases where there might exist a vocabulary mismatch between the user and the decision-maker (Sreedharan *et al.*, 2022c).

In such cases, one might need to translate the current system representation into forms that are easier for the user to follow and it may be quite hard to establish any form of equivalence between model components of the learned models and the original model. Instead in this chapter, we will define the validity of the explanations in terms of their impact on human expectations about the system. In particular, we will require that the transformed model doesn't satisfy ***any*** explanatory property not true in the original model. Effectively this would make sure that the transformation does not prevent the user from asking any question they might have about the system's capabilities. This will effectively ensure that the user doesn't place unwarranted trust in the system due to the explanation, a necessary property for generating effective posthoc explanations (Sreedharan *et al.*, 2022c). We will refer to such as *universally sound* transformations for a given model:

Definition 34. *A transformation function $\mathcal{F}$ is said to be* ***universally sound*** *for a model $\mathcal{M}$, if for any explanatory property $\mathcal{P}$, we have $\mathcal{F}(\mathcal{M}) \models \mathcal{P}$, then $\mathcal{M} \models \mathcal{P}$. Equivalently we will refer to the model $\mathcal{F}(\mathcal{M})$ a universally sound representation of $\mathcal{M}$.*

All the transformations we will discuss in Section 12.5 are examples of universally sound transformations. Additionally, we will require that every transformed model presented to the human be explicitly noted to be a simplification of the true model, while noting the exact relationship between the true model and transformed model when possible. Usually when we look at transformations that are generated through syntactic transformations of human readable model descriptions (as in the case of transformations discussed in Section 12.5), such relations are much easier to note.

With these basic definitions in place, we can describe the central explanatory problem as finding a minimal simplified representation sufficient that is sound for an explanatory sufficient to respond to the given explanatory query. In this chapter, we will look at generating simplified model for a given explanatory property, with assumption being that we will try to find an simplified model sound with respect the most preferred explanatory property in the original model.

Definition 35. *Given a model $\mathcal{M}^{R^*}$, a set of universally sound transformation functions $\mathbb{F} = \{\mathcal{F}_1, ..., \mathcal{F}_k\}$, the problem of generating a* ***minimal explanatory model*** *for a given explanatory property $\mathcal{P}$, corresponds to the problem of finding a sequence of transformation $\mathcal{T}_{min}$ which results in a simplified representation $\mathcal{M}$ that is sound for $\mathcal{P}$ and there exists no other sequence of model transformations that will result in a simpler yet sound representation, i.e,*

$$\nexists \mathcal{T}' \; Comp_H(\mathcal{T}_{min}(\mathcal{M}^{R^*}), \mathcal{P}) > Comp_H(\mathcal{T}'(\mathcal{M}^{R^*}), \mathcal{P})$$

The new model $\mathcal{T}_{min}(\mathcal{M}^{R^*})$ will form the basis of the explanation.

The basic algorithm for generating such minimal explanations is given in Algorithm 8, which correspond to an exhaustive search over all possible transformations and then returning the transformation sequence, explanatory property pair that meets the requirements provided in Definition 35.

Note that the above algorithm is an extremely computationally expensive one. The search space is exponential over the number of transformations possible and each search node evaluation in our case consists of solving a planning problem. However, as we will see in Section 12.8 by making commitments on the types of transformation and model classes, we can make use of a significantly more efficient search algorithm.

12.3.1 Explanatory Witness

Even after providing the model, the user may not be able to reason about the explanatory property on their own and as such may require additional assistance.

Algorithm 8 Basic Search

```
1: procedure SEARCH
2:     Input:  $\mathcal{M}^{R^*}$, $\mathbb{F}$, $\mathcal{P}$
3:     Output: The transformed model $\mathcal{M}$
4:     Procedure:
5:     $\mathcal{T}_{\text{best}} = \langle\rangle$
6:     Curr_Minimal_Effort $\leftarrow \infty$
7:     for Each subsequence $\mathcal{T}'$ of $\mathbb{F}$ do
8:         Comp_Effort $\leftarrow Comp_H(\mathcal{T}'(\mathcal{M}^{R^*}), \mathcal{P})$
9:         if Curr_Minimal_Effort $>$ Comp_Effort then
10:            $\mathcal{T}_{\text{best}} = \mathcal{T}'$
       return $\mathcal{T}_{\text{best}}(\mathcal{M}^{R^*})$
```

We will refer to such information as *Explanatory Witness.* XAI literature includes many examples of such information, and in fact, many works focus on identifying such information, while making implicit assumptions about the model information with which the user is supposed to make sense of this information. In general, from the current literature we can identify three categories of such information;

1. Proof of explanatory properties: A proof of the property being explained, which in our case can be provided in the minimal model. Though unfortunately, it is very hard to create exact interpretable proofs for most query properties. In general, most of these proofs would be incomplete or abstract in the sense of skipping some steps. An example of such abstract explanatory witnesses would be the use of Q-values to contrast the current choice against alternative (as in the case of (Juozapaitis *et al.*, 2019)). Here the explanatory witness doesn't provide information as to why the current action in a state or the alternative

has a specific Q-value.

2. Existential Information: This corresponds to presenting instances of potential solutions (plan/policies) or parts of solutions (a specific execution trace from a given policy) that acts as a demonstration of the property being explained.

3. Counterfactual Information: In the final category, the user is provided by examples of counterfactual solutions and even planning models, where the property being explained doesn't hold. The assumption being that these examples would help the human get a better sense of when the property might not hold.

While generating explanations for user studies we will use a sampled policy execution trace as the corresponding Explanatory Witness (a type of existential information). For example, when trying to explain why the cost of a policy may be above some threshold, then one could find a specific execution trace, i.e., a sequence of states, actions and resulting state that can be sampled from the current policy and show that the cost of that trace is above the current threshold. This is a particularly appealing for cost property ($\mathcal{P}_{\mathcal{E}_1}$). For properties related to goal reachability, possible explanatory witness include the use of qualitative occupancy frequency (similar to (Khan *et al.*, 2009)) and for unsolvability one could present the infeasibility of some example paths to the goal (as in the case of (Sreedharan *et al.*, 2020b)).
Section 12.10 includes a discussion of existing types of explanatory witnesses studied in the literature.

In the rest of the chapter, we will revisit each component of our framework, namely, explanatory properties, transformation and computational model ($Comp_H(\cdot,\cdot)$) and ground it for our specific setting of stochastic planning. As discussed, we will use our the properties of our specific properties, transformation and proxies for $Comp_H(\cdot,\cdot)$ to propose a much more efficient version of identifying minimal explanations.

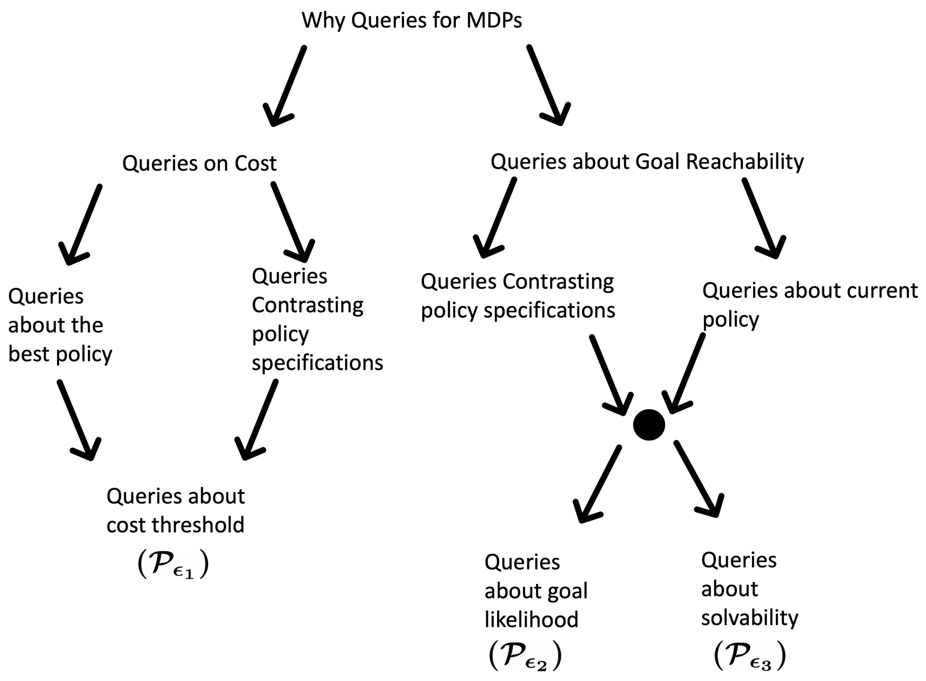

Why Queries for MDPs
Queries on Cost
Queries about Goal Reachability
Queries about the best policy
Queries Contrasting policy specifications
Queries Contrasting policy specifications
Queries about current policy
Queries about cost threshold
$(\mathcal{P}_{\epsilon_1})$
Queries about goal likelihood
$(\mathcal{P}_{\epsilon_2})$
Queries about solvability
$(\mathcal{P}_{\epsilon_3})$

Figure 12.2: A Hierarchy of *Why* Questions That Can Be Asked by a User Trying to Understand Decisions Generated by an Automated Decision-maker Using an Undiscounted Mdp That Meets the P Assumption.

12.4 Explanatory Queries and Explanatory Properties

Figure 12.2 presents the hierarchy of questions possible in this problem setting categorized over the two criteria. Now for each type, we further list two possible subcategories. The first category corresponds to queries exclusively about the current best solution provided by the system and the second corresponds to queries that contrast the current policy with an alternative the human had in their mind. In the former, the user would want to understand why the solution is worse off than what they were expecting (say in terms of goal reachability or cost), and in the latter, the user would want to know how the current policy compares against the alternative they had in mind. In the end, both categories can be mapped into a question that can be resolved by comparing a policy against a threshold. In the case of the former category, the threshold directly comes from the user's question and the system's policy is compared against it (for example, the kind of questions that might fall into this category would include instances like *Why does the policy cost more than X?* or *Why is the likelihood of reaching the goal less than Y?*), while in the latter the alternative posed by the user is compared against the cost or goal reachability of the current policy. In the end, the query hierarchy concretizes into three specific query types.

Each query type is characterized by a specific explanatory property,

1. $\mathcal{P}_{\mathcal{E}_1}$ stands for the fact that there exists no policy that has a cost less than a certain threshold for the initial state under the model

2. $\mathcal{P}_{\mathcal{E}_2}$ stands for the fact that no policy achieves the goal from the initial state with non-zero probability

3. $\mathcal{P}_{\mathcal{E}_3}$ stands for the fact that there exist no policy whose probability of achieving the goal from the initial state is above a certain threshold.

These properties classes correspond to a large set of specific explanatory properties and the exact threshold is determined by the specific query. For a specific model $\mathcal{M}$ and a threshold X, we will denote the fact that an instance of the property class $\mathcal{P}_{\cdot}$ holds for the threshold as $\mathcal{M} \models \mathcal{P}(X)$ (in the case of $\mathcal{P}_{\epsilon_2}$ X is limited to 0). While the properties themselves are described with respect to the initial state, one could generalize it to be considered against arbitrary states for the model (equivalent to considering whether the property holds in a model where the initial state is set to the new state in question).

A point to keep in mind is that even though the properties are defined over models, one could still use this convention to capture policy specific query. In such cases, we just need to consider a modified version of the model that only allows the policy in question. For example, if the user raises an alternative (by specifying actions to be performed in some of the states) and the system needs to explain why this alternative does not lead to the goal. Then the system can create a new model where the only action possible under the states mentioned are the ones provided by the user. Now the system needs to show how the likelihood of getting to the goal in this modified model is zero. Such a model-compilation strategy are particularly well suited to our setting, as it is quite unlikely that a user would provide one fully specified policy. At most, they partially specify some subset of the policy, a partially ordered set of subgoal or a specification in some formal language like LTL or LTL_f(Eifler *et al.*, 2020a). Such partial specifications maybe best thought of as imposing a constraint over the model in question (Sreedharan *et al.*, 2019b).

In this case, we have split the queries related to goal reachability into two distinct categories, even though, one could specify it as a comparison to a threshold. The reason why we chose not to do that, is two-fold;

1. Not only are queries about unsolvability a natural and commonly studied explanatory query type (Chapter 10), they are also a lot more accessible and more likely to be used by non-experts who may not be aware of or comfortable with the exact probability associated with the planning problem.

2. One can apply many more explanatory techniques to answer questions related to solvability, which is not necessarily applicable to questions that use direct comparison to probability threshold. For example, as we will see, the use of determinization or even the use of models with purely qualitative non-determinism is possible to answer queries related to solvability but doesn't apply to queries that explicitly use probabilistic thresholds.

Any possible contrastive query, which can be answered by decision-making system would correspond to one of these explanatory properties. following the conventions generally used in MDP solution strategies, we will establish a strict ordering between the properties, namely, $\mathcal{P}_{\epsilon_1} \prec \mathcal{P}_{\epsilon_2} \prec \mathcal{P}_{\epsilon_3}$.

12.5 Model Transformation Classes

In this section, we will look at some specific transformation classes. The classes were selected based on the fact that some restricted forms of these transformations were already studied in the wider XAI literature. For each class, we will introduce a general characterization of that transformation. Then look at a specific way to achieve it via syntactic transformation of the human readable model description (M^{D^*}) that takes polynomial time with respect to the size of the model description. That is, we will look at specific ways to generate updated model descriptions that will correspond to an MDP that satisfy the requirements of the corresponding transformation.

12.5.1 State Abstractions

The first transformation we could make use of is state abstraction that can help reduce the underlying state space of the MDP (Li *et al.*, 2006), which has been used for explaining deterministic plans (Sreedharan *et al.*, 2021d) and to summarize policies as by Topin and Veloso (2019). In particular, we will consider state aggregations that replace a set of states in the underlying MDP $\mathcal{M}^{R^*}$ with a single state (Van Roy, 2006). *In the example, it will be state abstraction that will let us ignore most of the state variables including those related to motion level constraints.* We will define state abstraction as

Definition 36. *For a given model $\mathcal{M} = \langle S, A, I, P, C, G\rangle$, we will define $\widetilde{\mathcal{M}} = \langle \widetilde{S}, \widetilde{A}, \widetilde{I}, \widetilde{P}, \widetilde{C}, \widetilde{G}\rangle$ is said to be an abstraction of $\mathcal{M}$, if there exists a surjective mapping from states and actions in $\mathcal{M}$ to $\widetilde{\mathcal{M}}$ (where $\mathcal{F}$ be the mapping), then $\forall s_1, s_2 \in S$ and for any $a \in A$, if $P(s_1, a, s_2) > 0$, then $P(\mathcal{F}(s_1), \mathcal{F}(a), \mathcal{F}(s_2)) > 0$.*

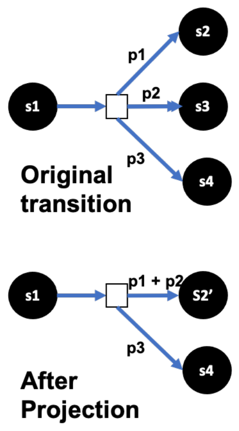


Figure 12.3: A Diagrammatic Representation of the Transformation Induced by the Abstraction Function on the Transition Probabilities. Here the Projection Operation Causes the States $s2$ and $s3$ to Collapse into a Single State $s2'$.

While state abstraction has been a popular topic of investigation in MDP planning/RL topics. We are particularly interested in its use to create sound representation for all the queries. Since we are looking at symbolic representation, it would be useful to have methods that create abstractions that are valid symbolic models. Specifically, we can make use of syntactic projections of the model descriptions, which have previously been used for generating heuristics (Klößner *et al.*, 2021).

12.5.2 State Abstractions Through Syntactic Projection

We will define the transformation as

Definition 37. *For a given model description $\mathcal{M}^D = \langle F^D, A^D, I^D, G^D \rangle$ and a set of propositional factors $\Lambda \subseteq F^D$ the syntactic projection (represented as a function $\mathcal{F}_{F\setminus\Lambda}$) results in a new model description $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M}^D) = \langle F^D \setminus \Lambda, A^{\mathcal{F}_{F\setminus\Lambda}}, I^D \setminus \Lambda, G^D \setminus \Lambda \rangle$. Where the new actions $A^{\mathcal{F}}_{F\setminus\Lambda}$ is given as follows: for each $a_i \in A^D$, there existing a corresponding action $\mathcal{F}_{F\setminus\Lambda}(a_i) = \langle prec_i \setminus \Lambda \rangle$ (To simplify discussion we will be overloading the notation $\mathcal{F}_{F\setminus\Lambda}$ to stand for any mapping from the components of the* *original model or description to those in the new model)*

$$\mathcal{F}_{F\setminus\Lambda}(E_i) = \{\langle add_i^j \setminus \Lambda, del_i^j \setminus \Lambda, p', c'\rangle | \ \langle add_i^j, del_i^j, p_i^j, c_j^i\rangle \in E_i\} \tag{12.1}$$

Where the new probability of the effect p' is given as

$$\begin{aligned} p' = \sum_{\langle add_i^k, del_i^k, p_i^k, c_i^k\rangle \in E_i \ \ ,where \ \ add_i^k \setminus \Lambda = add_i^j \setminus \Lambda \ \ and \ \ del_i^k \setminus \Lambda = del_i^j \setminus \Lambda} p_i^k \\ c' = min\{c_i^k | \langle add_i^k, del_i^k, p_i^k, c_i^k\rangle \in E_i \\ ,where \ add_i^k \setminus \Lambda = add_i^j \setminus \Lambda \ and \ del_i^k \setminus \Lambda = del_i^j\}\} \end{aligned} \tag{12.2}$$

So effectively the transformation removes every appearance of a fluent being projected out from the description, and if this results in mutually exclusive effects to be duplicated (including being empty), then only one version of the effect is included in the updated model, the probability of this effect becomes the sum of the individual effects and the cost of this effect becomes the minimum of the original effects. Figure 12.3, present a visual representation of the transformation.

Proposition 28. *One could create the transformed model description $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M}^D)$, in time polynomial over the size of the original model description and the number of factors in Λ.*

This complexity should be clear from the fact that one could do it by making two passes through the domain model. Once to remove the fluents in Λ and the second to merge duplicate effects. Now we will see that the MDP corresponding to updated model description $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M}^D)$, is in fact a valid state abstraction.

Proposition 29. *The model $\mathcal{M}^{\mathcal{F}_{F\setminus\Lambda}}$ represented by $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M}^D)$ is a state abstraction (per Definition 36) of the model $\mathcal{M}$. Additionally, for any states $i, j \in S$ for MDP $\mathcal{M}$, then $P(i, a_{a_i}^{mdp}, j) \leq P^{\mathcal{F}_{F\setminus\Lambda}}(\mathcal{F}_{F\setminus\Lambda}(i), a_{\mathcal{F}_{F\setminus\Lambda}(a_i)}^{mdp}, \mathcal{F}_{F\setminus\Lambda}(j))$ and $C(i, a, j) \geq C^{\mathcal{F}_{F\setminus\Lambda}}(\mathcal{F}_{F\setminus\Lambda}(i), a_{\mathcal{F}_{F\setminus\Lambda}(a_i)}^{mdp}, \mathcal{F}_{F\setminus\Lambda}(j))$, for states $\mathcal{F}_{F\setminus\Lambda}(i)$ and $\mathcal{F}_{F\setminus\Lambda}(j)$ in $\mathcal{M}^{\mathcal{F}_{F\setminus\Lambda}}$.*

Proof Sketch. First off $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M}^D)$ is a well formed model description, so there exists a unique model $\mathcal{M}^{\mathcal{F}_{F\setminus\Lambda}}$ induced by it. The surjective mapping from state space S of $\mathcal{M}$ to $S^{\mathcal{F}_{F\setminus\Lambda}}$ (Overloading the notation a bit, we will use $\mathcal{F}_{F\setminus\Lambda}$ to also capture this mapping) is given as

$$\mathcal{F}_{F\setminus\Lambda}(i) = \hat{i} \text{ if } \hat{s}_i = s_i \setminus \Lambda$$

It should be easy to verify that this is in fact a surjective function. The relationship between the probability functions and cost functions comes directly as a result of the transformation. $\square$

We can use the abstraction function to also define a relation among the states in the model description, such that $s_i \sim_{\mathcal{F}_{F\setminus\Lambda}} s_j$, if $\mathcal{F}_{F\setminus\Lambda}(s_i) = \mathcal{F}_{F\setminus\Lambda}(s_j)$. Note that this is an equivalent relation and thus help partition the state space into disjoint sets that map into the same abstract. If $\hat{s}_i$ is an abstract state for $\mathcal{F}_{F\setminus\Lambda}(\mathcal{M})$, the we will use the notation $\mathcal{F}^{-}_{F\setminus\Lambda}1(\hat{s}_i)$ to capture the quotient set for the relation $\sim_{\mathcal{F}_{F\setminus\Lambda}}$ that maps the state from the concrete model into the abstract state $\hat{s}_i$, i.e.,

$$\mathcal{F}^{-1}_{F\setminus\Lambda}(s) = \{s' | \mathcal{F}_{F\setminus\Lambda}(s') = s\}$$

Unless specified otherwise, the cost function for the concrete model are denoted as functions over just state indexes (for example $J(i)$), while that for the abstract model it is denoted with states where an abstraction function has been applied (for example $J(\mathcal{F}_{F\setminus\Lambda}(i))$).

Next we will detail some simple properties of the model and demonstrate the central properties we can use for explanation.

This takes us to the next result

Proposition 30. *Probability of a trace (a sequence of action control state tuples) from a given state to a goal state is either preserved or increases over the transformation.*

The result follows from earlier proposition.

Proposition 31. *Prob of an action causing a transition from an abstract state $\hat{i}$ to another abstract state $\hat{j}$, is equal to the sum of the probability of transitions from one of the states in $\mathcal{F}^{-1}_{F\setminus\Lambda}(\hat{i})$ to all the states in $\mathcal{F}^{-1}_{F\setminus\Lambda}(\hat{j})$ under that action.*

The result follows from the fact that given the declaration method, all the states that merged must have had similar effects, so choice of the state $\mathcal{F}^{-1}_{F\setminus\Lambda}(\hat{i})$ doesn't matter. The second part of the proposition follows from the transformation itself.

With this transformation in hand, we can show that the syntactic transformation results in a state aggregation We can now show the optimal cost function in the new model is lower than that of the original model.

Lemma 3. *For every state i in the model $\mathcal{M}$, $J^*(i) \geq J^*_{\mathcal{F}_{F\setminus\Lambda}}(\mathcal{F}_{F\setminus\Lambda}(i))$, where $J^*_{\mathcal{F}_{F\setminus\Lambda}}$ is the optimal cost function for $\mathcal{M}^{\mathcal{F}_{F\setminus\Lambda}}$.*

Proof Sketch. We can show this by initializing a cost function for the abstract model $J_{\mathcal{F}_{F\setminus\Lambda}}$ with cost from J^*, such that for any abstract state $\hat{i}$ as $J_{\mathcal{F}_{F\setminus\Lambda}}(\hat{i}) = \min_{i\in\mathcal{F}^{-1}_{F\setminus\Lambda}(\hat{i})}(J^*(i))$. If we now apply a bellman operator T, given propositions 29 and 31, we will have

$$TJ_{\mathcal{F}_{F\setminus\Lambda}}(\hat{i}) \leq J_{\mathcal{F}_{F\setminus\Lambda}}(\hat{i})$$

For P condition the bellman operator is still monotonic function (Bertsekas, 2005), and converges to the optimal cost function. Thus the oprimal cost function for $\mathcal{F}_{F\setminus\Lambda}(i)$ must be less than or equal to $J^*(i)$ □

Next in regards to the probability we can establish that

Lemma 4. *For every state i in the model $\mathcal{M}$, $P^*(i) \leq P^*_{\mathcal{F}_{F\setminus\Lambda}}(\mathcal{F}_{F\setminus\Lambda}(i)))$, where $P^*_{\mathcal{F}_{F\setminus\Lambda}}$ is the maxprob probability for the model $\mathcal{M}^{\mathcal{F}_{F\setminus\Lambda}}$.*

This result directly follows from Proposition 30 (a similar result was also established by Klößner *et al.* (2021)). With these two propositions we have established that state abstraction does in fact result in new models that underapproximated cost and overapproximates reachability. Thus establishing that *the syntactic projection described here results in a valid state abstraction per Definition 36 and the resultant transformation is sound with respect to all three explanatory properties.*

Theorem 7. *The domain description transformation $\mathcal{F}_{F\setminus\Lambda}$ corresponds to a universally sound transformation for any model that can be represented by a model description of the form $\mathcal{M}^{\mathcal{D}} = \langle F^{\mathcal{D}}, A^{\mathcal{D}}, I^{\mathcal{D}}, G^{\mathcal{D}}, C^{\mathcal{D}} \rangle$*

For the robot example, consider the original probabilistic effects of the mixing action, which says you might cake batter after mixing with probability 0.5, a cake batter with bubbles in it with probability 0.25, and with probability 0.25 no cake batter at all.

```
(probabilistic 1/2 (and (has-cake-batter))
      1/4 (and (has-cake-batter)
      (cake-batter-has-bubbles)))
```

After projecting out (cake-batter-has-bubbles) you get an effect that says the probability of having cake batter is 0.75

```
(probabilistic 3/4 (and (has-cake-batter)))
```

12.5.3 Problem Determinization

Note that the abstraction operation creates models with probabilistic effects unless the effects merge into a single effect. At least for some of the queries, the system could generate valid responses while using an optimistic determinization of the model that ignores all stochasticity of the model. *For example, in the robot chef example* *even if there were no undesired side-effects due to stochasticity (the 0.25 probability of the mixing action not forming a cake batter), the mix action could still not be executed as it has a missing precondition (has-cake-whisk).* One possible way to create such optimistic determinization could be to generate a new model where action with multiple stochastic effects is turned into multiple actions with deterministic effects. Determinization belongs to a larger class of transformations (which we will refer to as problem class simplification) that transforms the original problem to simpler decision-making problems for the sake of generating explanations. While there are some instances of problem class simplification for explanations for multi-objective explanations (cf. (Eifler *et al.*, 2020b)), we are unaware of any direct use of determinization to simplify explanations.

All Outcome Determinization

Definition 38. $\mathcal{M}^D = \langle F^D, A^D, I^D, G^D \rangle$ *a determinization is presented by an operator* $\mathcal{F}_\Delta$ *and generates a new model of the form* $\mathcal{F}_\Delta(\mathcal{M}^D) = \langle F^D, \mathcal{F}_\Delta(A^D), I^D, G^D \rangle$*. Where for every* $a_i = \langle prec_i, \{e_i^1, ..., e_i^k\} \rangle$*, there exists* k *actions in* $\mathcal{F}_\Delta(A^D)$ *such that* $a_i^j = \langle prec_i, add_i^j, del_i^j, 1, c_i^j \rangle$*, i.e., it generate the* j^{th} *effect with probability 1.*

Such determinization operations are sometimes referred to as all outcome determinization (Yoon *et al.*, 2007). Which is also closely connected to hindsight optimization techniques that have been studied in multiple fields including control theory (Yoon *et al.*, 2008). *The transformation could be used for both cost property* ($\mathcal{P}_{\epsilon_1}$) *and solvability property* ($\mathcal{P}_{\mathcal{E}_2}$)*.* Also note that for the class of models considered in this chapter the transformation of the model description can be carried out effectively.

Proposition 32. *The transformed model description* $\mathcal{F}_\Delta(\mathcal{M}^D)$ *can be created by performing a single pass through the original description* $\mathcal{M}^D$ *and the maximum number* *of actions in $\mathcal{F}_\Delta(\mathcal{M}^D)$ is upper-bounded by $K \times |A|$, where K is the maximum number of mutually exclusive effects that can appear in an action definition in A.*

The fundamental property of this determinization we can use here is the following

Proposition 33. *Let $\tau = \langle I, a_1, ..., a_k, g\rangle$ be the sequence of symbolic states and actions corresponding to a trace from the initial state to a goal state $g \in G$ with non-zero probability for the model defined by the description $\mathcal{M}^\mathcal{D}$. Then $A(\tau)$ (the sequence of actions appearing in τ) is a valid plan for the deterministic planning model $\mathcal{F}_\Delta$. That is when the sequence $A(\tau)$ is executed in I it will take you to the state g.*

This property directly comes from the form of the transformation is widely established in the determinization literature.

Lemma 5. *The cost of best possible plan in $\mathcal{F}_\Delta(\mathcal{M}^D)$ is guaranteed to be less than or equal to $J^*(I)$ for the model $\mathcal{M}$ corresponding to $\mathcal{M}^D$*

After all the lowest cost trace that can be sampled from any policy is a plan in the determinized model. As the cost of the policy should be higher than the cost of its lowest cost trace, it should be higher than the cost of the optimal plan in $\mathcal{F}_\Delta(\mathcal{M}^D)$.

Lemma 6. *If the problem $\mathcal{F}_\Delta(\mathcal{M}^D)$ is unsolvable then $P^*(I) = 0$.*

This follows directly from Proposition 33.

With Lemma 5 and 6 in place should be clear that this transformation could be used for both cost property ($\mathcal{P}_1$) and solvability property ($\mathcal{P}_2$). One can actually in fact make a stronger claim and show that it is even sound for ($\mathcal{P}_3$).

Theorem 8. *The domain description transformation $\mathcal{F}_\Delta$ corresponds to a universally sound transformation for any model that can be represented by a model description of the form $\mathcal{M}^\mathcal{D} = \langle F^\mathcal{D}, A^\mathcal{D}, I^\mathcal{D}, G^\mathcal{D}, C^\mathcal{D}\rangle$*

Proof. As mentioned earlier the soundness of the first two properties directly follow the earlier Lemmas. To see the soundness with respect to $\mathcal{P}_{\epsilon_3}$, note that $\mathcal{F}_{\Delta}(\mathcal{M}^{\mathcal{D}})$ is a well formed PPDDL model description, but one that only allows for deterministic transition. So in this model the maxprob probability must be either 1 or 0. This means that the only threshold for which $\mathcal{P}_{\epsilon_3}$ can be meaningfully established in the updated model is for threshold 0 (discounting 1 as it as $\mathcal{P}_{\epsilon_3}$ for threshold 1 a tautological statement). $\mathcal{P}_{\epsilon_3}$ limited to threshold 0 corresponds to $\mathcal{P}_{\epsilon_2}$, which per our earlier discussion the transformation $\mathcal{F}_{\Delta}$ is already sound for. Thus establishing the fact that $\mathcal{F}_{\Delta}$ is universally sound. $\square$

With regards to the example, such transformation will allow us to ignore the probability of mix action failure when giving the explanation.

12.5.4 Problem Decomposition

Even after applying all the above transformations, the plans generated from the resultant model could be extremely long. One way to address would be to decompose the original task into smaller subtasks. *For example, in the case of the robot chef, rather than talking about the problem of baking cake it can focus just on explaining the subproblem of making the cake batter.* In this section, we introduce problem decomposition with respect to an initial state that focuses on subproblems that reuse states and actions of the original problem and is either cheaper or it is easier to achieve the goal

Definition 39. *Given an atomic MDP $\mathcal{M} = \langle S, A, P, I, C, G \rangle$ with an optimal cost J^*, an MDP $\mathcal{M}' = \langle S, A, P, I, C, G' \rangle$ with an optimal cost J', is said to be a subproblem if it is guaranteed that $J'(I) \leq J^*(I)$ (where the decomposition is called cost based decomposition) or $P'(I) \leq P^*(I)$ (where the decomposition is called reachability-based* *decomposition).*

For queries related to $\mathcal{P}_{\mathcal{E}_1}$ if we can establish that the cost of the subproblem under cost-based decomposition is greater than the limit, then that automatically establishes that the original problem should be worse. For queries related to reachability, we can look for similar arguments for a subproblem under reachability-based decomposition. Now the question is how can we find such decompositions.

Subgoals

For goal-directed problems with an initial state, a natural idea could be subgoals, where subgoals are representations of intermediate states that the policy may need to achieve before getting to the final goal. A specific subgoal that could be useful here is landmarks, which were recently adapted for SSPs by Sreedharan *et al.* (2020c) (the specific methods will be discussed in Chapter 21). These are propositional facts that need to be satisfied by any path from the initial state to goal with non-zero probability. While that paper introduced these facts as a way to summarize policies, we use them to decompose problems and form simpler problems. Such landmarks can be automatically extracted from the model description $\mathcal{M}^{D^*}$ For example, in the case of the baking example, has-cake-batter is a landmark for the goal bake-cake. *A problem decomposition using landmarks as the new goal is both a cost and reachability decomposition and is sound for all three explanatory properties.*

Lemma 7. *If $f \in F^D$ is a fact landmark for the MDP corresponding $\mathcal{M}^D = \langle F^D, A^D, I^D, G^D \rangle$, then the model $\mathcal{F}_{\mathbb{D}}(\mathcal{M}^D) = \langle F^D, A^D, I^D, f \rangle$ is both a cost-based decomposition and probability-based decomposition. i.e., $J^*_{\mathcal{F}_{\mathbb{D}}}(I) \leq J^*(I)$ and $P^*_{\mathcal{F}_{\mathbb{D}}}(I) \leq P^*(I)$, where $J^*_{\mathcal{F}_{\mathbb{D}}}$ and $P^*_{\mathcal{F}_{\mathbb{D}}}$ are the optimal cost function and MAXPROB probabilities for the model corresponding to the description $\mathcal{F}_{\mathbb{D}}(\mathcal{M}^D)$*

Proof Sketch. Let $\tau = \langle I, a_1, ..., a_k, g \rangle$ be the sequence of symbolic states and actions corresponding to a trace from the initial state to a goal state $g \in G$ with non-zero probability for the model defined by the description $\mathcal{M}^{\mathcal{D}}$. From the definition of landmarks, we know that there must be state s_f in the sequence τ such that $f \in s_f$. Therefore any trace with nonzero probability can be decomposed into a prefix corresponding to the sequence that leads to a landmark state and then the sequence from landmark state to the final goal. Additionally we can see that the probability of this trace prefix must be greater than or equal to the probability of the full trace. Thus for all the traces sampled from a given policy the total probability of getting to all landmarks states must be higher than that of reaching the final goal. We can use a similar reasoning to show that the total expected cost of reaching the landmark states must be less than or equal to the cost of reaching the final goal state. □

The above lemma establishes the fact that problem decomposition through landmarks underapproximates costs and overapproximates reachability, thus its valid for all three explanatory properties.

Theorem 9. *The domain description transformation $\mathcal{F}_{\mathbb{D}}$ corresponds to a universally sound transformation for any model that can be represented by a model description of the form $\mathcal{M}^{\mathcal{D}} = \langle F^{\mathcal{D}}, A^{\mathcal{D}}, I^{\mathcal{D}}, G^{\mathcal{D}}, C^{\mathcal{D}} \rangle$*

Now coming to the complexity of the transformation, once given a fact landmark the model transformation can be performed quite effectively.

Proposition 34. *For a given fact landmark f and a model description $\mathcal{M}^{\mathcal{D}}$, the domain description transformation $\mathcal{F}_{\mathbb{D}}$ can be performed in constant time.*

This is because you just need to replace the goal description in $\mathcal{M}^{\mathcal{D}}$ to form the transformed description. Now coming to the complexity of generating a landmark, the problem of finding landmark in general is PSPACE-Complete (Hoffmann *et al.*, 2004). However, there are classes of landmarks which can be generated more effectively. One class of such landmarks is the causal landmarks which can be generated in time polynomial to the size of the model description (Zhu and Givan, 2003).

12.5.5 Local Approximation

The next transformation we will consider is that of local approximations. Local approximations (Ribeiro *et al.*, 2016), have been successfully used in the context of explaining machine learning decisions. The basic premise being that the model being used for explanation need not accurately reflect the entire decision-space, but only the parts that are relevant to the current query. One could translate the same intuition into the sequential decision-making settings, and look at creating simpler representations of the model that focus only on a part of the transition system. *Revisiting our motivating example, through local approximation we can establish the facts that the robot doesn't need to talk about any of its skills unrelated to cooking or baking cakes. Based on the fact that it is in a home, it can also figure out that it can skip providing any information to the user about it's ability to use industrial mixers to make cake batter.*

We can describe a local approximation (represented by the transformation $\mathcal{F}_{\mathcal{L}}$) for a set of states $\hat{S}$ and a set of actions $\hat{A}$ as a function that generates a new model that conserves the transition probabilities and cost functions related to states and actions that appear in $\hat{S}$ and $\hat{A}$.

Definition 40. *For a given model $\mathcal{M} = \langle S, A, P, C, I, G \rangle$ and a subset of states and actions $\hat{S} \subseteq S$ and $\hat{A} \subseteq A$, a well formed MDP model $\mathcal{M}' = \langle S', A', P', C', I', G' \rangle$, is said to be a* ***local approximation*** *(denoted as $\mathcal{F}_{\mathcal{L}}(\mathcal{M}) = \mathcal{M}'$) if there exists a mapping from $\hat{S}$ to S' and from $\hat{A}$ to A' (with a slight abuse of notation we will use*

$f_{\mathcal{L}}$ for both the state to state and action to action mapping), such that for any given states $i, j \in \hat{S}$ and an action $a \in \hat{A}$, we have $P(i,a,j) = P'(f_{\mathcal{L}}(i), f_{\mathcal{L}}(a)), f_{\mathcal{L}}(j))$, $C(i,a,j) = C'(f_{\mathcal{L}}(i), f_{\mathcal{L}}(a), f_{\mathcal{L}}(j))$ and there exists a mapping from $\hat{S} \cap G$ to G'.

This is a very permissive definition and not all instantiation of the transformation would necessarily lead to transformations that would lead us to ones that may preserve explanatory properties. So before, we consider an instance of the transformation, let us consider a specific subclass of the transformation, one that focuses on cases where the subset of states and actions consider form a closed transition system ,i.e., all states reachable from $\hat{S}$ under the policy is a subset of $\hat{S}$

Definition 41. *A set of states $\hat{S}$ and set of actions $\hat{A}$, is said to be **closed for a policy** π, if*

1. *Execution policy in $\hat{S}$, will never lead to a transition to a state outside the set, i.e., $\forall j \in S \setminus \hat{S}$ $\nexists i \in \hat{S}$ and $a \in \hat{a}$ such that $P(i,a,j) > 0$*
2. *The set $\hat{A}$ covers all actions assigned to states in $\hat{S}$, under the policy π, i.e., $\forall i \in \hat{S}, \pi(i) \in \hat{A}$*

We will now see how any local transformation defined over a closed set, will result in universally sound transformation. More formally, we can state this as;

Proposition 35. *If the sets $\hat{S}$ and $\hat{A}$ are closed for a policy π, then $\forall i \in \hat{S}, J'^{*}(f_{\mathcal{L}}(i)) \leq J^{\pi}(i)$ and $P'^{*}(f_{\mathcal{L}}(i)) \geq P^{*}(i)$, where J'^{*} and P'^{*} are the optimal cost function and the maxprob probability for the model $\mathcal{F}_{\mathcal{L}}(\mathcal{M})$.*

Proof sketch. This follows directly from the fact that the transformation introduces no new transitions for states and actions that are part of the set $\hat{A}$ and $\hat{S}$. Moreover the transformation conserves both cost and probabilities for those states and actions.

This means there exists multiple policies for $\mathcal{F}_{\mathcal{L}}(\mathcal{M})$ which has the same value and probability for states in $\hat{S}$ as the original policy π. This means the optimal policy and maxprob policy should lead to policy that are more cheaper and with higher likelihood of getting the goal. Note that this doesn't need to be equal as the local approximations allows for the fact that there may be new actions now applicable in the states that are part of $\hat{S}$. □

Proposition 35 ensures that local approximation results in models that underapproximates costs and overapproximates reachability and we can now state the.

Theorem 10. *The transformation $\mathcal{F}_{\mathcal{L}}$ for the given sets $\hat{S}$ and $\hat{A}$ is a universally sound transformation for a model $\mathcal{M}$, if,*

1. *$\hat{S}$ and $\hat{A}$ are closed with respect to an optimal policy and a MAXPROB policy.*
2. *$I \in \hat{S}$ and $\mathcal{F}_{\mathcal{L}}(I) = I'$, where I' is the new initial state in the model $\mathcal{F}_{\mathcal{L}}(\mathcal{M})$.*

The fact that the initial state is in $\hat{S}$ (and it's corresponding image remains the new initial state) and that it satisfies the requirement for Proposition 35 means that all three explanation properties $\mathcal{P}_{\epsilon_1}$, $\mathcal{P}_{\epsilon_2}$ and $\mathcal{P}_{\epsilon_3}$.

Reachability Analysis For Local Approximation

One way to create a local approximation is to perform reachability analysis to remove actions and fluents guaranteed to be not reachable from the initial state. To identify the non-reachable fluents we will consider the delete-relaxation of the all outcome determinization of the original model and try to identify the reachable fluents and actions by building a relaxed planning graph (Hoffmann and Nebel, 2001). The fluents and actions not present in the planning graph are removed from the model description. We will refer to the model description formed through this procedure as $\mathcal{F}^{+}_{\mathcal{L}}(\mathcal{M}^{\mathcal{D}})$, where + denotes the fact that the local approximation relies on a delete relaxation of the model. First thing to note is that $\mathcal{F}_{\mathcal{L}}^{+}$, can be formed rather efficiently. In particular, we have

Proposition 36. *The new model description $\mathcal{F}_{\mathcal{L}}^{+}(\mathcal{M}^{\mathcal{D}})$ can be created in time polynomial to the size of the original model description $\mathcal{M}^{\mathcal{D}}$.*

Proof. This follows from the fact that an all outcome determinization can be generated in polynomial time. The formation of the relaxed planning graph and subsequent identification of unreachable fluents and actions can again be performed in time polynomial to the size of the determinized model. With the fluents and actions to be removed identified, one can form the model description $\mathcal{F}_{\mathcal{L}}^{+}(\mathcal{M}^{\mathcal{D}})$ again in polynomial time. $\square$

Now the fact we need to show is that this model description transformation actually correspond to a local approximation for a certain subset of states and actions and moreover this approximation meets the requirements for Theorem 10. In particularly we will see that the transformation will create a local approximation defined over the original MDP where the subset of states and actions considered correspond to set of states that can be formed by the remaining fluent and remaining actions and it is in fact closed. More formally, we can state

Proposition 37. *Let $\mathcal{M}$ be the model corresponding to the description $\mathcal{M}^{D}$ and let $\mathcal{F}_{\mathcal{L}}^{+}(\mathcal{M}^{\mathcal{D}}) = \langle \hat{F}^{\mathcal{D}}, \hat{A}^{\mathcal{D}}, I^{\mathcal{D}}, G^{\mathcal{D}}, C^{\mathcal{D}} \rangle$ be the newly formed transformed model such that, $\hat{F}^{\mathcal{D}} \subseteq F^{\mathcal{D}}$ and $\hat{A}^{\mathcal{D}} \subseteq A^{\mathcal{D}}$. Then the MDP $\mathcal{M}'$ corresponding to the description $\mathcal{F}_{\mathcal{L}}^{+}(\mathcal{M}^{\mathcal{D}})$ is a local approximation of the model $\mathcal{M}$, defined over the state and action subset $\hat{S}$ and $\hat{A}$, such that*

1. *$\hat{S}$ corresponds to the state defined by $2^{|\hat{F}^{\mathcal{D}}|}$ and $\hat{A}$ correspond to actions in $\hat{A}^{\mathcal{D}}$ (both of which are subsets of S and A for the model $\mathcal{M}$)*

2. $\hat{S}$ and $\hat{A}$ meets the requirements specified in Theorem 10

Proof Sketch. Note that the mapping here is an identity mapping from states and actions in $\hat{S}$ and $\hat{A}$ to those in the model $\mathcal{M}'$. From the construction of the relaxed planning graph, every fluent that is true in initial state will be conserved and thus I must be part of $\hat{S}$. The fact that $\hat{S}$ and $\hat{A}$ are closed comes from the fact that we are considering a delete relaxation of an all outcome determinization. This means if the process removes a state from consideration (i.e. removes a fluent f that is part of the state), then there exists no non-zero probability trace that can reach that state from the initial state. Thus the states are closed under any policy not just optimal or MAXPROB ones. Similarly for actions, the procedure will never remove any action that is applicable for a state that is part of $\hat{S}$ and thus must be closed. □

Proposition 37 thus establishes the fact that $\mathcal{F}^{+}_{\mathcal{L}}$ also corresponds to a universally sound transformation.

12.6 Computational Model

One of the remaining components that we have yet to discuss in detail in this chapter is the function capturing the computational burden placed on the human to solve a specific task, i.e., the function $Comp_H(\cdot,\cdot)$. An exact characterization of $Comp_H(\cdot,\cdot)$ would be hard, since it would require accurately capturing the inferential capabilities of the human. However, one could still use a number of simpler representations of $Comp_H(\cdot,\cdot)$ to calculate useful representations, some of the possible choices here include

- Using computational model with psychological fidelity - This could correspond methods like, the ones that leverage computational implementation of psychological models like prospect theory (Kahneman and Tversky, 2013), use of decision-making algorithms that make use of limited memory (Nikolaidis *et al.*, 2016), use of various models of bounded-rationality including ones from behavioral game theory like the finite nested rational model (Wright and Leyton-Brown, 2017), etc. However, works in these models are still in preliminary stages and we are unaware of any existing techniques that could directly be applied to our problem framework.

- Directly learning $Comp_H$ - Another possibility might be to directly learn the $Comp_H$ function from data collected from human subjects. While we are unaware of any method that can currently learn the function of the form we require, some preliminary work in this direction include those presented by Zhi-Xuan *et al.* (2020).

- Using Computational Proxies - While we may not have access to $Comp_H$, we could directly measure the hardness of establishing the explanatory property using exact and sound methods and measuring the time. While this isn't expected to be equal to the actual inferential burden faced by the user, we could use this as an approximation of the exact value. In addition to exact time taken by an automated reasoner, one could also use other measures like the size of description, length of the most likely policy trace, etc.

- Using Human-Subject Studies to Establish Effectiveness of Individual Transformations - Another method might be to not directly learn $Comp_H$, but instead identify any preference use might have on various types of transformation that may be applicable for simplifying a given model. Then one could leverage the preference between the transformations to identify solutions that may be preferred by the user.

In this chapter, we will mostly focus on the latter two strategies, wherein we will look at generating simplified model that are simpler with respect to an automated reasoner. We also introduce an ordering between the various transformations to be applied (where the more preferred transformation are applied first). This ordering between the transformation are determined through a user subject study.

12.7 User Study

We performed an ablation study to establish the effectiveness of individual transformations to help users understand the explanation.
Study Objective: Identify the utility provided by the four transformation classes. We will compare the effectiveness of an explanation defined over a model containing all four transformations, against one where one of the transformation operations is missing.
We will also try to verify the secondary hypothesis
H1: Does the presence of stochasticity in explanatory information reduce the effectiveness of an explanation
We recruited a total of 150 participants through Prolific (Prolific, UK, 2021). The participants were paid \$3.30 for 15 minutes. The study took the form of a timed quiz (the quiz is automatically submitted at 15 minutes) where they read an explanatory dialogue and were asked to answer questions related to the explanation. There was also a bonus of \$5 offered to the top two fastest participants from a group, who get all the answers right (thus ensuring that people optimize for both completion time and correctness). We required that the participants were fluent in English and it was their first language. 75% of the participants were from the US. For the maximum education degree completed: 28% of all the participants who attempted the test (including those who dropped out in the middle) reported having a Bachelor's degree, 21% a high school degree, 17% having some college credits, and 15% having a master's de-

	all	**-determin**	**-locl-apprx**	**-decomp**	**-abs**
# Filtered	27	27	27	24	24
% satisfaction	85.18	81.48	81.48	66.66	66.66
Participants with correct answers	22	23	24	17	20
Avg Time Taken	237.59 ∓26	281.45∓48	299.6∓46	249.06 ∓40	391.75∓73
T-test against **all**	-	0.096	0.0288	0.343	0.0017

Table 12.1: Summary of User Study Results. Average Time Reported Is with 95% Confidence Interval. Last Row Reports the P-value Calculated from a Two Tailed Homoscedastic T-test

gree. In the group corresponding to **all-but-determinization**, 88% of participants reported that they understood probabilities. The study was performed on a simple travel planning domain, where the goal is to let the person board a flight. The domain consists of 11 actions, four of which have stochastic effects. We start with a model that contains all the transformations. It projects out all but six propositional fluents, has a subgoal of getting to the terminal, is determinized, and uses a regression-based local-approximation method to prune out actions that can't possibly contribute to the goal of getting to the terminal. The actual explanatory dialogue consists of a system stating the estimated time taken to reach the final destination (time being a stand-in for cost) and the user in the dialogue asking why it can't be done in an hour. The explanation consists of a description of the corresponding simplified model (generated by filling templates with descriptions of the propositions) and a single execution trace as the explanatory witness. The user is asked to read this dialogue and asked to answer a series of questions. Two questions of particular interest are a filter question that just checks whether the participant read the instructions and then a question that tests whether the user understood the explanation, by asking how they could speed up the travel plan. For the second question, the participant was given five options, only two of which are correct answers. One of which requires the participant to understand the current plan and the second requires them to reason about an alternate initial state. We ignored any participant who got the filter question wrong.

The five conditions we considered were (A) **all** - A condition consisting of a model that includes all four transformations, here the main explanatory text (model description plus explanatory witness) included 1493 characters. (B) **all-but-determinization** - as the name suggests all transformations except determinization are applied, the explatory text included 1805 characters. (C) **all-but-local-approximation** - the explanatory text includes 2274 character. (D) **all-but-decomposition** - the explanatory text includes 2429 characters. This group has the same model description as **all-but-local-approximation**, but the explanatory witness is longer (as opposed to just talking about reaching terminal A the trace takes you all the way to boarding the flight). (E) **all-but-abstraction** - the explanatory text includes 3,211 characters. We considered 30 participants per condition.

We measure the effectiveness of explanation both on a subjective level (do people find the explanation satisfying?) and on an objective one (do people find the explanations helpful?). We measure the former by directly asking the participants if they found the explanations satisfying and the latter by checking whether they found correct answers and how long they took to find those answers. Table 12.1, presents the results of the user study. The first interesting result to note is the fact that the transformation, whose removal makes the most difference seems to be the state abstraction. It causes a marked reduction in both subjective and objective front. The p value calculated from the t test (can be roughly interpreted as the probability that the samples come from the same population) is 0.0017. Which is lower than standard significance levels ($\alpha = 0.05$) used to establish statistical significance. Apart from this primary observation, the results also point to a number of other phenomena.

In regards to H1, we see that the use of probabilities seems to introduce a drop in the number of people who reported they are satisfied with the explanation, as compared to the condition **all** and more interestingly comparable to a condition that was actually more verbose (**all-but-local-approximation**). We also see an increase in time taken for probabilities against **all** and also against a more verbose condition (**all-but-local-approximation**). All these seem to point to the fact that the H1 holds even in this simple scenario where the abstraction and local approximation removes half of the probabilistic actions. Another interesting point is the difference between **all-but-local-approximation** and **all-but-decomposition**. They are near identical in the size of the explanation, but in one, you are asked to reason about a longer horizon (i.e. **all-but-decomposition**), and in the other, you have unnecessary actions that don't contribute to the goal being explained against (i.e. reaching the terminal). We see that in **all-but-local-approximation** a lot of people are able to solve the problem but at the cost of much higher time (the statistical test between the time taken under condition **all** and **all-but-local-approximation** is again under the significance level of 0.05). One possible explanation for the higher time could be the selectivity principle that has been discussed in that extra time need to be spent sifting through the unrelated details (Miller, 2017a). While in the case of **all-but-decomposition**, there are no irrelevant details but possibly the longer planning horizon is leading to people overlooking or making mistakes during reasoning.

Takeaways. The results show the utility of abstraction, particularly when it results in a considerable reduction in model size. Though one can't just rely on model-description sizes, as the comparison between **all-but-local-approximation** and **all-but-decomposition** shows there are multiple other factors that could influence the utility of a transformation.

12.8 Stratified Search

Now let us return back to the question of how to effectively generate the transformation sequences. Even if we limit our transformation classes to the one we discussed in Section 12.5, a search for the minimal transformation sequence using Algorithm 8 would be an expensive problem. For one the search space could be large and even if one were to introduce a more informed version of the search any search heuristic we employ will have to compute the effect of a transformation on the computational hardness of the problem. A more useful approach may be to set up a hard priority between the transformation classes. If we are able to set a primary transformation class, then we can directly try to find the maximal number of transformations we can apply from that class. The other transformations need only be considered to the degree that they can be applied to the models generated by applying a maximal sequence of primary transformations possible. An excellent candidate for primary transformation is state abstractions, which are demonstrated both by our user studies (Section 12.7) and makes sense from a computational point of view. After all, removing each binary state fluent reduces the state space by half. Among the most abstract models that support the given explanatory query, we can look at applying the other transformations provided they improve the secondary characteristics under consideration.

However, one could also additionally exploit the specifics of the transformations to get additional improvement in search. Starting with state abstraction transformation ($\mathcal{F}_{F\setminus\cdot}$), there are two immediate properties we can exploit, namely the fact that the transformation is *commutative* and *compositional*, or more formally

Proposition 38. *For any model $\mathcal{M}^{\mathcal{D}}$ and for any proposition set $\hat{F}_1 \subseteq F$ and $\hat{F}_2 \subseteq F$, we have*

1. *$\mathcal{F}_{F\setminus\hat{F}_1}(\mathcal{F}_{F\setminus\hat{F}_2}(\mathcal{M}^{\mathcal{D}}))$ and $\mathcal{F}_{F\setminus\hat{F}_2}(\mathcal{F}_{F\setminus\hat{F}_1}(\mathcal{M}^{\mathcal{D}})$, i.e. the state abstraction transformation is commutative*

2. *$\mathcal{F}_{F\setminus\hat{F}_1}(\mathcal{F}_{F\setminus\hat{F}_2}(\mathcal{M}^{\mathcal{D}})$ and $\mathcal{F}_{F\setminus(\hat{F}_1\cup\hat{F}_2)}(\mathcal{M}^{\mathcal{D}})$, i.e. the state abstraction transformation is compositional*

Proposition 38 follows directly from the definition and implies that we can apply state abstraction one fluent at a time and can be applied in any order.

Next moving onto the next three transformations, we see another fascinating property. Namely that the order in which the determinization ($\mathcal{F}_{\Delta}$) and subgoal decomposition ($\mathcal{F}_{\mathbb{D}}$) is applied doesn't matter. Similarly, the applying determinization and local approximation via reachability analyses ($\mathcal{F}^{+}_{\mathcal{L}}$) in any order also results in the same model. However, applying local approximation after subgoal decomposition is always guaranteed to result in removal of more elements, more formally,

Proposition 39. *For any model $\mathcal{M}^{\mathcal{D}}$*

1. *$\mathcal{F}_{\Delta}(\mathcal{F}_{\mathbb{D}}(\mathcal{M}^{\mathcal{D}})) = \mathcal{F}_{\mathbb{D}}(\mathcal{F}_{\Delta}(\mathcal{M}^{\mathcal{D}}))$ and $\mathcal{F}_{\Delta}(\mathcal{F}^{+}_{\mathcal{L}}(\mathcal{M}^{\mathcal{D}})) = \mathcal{F}^{+}_{\mathcal{L}}(\mathcal{F}_{\Delta}(\mathcal{M}^{\mathcal{D}}))$*

2. *If $M' = \mathcal{F}^{+}_{\mathcal{L}}(\mathcal{F}_{\mathbb{D}}(\mathcal{M}^{\mathcal{D}}))$, $M'' = \mathcal{F}_{\mathbb{D}}(\mathcal{F}^{+}_{\mathcal{L}}(\mathcal{M}^{\mathcal{D}}))$, then we can guarantee that $F' \subseteq F''$ and $A' \subseteq A''$.*

Proof. The first result follows from the facts that (a) determinization will not change the goal description and subgoal decomposition only changes the goal description and thus are independent of each other and (b) the reachability is already calculated on an all outcome determinization of the model. For the second result, remember that the relaxed planning graph is always only built until the goal is achieved, as such using a subgoal that is easier to reach could let us prune out more actions and fluents. $\square$

Algorithm 9 presents the pseudo-code for the stratified search. The algorithm starts with the most abstract model (which only includes the goal fluents) and then searches for the minimal number of fluents to be introduced into the model so that the query property $\mathcal{P}$ is satisfied. Let $Comp_{sys}$ be the computational proxy we are using. The successor procedure will only consider generating a model generated through

Algorithm 9 Stratified Search

```
1: procedure SEARCH
2:     Input:  $\mathcal{M}^D, \mathcal{P}, \mathbb{F}, \mathcal{C}$
3:     Output: Updated model description $\hat{\mathcal{M}}^D$
4:     Procedure:
5:     $\lambda = F^D \setminus G^D$
6:     curr_model = $\mathcal{F}_{F \setminus \lambda}(\mathcal{M}^D)$
7:     Fringe.push(curr_model) (Where Fringe is a Queue)
8:     $\mathcal{C}_{min} = \infty$
9:     Abs_min = $|F^D|$
10:    while Fringe is not empty do
11:        curr_model = Fringe.pop()
12:        Property_holds = False
13:        if test_property(curr_model,$\mathcal{P}$) then
14:            Property_holds = True
15:            if Abs_min is less than number of fluents in curr_model then
16:                set Abs_min to number of fluents in curr_model
17:            $\mathcal{C}_{new} = Comp_{sys}$(curr_model, $\mathcal{P}$)
18:            if $\mathcal{C}_{min} > \mathcal{C}_{new}$ then
19:                $\mathcal{C}_{min} = \mathcal{C}_{new}$
20:                best_model = curr_model
21:        Fringe.push(successor(curr_model, $\mathbb{F}$, Abs_min, Property_holds))
22:    if test_property($\mathcal{F}_{\Delta}$(best_model)) then
23:        best_model ← $\mathcal{F}_{\Delta}$(best_model)
24:    if test_property($\mathcal{F}_{\mathbb{D}}$(best_model)) then
25:        best_model ← $\mathcal{F}_{\mathbb{D}}$(best_model)
26:    if test_property($\mathcal{F}^{+}_{\mathcal{L}}$(best_model)) then
27:        best_model ← $\mathcal{F}^{+}_{\mathcal{L}}$(best_model)
       return best_model
```

non-concretization transformation (i.e. transformation that adds new fluents in the model) if the number of fluents in the model is equal to Abs_min and the property holds in the model. Abs_min starts initialized to the total number of fluents in the original model, but as soon as an abstract model is found then Abs_min gets set to the number of fluents in the model. After that, the search will keep concretizing the models to the same level of abstraction (in terms of the number of fluents) and then try to see if further transformations help improve the secondary characteristic being optimized for.

12.9 Synthetic Experiments

Here we will look at the ability of Stratified search to generate simplified models. Here the performance of the simplified model is measured by the time taken to solve the problem using standard solvers and the size of the description. We considered three different solvers, a MAXPROB solver (Klößner *et al.*, 2021), an implementation of LAO* solver (Hansen and Zilberstein, 2001) for SSP problems (for the cost queries) and the fast-downward planner (ran with A* search and LM-cut heuristic) for deterministic problems (Helmert, 2006). The problems were tested on problems from IPPC problems from 2006 and 2008 (Bryce and Buffet, 2008) and some additional problems for unsolvability. We tested problems corresponding to all three property classes. For property corresponding to cost (i.e. $\mathcal{P}_{\mathcal{E}_1}$) we only consider domains which is guaranteed to have proper policies (i.e. goal achievement probability is 1). For each domain considered, we selected only problems that could be solved by the solvers within 30 minutes and was appropriate for the specific property (i.e. was unsolvable for $\mathcal{P}_{\mathcal{E}_2}$ and had the max probability of less than 1 for $\mathcal{P}_{\mathcal{E}_3}$). Since we are unaware of any unsolvability benchmarks (for property $\mathcal{P}_{\mathcal{E}_2}$) for probabilistic planning, we took a deterministic domain (from Hoffmann *et al.* (2014)) and turned them into probabilistic domains by randomly changing some of the add effects to stochastic effect with probability 0.5. For $\mathcal{P}_{\mathcal{E}_1}$) we created the query by considering the cost threshold to be 5 (note the SSP solver ignores action cost) and for $\mathcal{P}_{\mathcal{E}_3}$) we

Property	Domain	Problem Count	Original Prob Size	Average Simplified Prob Size	Average Solver Time of Original Model	Average Solver Time of Simplified Model
$\mathcal{P}_{\mathcal{E}_1}$	Blocksworld	5	67.4	25.4	1.647	1.22
	Elevators	12	75.36	7.86	58.01	1.09
	Zenotravel	5	81	26.2	534.06	1.26
$\mathcal{P}_{\mathcal{E}_2}$	Bottleneck	12	145	116.25	60.38	1.62
$\mathcal{P}_{\mathcal{E}_3}$	Horizon Constrained Blocksworld	5	115	63.2	1.03	0.88
	Dirve	15	421.53	51	0.06	0.13
	Exploding Blocksworld	9	77.44	19.11	53.59	0.17

Table 12.2: The Results of The Simulated Experiments

used a probability threshold of 1. We also used a constrained version of Blocksworld domain for $\mathcal{P}_{\mathcal{E}_3}$ that was introduced by (Klößner *et al.*, 2021). As clear from Table 12.2 shows in everyday domain the transformation results in a smaller domain and in all but constrained blocksworld domain, results in shorter solution time. All experiments were run on an Ubuntu 14.04 machine with 12 cores and 64 GB Ram.

12.10 Related Works

While XAI as a field has been getting a lot of attention (Lakkaraju *et al.*, 2020), explaining sequential decisions is relatively under explored. Though there is a growing recognition that explaining sequential decisions presents unique challenges. Particularly there has been a number of recent works that have looked at explaining visual RL agents (for a recent survey for RL explanations in general please refer to (Alharin *et al.*, 2020)). A lot of explanation works from the RL space seem to focus either on feature attribution explanation (cf. (Greydanus *et al.*, 2018; Huber and André, 2019)) or generating policy summaries (cf. (Topin and Veloso, 2019; Hayes and Shah, 2017; Amir and Amir, 2018; Lage *et al.*, 2019; Koul *et al.*, 2019)). Most of the works that try to answer 'why' a specific decision was made, focus on the choice of a specific action at a state (while keeping the rest of the policy the same) (c.f (Lin *et al.*, 2021; Khan *et al.*, 2009; Juozapaitis *et al.*, 2019)) rather than contrasting the current policy/behavior over whole behaviors/alternate policies. In regards to explanations for model-based sequential decisions, Chakraborti *et al.* (2020) presents a rather comprehensive survey (the thesis also presents a parallel survey in Chapter 22). As mentioned most work in model simplification for explanations comes from deterministic planning community. With state abstraction being a popular method investigated by (Sreedharan *et al.*, 2021d) and (Sreedharan *et al.*, 2019b) (many placed in the framework of model reconciliation (Chakraborti *et al.*, 2017)). (Sreedharan *et al.*, 2020b) also tried to use the ideas from these papers in the context of FOND planning (Cimatti *et al.*, 2003). (Sreedharan *et al.*, 2019b) also talks about finding the first unsolvable subgoal. There are also works that try to map inscrutable black box models into models expressed in terms of interpretable features or force RL algorithms to use interpretable features. Representative works in this direction include (Lin *et al.*, 2021; Sreedharan *et al.*, 2022c; Waa *et al.*, 2018). Though unless these new features are hand-selected they don't necessarily have to lead to simpler models. In addition to XAI works, the transformation methods used here also have roots in generating model approximations to speed up planning and generating heuristics. Some relevant works include those by Srivastava *et al.* (2016); Van Roy (2006); Klößner *et al.* (2021); Givan *et al.* (2000) for abstraction, Richter and Westphal (2010) for landmarks, Yoon *et al.* (2007) for determinization.

One of the points just referred to, but not expanded upon was the use of model class simplification transformation. One example is the method discussed by Eifler *et al.* (2020a), where the generate contrastive explanation generation for the oversubscription planning problem. Here the explanation provides the various constraints between the different possible objectives. We could see these works as performing model class simplification, where they effectively convert a multi-objective planning problem into a constraint satisfaction problem where there are specified constraints among various objectives. A similar approach was also followed by Sukkerd *et al.* (2018, 2020).

Table 12.3 and 12.4, presents some example works from the explanation for sequential-decision-making literature that uses explanatory witnesses and the type of information provided by each. Note that most works that use some form of proof-based explanatory witness do so by skipping some information from the proof. For example, to establish the choice of one action over another they may report the Q values of the other actions without establishing why the Q values of the other actions have those specific values. As such we have referred to the information generated from these works as Abstract proof.

12.11 Concluding Remarks

This chapter presents a framework for generating a simplified model representation for the purposes of explanation. In particular, we look at transformations over model descriptions that preserve some property being queried by the user. As part of defining this framework, we also establish the space of possible explanatory properties that can be queried by the user in this setting, perform analysis over some general class of transformations and formalize the idea of explanatory witness. We perform user studies to validate the specific transformations studied in the chapter. Our user study results show that the transformations do help improve comprehensibility of the

Paper	Type of Witness	Actual Information
(Seegebarth *et al.*, 2012; Bercher *et al.*, 2014)	Existential Information	Causal chains
(Sreedharan *et al.*, 2018c)	Proof	Discusses providing plan trace and failure points of the foils
(Sreedharan *et al.*, 2019b)	Abstract Proof	Unreachable subgoal
(Göbelbecker *et al.*, 2010)	Counterfactual Information	Alternative Initial state
(Khan *et al.*, 2009)	Abstract Proof	Reachability information under current policy
(Dodson *et al.*, 2013)	Existential Information	Provides action factored differential values - Describes how much better the actions are in the next state given the current optimal action in comparison to the other actions.
(Madumal *et al.*, 2020b)	Existential Information	They contrast the outcomes of two actions over an action influence diagram, which is a modified form of structural causal model
(Krarup *et al.*, 2021, 2019)	Existential Information	Produces a plan that satisfy the user specified alternative
(Lin *et al.*, 2021)	Abstract Proof	The preference of one action over another is represented in terms of some accumulation of the policy execution after the current action (as opposed to the foil case)

Table 12.3: Some Examples of Explanatory Witness Used in the Literature (Part 1).

Paper	Type of Witness	Actual Information
(Juozapaitis *et al.*, 2019)	Abstract Proof	The model consists of an interpretable value function, where the value is decomposed into multiple human interpretable components. Thus the auxiliary information consists of comparing the values across these components.
(Madumal *et al.*, 2020a)	Existential Information	Here explanation includes opportunity chains, i.e., information derived from a decision-tree based representation of policies and also the information part of (Madumal *et al.*, 2020b)
(Waa *et al.*, 2018)	Existential Information	Describes the negative outcomes that occur as part of the foil
(Valmeekam *et al.*, 2020)	Abstract Proof	Provides the closest plan that may be feasible, conflict set and most likely set of actions that can be satisfied
(Kasenberg *et al.*, 2020)	Existential Information	If the queried formula (ϕ) could have been achieved, it generates a trajectory that satisfied ϕ and presents the outcome of following that trajectory
(Olson *et al.*, 2019)	Counterfactual Information	Counterfactual states

Table 12.4: Some Examples of Explanatory Witness Used in the Literature (Part 2).

explanations. Though we can't just rely on computational intuitions to decide the most useful transformations. Thus more work needs to be done in both identifying the strengths of these transformations and even developing novel transformations better suited for explanations.

Part III

ADDRESSING VOCABULARY ASYMMETRY

Chapter 13

PART-III OVERVIEW

In the last two parts of the thesis, we look at the two dimensions of explanations that could correspond to the reason why the user may be confused by the decisions being made by the system. We also saw, how in each case we could try to provide information about the robot's model to update the human mental model and by extension their expectation about robot behavior. However, each method discussed in the previous chapters makes one common assumption, namely that the robot can provide the information in a form the user can understand. In fact, most of the time we made the implicit assumption that the robot's own model is already specified in a form that can be understood by the human. This need not always be true, especially if the robot is using a learned model. In such cases, the robot's native model may be represented in a form that may be inscrutable to the human. As such, the explanation system would first need to translate the robot model to terms the user can understand before it can be presented to the user. This part of the thesis will focus on methods that are designed to generate explanations in the presence of such *vocabulary mismatch* between the robot and the user.

13.1 Structure for Part III and Technical Contributions

This part will be divided into two chapters

1. Chapter 14: This chapter will introduce the basic problem of vocabulary mismatch and present the case for using post hoc learned symbolic models as the basis for generating explanations in the presence of such asymmetry. This chap-

ter will be particularly focused on highlighting some of the central research challenges related to this direction. It will also layout a research agenda for future works related to addressing vocabulary mismatch.

2. Chapter 15: In this chapter, we will look at a specific explanation generation method that will try to address many of the challenges outlined in the previous chapter. In particular, we will look at the most basic contrastive explanation setting, where the agent presents a solution and the human raises an alternative plan they were expecting. The method will assume they have access to a set of predefined concepts that the human understands and use those concepts as the basis for constructing symbolic model fragments that could act as the basis for the explanation to the user. The method also allows for a way to quantify any uncertainty the agent may have about the post-hoc model it constructed. The chapter will also present the results from a user study that we ran to compare the effectiveness of post hoc symbolic explanation against methods like saliency maps.

13.2 Important Takeaways

One of the takeaways from the current method presented in Chapter 15, is the access to potential concepts the human understands. Going forward, effective methods to address vocabulary mismatch would require us to develop more efficient methods for vocabulary acquisition and as Chapter 14 discussed methods to teach new concepts to humans. In regards to connections to wider XAI, we can see many methods from explainable machine learning adopting similar methods. The obvious one being TCAV (Kim *et al.*, 2018), however even methods like LIME (Ribeiro *et al.*, 2016) are trying to bridge vocabulary differences when it is making use of alternative features like superpixels. Even outside the context of explanation generation, the need to bridge the vocabulary mismatch has been highlighted as a requirement for more effective human-AI collaboration (Kambhampati *et al.*, 2022) and as a requirement for value-alignment (Kim, 2022).

Chapter 14

ADDRESSING VOCABULARY MISMATCH THROUGH POST HOC LEARNED PDDL MODELS

In this chapter, we will look at the basic problem of vocabulary mismatch and present a case for using post hoc learned symbolic models as the basis for generating explanations in the presence of vocabulary mismatch. In particular, we will make the case for using PDDL style action centered model descriptions as the basis for such symbolic representations. The reason for considering PDDL-like representations for the model is multifold. For one, models like PDDL provide a very intuitive representation for planning problems as they are built around concepts from folk psychology (Miller, 2017a). Secondly, once we have a PDDL representation for the task, we can leverage many of the existing methods for explanation generation that have been developed specifically for PDDL like models (many of which are covered in this thesis). In the end, this work follows a growing consensus that while it is unclear whether AI systems themselves would need to use symbols in their internal processing for effective decision-making, there is no doubt that people are comfortable with and expect to communicate with these systems in terms of symbols that are meaningful to them. PDDL provides a particularly expressive, intuitive, and well-studied representation for sequential decision-making problems which could be used for such communication purposes.

14.1 Vocabulary Mismatch

We will start by defining vocabulary mismatch

Definition 42. *Consider a robot model $\mathcal{M}^R$ and the corresponding human mental model $\mathcal{M}^R_h$, such that there exists a mapping $\mathcal{F}_H$ from $\mathcal{M}^R$ states ($S^{\mathcal{M}^R}$) to $\mathcal{M}^R_h$ states $S^{\mathcal{M}^R_h}$. Where $\mathcal{F}_H(s)$ is a state in $S^{\mathcal{M}^R_h}$ that corresponds to state $s \in S^{\mathcal{M}^R}$ and we have $\mathcal{F}_H(I^{\mathcal{M}^R}) = I^{\mathcal{M}^R_h}$. In this setting, a **vocabulary mismatch** is said to exist between the two models if one of the following conditions are met*

1. $|F^{\mathcal{M}^R} \Delta F^{\mathcal{M}^R_h}| > 0$
2. $|A^{\mathcal{M}^R} \Delta A^{\mathcal{M}^R_h}| > 0$
3. $\exists f \in (F^{\mathcal{M}^R} \cap F^{\mathcal{M}^R_h})$ *and* $s \in S^{\mathcal{M}^R}$, *Such that* $f \in (s\ \Delta\ \mathcal{F}_H(s))$

The above definition, lays out two broad reasons for the existence of a vocabulary mismatch between the robot and the human model. First off, the robot or human model contains fluents or actions the other model doesn't contain (corresponds to conditions 1 and 2). An immediate consequence of this mismatch is the fact that we can no longer define the model parameterization function Γ (Definition 1) which requires a common fluent and action label space. This makes the communications of model information challenging.

The second and more nefarious way in which a vocabulary mismatch may manifest is through a difference in fluent grounding. In other words, both models may contain the same fluent label though they may refer to different things. In the above definition, this is denoted by cases where there are states whose representation in the robot and human model may use different fluents.

The above definition presents the basic version of the vocabulary mismatch problem when the robot model corresponds to a symbolic model. However, in many cases the robot model in its native form may correspond to a simulator or the individual model components may be represented by parameterized functions or inscrutable neural networks.

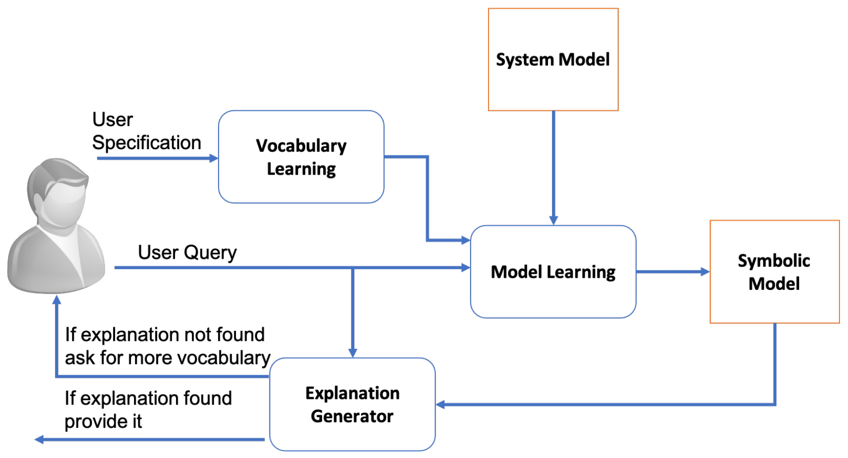


Figure 14.1: The Diagrammatic Representation of the Various Steps Involved in the Generation of Post Hoc Symbolic Explanations. The Explanatory Process Starts with the End User Specifying a Set of Vocabulary Terms They Understand and an Explanatory Query. The System Then Uses the Query along with the Specified Vocabulary to Create a Model Approximation of the Actual Model Used by the System.

Our general strategy to address vocabulary mismatch as laid out in this chapter and the next would be to relearn a representation of the robot model (or at least parts of the model) using the fluent, action labels and the fluent groundings that are part of the human model. We can then use this newly learned model to provide various explanations to the human.

14.2 Post Hoc Symbolic Explanation

Figure 14.1, presents an overview of the learning/explanation generation process that could be used to generate such post symbolic explanations. The overall process starts with the user of the system providing a set of vocabulary items (i.e. labels for actions and state fluents that the user understands) and an explanatory query (for example a contrastive query (Miller, 2017a)). Within the context of an AI system, the user described here could correspond to people interacting with the system under many capacities, including system designers, end-users, and even domain experts. In this chapter, we will mostly be agnostic to the specific user types, though one could easily see the role and background of the user could change the type of approaches used in each step. Also in the most general cases, the one specifying the vocabulary and the one raising the query need not be the same, but we will also ignore this distinction for now. With the vocabulary set and the query in place, the explanation generation procedure can interact with the actual system to generate a symbolic approximation that is sufficient to provide a response to the current query. If the system fails to identify an appropriate explanation, this is a good indicator that the original vocabulary set is incomplete and would require additional vocabulary items to create a higher fidelity symbolic model to generate the required explanations. In the rest of the section, we will look at each of these individual steps of the overall flow. One of the requirements for this method is the ability to leverage the internal models of the agent. Here we are using the term model in a very general sense. These could correspond to learned models being used by the agent (for example neural network models learned over latent state representations), procedural models, internal simulators, or even non-parametric models consisting of original experience being used by the agent to form its policy. The only requirement we place on the model is that we are able to interact with it and potentially sample experiences from it. In regards to the symbolic model, we will generally assume some variant of PDDL.

14.2.1 Vocabulary Learning

One of the core research challenges we are trying to address here is that of vocabulary mismatch. Plainly put, we need to overcome the fact that the system may be reasoning about the task in terms that a user of the desired background may not understand. Thus, mapping these models into any symbolic representation isn't enough as these may still be defined in terms that carry no real significance to the users. After all, a single atomic transition system could be synthesized from quite a different set of PDDL models, defined over various state factors. This makes many of the automatic model synthesis methods like that from Bonet and Geffner (2019) or Asai and Fukunaga (2018), which also try to automatically generate symbols, ill-suited for our purposes. One of our core proposals in this chapter is the need to include human input in some stage of the learning pipeline to determine the factors on which the model will be built. Basically, we would need people to specify at least parts of the action space and the state fluents over which the model would be learned. The core requirement for each vocabulary item is for the system to learn a way in which it can detect when the state contains a specific fluent value or when the action (or a trajectory) performed by the agent corresponds to an action specified by the user. One way to represent such mappings from system representations to human vocabulary items would be to learn binary classifiers for each item from data collected by interacting with the user. In general, all the data collection strategies discussed below will assume that the user can visually observe the current state and agent actions. This doesn't necessarily mean the agent actually reasons about the world in terms of raw visual information. It may well be that the human can observe an embodied agent acting in the world or the agent can expose some visual representation of its internal state (like in the case of ATARI agents, where the agent presents a visual representation of its ram state).

Actions The first obvious task would be to let the system identify the set of actions. This would be relatively straightforward in many RL tasks, including games where the action set of the original problem set is limited, and there exists a natural label of each of the possible actions. Though for many domains like robotics, the actual action space may be too complex (or even continuous) for the human to identify each action name. Instead, the human may think about the agent's action in terms of temporally extended actions (compared to the original actions). So instead of a series of joint angle changes, they may instead think in terms of abstract actions like picking and putting down objects. The person could communicate such abstract actions by labeling a set of agent demonstrations or by even providing demonstrations for each action. In either case, one could obtain a sequence of trajectories that correspond to each higher-level action, and one could learn classifiers that map trajectories to high-level action labels.

Fluents The other vocabulary item of interest to us is the state fluents. This could consist of propositional or relational factors the user believes is relevant to the given task. In the case of propositional fluents, the user could specify the set of important fluents to the system by providing a set of states where the fluent is true and a set of states where the fluent is false. These examples could then later be used to train classifiers for each concept. For relational concepts, the user could start by labeling relevant objects and then, similar to the propositional case, provide positive and negative examples for each predicate of interest.

Note that a core flexibility provided by the setting is the fact that it allows the original vocabulary set provided by the user to be *incomplete*. This makes it a fundamentally different enterprise from all the other works that try to force the decision-making algorithms to use interpretable features (cf. (Koh *et al.*, 2020) for single-step decision making, and (Lin *et al.*, 2021) for sequential problems). On the one hand, these methods can guarantee that the system is considering these features, on the other, they are also inherently limited by the original vocabulary set. It can do no better than what is possible under the original vocabulary set. In contrast, under this method, the system is free to choose the best representation of the problem that allows it to come up with solutions efficiently. While this is still a widely debated topic, at least the dogma in mainstream RL seem to be that while symbolic representations are useful for end-users to understand and interact with the system, forcing the system to reason over human-engineered representations and knowledge would hamper the system in many practical scenarios as they preclude the use of more general and scalable methods (cf. (Sutton, 2019; Silver *et al.*, 2021)). In this case, vocabulary is only used for explanations. Additionally, given the fact that the system has access to the more complex internal model, the explanatory system is also able to tell when the given vocabulary set is insufficient to explain the given decision. Once the system has detected this, it can query the user for more vocabulary items. We would expect the system should to do it in a directed way, though performing directed concept acquisition is very much an open problem.

14.2.2 Model Learning

The next important aspect of the entire process is to take the vocabulary item and try to build a symbolic approximation of the overall model. There are multiple ways one could go about learning the models. One might be to generate a bunch of plan traces, use the learned vocabulary items to match into symbolic terms, and then use any of the existing model learning methods (cf. (Stern and Juba, 2017)) to learn the final model. Alternatively, one could also employ a more active learning process in which the agent actively interacts with the environment until it finds a model that meets the required criteria (this is similar to the strategy employed by Chapter 15). Given all the existing work, we won't delve too much into the learning problem itself but rather look at some of the more unique possibilities that arise in this specific problem setting.

Local Approximations: While there aren't many explanation generation methods for sequential-decision making problems that look at post hoc explanations (particularly ones that try to build alternate models), post hoc representation learning is a very popular method in generating explanations for single shot decisions (Lakkaraju *et al.*, 2020). A common technique used by many of these methods to simplify the explanations is to focus on creating local approximations of the original models. Popularized by Ribeiro *et al.* (2016), under this technique, rather than generating post hoc models that try to approximate the full model, they try only to capture the behavior of the model in a region of interest over the input space. Usually, this may correspond to data points close to the one that needs to be explained. We can translate the idea of local approximations also to our setting, where we can choose to learn a symbolic model that approximates the true model only for a subset of states and actions. The next natural question would be how to decide this set of actions. One approach would be to follow Ribeiro *et al.* (2016) and the earlier machine learning explanation works and choose distance as the deciding factor. In particular, consider only states within some distance from the initial state or the states in the current plan, and consider only actions that are possible in those states. A very natural distance measure for planning problems would be reachability or, in particular, reachability within a specified number of steps. Though rather than just blindly focusing on reachability/distance, one could also select the state and action subset more effectively if we are aware of the user's intentions for asking the query. For the previously discussed use case, if we restrict reachability to only states that are part of screen 1, the precondition becomes not_next_to_skull (as Joe can't acquire a sword in that screen).

Learning Abstract Models vs Model Components: Unlike the traditional use cases of learning planning models, here, we may not need to learn the entire model. Instead, depending on the explanatory query, we may need to only learn an abstract version of the model or even just identify parts of the model. For example, the methods discussed in Chapter 15 looks at identifying explanations meant to refute alternate plans provided by the user. In this case, they identify only the required preconditions and an abstraction of the cost functions needed to refute the user queries.

Post Hoc Explanatory Confidence: Another important factor that is worth considering in this setting is measuring how accurately the learned model approximates the system model. Unless the explanatory system is exhaustively generating all possible transitions and behavior possible in the region of interest, there is a possibility that the learned model may not accurately reflect the actual behavior. If the model is wildly different from the true model, it could end up inducing incorrect beliefs in the end-user about the task and the system's understanding of the task. One way to try addressing such issues may be to provide the system with the ability to quantify its uncertainty about the model. Then it could use those measures to decide when it may be safe to provide explanations or even surface its uncertainty to the end-user. While there are some existing works on quantifying PAC guarantees for model learning (Stern and Juba, 2017), this generally is an underexplored problem. Additionally, if the learned vocabulary mapping (from system's representation to user's vocabulary) is noisy, the symbolic traces that the explanation system collects may be incorrect and this should also be reflected in the confidence it assigns to the learned model. Chapter 15 presents some methods for creating such confidence measures under certain assumptions about the task.

Reusing Previously Learned Model Components: In the end, interacting with the complete model will be an expensive process and we would want to avoid performing this unless it is completely required. This means, being able to recognize cases where any new queries raised by the user can be resolved by a previously learned model representation. Also developing methods that are able to stitch together previously learned model components and abstractions to more complete model representations and checking if they suffice to address the user queries.

14.2.3 Explanation Generation

We won't delve too much into the exact explanation methods that could be used to generate the target explanations. But will bring up the two factors that may be worth considering in this question.

Explanatory Queries: As mentioned earlier the model learning is driven by the explanatory query. Keeping with most mainstream works in XAIP, we will assume most of these queries are contrastive in the sense that the user is trying to understand why a specific decision was made against a possible alternate decision the human was expecting. Though in this case, we have to make an additional level of distinction, namely what the explanation is trying to establish why the current decision is better than the alternative raised by the user, or why and how the system decided to make the decision. This was a distinction established in Langley (2019), where the author refers to the former as preference accounts and the latter as process accounts. The post hoc explanatory methods are particularly well suited for preference accounts. As they can be evaluated on these post hoc models independent of the original decision-making process used to derive the system's decisions.

Identifying When the Model is Incomplete: The next important feature required for the explanation generation, is to explicitly allow for the fact that the model being used to generate the explanation may be incomplete. So the explanation generation method needs to be able to identify cases where the current learned model may be incapable of generating the required explanation and as such the system needs to query the user to acquire new vocabulary items which may be used to augment the given model. Access to a system model could be extremely useful in such scenarios, particularly for evaluating user-specified alternatives in the context of contrastive queries. As the system model could be used to test the validity and cost of these alternatives.

14.3 Research Challenges and Opportunities

This section will discuss some of the big open questions and research opportunities posed by this direction. This is in addition to smaller problems like tailoring explanations to specific types of models that may be popular in fields like RL, considering stochastic models, etc.

Acquiring new Concepts: One of the open challenges is to address cases where the original vocabulary set is incomplete. The agent now needs to query the human to expand its vocabulary. One obvious strategy may be to ask for more concepts, though this could be a very inefficient way to collect more concepts as the ones the human may provide may be completely irrelevant to the given problem. An advantage the agent has is that it has access to its own representation of the task and thus may be able to provide some hints to the human as to what concepts may be relevant to the current query. One way to accomplish this may be to leverage low-level visual explanations (when a common visual channel is available). While we are unaware of any works in sequential-decision explanation that have leveraged such methods to collect concepts, a closely related work that has looked at collecting such concepts is the method presented by Hamidi-Haines *et al.* (2018), where they developed an interface that allows users to name certain regions of the state highlighted according to their relevance to the decision-making process. One would want to build on such interfaces for more general sequential decision-making systems and also possibly relax various assumptions like the fact that each concept corresponds to specific parts of the image. Another possibility is to revisit the end-to-end model learning methods similar to Bonet and Geffner (2019), that also learns symbols. An open research question here is developing methods that check if any of the automatically discovered concepts or composition of such concepts could potentially map to concepts at the user's end. Additionally, we could also check if one could introduce inductive biases into these systems that allow the generation of naturally interpretable concepts (Yeh *et al.* (2020) discussed how the assumption of the locality could be used in single-shot decisions).

Incorporating Possible Noisy Decision-Making: One of the questions that we have ignored in this chapter is whether the agent's decision-making process can correctly use their internal model. Even in the best case, the symbolic models will only reflect what is present in the internal model. Though given the realities of the state of the art RL methods, in many cases, it is hard to guarantee that the decision-making processes can correctly use their own internal models to generate their decisions. If, in fact, these explanations are presented as the reasoning behind the agent's decisions (i.e., a process account per Langley (2019)), it could lead to the user forming incorrect beliefs about the agent's reasoning capabilities. It is still very much an open question how we could leverage the specifics of the agent's decision-making processes to select more precise explanations. Some initial strategies we could employ include performing additional tests like checking whether perturbation of some concept identified as part of the explanation in the current state leads to the system choosing a different action, or using the system's own internal representations (for example, looking at intermediate layer activations in the case of neural networks) to build concept classifiers, etc.

Allowing For Collaboration: We strongly believe that explanatory systems should be evaluated in the context of the overall application. A common use case for such systems may be scenarios where the user and the system are collaborating to come up with better solutions (as in the case of iterative planning (Smith, 2012)). In the most general case, we would want this to be a bidirectional interaction wherein both the agent and the human can influence each other's beliefs on what constitutes an ideal solution. While there have been recent works on developing methods that allow people to specify preferences/objectives for agent behavior (cf. (Illanes *et al.*, 2020; De Giacomo *et al.*, 2019)), they have generally focused only on developing interfaces to let the user specify constraints over the behavior the agent can generate using its internal models. For these systems to be truly successful, we need to not only allow the agent to provide explanations over why it may be performing certain behaviors, in terms of its beliefs about the model of the task, but also provide the user the ability to override the agent's belief about the task. We strongly believe that symbolic models can provide us with an interpretable interface to facilitate such bidirectional interactions. However, it's very much an open question on how to effectively take these post hoc model updates and fold them into the agent's decision-making process that may be using inscrutable models.

Few Shot Learning of Concepts: Unless the concepts are being pre-specified by a domain expert/designer where they could invest in collecting a large number of examples, these concepts would need to be learned from a few examples. If the agent is using learning methods that can compute its own representations of the state for coming up with decisions. Such simplified representations could then be used as the input to our vocabulary item classifiers.

Teaching Concepts: One of the possible scenarios we have not quite considered in this chapter is what happens when the human's vocabulary has no concept that could potentially explain the current decision. For example, there may exist no concepts that can describe the kind of patterns AlphaGo may be looking for to decide novel moves. In such cases, the agents would need to teach the human new concepts. We are unaware of any works that have even begun to look at these problems, though one can imagine effective solutions to this problem would need to make use of strategies from intelligent tutoring systems (ITS) (Aeronautiques *et al.*, 1998), natural language generation, and many other subareas of AI.

Chapter 15

A SAMPLING BASED METHOD TO LEARN SYMBOLIC MODEL FRAGMENTS FOR CONTRASTIVE EXPLANATIONS

The previous chapter looked at the basic problem of vocabulary mismatch and introduced some of the basic challenges of generating explanations in this setting. This chapter on the other hand will introduce a specific method for providing contrastive explanations in terms of user-specified concepts for sequential decision-making settings where the system's model of the task may be best represented as an inscrutable model. We do this by building partial symbolic models of a local approximation of the task that can be leveraged to answer the user queries. We test these methods on a popular Atari game (Montezuma's Revenge) and variants of Sokoban (a well-known planning benchmark) and report the results of user studies to evaluate whether people find explanations generated in this form useful.

Figure 15.1 presents the flow of the proposed explanation generation process in the context of Montezuma's Revenge (Wikipedia contributors, 2019). In this chapter, we will focus on deterministic settings (Section 15.2), though we can easily extend the ideas to stochastic settings, and will ground user-specified vocabulary terms in the AI system's internal representations via learned classifiers (Section 15.3). The model components required for explanations are learned by using experiences (i.e. state-action-state sets) sampled from the agent model (Section 15.3 & 15.4). Additionally, we formalize the notion of local symbolic approximation for sequential decision-making models, and introduce the idea of explanatory confidence (Section 15.5). The exaplanatory confidence captures the fidelity of explanations and help ensure the system only provides explanation whose confidence is above a given thresh-

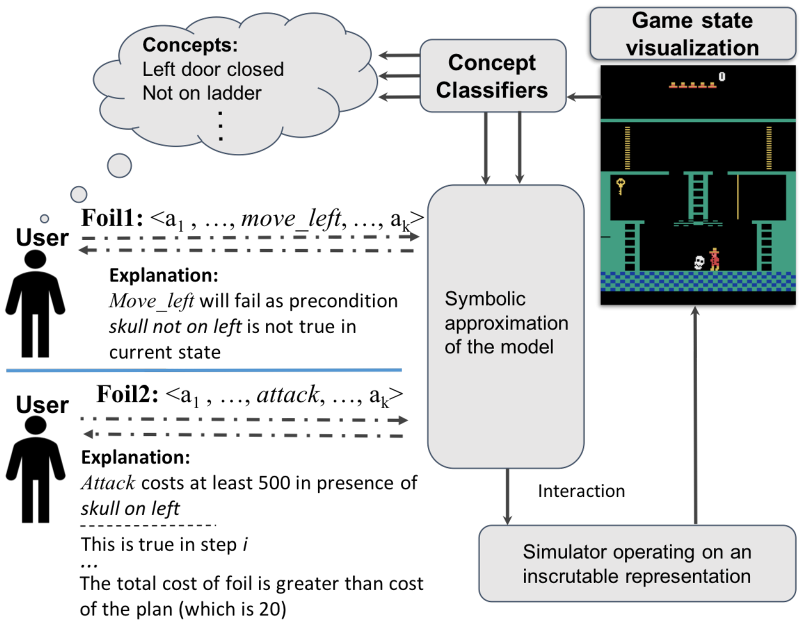

Concepts:
Left door closed
Not on ladder
⋮
Concept Classifiers
Game state visualization
User
Foil1: <a1 , …, move_left, …, ak>
Explanation:
Move_left will fail as precondition skull not on left is not true in current state
Symbolic approximation of the model
User
Foil2: <a1 , …, attack, …, ak>
Explanation:
Attack costs at least 500 in presence of skull on left
This is true in step i
…
The total cost of foil is greater than cost of the plan (which is 20)
Interaction
Simulator operating on an inscrutable representation

Figure 15.1: Explanatory Dialogue Starts When the User Presents a Specific Alternate Plan. The System Explains the Preference over This Alternate Plan in Terms of Model Information Expressed in Terms of Propositional Concepts Specified by the User, Operationalized as Classifiers.

old. We will evaluate the effectiveness of the method through both systematic (IRB approved) user studies and computational experiments (Section 15.6). As we will discuss in Section 15.1, our approach has some connections to the method presented by Kim *et al.* (2018), which while focused on one-shot classification tasks, also advocates explanations in terms of concepts that have meaning to humans in the loop.

15.1 Related Work

The representative works in the direction of using concept level explanation include works like those presented by Bau *et al.* (2017), TCAV (Kim *et al.*, 2018) and its various offshoots (Luss *et al.*, 2019) that have focused on one-shot decisions. These works take a line quite similar to us in that they try to create explanations for current decisions in terms of a set of user-specified concepts. While these works don't explicitly reason about explanatory confidence they do discuss the possibility of identifying inaccurate explanation, and tries to address them through statistical tests. Another thread of work, exemplified by works like those presented by Koh *et al.* (2020); Lin *et al.* (2021), tries to force decisions to be made in terms of user-specified concepts, which can then be used for explanations. There have also been recent works on automatically identifying human-understandable concepts like the ones presented by Ghorbani *et al.* (2019) and Hamidi-Haines *et al.* (2018). We can also leverage these methods when our system identifies scenarios with insufficient vocabulary.

Most works in explaining sequential decision-making problems either use a model specified in a shared vocabulary as a starting point for explanation or focus on saliency-based explanations (cf. (Chakraborti *et al.*, 2020)), with very few exceptions. Hayes and Shah (2017) have looked at the use of high-level concepts for policy summaries. They use logical formulas to concisely characterize various policy choices, including states where a specific action may be selected (or not). Unlike our work, they are not trying to answer why the agent chooses a specific action (or not). Waa *et al.* (2018) looked at addressing the suboptimality of foils while supporting interpretable features, but it requires the domain developer to assign positive and negative outcomes to each action. In addition to not addressing possible vocabulary differences between a system developer and the end-user, it is also unclear if it is always possible to attach negative and positive outcomes to individual actions.

Another related work is the approach studied by Madumal *et al.* (2020c). Here, they are also trying to characterize dynamics in terms of high-level concepts but assume that the full structural relationship between the various variables is provided upfront. The explanations discussed in this chapter can also be seen as a special case of Model Reconciliation explanation (cf. (Chakraborti *et al.*, 2017)), where the human model is considered to be empty. The usefulness of preconditions as explanations has also been studied by works like the ones by Winikoff (2017); Broekens *et al.* (2010). Our effort to associate action cost to concepts could also be contrasted to efforts by Juozapaitis *et al.* (2019) and Anderson *et al.* (2019) which leverage interpretable reward components. Another group of works popular in RL explanations is the ones built around saliency maps (Greydanus *et al.*, 2018; Iyer *et al.*, 2018; Puri *et al.*, 2019), which tend to highlight parts of the state that are important for the current decision. In particular, we used the method proposed by Greydanus *et al.* (2018) as a baseline because many follow-up works have shown their effectiveness (cf. (Zhang *et al.*, 2020)). Readers can refer to Alharin *et al.* (2020) for a recent survey of explanations in RL.

Another related thread of work, is that of learning models (Carbonell and Gil (1990); Stern and Juba (2017); Wu *et al.* (2007) provide some representative works). To the best of our knowledge none of the works in this direction allow for noisy observations of state and none focused on identifying specific model components. While we are unaware of any works that provide confidence over learned model components, Stern and Juba (2017) provides loose PAC guarantees over the entire learned model.

15.2 Problem Setting

Our setting consists of an agent, be it programmed or RL-based, that has a model of the dynamics of the task that is inscrutable to the user (in so far that the user can't directly use the model representation used by the agent) in the loop and uses a decision-making process that is sound for this given model. Note that here the term *model* is being used in a very general sense. These could refer to tabular models defined over large atomic state spaces, neural network-based dynamics models possibly learned over latent representation of the states and even simulators. The only restriction we place on the model is that we can sample possible experiences from it. Regardless of the true representation, we will denote the model by the tuple $\mathcal{M} = \langle S, A, T, \mathcal{C} \rangle$. Where S and A are the state and action sets and $T : S \times A \rightarrow S \cup \{\perp\}$ ($\perp$ the absorber failure state) and $\mathcal{C} : S \times A \rightarrow \mathbb{R}$ capture the transition and cost function. We will use $\perp$ to capture failures that could occur when the agent violates hard constraints like safety constraints or perform any invalid actions. We will consider goal-directed agents that are trying to drive the state of the world to one of the goal states ($\mathbb{G}$ being the set) from an initial state I. The solution takes the form of a sequence of actions or a plan π.

We will use model similar to the one described in Section 2.1 to approximate the problem for explanations. Such a model can be represented by the tuple $\mathcal{M}_{\mathcal{S}} = \langle F_{\mathcal{S}}, A_{\mathcal{S}}, I_{\mathcal{S}}, G_{\mathcal{S}}, \mathcal{C}_{\mathcal{S}} \rangle$, where $F_{\mathcal{S}}$ is a set of propositional state variables defining the state space, $A_{\mathcal{S}}$ is the set of actions, $I_{\mathcal{S}}$ is the initial state, $G_{\mathcal{S}}$ is the goal specification. Each valid problem state is uniquely identified by the subset of state variables that are true in that state (so for any state $s \in S_{\mathcal{M}_{\mathcal{S}}}$, where $S_{\mathcal{M}_{\mathcal{S}}}$ is the set of states for $\mathcal{M}_{\mathcal{S}}$, $s \subseteq F_{\mathcal{S}}$). Each action $a \in A_{\mathcal{S}}$ is further described in terms of the preconditions $prec_a$ (specification of states in which a is executable) and the effects of executing the action. We will denote the state formed by executing action a in a state s as $a(s)$. We will focus on models where the preconditions are represented as a conjunction of propositions. If the action is executed in a state with missing preconditions, then the execution results in the invalid state $\perp$. Unlike standard model described in Section 2.1, where the cost of executing action is independent of states, we will use a state-dependent cost function of the form $\mathcal{C}_{\mathcal{S}} : 2^F \times A_{\mathcal{S}} \rightarrow \mathbb{R}$ to capture forms of cost functions popular in RL benchmarks.

15.3 Contrastive Explanations

The specific explanatory setting, illustrated in Figure 15.1, is one where the agent comes up with a plan π (to achieve one of the goals specified in $\mathbb{G}$ from I) and the user responds by raising an alternate plan π_f (*the foil*) that they believe should be followed instead. Now the system needs to explain why π may be preferred over π_f, by showing that π_f is invalid (i.e., π_f doesn't lead to a goal state or one of the action in π_f results in the invalid state $\perp$) or π_f is costlier than π ($\mathcal{C}(I, \pi) < \mathcal{C}(I, \pi_f)$).[1]

To concretize this interaction, consider the modified version of Montezuma's Revenge (Figure 15.1). The agent starts from the highest platform, and the goal is to get to the key. The specified plan π may require the agent to make its way to the lowest level, jump over the skull, and then go to the key with a total cost of 20. Now the user raises two possible foils that are similar to π but use different strategies in place of jumping over the skull. *Foil1*: instead of jumping, the agent just moves left (i.e., it tries to move *through* the skull) and, *Foil2*: instead of jumping over the skull, the agent performs the `attack` action (not part of the original game, but added here for illustrative purposes) and then moves on to the key. Using the internal model, the system can recognize that in the first case, moving left would lead to an invalid state, and in the second case, the foil is costlier, though effectively communicating this to the user is a different question. If there exists a shared visual communication channel, the agent could try to demonstrate the outcome of following these alternate strategies. Unfortunately, this would not only necessitate additional cognitive load on the user's end to view the demonstration, but it may also be confusing to the user in so far that they may not be able to recognize why in a particular state the move left action was invalid and attack action costly. As established in our user study

[1]If the foil is as good as the original plan or better, then the system could switch to the foil.

and pointed out by Atrey *et al.* (2019), even highlighting of visual features may not effectively resolve this confusion. This scenario thus necessitates the use of methods that are able to express possible explanations in terms that the user may understand.

Learning Concept Maps: The input to our system is *the set of propositional concepts the user associates with the task.* For Montezuma, this could involve concepts like *agent being on a ladder, holding onto the key, being next to the skull.* Each concept corresponds to a propositional fact that the user associates with the task's states and believes their presence or absence in a state could influence the dynamics and the cost function. We can collect such concepts from subject matter experts as by Cai *et al.* (2019), or one could just let the user interact with or observe the agent and then provide a possible set of concepts. We used the latter approach to collect the propositions for our evaluation of the Sokoban domains (Section 15.6 and A). Each concept corresponds to a binary classifier, which detects whether the proposition is present or absent in a given internal state (thus allowing us to convert the atomic states into a factored representation). Let $\mathbb{C}$ be the set of classifiers corresponding to the high-level concepts. For state $s \in S$, we will overload the notation $\mathbb{C}$ and specify the concepts that are true as $\mathbb{C}(s)$, i.e., $\mathbb{C}(s) = \{c_i | c_i \in \mathbb{C} \wedge c_i(s) = 1\}$ (where c_i is the classifier corresponding to the i^{th} concept and we overload the notation to also stand for the label of the i^{th} concept). The training set for such concept classifiers could come from the user (where they provide a set of positive and negative examples per concept). Classifiers can be then learned over the model states or the internal representations used by the agent decision-making (for example activations of intermediate neural network layers).

Explanation using concepts: To explain the preference of plan π over foil π_f, we will present model information to the user taken from a symbolic representation of the agent model. But rather than requiring this model to be an exact representation of the complete agent model, we will instead focus on accurately capturing a subset of the model by instead trying to learn a local approximation

Definition 43. *A symbolic model* $\mathcal{M}^{\mathbb{C}}_{\mathcal{S}} = \langle \mathbb{C}, A^{\mathbb{C}}_{\mathcal{S}}, \mathbb{C}(I), \mathbb{C}(\mathbb{G}), \mathcal{C}^{\mathbb{C}}_{\mathcal{S}} \rangle$. *is said to be a* local symbolic approximation *of the model* $\mathcal{M}^R = \langle S, A, T, \mathcal{C} \rangle$ *for regions of interest* $\hat{S} \subseteq S$ *if* $\forall s \in \hat{S}$ *and* $\forall a \in A$*, we have an equivalent action* $a^{\mathbb{C}} \in A^{\mathbb{C}}_{\mathcal{S}}$*, such that (a)* $a^{\mathbb{C}}(\mathbb{C}(s)) = \mathbb{C}(T(s, a))$ *(assuming* $\mathbb{C}(\bot) = \bot$*) and (b)* $\mathcal{C}^{\mathbb{C}}_{\mathcal{S}}(\mathbb{C}(s), a) = \mathcal{C}(s, a)$ *and (c)* $\mathbb{C}(\mathbb{G}) = \bigcap_{s_g \in \mathbb{G} \cap \hat{S}} \mathbb{C}(s_g)$.

Following Section 15.2, this is a PDDL-style model with preconditions and conditional costs defined over the conjunction of positive propositional literals. A establishes the sufficiency of this representation to capture arbitrarily complex preconditions (including disjunctions) and cost functions expressed in terms of the proposition set $\mathbb{C}$. Also to establish the preference of plan does not require informing the users about the entire model $\mathcal{M}^{\mathbb{C}}_{\mathcal{S}}$, but rather only the relevant parts. To establish the invalidity of π_f, we only need to explain the failure of the first failing action a_i, i.e., the one that resulted in the invalid state (for *Foil1* this corresponds to move-left action at the state visualized in Figure 15.1). We explain the failure of action in a state by pointing out a proposition that is an action precondition which is absent in the given state. Thus a concept $c_i \in \mathbb{C}$ is considered an explanation for failure of action a_i at state s_i, if $c_i \in prec_{a_i} \setminus \mathbb{C}(s_i)$. For *Foil1*, the explanation would be – ***the action* move-left *failed in the state as the precondition* skull-not-on-left *was false in the state***.

This formulation can also capture failure to achieve goal by appending an additional goal action at the end of the plan, which causes the state to transition to an end state, and fails for all states except the ones in $\mathbb{G}$. Note that instead of identifying all the missing preconditions, we focus on identifying a single precondition, as this closely follows prescriptions from works in social sciences that have shown that selectivity or minimality is an essential property of effective explanations (Miller, 2017a).

For explaining suboptimality, we inform the user about the symbolic cost function $\mathcal{C}_{\mathcal{S}}^{\mathbb{C}}$. To ensure minimality, rather than provide the entire cost function, we will instead try to learn and provide an abstraction of the cost function $\mathcal{C}_s^{abs}$

Definition 44. *For the symbolic model $\mathcal{M}_{\mathcal{S}}^{\mathbb{C}} = \langle \mathbb{C}, A_{\mathcal{S}}^{\mathbb{C}}, \mathbb{C}(I), \mathbb{C}(\mathbb{G}), \mathcal{C}_{\mathcal{S}}^{\mathbb{C}} \rangle$, an abstract cost function $\mathcal{C}_{\mathcal{S}}^{abs} : 2^{\mathbb{C}} \times A_{\mathcal{S}}^{\mathbb{C}} \rightarrow \mathbb{R}$ is specified as: $\mathcal{C}_{\mathcal{S}}^{abs}(\{c_1, .., c_k\}, a) = min\{\mathcal{C}_{\mathcal{S}}^{\mathbb{C}}(s, a) | s \in S_{\mathcal{M}_{\mathcal{S}}^{\mathbb{C}}} \wedge \{c_1, .., c_k\} \subseteq s\}$.*

Intuitively, $\mathcal{C}_{\mathcal{S}}^{abs}(\{c_1, .., c_k\}, a) = k$ can be understood as stating that *executing the action a, in the presence of concepts $\{c_1, .., c_k\}$ costs at least k.* We can use $\mathcal{C}_{\mathcal{S}}^{abs}$ in an explanation by identifying a sequence of concept set $\mathbb{C}_{\pi_f} = \langle \hat{\mathbb{C}}_1, ..., \hat{\mathbb{C}}_k \rangle$, corresponding to each step of the foil $\pi_f = \langle a_1, .., a_k \rangle$, such that (a) $\hat{\mathbb{C}}_k$ is a subset of concepts in the corresponding state reached by the foil and (b) the total cost of abstract cost function defined over the concept subsets are larger than the plan cost $\sum_{i=\{1..k\}} \mathcal{C}_{\mathcal{S}}^{abs}(\hat{\mathbb{C}}_i, a_i) > \mathcal{C}(I, \pi)$. For *Foil2*, the explanation would include the information – ***executing the action* attack *in the presence of the concept* skull-on-left*, will cost at least 500*.**

15.4 Identifying Explanations through Sample-based Trials

For identifying the model parts, we will rely on the internal model to build symbolic estimates. Since we can separate the two explanation cases using the agent's internal model, we will only focus on the problem of identifying the model parts given the required explanation type.

Identifying failing precondition: The first case we will consider is the one related to explaining plan failures. In particular, given the failing state s_{fail} and failing

a Algorithm for Finding Missing Precondition

```
1: procedure PRECONDITION-SEARCH
2:     Input:   $s_{\text{fail}}, a_{\text{fail}}$, Sampler, $\mathcal{M}^R, \mathbb{C}, \ell$
3:     Output: Missing precondition $C_{\text{prec}}$
4:     Procedure:
5:     $\mathcal{H}_{\mathbb{P}} \leftarrow$ Missing concepts in $s_{fail}$
6:     sample_count = 0
7:     while sample_count < $\ell$ do
8:         $s \sim$ Sampler
9:         if $T(s, a_{\text{fail}}) \neq \perp$ then
10:            $\mathcal{H}_{\mathbb{P}} \leftarrow \mathbb{C}(s) \cap \mathcal{H}_{\mathbb{P}}$
11:         if $|\mathcal{H}_{\mathbb{P}}| = 0$ then return Signal that concept list is incomplete
12:         sample_count += 1
     return any $C_i \in$poss_prec_set
```

b Algorithm for Finding Cost Function

```
1: procedure COST-FUNCTION-SEARCH
2:     Input:   $\pi_f, \mathcal{C}_\pi$, Sampler, $\mathcal{M}^R, \mathbb{C}, \ell$
3:     Output: $\mathbb{C}_{\pi_f}$
4:     Procedure:
5:     for conc_limit in 1 to $|\mathbb{C}|$ do
6:         current_foil_cost = 0, conc_list = [], $s \leftarrow I$
7:         for i in 1 to k (the length of the foil) do
8:             $s \leftarrow T(s, a_i)$
9:             $\hat{\mathbb{C}}_i$, min_cost =
10: find_min_conc_set($\mathbb{C}(s), a_i$, conc_limit, $\ell$)
11:             current_foil_cost += min_cost
12:             conc_list.push($\hat{\mathbb{C}}_i$, min_cost)
13:         if current_foil_cost > $\mathcal{C}_\pi$ then return conc_list
     return Signal that the concept list is incomplete
```

Figure 15.2: Algorithms for Identifying (a) Missing Precondition and (b) Cost Function

foil action a_{fail}, we want to find a concept that is absent in s_{fail} but is a precondition for the action a_{fail}. We will try to approximate whether a concept is a precondition or not by checking whether they appear in all the states sampled from the model, where a_{fail} is executable ($\mathbb{S}$ - set of all sampled states). Figure 15.2(a) presents the pseudo-code for finding such preconditions. We rely on the sampler (denoted as Sampler) to create the set $\mathbb{S}$, where the number of samples used is upper-bounded by a sampling budget ℓ. poss_prec_set captures the set of hypotheses regarding the missing precondition maintained over the course of the search.

We ensure that models learned are local approximations by considering only samples within some distance from the states in the plan and foils. For reachability-based distances, we can use random walks to generate the samples. Specifically, we can start a random walk from one of the states observed in the plan/foil and end the sampling episode whenever the number of steps taken crosses a given threshold. The search starts with a set of hypotheses for preconditions $\mathcal{H}_{\mathbb{P}}$ and rejects individual hypotheses whenever the search sees a state where the action a_{fail} is executable and the concept is absent. The worst-case time complexity of this search is linear on the sampling budget.

If the hypotheses set turns empty, *it points to the fact that the current vocabulary set is insufficient to explain the given plan failure.* Focusing on a single model-component not only simplifies the learning problem but as we will see in Section 15.5 allows us to quantify uncertainty related to the learned model components and also allow for noisy observation. These capabilities are missing from previous works.

Identifying cost function: We will employ a similar sampling-based method to identify the cost function abstraction. Unlike the precondition failure case, there is no single action we can choose, but rather we need to choose a level of abstraction for each action in the foil (though it may be possible in many cases to explain the suboptimality of foil by only referring to a subset of actions in the foil). For the concept subset sequence ($\mathbb{C}_{\pi_f} = \langle \hat{\mathbb{C}}_1, ..., \hat{\mathbb{C}}_k \rangle$) that constitutes the explanation, we will also try to minimize the total number of concepts used in the explanation ($\sum_{i=1..k} \|\hat{\mathbb{C}}_i\|$). Figure 15.2(b) presents a greedy algorithm to find such cost abstractions. The procedure find_min_conc_set, takes the current concept representation of state s_i in the foil and searches for the subset $\hat{\mathbb{C}}_i$ of $\mathbb{C}(s_i)$ with the maximum value for $\mathcal{C}_{\mathcal{S}}^{abs}(\hat{\mathbb{C}}_i, a_i)$, where the value is approximated through sampling (with budget ℓ), and the subset size is upper bounded by conc_limit. As mentioned in Definition 44, the value of $\mathcal{C}_{\mathcal{S}}^{abs}(\hat{\mathbb{C}}_i, a_i)$ is given as the minimal cost observed when executing the action a_i in a state where $\hat{\mathbb{C}}_i$ are true. The algorithm incrementally increases the conc_limit value till a solution is found or if it crosses the vocabulary set size. Note that this algorithm is not an

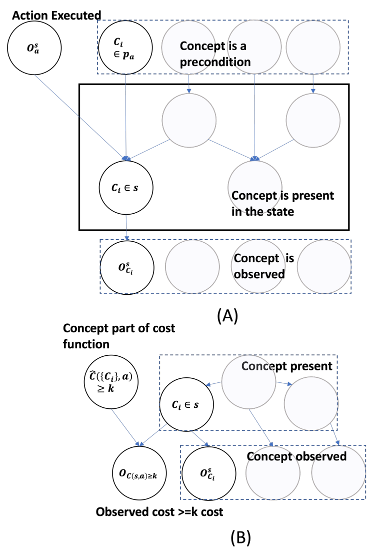


Figure 15.3: Simplified Probabilistic Graphical Models for Confidence of Learned (A) Preconditions and (B) Cost).

optimal one, but we found it to be effective enough for the scenarios we tested. We can again enforce the required locality within the sampler and similar to the previous case, *we can identify the insufficiency of the concept set by the fact that we aren't able to identify a valid explanation when conc_limit is at vocabulary size.* The worst case time complexity of the search is $\mathcal{O}(|\mathbb{C}| \times |\pi_f| \times |\ell|)$. We are unaware of any existing works for learning such abstract cost-functions.

15.5 Quantifying Explanatory Confidence

One of the critical challenges faced by any post hoc explanations is the question of fidelity and how accurately the generated explanations reflect the decision-making process. In our case, we are particularly concerned about whether the model component learned using the algorithms mentioned in Section 15.4 is part of the exact symbolic representation $\mathcal{M}_{\mathcal{S}}^{\mathbb{C}}$ (for a given set of states $\hat{S}$ and actions $\hat{A}$). Given that we are identifying model components based on concepts provided by the user, it would be easy to generate explanations that feed into the user's confirmation biases about the task.

Our confidence value associates a probability of correctness with each learned model component by leveraging learned relationships between the concepts. Additionally, our confidence measure also captures the fact that the learned concept classifiers will be noisy at best. So we will use the output of the classifiers as noisy observations of the underlying concepts, with the observation model $P(O^s_{c_i}|c_i \in S)$ (where $O^s_{c_i}$ captures the fact that the concept was observed and $c_i \in S$ represents whether the concept was present in the state) defined by a probabilistic model of the classifier prediction.

Figure 15.3, presents a graphical model (with notations defined inline) for measuring the posterior of a model component being true given an observation generated from the classifier on a sampled state. For a given observation of executions of action a in state s ($O^s_a = True$ or just O^s_a), the positive or negative observation of a concept c_i ($O^s_{c_i} = x$, where $x \in \{True, False\}$) and an observation that action's cost is greater than k ($O_{\mathcal{C}(s,a)\geq k}$), the updated explanatory confidence for each explanation type is provided as *(1)* $P(c_i \in p_a|O^s_{c_i} = x \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$*, where* $c_i \in p_a$ *captures the fact that the concept is part of the precondition of* a *and* $O_{\mathbb{C}(s)\setminus c_i}$*is the observed status of the rest of the concepts*, and (2) $P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k|O^s_{c_i} = x \wedge O_{\mathcal{C}(s,a)>=k} \wedge O_{\mathbb{C}(s)\setminus c_i})$, where $\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k$ asserts that the abstract cost function defined over concept c_i is greater than k

The final probability for the explanation would be the posterior calculated over all the state samples. Additionally, we can also extend these probability calculations to multi-concept cost functions. We can now incorporate these confidence measures directly into the search algorithms described in Section 15.4 to find the explanation that maximizes these probabilities. In the case of precondition identification, given a noisy classifier, we can no longer use the classifier output to directly remove a concept from the hypothesis set $\mathcal{H}_{\mathbb{P}}$. At the start of the search, we associate a prior to each hypothesis and use the observed concept value at each executable state to update the posterior. We only remove a precondition from consideration if the probability associated with it dips below a threshold κ. For cost function identification, if we have multiple cost functions with the same upper bound, we select the one with the highest probability. We also allow for noisy observations in the case of cost identification. The updated pseudo codes are provided in A.

We can simplify the probability calculations by making the following assumptions: (1) the distribution of all non-precondition concepts in states where the action is executable is the same as their overall distribution (which can be empirically estimated), (2) The cost distribution of action over states corresponding to a concept that does not affect the cost function is identical to the overall distribution of cost for the action (which can again be empirically estimated). The first assumption implies that the likelihood of seeing a non-precondition concept in a sampled state where the action is executable is equal to the likelihood of it appearing in any sampled state. In the most general case, this equals $P(c_i \in s|O_{\mathbb{C}(s)\setminus c_i})$, i.e., the likelihood of seeing this concept given the other concepts in the state. The second assumption implies that for a concept that has no bearing on the cost function for an action, the likelihood that executing the action in a state where the concept is present will result in a cost greater than k will be the same as that of the action execution resulting in a cost greater than k for a randomly sampled state ($p_{\mathcal{C}(.,a)\geq k}$). This assumption can be further relaxed by considering the distribution of the action cost given the rest of the observed set of concepts (though this would require more samples to learn). *The full derivation of the probability calculations with the assumptions is provided in A and an empirical evaluation of the assumptions in A. All results reported in the next section*

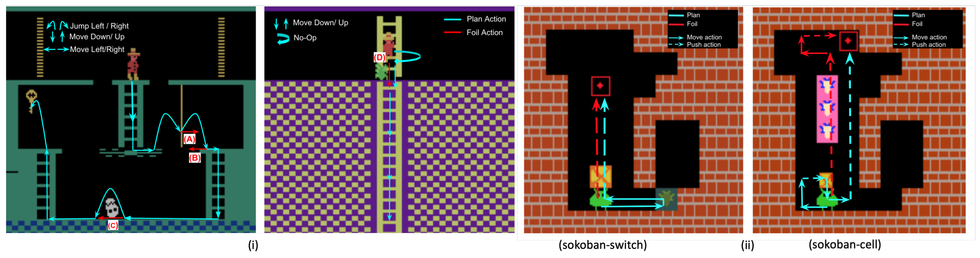


Figure 15.4: The Plan and Foils Used in the Evaluation, Plans Are Highlighted in Blue and Foils in Red: (i) Shows the Two Montezuma's Revenge Screens Used (ii) the Two Variations of the Sokoban Game Used. For Montezuma the Concepts Corresponding to Missing Precondition Are Not_on_rope, Not_on_left_ledge, Not_skull_on_left and Is_clear_down_of_crab for Failure Points a, B, C and D Respectively (Note All Concepts Were Originally Defined as Positives and We Formed the Not_ Concepts by Negating the Output of the Classifier). Concepts Switch_on and On_pink_cell for Sokoban Cost Variants That Results in Push Action Costing 10 (Instead of 1)

were calculated using the probabilistic version of the search while allowing for noisy concept maps.

15.6 Evaluation

We tested the approach on the open-AI gym's deterministic implementation of Montezuma's Revenge (Brockman *et al.*, 2016) for precondition identification and two modified versions of the gym implementation of Sokoban (Schrader, 2018) for both precondition and cost function identification. *To simplify calculations for the experiments, we made an additional assumption that the concepts are independent.* All hyperparameters used by learning algorithms are provided in A.

Montezuma's Revenge: We used RAM-based state representation here. To introduce richer preconditions to the settings, we marked any non-noop action that fails to change the current agent state and action that leads to midair states where the agent is guaranteed to fall to death (regardless of actions taken) as failures. We selected four invalid foils (generated by the authors by playing the game), three from screen-1 and one from screen-4 of the game (Shown in Figure 15.4 (i)). The plan for screen-1 involved the player reaching a key and for screen-2 the player has to reach the bottom of the screen. We specified ten concepts, which we believed would be relevant to the game, for each screen and collected positive and negative examples using automated scripts. We used AdaBoost Classifiers (Freund *et al.*, 1999) for the concepts and had an average accuracy of 99.72%.

Sokoban Variants: The original sokoban game involves the player pushing a box from one position to a target position. We considered two variations of this basic game. One that requires a switch (green cell) the player could turn on before pushing the box (referred to as Sokoban-switch), and a second version (Sokoban-cells) included particular cells (highlighted in pink) from which it is costlier to push the box. For Sokoban-switch, we had two variations, one in which turning on the switch was a precondition for push actions and another one in which it merely reduced the cost of pushing the box. The plan and foil (Shown in Figure 15.4 (ii)) were generated by the authors by playing the game. *We used a survey of graduate students unfamiliar with our research to collect the set of concepts for these variants.* The survey allowed participants to interact with the game through a web interface (the cost-based version for Sokoban-switch and Sokoban-cell), and at the end, they were asked to describe game concepts that they thought were relevant for particular actions. We received 25 unique concepts from six participants for Sokoban-switch and 38 unique concepts from seven participants for Sokoban-cell. We converted the user descriptions of concepts to scripts for sampling positive and negative instances. We focused on 18 concepts and 32 concepts for Sokoban-switch and Sokoban-cell, respectively, based on the frequency with which they appear in game states, and used Convolutional Neural Networks

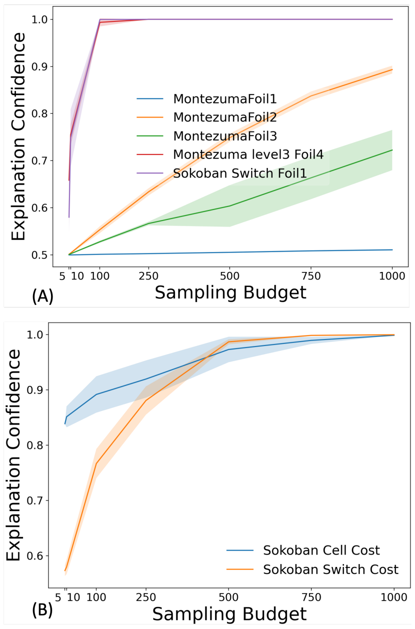


Figure 15.5: The Average Probability Assigned to the Correct Model Component by the Search Algorithms, Calculated over Ten Random Search Episodes with Std-deviation.

(CNNs) for the classifier (which mapped state images to concepts). The classifiers had an average accuracy of 99.46% (Sokoban-switch) and 99.34% (Sokoban-cell).

Computational Experiments: We ran the search to identify preconditions for Montezuma's foils and Sokoban-switch and cost function for both Sokoban variants. From the original list of concepts, we doubled the final concept list used by including negations of each concept (20 each for Montezuma and 36 and 64 for Sokoban variants). The probabilistic models for each classifier were calculated from the corresponding test sets. For precondition identification, the search was run with a cutoff probability of 0.01 for each concept. In each case, our search-based methods were able to identify the correct model component for explanations. Figure 15.5(A), presents the probability our system assigns to the correct missing precondition plotted against the sampling budget. It presents the average calculated across ten random episodes

	Prefers symbols	Average Likert-score	P-value
Precondition	19/20	3.47	1.0×10^{-8}
Cost	16/20	3.21	0.03

Table 15.1: Results from the user study for hypothesis H1

(which were seeded by default using urandom (Linux, 2013)). We see that in general, the probability increases with the sampling budget with a few small dips due to misclassifications. We had the smallest explanation likelihood for foil1 in screen 1 in Montezuma (with 0.511 ± 0.001), since it used a common concept, and its presence in the executable states was not strong evidence for them being a precondition. Though even in this case, the system correctly identified this concept as the best possible hypothesis for precondition as all the other hypotheses were eliminated after around 100 samples. Similarly, Figure 15.5(B) plots the probability assigned to the abstract cost functions.

User study: With the basic explanation generation methods in place, we were interested in evaluating if users would find such an explanation helpful. All study designs followed our local IRB protocols. We were interested in measuring the effectiveness of the symbolic explanations over two different dimensions: (a) whether people prefer explanations that refer to the symbolic model components over a potentially more concise and direct explanation and (b) whether such explanations help people get a better understanding of the task (this mirrors the recommendation established by works like Hoffman *et al.* (2018)). All study participants were graduate students (different from those who specified the concepts), the demographics of participants can be found in A.

For measuring the preference, we tested the hypothesis

Method	# of Participant	Average Time Taken (sec)	Average # of Steps
Concept-Based	23	43.78 ± 12.59	35.87 ± 9.69
Saliency Map	25	134.24 ± 61.72	52.64 ± 11.11

Table 15.2: Results from the user study for hypothesis H2

H1*: People would prefer explanations that establish the corresponding model component over ones that directly presents the foil information (i.e. the failing action and per-step cost)*

This is an interesting hypothesis, as in our case, the explanatory text for the baseline is a lot more concise (in terms of text-size). As per the selectivity principle (Miller, 2017a), people prefer explanations that doesn't contain any extraneous information and thus we are effectively testing here whether people find the model information extraneous. The exact screenshots of the conditions are provided in A. We used a *within-subject* study design where the participants were shown an explanation generated by our method along with a simple baseline. Precondition case involved pointing out the failing action and the state it was executed and for the cost case the exact cost of executing each action in the foil was presented. The users were asked to choose the one they believed was more useful (*the choice ordering was randomized to ensure the results were counterbalanced*) and were also asked to report on a five-point Likert scale the completeness of the chosen explanation (1 being not at all complete and 5 being complete). For precondition, we collected 5 responses per each Montezuma foil, and for cost we did 10 per each sokoban variant (40 in total). Table 15.1, presents the summary of results from the user study and supports H1. In fact, a binomial test shows that the selections are statistically significant with respect to $\alpha = 0.05$.

For testing the effectiveness of explanation in helping people understand the task, we studied

H2: *Concept-based precondition explanations help users understand the task better than saliency map based ones.*

We focused saliency maps and preconditions, as saliency map based explanation could highlight areas corresponding to failed precondition concepts (especially when concepts correspond to local regions within the image). *Currently we are unaware of any other existing explanation generation method which can generate explanation for this scenario without introducing additional knowledge about the task.* We measured the user's understanding of the task by their ability to solve the task by themselves. We used a between-subject study design, where each participant was limited to a single explanation type. Each participant was allowed to play the precondition variant of the sokoban-switch game. They were asked to finish the game within 3 minutes and were told that there would be bonuses for people who finish the game in the shortest time. They were not provided any exact instructions about the dynamics of the game. During the game, if the participant performs an action whose preconditions are not met, they are shown their respective explanation. Additionally, the current episode ends and they will have to restart the game from the beginning. One group of users were provided the precondition explanations generated through our method, while the rest were presented with a saliency map. For the latter, we used a state of the art saliency-map based explanation method (Greydanus *et al.*, 2018). The participants were told that highlighted regions are parts of the state an AI agent would focus on if it was acting in that state. In all the saliency map explanations, the highlighted region included the precondition region of switch (shown in A). In total, we collected 60 responses but had to discard 12 due to the fact they reported not seeing the explanations. Table 15.2 presents the average time taken and steps taken to complete the task (along with 95% confidence intervals). As seen, the average value is much smaller for concept explanation. Additionally, a two-tailed t-test shows the results are statistically significant (p-values of 0.021 and 0.038 against a significance level $\alpha = 0.05$ for time and steps respectively).

15.7 Concluding Remarks

We view the approaches introduced in the chapter as the first step towards designing more general post hoc symbolic explanation methods for sequential decision-making problems. We facilitate the generation of explanations in user-specified terms for contrastive user queries. We implemented these methods in multiple domains and evaluated the effectiveness of the explanation using user studies and computational experiments. As discussed in (Kambhampati *et al.*, 2022), one of the big advantages of creating such symbolic representation is that it provides a symbolic middle layer that can be used by the users to not only understand agent behavior, but also to shape it. Thus the user could use the model as a starting point to provide additional instructions or to clarify their preferences. Section A contains more detailed discussion on various future works and related topics, including its applicability to settings with stochasticity, partial observability, temporally extended actions, the process of collecting more concepts, identifying confidence threshold and possible ethical implications of using our post-hoc explanation generation methods.

Part IV

INTERPRETABLE BEHAVIOR GENERATION

Chapter 16

PART-IV OVERVIEW

The previous three parts of the thesis were focused on the question of how the robot can explain its decisions, in many cases the optimal one with respect to its task cost, when it doesn't align with the human's expectation about the most preferred decision. However, the robot being an autonomous decision-maker is free to choose what plan will follow and in cases where the robot is interacting with the human, the system may need to make considerations that go beyond the cost of the plan. One particular aspect the system may need to look at while choosing its plans is the overhead of explaining a given decision. Not all robot plans, even if one were to limit themselves to optimal plans, are equivalent in the effort required to explain them. This means that when the robot is choosing its plans, it could help simplify the interaction by choosing to follow plans that are easier to explain. However, in many cases the choice to follow an easier to explain plan may come at a cost to the robot, as it may be harder for the robot to follow such an easier to explain plan as compared to an optimal plan. This part of the thesis will be focused on introducing a class of planning algorithms that are able to choose plans, while being able to make the tradeoff between the inherent cost of the plan and the overhead associated with explaining the plan. We will end this part of the thesis by revisiting the general notion of generating interpretable behavior. We will establish a unified Bayesian model to understand the entire landscape of interpretable behavior generation works. This framework could act as a basis for any future methods that can generate similar planners that can balance communication with consideration across any interpretability measures.

16.1 Structure for Part III and Technical Contributions

This part will be divided into three chapters

1. Chapter 17: This chapter will introduce the basic problem of planning in the presence of diverging expectations from an observer. This chapter will formalize the notion of balanced plans and discuss various forms of balanced plans with differing characteristics. We will establish the complexity of planning in this context. We will in particular put forth and evaluate an approximate algorithm to generate balanced optimal plans. This chapter will also present some preliminary user studies that were performed to evaluate balanced plans.

2. Chapter 18: In this chapter, we introduce a novel planning formulation that will present a single unified planning formulation for balanced planning. Under this formulation, explanations are treated simply as another set of actions, but ones with epistemic effects. This means the tradeoff between the cost of explanations and the cost of a plan can be performed directly by the planner. We also empirically show that this compilation is more computationally efficient than the formulation introduced in Chapter 17.

3. Chapter 19: In this chapter, we will take a step back to take a look at the entire landscape of interpretable behavior generation. So in addition to explicability we will look at the measures like predictability and legibility. In this chapter, we will introduce a single Bayesian reasoning framework and by extension a planning formalism that allows the agent to consider all these diverse interpretability measures simultaneously.

16.2 Important Takeaways

One of the important takeaways from the series of works is the placement of balanced planning and by extension model-reconciliation explanations in the realm of epistemic planning. As discussed earlier, since the introduction of this relation between the two, other works have tried to connect explanations in general to epistemic reasoning (cf. (Shvo *et al.*, 2020)). While one would be hard pressed to find an exact analogy to balanced planning within XAI literature, there are works that argue for choosing models with potentially lower accuracy while improving the overall efficiency of the human-AI team (cf. (Bansal *et al.*, 2021)). One could argue that when one uses inherently interpretable models at the cost of overall system performance, one is engaged in performing a similar trade-off.

Another point that is worth keeping in mind is that the methods discussed in this part, particularly in Chapters 17 and 18, assume that the human is an optimal reasoner and there exists no vocabulary mismatch between the human and the robot. So there is both a scope and need to develop versions of these methods that can handle cases where these assumptions are not necessarily met. Also while Chapter 19 establishes a starting point for developing similar balancing algorithms for other interpretability measures, the problem of developing effective planning systems for those settings is an open one.

Chapter 17

BALANCING EXPLANATIONS AND EXPLICABILITY

In the previous chapters, we looked at cases where when the robot's optimal decision doesn't match the human's expectations then the robot can try to update the human expectations through explanations. However, explanation generation only constitutes one strategy the robot could pursue. The other being, the possibility for the robot to follow the plan that aligns with the human's expectations. This has been traditionally referred to as explicable planning. Usually, these two processes of plan explanations and explicability have been treated as two independent ways of addressing this mismatch in expectations. Following the convention set by Chapter 1, we have focused on cases where the robot selects a plan π^*_R, such that there exist $\pi' \prec_{\mathcal{M}^R_h} \pi^*_R$. Under explicable planning we will try to select π^H or plans close to π^H, such that $\forall \pi, \pi^H \preceq_{\mathcal{M}^R_h} \pi$.

Additionally, there are situations where a combination of both provide a much better course of action – if the expected human plan is too costly in the planner's model (e.g. the human might not be aware of some safety constraints) or the cost of communication overhead for explanations is too high (e.g. limited communication bandwidth). Consider, for example, a human working with a robot that has just received a software update allowing it to perform new complex maneuvers. Instead of directly trying to conceive all sorts of new interactions right away that might end up spooking the user, the robot could instead reveal only certain parts of the new model while still using its older model (even though suboptimal) for the rest of the interactions so as to slowly reconcile the drifted model of the user. In this chapter, we will thus define a planning framework that will bring these two techniques together.

Following Chapter 4, we will assume that the human is a perfect reasoner and capable of identifying the optimal plan for a given plan. Additionally, we will assume that there doesn't exist any vocabulary mismatch between the human and the robot and we will focus on deterministic planning problems.

17.1 Illustrative Example

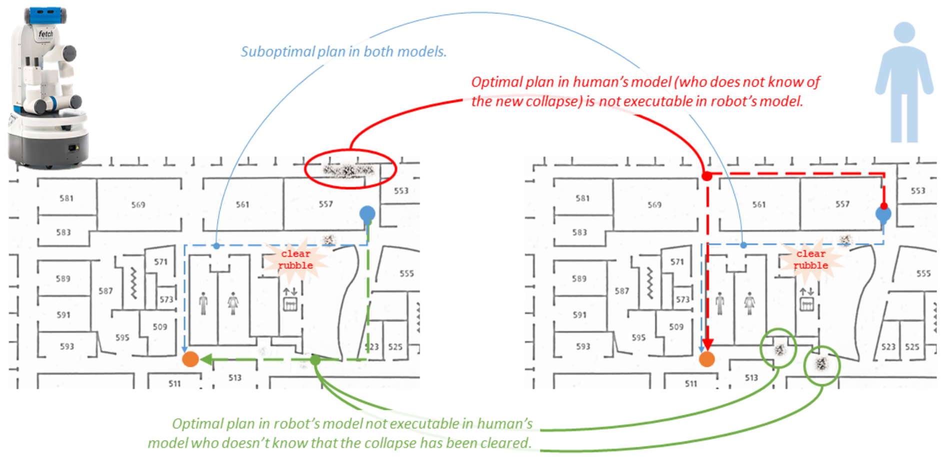

Suboptimal plan in both models.
Optimal plan in human's model (who does not know of the new collapse) is not executable in robot's model.
clear rubble
clear rubble
Optimal plan in robot's model not executable in human's model who doesn't know that the collapse has been cleared.

Figure 17.1: A Demonstration of the Explicability-explanation Trade-off. Video Link: https://youtu.be/Yzp4FU6Vn0M.

We illustrate our approach on a robot performing an Urban Search And Reconnaissance (USAR) task – here a remote robot is put into disaster response operation controlled partly or fully by an external human commander. This is a typical USAR setup (Bartlett, 2015) where the robot's job is to infiltrate areas that may be otherwise harmful to humans, and report on its surroundings as and when required or instructed by the external. The external usually has a map of the environment, but this map is no longer accurate in a disaster setting – e.g. new paths may have opened up or older paths may no longer be available due to rubble from collapsed structures like walls and doors. The robot however may not need to inform the external of all these changes so as not to cause information overload of the commander who may be otherwise engaged in orchestrating the entire operation. This requires the robot to reason about the model differences due to changes in the map, i.e. the initial state of the planning problem.

Figure 17.1 shows a relevant section of the map of the environment where this whole scenario plays out. A video demonstration can be viewed at `https://youtu.be/Yzp4FU6Vn0M`. The dark marks indicate rubble that has blocked a passage. A lot of rubble cannot be removed. The robot (Fetch), currently located at the position marked with a blue **O**, is tasked with taking a picture at location marked with an orange **O**. The commander expects the robot to take the path shown in red, which is no longer possible. The robot has two choices – it can either follow the green path and explain the revealed passageway due to the collapse, or compromise on its optimal path, clear the rubble and proceed along the blue path. The first part of the video demonstrates the plan that requires the least amount of explanation, i.e. the most explicable plan. The robot only needs to explain a single initial state change to make its plan optimal in the updated map of the commander:

```
remove-has-initial-state-clear_path p1 p8
```

This is an instance where the plan closest to the human expectation, i.e. the most explicable plan, still requires an explanation, which previous approaches in the literature cannot provide. Moreover, in order to follow this plan, the robot must perform the costly `clear_passage p2 p3` action to traverse the corridor between `p2` and `p3`, which it could have avoided in its optimal (green) path. Indeed, the robot's optimal plan requires the following explanation:

```
add-has-initial-state-clear_path p6 p7
add-has-initial-state-clear_path p7 p5
```

```
remove-has-initial-state-clear_path p1 p8
```

By providing this explanation, the robot is able to convey to the human the optimality of the current plan as well as the infeasibility of the human's expected plan (shown in red).

17.2 Expectation-Aware Planning

We will follow the model definition established in Section 2.1, where a model is given by a tuple $\mathcal{M} = \langle F, A, I, G, C \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, A is a set of actions - and initial and goal states are $I, G \subseteq F$. Action $a \in A$ is a tuple $\langle \mathit{pre}(a), \mathit{add}(a), \mathit{del}(a) \rangle$ where $\mathit{pre}(a), \mathit{add}(a), \mathit{del}(a) \subseteq F$ are the preconditions and add/delete effects, i.e. $\delta_{\mathcal{M}}(s, a) \models \perp \textit{ if } s \not\models \mathit{pre}(a);\ \textit{else } \delta_{\mathcal{M}}(s, a) \models s \cup \mathit{add}(a) \setminus \mathit{del}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. Finally, C is the cost function

The "model" $\mathcal{M}$ of a planning problem includes the action model *as well as the initial and goal states of an agent.* The solution to $\mathcal{M}$ is a sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(I, \pi) \models G$. The cost of a plan π is $C(\pi) = \sum_{a \in \pi} C(a)$ if $\delta_{\mathcal{M}}(I, \pi) \models G$; ∞ otherwise. The optimal plan has cost $C^*_{\mathcal{M}}$. We will use $\Pi^*_{\mathcal{M}}$ to represent the set of all optimal plans for $\mathcal{M}$.

17.2.1 Expectations and Mental Models in Planning

Interfacing with humans adds a new component to classical planning – the mental model of the human. This manifests itself in the form of expectations that the human has of the agent. Such a mental model can be represented as a version of the problem at hand which the agent believes the human is operating with.[1] (Chakraborti *et al.*,

[1]The actual decision making problem may be over a graph, a planning problem, a logic program, etc. Many of the concepts discussed in the chapter, though confined to automated planning, do in fact carry over beyond the actual representation.

2017) This brings us the to an extension to the classical planning paradigm that accepts as inputs the agent model as well as the mental model of the human, in order to account for the expectations of the human in the planning process.

Definition 45. An Expectation-Aware Planning Problem (EA) *is the tuple* $\Psi = \langle \mathcal{M}^R, \mathcal{M}_h^R \rangle$ *where* $\mathcal{M}^R = \langle F^{\mathcal{M}^R}, A^{\mathcal{M}^R}, I^{\mathcal{M}^R}, G^{\mathcal{M}^R}, C^{\mathcal{M}^R} \rangle$ *is the agent's model and* $\mathcal{M}_h^R = \langle F^{\mathcal{M}_h^R}, A^{\mathcal{M}_h^R}, I^{\mathcal{M}_h^R}, G^{\mathcal{M}_h^R}, C^{\mathcal{M}_h^R} \rangle$ *is the human's understanding of the same.*[2]

While the human is under the assumption that $\mathcal{M}_h^R$ is an accurate representation of the task at hand, the model could be different from $\mathcal{M}^R$ in terms of action definitions, the initial state, and the goal. This difference means that plans generated for the model $\mathcal{M}^R$ may have different properties in the mental model $\mathcal{M}_h^R$. For example, a plan π^* that is optimal in $\mathcal{M}^R$ may be considered suboptimal or even un-executable by the human. Existing literature has explored two kinds of solutions to EA problems, as discussed below.

17.2.2 Explicable Plans

An explicable solution to an EA is a plan π (1) executable in the robot's model and (2) closest to the expected (optimal) plan in the human's model –

(1) $\delta_{\mathcal{M}^R}(I^{\mathcal{M}^R}, \pi) \models G^{\mathcal{M}^R}$; and

(2) $C^{\mathcal{M}_h^R}(\pi) \approx C^*_{\mathcal{M}_h^R}$.

[2]This **does not** assume that humans use an explicitly represented symbolic domain to plan. The robot only uses this to represent the information content of that model. It cannot, of course, have direct access to it. There is extensive work on learning such models (cf. (Zhang *et al.*, 2017; Kulkarni *et al.*, 2019a)) and reasoning with uncertainty over them (Sreedharan *et al.*, 2018a). It is true that this estimate might be different from the ground truth. However, an agent can only plan and explain with what it knows.

"Closeness" or distance to the expected plan is modeled here in terms of cost optimality, but in general this can be any metric such as plan similarity. In existing literature (Zhang *et al.*, 2017; Kulkarni *et al.*, 2019a) this has been achieved by modifying the search process so that the heuristic that guides the search is driven by the robot's knowledge of the human mental model. Such a heuristic can be either derived directly from the mental model (Kulkarni *et al.*, 2019a) or *learned* (Zhang *et al.*, 2017) through interactions in the form of affinity functions between plans and their purported goals.

17.2.3 Plan Explanations

The other approach would be to (1) compute optimal plans in the planner's model as usual, but also provide an explanation (2) in the form of a model update to the human so that (3) the same plan is now also optimal in the updated mental model. Thus, a solution involves a plan π and an explanation $\mathcal{E}$ –

(1) $C^{\mathcal{M}^R}(\pi) = C^*_{\mathcal{M}^R}$;

(2) $\widehat{\mathcal{M}}^R_h \longleftarrow \mathcal{M}^R_h + \mathcal{E}$; and

(3) $C^{\widehat{\mathcal{M}}^R_h}(\pi) = C^*_{\widehat{\mathcal{M}}^R_h}$.

A model update, as indicated by the + operator, may include a correction to the belief (goals or state information) as well as information pertaining to the action model itself, as illustrated by Chakraborti *et al.* (2017). As a result of this explanation, the human and the agent both agree that the given plan is the best possible the latter could have come up with. Note that whether there is no solution in the human model, or just a different one, does not make any difference. The solution is still an explanation so that the given plan is the best possible in the updated human model.

On the other hand, if there is no plan in the robot model, the explanation ensures that there is no plan in the updated human model either.

In Chapter 4, we explored many such solutions – including ones that minimize length, called **minimally complete explanations** or MCEs:

$$\min |\mathcal{E}| \text{ such that } \pi \in \Pi^*_{\mathcal{M}_h^R + \mathcal{E}}$$

However, this was done post facto, i.e. the plan was already generated and it was just a matter of finding the best explanation for it. This not only ignores the possibility of finding better plans that are also optimal but with smaller explanations, but also misses avenues of compromise whereby the planner sacrifices its optimality to reduce the overhead of the explanation process. Our approach is capable of both explaining its plans as well as choosing plans that align with the user expectations.

17.3 Balancing Explanation and Explicable Behavior Generation

In this chapter, we propose a "balanced solution" to an EA, with components of both explicability and explanation. This means that a solution may consist of model information to be provided to the observer along with the plan that will be followed by the agent. By allowing the planner to reason about explanations relevant to a given plan, we will effectively create agents that are able to combine the strengths of explicable planning and explanation generation. This method would even allow the agent to fall back to pure forms of these strategies when the situation demands for it. We will call such plans *Balanced Plans* and represent them as a tuple of the form $(\pi, \mathcal{E})$, where π is the plan the agent will be following and $\mathcal{E}$ is the explanatory information the robot will be providing the human.

We will start by establishing the complexity of generating valid balanced plans, i.e., generate $(\pi, \mathcal{E})$ for a given EA problem such that the plan π is valid in both the robot model and the updated human model.

Theorem 11. *For a given EA problem $\Psi = \langle \mathcal{M}^R, \mathcal{M}^R_h \rangle$, where both $\mathcal{M}^R$ and $\mathcal{M}^R_h$ are represented as classical planning problems, the problem of identifying a valid blanced plan for Ψ is PSPACE-complete.*

Proof Sketch. The PSPACE-hardness for creating a balanced plan for EA is easy to establish since the problem of planning with just agent model can be mapped to a specific EA planning scenario where both agent and user have the same model (thus possible explanations are empty). We can establish membership in PSPACE class by showing that there exist a sound and complete compilation from creating balanced plans for EA to a planning problem with conditional effects and disjunctive/negative preconditions that is linear in size of the original planning problems. We can then follow the same proof specified by (Bylander, 1994) to show that the problem of plan existence is still in PSPACE for this class of planning problems. The exact details of the compilation along with the soundness and completeness proofs will be discussed in Theorem 12. □

But in most cases, we need to go beyond just generating valid plans and talk about generating cost-minimizing and even optimal solutions. The general problem of creating balanced plans involves optimizing for the following cost terms.

1. **Plan Cost** $C(\pi)$: The first objective is the cost of the plan that the robot is going to follow.

2. **Communication Cost** $C(\mathcal{E})$: The next objective is the cost of communicating the explanation. Ideally this cost should reflect both the cost accrued at the agent's end (corresponding to the cost of actions that need to be performed to communicate the information) and a cost relating to the difficulty faced by

the human to understand the explanation. For the majority of this chapter, we will use explanation length as a proxy for both aspects of explanation costs ($C(\mathcal{E}) = |\, \mathcal{E} \,|$): i.e. the larger an explanation, the harder it may be to understand for the human.

3. **Penalty of Inexplicability** **($C_{IE}(\pi, \mathcal{M}_h^R + \mathcal{E})$)**: This corresponds to the cost the human attaches to the inexplicability of the generated plan in their updated mental model. We will generally assume this cost to be directly proportional to the inexplicability score related to the plan. As in the case of explicable planning, we could measure inexplicability in terms of several different metrics, but this chapter will mostly focus on measuring inexplicability in terms of the difference between the cost of the current plan in the human's updated model and the cost of the expected plan. We will assume the penalty itself to be directly proportional to the absolute value of this difference.

While in the most general case generating a balanced plan would be a multi-objective optimization, for simplicity, we will assume that we have weight parameters that allow us to combine the individual costs into a single cost. Thus the cost of a balanced plan (which includes both an explanation and a plan), would be given as

$$C((\mathcal{E}, \pi)) = C(\mathcal{E}) + \alpha * C(\pi) + \beta * C_{IE}(\pi, \mathcal{M}_h^R + \mathcal{E}) \tag{17.1}$$

Property 1 A balanced solution is non-unique. This is similar to standalone explicable plans and plan explanations – i.e. there can be many solutions to choose from, even for a given set of weight parameters α and β. Interestingly, solutions that are equally good according to the cost model can turn out to be different in usefulness to the human, as investigated recently by (Zahedi *et al.*, 2019) in the context of plan explanations. This can have similar implications to balanced solutions as well.

Apart from Section 18.3.2, we will rarely look at optimizing this full cost term. Instead, we will look at some special cases of this general optimization problem. Below we will look at three classes of balanced planning problems, each looking at optimizing a successively more constrained version of the cost function defined above.

1. **Optimal Balanced Planning:** The first group obviously correspond to generating plans that optimizes for the cost function provided in Equation 17.1. Obviously the nature of the final solution still rely heavily on the values of the parameters α and β. We will take a look at the impact of these values on the nature of the solution generated in Section 18.3.2.

2. **Balanced Explicable Plans:** The first special case we could consider are the ones where we restrict ourselves to cases where the plan is perfectly explicable, i.e., optimal, in the resultant model. Therefore in our objective we can ignore the inexplicability penalty term and the optimization objective becomes.

$$\min_{(\pi,\mathcal{E})} C(\mathcal{E}) + \alpha * C(\pi)$$

$$\text{subject to } IE(\pi, \mathcal{M}_h^R + \mathcal{E}) = 0$$

   Most of this chapter will focus on generating this type of balanced plans. Note that while the plan may be optimal in the human's updated model, the plan need not be optimal for the robot. Which brings us to the next group of behavior.

3. **Balanced Optimal Plans:** In this case, we constrain ourselves to identifying not only plans and explanations that will ensure perfect explicability, but we also try to ensure that the plans are in fact optimal in the robot model (note that this carries an inherent bias that robot's task level actions are always more

expensive than communication, which need not be true). Thus the objective in this case becomes just

$$\min_{(\pi,\mathcal{E})} C(\mathcal{E})$$

$$\text{subject to } IE(\pi, \mathcal{M}_h^R + \mathcal{E}) = 0$$

$$\text{and } C(\pi, \mathcal{M}^R) = C^*_{\mathcal{M}^R}$$

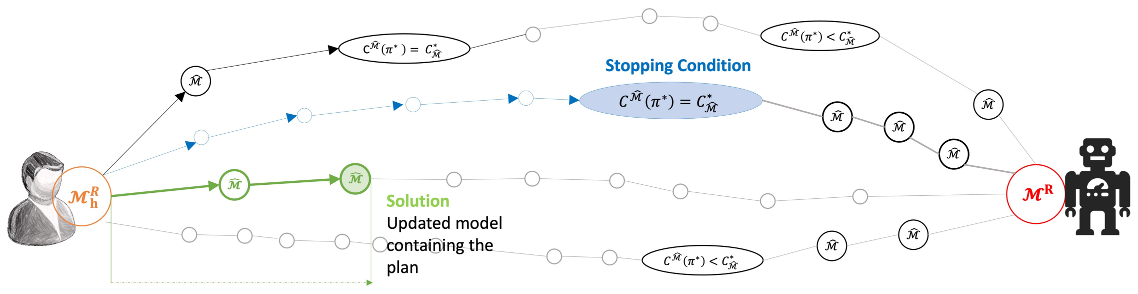
Interestingly, this means identifying the optimal in the robot's model with the MCE with the minimal cost.

In the following sections, we will see how one could generate such balanced plans.

17.3.1 The `MEGA` *Algorithm: Model Space Search*

We employ a *model space* A^* search to compute the optimal Balanced Explicable Plan for a given α.[3] We call this novel planning technique `MEGA` (Multi-model Explanation Generation Algorithm). Similar to Chakraborti *et al.* (2017) we define a state representation over planning problems with a mapping function $\Gamma : \mathcal{M} \mapsto \mathcal{F}$ which represents a planning problem by transforming every condition in it into a predicate. The set Λ of actions contains unit model change actions which make a single change to a domain at a time. Note that our use of '+' operator does not imply that all model reconciliation explanations are additive as $\mathcal{E}$ could include information aimed at correcting user's misconceptions about additional effects or even additional actions that the robot is capable of. We will follow the conventions set by Chakraborti *et al.* (2017) and focus on three main types of model updates:

[3]As in Chapter 4 we assume that the mental model is known and has the same computation power: Chapter 4 also suggests possible ways to address this, the same discussions apply here as well.



$\mathcal{M}^R_h$
$\mathcal{M}^R$
Stopping Condition
$C^{\widehat{\mathcal{M}}}(\pi^*) = C^*_{\widehat{\mathcal{M}}}$
$C^{\widehat{\mathcal{M}}}(\pi^*) < C^*_{\widehat{\mathcal{M}}}$
Solution
Updated model containing the plan

Figure 17.2: Model Space Search to Determine the Best Model to Plan in W.R.T. The Explanation Versus Explicability Trade-off. The Search Stops at the Blue Node Which Houses a Model Where the Generated Plan Is Optimal. The Green Node with the Best Value of the Objective Function Is Then Selected as the Solution.

1. Turn a fluent p true or false in initial state. Represented by the operator {add/remove}-p-from-I.

2. Add or remove a fluent p from the precondition (or add or delete effect) list of an action a. Represented by the operator {add/remove}-p-from-{prec/adds/dels}-of-a.

3. Add or remove a fluent p from the goal list. Represented by the operator {add/remove}-p-from-G.

The algorithm starts by initializing the min node tuple ($\mathcal{N}$) with the human mental model M_h^R and an empty explanation. For each new possible model $\widehat{M}$ generated during model space search, we test if the objective value of the new node is smaller than the current min node. Note that we will specifically focus on cases where the cost of an explanation $\mathcal{E}$ equals to $\mid \mathcal{E} \mid$. We stop the search once we identify a model that is capable of producing a plan that is also optimal in the robot's own model. This is different from the original MCE-search (Chakraborti *et al.*, 2017) where we were trying to find the *first* node where a given plan is optimal. Finally, we select the node with the best objective value as the solution. Algorithm 10 provides details of the model space search process, while Figure 17.2 provides a visual depiction of the same. The search stops upon reaching the blue node (which houses a possible model update where the generated plan is optimal). The green node with the best value of the objective function is then selected as the solution.

17.3.2 Properties of Balanced Explicable Plans and `MEGA`Algorithm

In the following discussion, we will compare and contrast some properties of this balanced approach to planning, especially as it relates to the explanation or explicability only point of views.

Property 2 `MEGA` yields the smallest possible explanation for any plan generated as part of the solution $(\mathcal{E}, \pi)$.

This means that with a high enough α the algorithm is guaranteed to compute the best possible plan for the planner as well as the smallest explanation associated with it. This is by construction of the search process itself, i.e. the search only terminates after the all the nodes that allow $C^{\widehat{\mathcal{M}}_h^R}(\pi) = C^*_{\widehat{\mathcal{M}}_h^R}$ have been exhausted. This is beyond what is offered by the model reconciliation search (cf. Chapter 4), which only computes the smallest explanation *given* a plan that is optimal in the planner's model.

Property 3 $\alpha = \mid \mathcal{M}^R \,\Delta\, \mathcal{M}_h^R \mid$ (i.e. the total number of differences in the models) yields the most optimal plan in the planner's model along with the minimal explanation possible.

This is easy to see, since with $\forall \mathcal{E},\ |\mathcal{E}| \leq \mid \mathcal{M}^R \,\Delta\, \mathcal{M}_h^R \mid$, the latter being the total model difference, the penalty for departure from explicable plans is high enough that the planner must choose from possible explanations only (note that the explicability

Algorithm 10 `MEGA`

```
1: procedure MEGA-SEARCH
2:     Input: EP $\Psi = \langle \mathcal{M}^R, \mathcal{M}_h^R \rangle$, $\alpha$
3:     Output: Plan $\pi$ and Explanation $\mathcal{E}$
4:     fringe $\leftarrow$ Priority_Queue()
5:     c_list $\leftarrow \{\}$                                  ▷ Closed list
6:     $\mathcal{N}_{min} \leftarrow \langle \mathcal{M}_h^R, \{\} \rangle$        ▷ Track node with min. value of obj.
7:     fringe.push($\langle \mathcal{M}_h^R, \{\} \rangle$, priority = 0)
8:     while True do
9:         $\langle \widehat{\mathcal{M}}, \mathcal{E} \rangle, c \leftarrow$ fringe.pop($\widehat{\mathcal{M}}$)
10:        if OBJ_VAL($\langle \widehat{\mathcal{M}}, \mathcal{E} \rangle$) $\leq$ OBJ_VAL($\mathcal{N}_{min}$) then
11:            $\mathcal{N}_{min} \leftarrow \langle \widehat{\mathcal{M}}, \mathcal{E} \rangle$                ▷ Update min node
12:        for $\forall \pi^*_{\widehat{\mathcal{M}}}$ do                    ▷ This is relaxed in optimistic version
13:            if $C^{\mathcal{M}^R}(\pi^*_{\widehat{\mathcal{M}}}) = C^*_{\mathcal{M}^R}$ then    ▷ Search is complete when $\pi^*_{\widehat{\mathcal{M}}}$ is optimal in $\mathcal{M}^R$
14:                $\langle \mathcal{M}_{min}, \mathcal{E}_{min} \rangle \leftarrow \mathcal{N}_{min}$
15:                return $\langle \pi_{\mathcal{M}_{min}}, \mathcal{E}_{min} \rangle$
16:            else
17:                c_list $\leftarrow$ c_list $\cup\ \widehat{\mathcal{M}}$
18:                for $f \in \Gamma(\widehat{\mathcal{M}}) \setminus \Gamma(\mathcal{M}^R)$ do    ▷ Misconceptions in the mental model
19:                    $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{\}, \{f\} \rangle$            ▷ Remove from $\widehat{\mathcal{M}}$
20:                    if $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda) \notin$ c_list then
21:                        fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda)\mathcal{E} \cup \lambda \rangle$, $c+1$)
22:                for $f \in \Gamma(\mathcal{M}^R) \setminus \Gamma(\widehat{\mathcal{M}})$ do    ▷ Missing conditions in the mental model
23:                    $\lambda \leftarrow \langle 1, \{\widehat{\mathcal{M}}\}, \{f\}, \{\} \rangle$            ▷ Add to $\widehat{\mathcal{M}}$
24:                    if $\delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda) \notin$ c_list then
25:                        fringe.push($\langle \delta_{\mathcal{M}_h^R, \mathcal{M}^R}(\Gamma(\widehat{\mathcal{M}}), \lambda)\mathcal{E} \cup \lambda \rangle$, $c+1$)
26:    procedure OBJ_VAL($\langle \widehat{\mathcal{M}}, \mathcal{E} \rangle$)
27:        return $|\mathcal{E}| + \alpha \times |\min_{\pi^*_{\widehat{\mathcal{M}}}} C^{\mathcal{M}^R}(\pi^*_{\widehat{\mathcal{M}}})|$
28:                        ▷ Consider optimal plan in $\widehat{\mathcal{M}}$ that is cheapest in $\mathcal{M}^R$
```

penalty is always positive until the search hits the nodes with $C^{\widehat{\mathcal{M}}_h^R}(\pi) = C^*_{\widehat{\mathcal{M}}_h^R}$, at which point onwards the penalty is exactly zero). In general this works for any $\alpha \geq |MCE|$ but since an MCE will only be known retrospectively after the search is complete, the above condition suffices since the entire model difference is known up front and is the largest possible explanation. In other words, setting $\alpha = \mid \mathcal{M}^R \Delta \mathcal{M}_h^R \mid$ will force `MEGA` to generate Balanced optimal plans.

Property 4 $\alpha = 0$ yields the plan that requires least amount of explanations.

Under this condition, the planner minimizes the cost of explanations only – i.e. it will produce the plan that requires the shortest explanation. This brings us close to the original "explicability only" view of planning by Zhang *et al.* (2017); Kulkarni *et al.* (2019a), where the cost of robot plan is secondary to generating plans that are easier for the human to understand. Though unlike the earlier works in this case we limit ourselves to plans that are perfectly explicable, that is plans that are optimal in the (possibly updated) human model. So if the

Property 5 `MEGA`-search is required only once per problem, and is independent of the hyperparameter α.

The algorithm terminates only after all the nodes containing a minimally complete explanation have been explored. This means that for different values of α, the agent only needs to post-process the nodes with the new objective function in mind. Thus, a large part of the reasoning process for a particular problem can be pre-computed.

Approximate `MEGA`

`MEGA` evaluates executability (in the robot model) of all optimal plans within each intermediate model during search. This is quite expensive. Instead, we implement `MEGA-approx` that does this check only for the first optimal plan that gets computed.

This means that, in Alogirthm 10, we drop the loop (line 16) and have a single opitmality cost (line 38). This has the following consequence.

Property 6 `MEGA-approx` is not complete.

`MEGA-approx` is an optimistic version of `MEGA` and is not guaranteed to find all balanced solutions. This is because in each search node we are checking for whether an optimal plans – the first one that gets computed – is executable in the robot model, and moving on if not. In models where multiple optimal plans are possible, and some are executable in the robot model while others are not, this will result in `MEGA-approx` discarding certain models as viable solutions where a balanced plan was actually possible. The resulting incompleteness of the search means we lose Properties 1 and 3, but it also allows us to compare directly to the methods discussed by Chakraborti *et al.* (2017) where the optimal plan is fixed.

17.4 Empirical Results

We will now provide evaluations of `MEGA-approx` demonstrating the trade-off in the cost and computation time of plans with respect to varying size of the model difference and the hyper-parameter α. We will then report on human factor studies on how this trade-off is received by users. Note that the two flavors of evaluations are done with different motivations. The former evaluates the explicability-explanation trade-off from the perspective of the robot which is able to minimize communication but also the penalty due to explicability. The user study instead evaluates the effect of this on the human.

17.4.1 Illustration of the Cost Trade-off in Balanced Plans

The hyperparameter α determines how much an agent is willing to sacrifice optimality versus the explanation cost. We will illustrate this trade-off on modified versions of two popular IPC[4] domains.

The Rover (Meets a Martian) Domain Here the IPC Mars Rover has a model as described in the IPC domain, but has undergone an update whereby it can carry the rock and soil samples needed for a mission at the same time. This means that it does not need to empty the store before collecting new rock and soil samples anymore so that the new action definitions for `sample_soil` and `sample_rock` no longer contain the precondition `(empty ?s)`.

During its mission it runs across a Martian who is unaware of the robot's expanded storage capacity, and has an older, extremely cautious, model of the rover it has learned while spying on it from its cave. It believes that any time the Rover collects a rock sample, it also needs to collect a soil sample and need to communicate this information to the lander. The Martian also believes that before the rover can perform `take_image` action, it needs to send the soil data and rock data of the waypoint from where it is taking the image. Clearly, if the rover was to follow this model, in order not to spook the Martian it will end up spending a lot of time performing unnecessary actions (like dropping old samples and collecting unnecessary samples) – e.g. if the rover is to communicate an image of an objective `objective2`, all it needs to do is move to a waypoint (`waypoint3`) from where `objective2` is visible and perform the action:

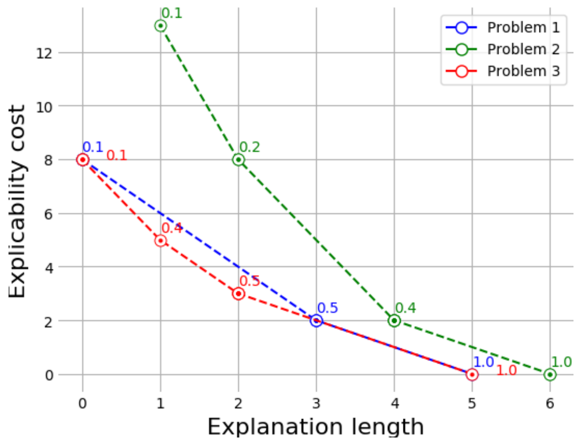
```
(take_image waypoint3 objective2 camera0 high_res)
```

[4]From the International Planning Competition (IPC) 2011: `http://www.plg.inf.uc3m.es/ipc2011-learning/Domains.html`

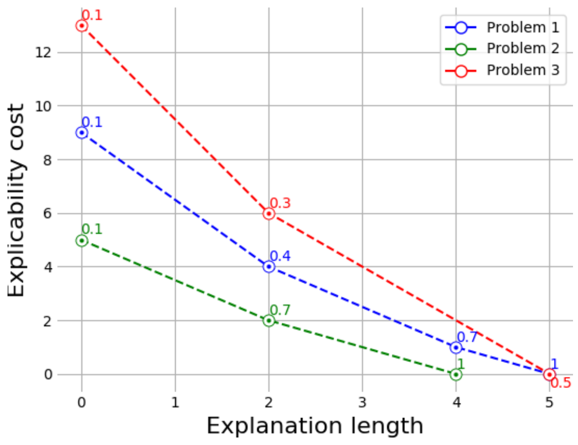
If the rover was to produce a plan that better represents the Martian's expectations, it would look like:

```
(sample_soil store waypoint3)
(communicate_soil_data waypoint3 waypoint3 waypoint0)
(drop_off store)
(sample_rock store waypoint3)
(communicate_rock_data waypoint3 waypoint3 waypoint0)
(take_image waypoint3 objective1 camera0 high_res)
```

If the rover uses an MCE here, it ends up explaining 6 model differences. In some cases, this may be acceptable, but in others, it may make more sense for the rover to bear the extra cost rather than laboriously walk through all updates with an impatient Martian. Figure 17.3 shows how the explicability and explanation costs vary for problem instances in this domain. The algorithm converges to the smallest possible MCE, when α is set to 1. For smaller α, MEGA saves explanation cost by choosing more explicable (and expensive) plans.



(a) The Rover (Meets a Martian) Domain



(b) The Barman (in a Bar) Domain

Figure 17.3: The Explicability Versus Explanation Costs W.R.T. α.

		$\Delta = 2$		$\Delta = 7$		$\Delta = 10$	
		$\lvert\mathcal{E}\rvert$	Time	$\lvert\mathcal{E}\rvert$	Time	$\lvert\mathcal{E}\rvert$	Time
Rover	p1	0	1.22	1	5.83	3	143.84
	p2	1	1.79	5	125.64	6	1061.82
	p3	0	8.35	2	10.46	3	53.22
Barman	p1	2	18.70	6	163.94	6	5576.06
	p2	2	2.43	4	57.83	6	953.47
	p3	2	45.32	5	4183.55	6	5061.50

Table 17.1: Runtime (Sec) and Size of Explanations $\mathcal{E}$ W.R.T. The Size of Model Difference Δ.

The Barman (in a Bar) Domain Here, the brand new two-handed Barman robot is wowing onlookers with its single-handed skills, even as its admirers who may be unsure of its capabilities expect, much like the standard IPC domain, that it needs one hand free for actions like `fill-shot`, `refill-shot`, `shake` etc. This means that to make a single shot of a cocktail with two shots of the same ingredient with three shots and one shaker, the human expects the robot to:

```
(fill-shot shot2 ingredient2 left right dispenser2)
(pour-shot-to-used-shaker shot2 ingredient3 shaker1 left)
(refill-shot shot2 ingredient3 left right dispenser3)
(pour-shot-to-used-shaker shot2 ingredient3 shaker1 left)
(leave left shot2)
(grasp left shaker1)
```

The robot can, however, directly start by picking both the shot and the shaker and does not need to put either of them down while making the cocktail. Similar to the Rover domain, we again illustrate (Figure 17.3) how at lower values of α the robot generates plans that require less explanation. As α increases the algorithm produces plans that require larger explanations with the explanations finally converging at the smallest MCE required for that problem.

Table 17.1 illustrates how the length of explanations computed square off with the total model difference Δ. Clearly, there are significant gains to be had in terms of minimality of explanations and the reduction in cost of explicable plans as a result of it. This is something the robot trades off internally by considering its limits of communication, cost model, etc. We will discuss the external effect of this (on the human) later in the human factors study.

17.5 Results from the User Study

In this section, we will use the USAR domain introduced before to analyze how participants in a user study responded to the explicability versus explanations trade-off. The experimental setup (reproduced here in part of clarity) derives from those used to study the broader details of the model reconciliation process by Chakraborti *et al.* (2019e). Specifically, we set out to test two key hypothesis –

H1. Subjects would require explanations when the robot comes up with suboptimal plans in the mental model.

H1a. Response to balanced plans should be indistinguishable from inexplicable / robot optimal plans.

H2. Subjects would require less explanations for explicable plans as opposed to balanced or robot optimal plans.

As we explained before, **H1** is the key thesis from Chapter 4 that we build on here; it formulates the process of explanation as one of *model reconciliation* to achieve common grounds with respect to a plan's optimality. This forms the basis of incorporating considerations of explanations in the plan generation process as well, as done in the chapter, in the event of model differences with the human in the loop. **H2** forms the other side of this coin and completes the motivation of computing balanced plans. Note that balanced plans would still appear suboptimal (and hence inexplicable) to the human even though they afford opportunities to the robot to explain less or perform a more optimal plan. Thus, we expect (**H1a**) their behavior to be identical in case of both robot optimal and balanced plans.

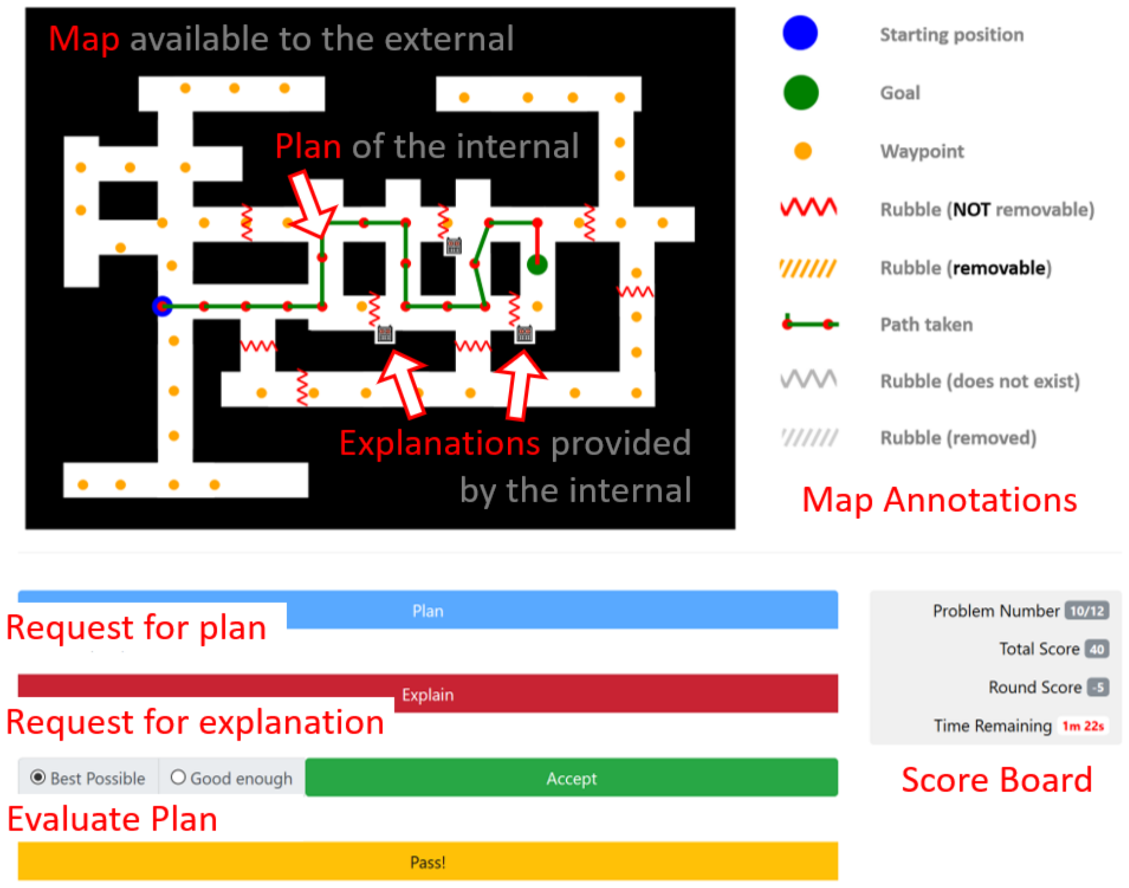


Figure 17.4: Interface for the External Commander in the USAR Domain.

17.5.1 Experimental Setup

The experimental setup exposes the external commander's interface to participants who get to analyze plans in a mock USAR scenario. The participants were incentivized to make sure that the explanation does indeed help them understand the optimality of the plans in question by formulating the interaction in the form of a game. This is to make sure that participants were sufficiently invested in the outcome as well as mimic the high-stakes nature of USAR settings to accurately evaluate the explanations. Figure 17.4 shows a screenshot of the interface which displays to each participant an initial map (which they are told may differ from the robot's actual map), the starting point and the goal. A plan is illustrated in the form of a series of paths through various waypoints highlighted on the map. The participant had to identify if the plan shown is optimal. If unsure, they could ask for an explanation. The explanation was provided in the form of a set of changes to the player's map. The player was awarded 50 points for correctly identifying the plan as either optimal or satisficing. Incorrect identification cost them 20 points. Every request for explanation further cost them 5 points, while skipping a map did not result in any penalty. Even though *there were no incorrect plans in the dataset*, the participants were told that selecting an inexecutable plan as either feasible or optimal would result in a penalty of 400 points, in order to deter them from guessing when they were unsure.

Each subject was paid \$10 as compensation for their participation and received additional bonuses depending on how well the performed (≤ 240 to ≥ 540 points). This was done to ensure that participants only ask for an explanation when they are unsure about the quality of the plan (due to small negative points on explanations) while they are also incentivized to identify the feasibility and optimality of the given plan correctly, with a reward for identifying the correct properties and a penalty for doing this wrongly.

Each participant was shown 12 maps. For 6 of them, they were was shown the optimal robot plan, and when they asked for an explanation, they were randomly shown different types of explanations from Chapter 4. For the rest, they were either shown a (explicable) plan that is optimal in their model with no explanation or a balanced plan with a shorter explanation. We had 27 participants, 4 female and 22 male of age 19-31 (1 participant did not reveal their demographic) with a total of 382 responses across all maps.

17.5.2 Salient Findings

Figure 17.5 shows how people responded to different kinds of plans and their associated explanations. These results are from the two problem instances that included both a balanced and a fully explicable plan. Out of 54 user responses to these, 13 were for explicable plans and 12 for the balanced ones. From the perspective of the human, the balanced plan and the robot optimal plan do not make any difference since both of them appear suboptimal. This is evident from the fact that the click-through rate for explanations in these two conditions are similar (**H1a**) – the high click-through rates for perceived suboptimality conform to the expectations of **H1a**. Furthermore, the rate of explanations is much less for explicable plans as desired (**H2**).

Table 17.2 shows the statistics of the explanations / plans. These results are from 124 problem instances that required MCEs as per Chapter 4, and 25 and 40 instances that contained balanced and explicable plans respectively. As desired, the robot gains in the reduced length of explanations but loses out in the cost of plans produced as it progresses along the spectrum of optimal to explicable plans. Thus, while Table 17.2 demonstrates the explanation versus explicability trade-off from the robot's point of view, Figure 17.5 shows how this trade-off is perceived from the human's perspective.

It is interesting to see that in Figure 17.5 almost a third of the time, participants still asked for explanations even when the plan was explicable, i.e. optimal in their map. This may be an artifact of the risk-averse behavior incentivized by the gamification of the explanation process as well as an indication of the cognitive burden on the

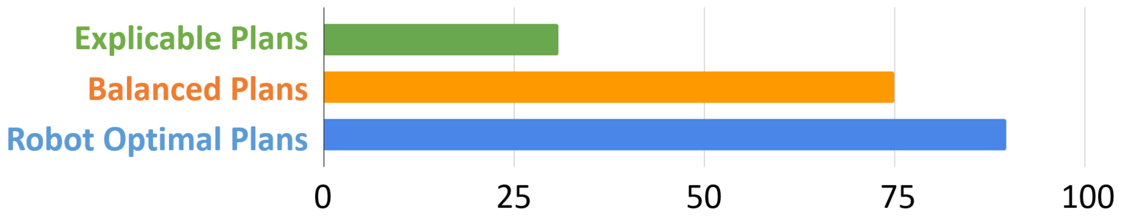


Figure 17.5: Percentage of times Participants in the Study Asked for Explanations for Different Types of Plans, Illustrating Reduced Demand for Explanations for Explicable Plans with No Significant Difference for Robot Optimal and Balanced Plans.

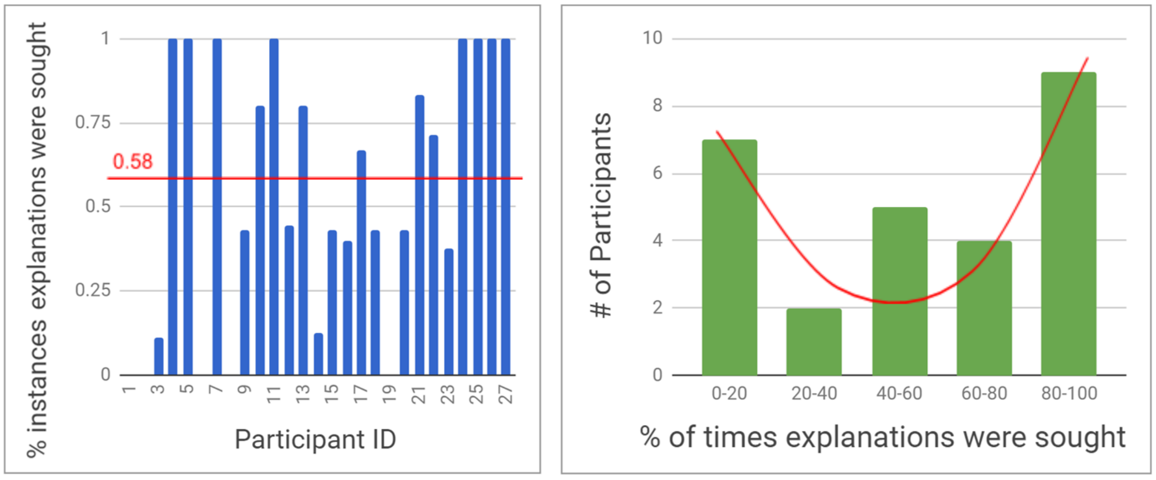


Figure 17.6: Click-through Rates for Explanations.

Optimal Plan		Balanced Plan		Explicable Plan	
$\|\mathcal{E}\|$	$C^{\mathcal{M}^R}(\pi)$	$\|\mathcal{E}\|$	$C^{\mathcal{M}^R}(\pi)$	$\|\mathcal{E}\|$	$C^{\mathcal{M}^R}(\pi)$
2.5	5.5	1	8.5	-	16

Table 17.2: Statistics of Explicability Vs. Explanation Trade-off.

humans who are never (cost) optimal planners. Furthermore, the participants also did *not* ask for explanations around 20-25% of the time when they "should have" (i.e. suboptimal plan in the human model). There was no clear trend here (e.g. decreasing rate for explanations asked due to, perhaps, increasing trust during the course of the experiment) and was most likely due to limitations of inferential capability of humans. Thus, going forward, the objective function must look to incorporate the cost or difficulty of analyzing the plans and explanations from the point of view of the human in addition to that in `MEGA`(4) and Table 17.2 modeled from the perspective of the robot.

Finally, in Figure 17.6, we show how the participants responded to inexplicable plans, in terms of their click-through rate on the explanation request button. Figure 17.6(left) shows the % of times subjects asked for explanations while Figure 17.6(right) shows the same w.r.t. the number of participants. They indicate the variance of human response to the explicability-explanations trade-off. Such information can be used to model the α parameter to situate the explicability versus explanation trade-off according to preferences of individual users. It is interesting to see that the distribution of participants (right) seem to be bimodal indicating that subjects are either particularly skewed towards risk-averse behavior or not, rather than a normal distribution of responses to the explanation-explicability trade-off. This is somewhat counter-intuitive and against expectations (**H1**) and further motivates the need for learning α interactively.

17.6 Concluding Remarks

This chapter introduces the basic framework of expectation-aware planning and introduces the notion of a balanced solution. We looked at how one could generate such balanced solutions, however the central focus in this chapter was introducing and empirically evaluating an approximate solution. However, in the next chapter we will see a method that is a complete approach for generating balanced solutions. This chapter also presents the results from a user study to evaluate the balanced solutions. One of the limitations of this method and one that will also be present in the techniques discussed in the next section is the fact that the explanation and explicability techniques considered are described only for cases where the human confusion comes purely from an asymmetry in knowledge. Going forward, we will need to develop methods that can also take into account asymmetry in inferential capabilities and can also support cases where there may be a vocabulary mismatch.

Chapter 18

SYNTHESIZING AND EXECUTING SELF-EXPLAINING PLANS

In the previous chapter, we established a basic expectation-aware planning framework and introduced the notion of balanced planning. In this chapter, we will look at an alternate solution technique to solving expectation-aware planning problems. One that leads to what may be best described as *self-explaining plans* with the plan now containing actions that are responsible for explaining the rest of the plan. Such explanations may be delivered by purely communicative actions (thereby allowing for explanations as studied by Chakraborti *et al.* (2017)) that are meant to update the human's mental model or task level actions that could also have epistemic side effects (thereby allowing for actions of the type studied by Kwon *et al.* (2018)). Additionally, the framework allows for selecting plans that aligns with human expectations whenever possible. The method will leverage previous formulations that have been proposed to compile epistemic planning problems into classical planning problems. We will also empirically show how this approach provides a computational advantage over our earlier approaches that rely on search in the space of models.

18.1 Background

We will again focus on deterministic planning problems defined by the tuple $\mathcal{M} = \langle F, A, I, G, C \rangle$ (Section 2.1). We will use $\Pi^*_{\mathcal{M}}$ to represent the set of all optimal plans for $\mathcal{M}$. To simplify the notations, we will use $a(s)$ to capture the effect of executing an action at a state and $\pi(s)$ to capture the effect of executing the plan π. Similar to Chapter 9, we will capture the different action definitions in different models using a parameterized entailment notation $\models_{\cdot}$.

Our setting involves an agent that makes decisions using its own model $\mathcal{M}^R = \langle F, A^R, I^R, G_R, C\rangle$ while a human evaluates the plan using their mental model $\mathcal{M}^R_h = \langle F, A^R_h, I^R_h, G^R_h, C\rangle$. For ease of discussion, we concentrate on the specific case where conditions for actions only consist of conjunction of positive literals and the actions have the same cost in both models. While the human is under the assumption that $\mathcal{M}^R_h$ is an accurate representation of the task at hand, the model could be different from $\mathcal{M}^R$ in terms of action definitions, the initial state, and the goal. This difference means that plans generated for the model $\mathcal{M}^R$ may have different properties in the mental model $\mathcal{M}^R_h$. For example, a plan π^* that is optimal in $\mathcal{M}^R$ may be considered suboptimal or even un-executable by the human.

When model asymmetry becomes a source of confusion for the observer, explaining the plan must involve bridging this gap. We can leverage *explanations as model reconciliation* (cf. Chapter 4) that focus on providing enough information that the current plan has required properties (such as executability, optimality, etc.) in the updated model. When the agent is aware of $\mathcal{M}^R_h$, it can use this knowledge to figure out the minimal (where minimality of explanations is defined with respect to an explanation cost C_E) information it needs to provide to achieve the required properties. For example, the problem of identifying explanations for establishing optimality of a given plan π thus becomes:

$$\text{argmin}_{\mathcal{E}}(C_E(\mathcal{E}))$$

$$\text{such that } \pi \in \Pi^*_{\mathcal{M}^R_h+\mathcal{E}}$$

where $\mathcal{E}$ is a set of model information about the agent to be provided to the user as explanation (this could include truth value of fluents in initial state, presence or absence of literals in preconditions/effects, etc.) and $\mathcal{M}^R_h + \mathcal{E}$ is the updated user model after the explanation. Note that our use of '+' operator does not imply that all model reconciliation explanations are additive as $\mathcal{E}$ could include information aimed at correcting user's misconceptions about additional effects or even additional actions that the robot is capable of. We will follow the conventions set by Chakraborti *et al.* (2017) (and Chapter 4) and as in the previous chapter focus on three main types of model updates

(1) {add/remove}-p-from-I

(2) {add/remove}-p-from-{prec/adds/dels}-of-a

(3) {add/remove}-p-from-G

This focuses on cases where the agent is explaining its plan to the human after generating it. The flip side would be to try generating plans that are tailored for the human model. This is referred to as explicable planning and the most basic version of this problem can be formulated as:

$$\text{argmin}_{\pi}(C(\pi))$$

$$\text{such that } \pi(I^R) \models G_R \text{ and } \pi(I_h^R) \models G_h^R$$

This computes a plan that is executable in the agent model and the human mental model with the lowest cost. Our approach is capable of both explaining its plans as well as choosing plans that align with the user expectations. Before delving into details, we briefly introduce the search and rescue domain from Chakraborti *et al.* (2019f) which we will use as an illustrative example for the rest of the chapter.

18.2 Our Running Example: Search & Rescue

A typical Urban Search and Rescue (USAR) scenario consists of an autonomous robot deployed to a disaster scene with an external commander who is monitoring its activities. Both agents start with the same model of the world (i.e the map of

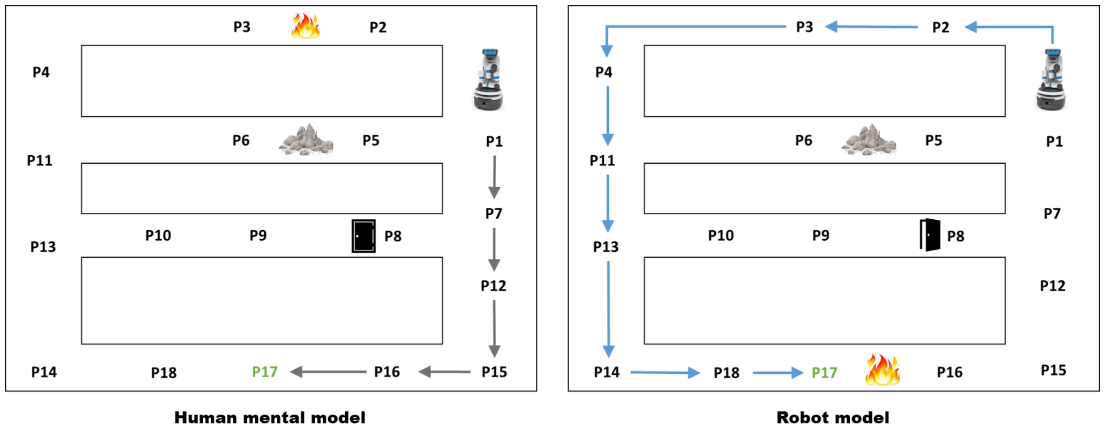


Figure 18.1: Illustration of the Robot Model and the Corresponding Mental Model of the Human. The Robot Starts at P1 and Needs to Go to P17. The Human Incorrectly Believes That the Path from P16 to P17 Is Clear and the One from P2 to P3 Is Blocked Due to Fire. Both Agents Know That There Is Movable Rubble Between P5 and P6 Which Can Be Cleared with a Costly Clear_passage Action. Finally, in the Mental Model, the Door at P8 Is Locked While It Is Unlocked in the Model for the Robot Which Cannot Open Unlocked Doors.

the building before the disaster) but the models diverge over time since the robot, being internal to the scene, has access to updated information about the building. This model divergence could lead to the commander incorrectly evaluating valid plans from the robot as sub-optimal or even unsafe. One way to satisfy the commander would be to communicate or explain changes to the model that led the robot to come up with those plans in the first place.

Figure 18.1 illustrates a scenario where the robot needs to travel from P1 to its goal at P17. The optimal plan expected by the commander is highlighted in grey in their map and involves the robot moving through waypoint P7 and follow that corridor to go to P15 and then finally to P16. The robot knows that it should in fact be moving to P2 – its optimal plan is highlighted in blue. This disagreement rises from the fact that the human incorrectly believes that the path from P16 to P17 is clear while that from P2 to P3 is blocked.

If the robot were to follow the explanation scheme established by Chakraborti *et al.* (2017), it would stick to its own plan and provide the following explanation:

```
> remove-(clear p16 p17)-from-I
    (i.e. Path from P16 to P17 is blocked)
> add-(clear p2 p3)-to-I
    (i.e. Path from P2 to P3 is clear)
```

If the robot were to stick to a purely explicable plan (Zhang *et al.*, 2017) then it can choose to use the passage through P5 and P6 after performing a costly clear_passage action (this plan is not optimal in either of the models).

18.2.1 Self-explaining Plans as Solutions to EA

One of the main challenges of compiling an EA problem to a traditional planning problems is to allow for a way to handle the identification of model updates and to account for the effect of these model updates on the user's expectation. A good way to go about this would be by acknowledging that that if the observer is actually watching the agent executing a plan, these explanations can delivered through and hence modeled as communicative or *explanatory actions*. These actions can, in fact, be seen as actions with epistemic effects in as much as they are aimed towards modifying the human mental model (knowledge state). This means that a solution to an EA planning problem can be seen as *self-explaining plans*, in the sense that some of the actions in the plan are aimed at helping people better understand the rest of it.

This puts EA planning squarely in the purview of epistemic planning, but the additional constraints enforced by the setting allow us to leverage relatively efficient methods to solve the problem at hand. These constraints include facts such as: all epistemic actions are public, modal depth is restricted to one, modal operators only applied to literals, for any literal the observer believes it to be true or false and the robot is fully aware of all of the observer beliefs.

Model updates in the form of epistemic effects of communication actions also open up the possibility of other actions having epistemic *side effects*. The definition of EA makes no claims as to how the model update information is delivered. It is quite possible that actions that the agent is performing to achieve the goal (henceforth referred to as task-level actions to differentiate it from primary epistemic communication actions) itself could have epistemic side-effects. This is something people leverage to simplify communication – e.g. one might avoid providing prior description of some skill they are about to use when they can simply demonstrate it. So one of our goals with the compilation is to allow for such epistemic side effects; a factor that has previously been not considered in any of the earlier works. This consideration also enables us to also capture task level constraints that may be imposed on the communication actions.

Compilation to classical planning.

To support such self-explaining plans, we adopt a formulation that is similar to the one introduced by Muise *et al.* (2015) to compile reasoning about epistemic states into a classical planning problem. In our setting, each explanatory action can be viewed as an action with epistemic effects. One interesting distinction to make here is that the mental model now not only includes the human's belief about the task state but also their belief about the robot's model. This means that the planning model will need to separately keep track of (1) the current robot state, (2) the human's belief regarding the current state, (3) how actions would effect each of these (as humans may have differing expectations about the effects of each action) and (4) how those expectations change with explanations.

Given the model reconciliation planning problem $\Psi = \langle \mathcal{M}_R, \mathcal{M}_H \rangle$, we will generate a new planning model $\mathcal{M}_\Psi = \langle F_\Psi, A_\Psi, I_\Psi, G_\Psi, C_\Psi \rangle$ as follows $F_\Psi = F \cup F_\mathcal{B} \cup F_\mu \cup \{\mathcal{G}, \mathcal{I}\}$, where $F_\mathcal{B}$ is a set of new fluents that will be used to capture the human's belief about the task state and F_μ is a set of meta fluents that we will use to capture the effects of explanatory actions and $\mathcal{G}$ and $\mathcal{I}$ are special goal and initial state propositions. We will use the notation $\mathcal{B}(p)$ to capture the human's belief about the fluent p. We are able to use a single fluent to capture the human belief for each (as opposed to introducing two new fluents $\mathcal{B}(p)$ and $\mathcal{B}(\neg p)$) as we are specifically dealing with a scenario where the human's belief about the robot model is fully known and human either believes each of the fluent to be true or false. In this case, we also do not require any of the additional rules that were employed by Muise *et al.* (2015) to ensure that the state captures the deductive closure of the agent beliefs.

F_μ will contain an element for every part of the human model that can be changed by the robot through explanations. A meta fluent corresponding to a literal ϕ from the precondition of an action a takes the form of $\mu^+(\phi^{pre(a)})$, where the superscript $+$ refers to the fact that the clause ϕ is part the precondition of the action a in the robot model (for cases where the fluent represents an incorrect human belief we will be using the superscript $-$).

For every action $a = \langle pre(a), add(a), del(a) \rangle \in A_R$ and its human counterpart $a_h = \langle pre(a_h), add(a_h), del(a_h) \rangle \in A_H$, we define a new action

$$a_\Psi = \langle pre(a_\Psi), add(a_\Psi), del(a_\Psi) \rangle \in \mathcal{M}_\Psi$$

whose precondition is given as:

$$pre(a_\Psi) = pre(a_R) \cup \{\mu^+(\phi^{pre(a)}) \rightarrow \mathcal{B}(\phi) \\ |\phi \in pre(a_R) \setminus pre(a_H)\} \\ \cup \{\mu^-(\phi^{pre(a)}) \rightarrow \mathcal{B}(\phi)|\phi \in pre(a_H) \setminus pre(a_R)\} \\ \cup \{\mathcal{B}(\phi)|\phi \in pre(a_H) \cap pre(a_R)\}$$

The important point to note here is that at any given state, an action in the augmented model is only applicable if the action is executable in robot model and the human believes the action to be executable. Unlike the executability of the action in the robot model (captured through unconditional preconditions) the human's beliefs about the action executability can be manipulated by turning the meta fluents on and off. The effects of these actions can also be defined similarly by conditioning them on the relevant meta fluent. In addition to these task level actions (represented by the set A_τ), we can also define explanatory actions (A_μ) that either add $\mu^+(*)$ fluents or delete $\mu^-(*)$.

Special actions a_0 and a_∞ that are responsible for setting all the initial state conditions true and checking the goal conditions are also added into the domain model. a_0 has a single precondition that checks for $\mathcal{I}$ and has the following add and delete effects:

$$add(a_0) = \{\top \rightarrow p \mid p \in I_R\} \cup \{\top \rightarrow \mathcal{B}(p) \mid p \in I_H\} \\ \cup \{\top \rightarrow p \mid p \in F_{\mu^-}\}$$

$$del(a_0) = \{\mathcal{I}\}$$

where F_{μ^-} is the subset of F_μ that consists of all the fluents of the form $\mu^-(*)$. Similarly, the precondition of action a_∞ is set using the original goal and adds the proposition $\mathcal{G}$.

$$pre(a_\infty) = G_R \cup \{\mu^+(p^G) \rightarrow \mathcal{B}(p) \mid p \in G_R \setminus G_H\} \cup$$
$$\{\mu^-(p^G) \rightarrow \mathcal{B}(p) \mid p \in G_H \setminus G_R\} \cup \{\mathcal{B}(p) \mid G_H \cap G_R\}$$

Finally the new initial state and the goal specification becomes $I_\mathcal{E} = \{\mathcal{I}\}$ and $G_\mathcal{E} = \{\mathcal{G}\}$ respectively. To see how such a compilation would look in practice, consider an action (move_from p1 p2) that allows the robot to move from point p1 to p2 only if the path is clear. The action is defined as follows in the robot model:

```
(:action move_from_p1_p2
    :precondition (and (at_p1)
                       (clear_p1_p2))
    :effect (and (not (at_p1)) (at_p2) ))
```

Let us assume the human is aware of this action but does not care about the status of the path (as they assume the robot can move through any debris filled path). In this case, the corresponding action in the augmented model and the relevant explanatory action will be:

```
(:action move_from_p1_p2
  :precondition
  (and
    (at_p1)
    (B((at_p1))) (clear_p1_p2)
    (implies
     (μ+_prec(move_from_p1_p2, (clear_p1_p2)))
     (B((clear_p1_p2))) ))
    :effect (and  (not (at_p1)) (at_p2)
           (not B(at_p1)) B(at_p2)) ))
```

```
(:action  explain_$\mu^{+}_{prec}$_move_from_clear
   :precondition  (and  )
   :effect  (and  $\mu^{+}_{prec}$(move_from_p1_p2 ,
                       (clear_p1_p2))  ))
```

Finally C_Ψ captures the cost of all explanatory and task level actions. For now we will assume that the cost of task-level actions are set to the original action cost in either robot or human model and the explanatory action costs are set according to C_E. Later, we will discuss how we can adjust the explanatory action costs to generate desired behavior.

We will refer to an augmented model that contains an explanatory action for each possible model updates and has no actions with effects on both the human's mental model and the task level states as the *canonical augmented model.*

Given an augmented model, let $\pi_\mathcal{E}$ be a plan that is valid for this model ($\pi_\mathcal{E}(I_\Psi) \subseteq G_\Psi$). From $\pi_\mathcal{E}$, we extract two types of information – the model updates induced by the actions in the plan (represented as $\mathcal{E}(\pi_\mathcal{E})$) and the sequence of actions that have some effect of the task state represented as $\mathcal{D}(\pi_\mathcal{E})$ (we refer to the output of $\mathcal{D}$ as the task level fragment of the original plan $\pi_\mathcal{E}$)). $\mathcal{E}(\pi_\mathcal{E})$ may contain effects from action in $\mathcal{D}(\pi_\mathcal{E})$. This brings us to our next theorem.

Theorem 12. *For a given EA problem $\Psi = \langle \mathcal{M}_R, \mathcal{M}_H \rangle$ the corresponding augmented model $\mathcal{M}_\Psi$ is a sound and complete formulation: (1) for every valid π for $\mathcal{M}_\Psi$ the tuple $\langle \mathcal{E}(\pi), \mathcal{D}(\pi) \rangle$ is a valid solution for Ψ and (2) for every valid solution $\langle \mathcal{E}^\Psi, \pi \rangle$, there exists a corresponding valid plan for π' for $\mathcal{M}_\Psi$ such that $\mathcal{D}(\pi') = \pi$ and $\mathcal{E}(\pi') = \mathcal{E}^\Psi$.*

Proof Sketch. The soundness of plans generated from $\mathcal{M}_\Psi$ are guaranteed by the construction of the model as all the preconditions of the actions in the updated user model have to be met in the current plan. To see why the formulation is complete, consider a solution $< \mathcal{E}^\Psi, \pi >$ for Ψ. From the procedure for constructing $\mathcal{M}_\Psi$ we know that there must exist an explanatory action for each possible model difference. This means that there should exist a sequence of explanatory actions $\langle a_1, .., a_k \rangle$ that results in the same model updates captured by $\mathcal{E}^\Psi$. It is easy to see that $\langle a_1, .., a_k \rangle + \pi$ is a valid plan for $\mathcal{M}_\Psi$ hence proving the assertion. $\square$

The planner can automatically find positions of the explanatory actions, but to avoid any confusion that may arise from belief revisions on the users' end, we can enforce some common sense ordering like making any explanation related to an action to appear before the first instance of that action. This ordering will make sure that users are not confused about earlier action effects and also helps reduce branching, making planning more efficient.

Stage of Interaction and Epistemic Side Effects:

One of the important parameters of the problem setting that we have yet to discuss is whether the explanation is meant for a plan that is proposed by the system (i.e the system presents a sequence of actions to the user) or are we explaining some plan that is being executed either in the real world or some simulation the user (observer) has access to. Even though the above formulation can be directly used for both scenarios, we can use the fact that the human is observing the execution of the plans to simplify the explanatory behavior by leveraging the fact that many of these actions may have epistemic side effects. This allows us to not explain any of the effects of the actions that the human can observe (for those effects we can directly update the believed value of the corresponding state fluent and the meta-fluent).[1]

[1]This means that when the plan is being executed, the problem definition should include the observation model of the human (which we assume to be deterministic). To keep the formula-

This is beyond the capability of any of the existing algorithms in this space of the explicability-explanation dichotomy.

This consideration also allows for the incorporation of more complicated epistemic side-effects wherein the user may infer facts about the task that may not be directly tied to the effects of actions. Such effects may be specified by domain experts or generated using heuristics. Once identified, adding them to the model is relatively straightforward as we can directly add the corresponding meta fluent into the effects of the relevant action. An example for a simple heuristic would be to assume that the firing of a conditional effect results in the human believing the condition to be true. For example, if we assume that the robot had an action (open_door_d1_p3) that had a conditional effect:

```
(when (and (unlocked_d1)) (open_d1))
```

Then in the compiled model, we can add a new effect:

```
(when (and (unlocked_d1))
   (and B(open_d1) B(unlocked_d1)))
```

Even in this simple case, it may be useful to restrict the rule to cases where the effect is conditioned on previously unused fluents so the robot does not expect the observer to be capable of regressing over the entire plan.

18.3 Optimality of the Agent

The compilation explored so far only takes into consideration the expectations the agent has about the safety of the plans (i.e the user would expect any plans generated to be valid and executable) and does not account for the user's expectation

tion simple, we ignore this for now. Including this additional consideration is straightforward for deterministic sensor models.

on whether the agent should act optimally. In the earlier example, if the agent just followed the plan that takes the robot through P5 and P6 with a clear_passage_P5_P6 action with no additional explanatory actions then the user may still be confused why the agent does not just follow the plan that involves going through P16 to P17 that it believes to be cheaper (marked in grey in the human's map).

Even in cases where the action costs are the same for the agent and the human, we cannot account for such expectations by merely generating optimal plans in the augmented model. For example, the optimal plan in the augmented model would be the one through P2 and P3 (the full plan is marked in blue in the robot map) with one extra explanatory action explain_μ_I^+_clear_P2_P3. While the above plan provides an explanation to ensure validity, ensuring the optimality of the resultant plan would require the agent to also explain that the passage from P16 to P17 is blocked, which would clearly be more expensive than choosing the valid plan for any non-zero cost for explanatory actions.

One approach to address this would be to prune all solutions where the task level fragment of the plan ($\mathcal{D}(\pi)$) is suboptimal in the updated human model. A simple way to enforce this would be to extend the planner to perform an optimality test for the current plan during the goal test. It may be possible to use more intelligent pruning to reduce the number of goal tests (e.g. one could leverage the fact that the optimality test never needs to be repeated for the same set of model updates) and we could design heuristics that take into account optimality aspects. In this chapter, we adopt this simple approach as a first step towards modeling these novel behaviors.

18.3.1 Balanced Plans Vs. Agent Optimal Plans

Even when generating plans that preserve the user's expectations about agent optimality, the agent could generate two types of plans: agent optimal plans (Chakraborti *et al.*, 2017) or balanced plans (Chakraborti *et al.*, 2019f). In the first scheme, the agent chooses to select self-explanatory plans whose task level fragment is going to be optimal in the original agent model and then choose the minimal explanations that justifies the optimality plan (i.e the plan is optimal in the user's updated model). Such explanations are referred to as Minimally Complete Explanation or MCE (the agent could also choose among the optimal plans the one that requires the cheapest MCE). An example would be choosing the plan highlighted in blue in robot model and then explaining that the path from P2 to P3 is clear and P16 to P17 is blocked. In the latter scheme, the agent could choose plans that are easiest to explain (here again we need to ensure that after the explanation the plan is optimal in the updated model). For example, in the USAR scenario if communication is expensive, it may be easier to choose the plan to move through P5 and P6 with a clear passage action since we only need to explain that the passage P16 to P17 is blocked.

In the first case, the agent is effectively prioritizing any loss of optimality over any overhead accrued by communicating the explanation, while in the second case the agent accounts for the cost of both the plan it is performing and the explanation cost (the cost of communication and possibly the computational overhead experienced by the user on receiving the explanation). By assigning explanatory costs to explanatory actions we are essentially generating balanced plans but there may be scenarios where the agent needs to stick to its optimal plan. We can generate such agent optimal plans by setting lower explanatory action costs. Before we formally state the bounds for explanatory costs, let us define the concept of *optimality delta* (denoted as $\Delta\pi_{\mathcal{M}}$) for a planning model, which captures the cost difference between the optimal plan and the second most optimal plan. More formally $\Delta\pi_{\mathcal{M}}$ can be specified as:

$$\begin{aligned}\Delta\pi_{\mathcal{M}} = \min\{v \mid v \in \mathbb{R} \wedge \\ &\nexists \pi_1, \pi_2((0 < (C(\pi_1) - C(\pi_2)) < v) \\ &\qquad \wedge \pi_1(I_{\mathcal{M}}) \models_{\mathcal{M}} G_{\mathcal{M}} \wedge \pi_2(I_{\mathcal{M}}) \in \Pi^*_{\mathcal{M}}\}\end{aligned}$$

Theorem 13. *In a canonical augmented model $\mathcal{M}_\Psi$ for an EA planning problem Ψ, if the sum of costs of all explanatory actions is $\leq \Delta\pi_{\mathcal{M}_R}$ and if π is the cheapest valid plan for $\mathcal{M}_\Psi$ such that $\mathcal{D}(\pi) \in \Pi^*_{\mathcal{M}_\Psi+\mathcal{E}(\pi)}$, then:*

(1) $\mathcal{D}(\pi)$ is optimal for $\mathcal{M}_R$

(2) $\mathcal{E}(\pi)$ is the MCE for $\mathcal{D}(\pi)$

(3) There exists no plan $\hat{\pi} \in \Pi^*_R$ such that MCE for $\mathcal{D}(\hat{\pi})$ is cheaper than $\mathcal{E}(\pi)$, i.e. the search will find an the plan with the smallest MCE.

Proof Sketch. We observe that there exists no valid plan π' for the augmented model ($\mathcal{M}_\Psi$) with a cost lower than that of π and where the task level fragment ($\mathcal{D}(\pi')$) is optimal for the human model. Let's assume $\mathcal{D}(\pi) \notin \Pi^*_{\mathcal{R}}$ (i.e current plan's task-level fragment is not optimal in robot model) and let $\hat{\pi} \in \Pi^*_{\mathcal{R}}$. Now let's consider a plan $\hat{\pi}_{\mathcal{E}}$ for augmented model that corresponds to the plan $\hat{\pi}$, i.e, $\mathcal{E}(\hat{\pi}_{\mathcal{E}})$ is the MCE for the plan $\hat{\pi}$ and $\mathcal{D}(\hat{\pi}_{\mathcal{E}}) = \hat{\pi}$. Then the given augmented plan $\hat{\pi}_{\mathcal{E}}$ is a valid solution for our augmented planning problem $\mathcal{M}_\Psi$ (since the $\hat{\pi}_{\mathcal{E}}$ consists of the MCE for $\hat{\pi}$, the plan must be valid and optimal in the human model), moreover the cost of $\hat{\pi}_{\mathcal{E}}$ must be lower than π. This contradicts our earlier assumption hence we can show that $\mathcal{D}(\pi)$ is in fact optimal for the robot model.

Using a similar approach we can also show that no cheaper explanation exists for $\pi_{\mathcal{E}}$ and there exists no other plan with a cheaper explanation. □

Note that while it is hard to find the exact value of the optimality $\Delta\pi_{\mathcal{M}}$, it is guaranteed to be ≥ 1 for domains with only unit cost actions or $\geq (C_2 - C_1)$, where C_1 is the cost of the cheapest action and C_2 is the cost of the second cheapest action, i.e. $\forall a(C_{\mathcal{M}}(a) < C_2 \rightarrow C_{\mathcal{M}}(a) = C_1)$. Thus allowing us to easily scale the cost of the explanatory actions to meet this criteria.

18.3.2 Disallowing Explicable Plans That Are Too Costly

There could be scenarios where forcing the agent to always choose plans that are optimal in the human model may not be the best strategy. For example, there may be cases where we would prefer the agent to deviate from the optimal expected plan if it results in significant gains. The penalty for deviating from the expected optimal plan should thus be another optimization criteria. To avoid complexities arising due to multi-criteria optimization, we will assume the agent is optimizing a single objective function of the form

$$C(\pi) + \alpha * (C_{\mathcal{M}_H+\mathcal{E}(\pi)}(\pi) - C^*_{\mathcal{M}_H+\mathcal{E}(\pi)})$$

where the second term basically captures the difference between the cost of the current plan in the resultant model and the cost of the expected plan and α is some scaling factor to allow linear combination of the two terms. Now with this new objective function we can define an *Optimally Balanced Plan* as a plan that is executable in both robot and the resultant human model and minimizes the above objective.

Definition 46. For a problem Ψ, the tuple $\langle \mathcal{E}^\Psi, \pi^\Psi \rangle$ is said to be the optimally balanced solution if,

1. $\pi^\Psi(I_R) \models G_R$.
2. $\pi^\Psi(I_H) \models_{\mathcal{M}_H} G_H$.

3. $\nexists \langle \hat{\mathcal{E}}, \hat{\pi} \rangle$, such that the tuple satisfies (1.) and (2.), and $C(\hat{\mathcal{E}}) + C_{\mathcal{M}_R}(\hat{\pi}) + \alpha * (C_{\mathcal{M}_h+\hat{\mathcal{E}}}(\hat{\pi}) - C^*_{\mathcal{M}_H+\hat{\mathcal{E}}}) < C(\mathcal{E}^\Psi) + C_{\mathcal{M}_R}(\pi^\Psi) + \alpha * (C_{\mathcal{M}_h+\mathcal{E}^\Psi}(\pi) - C^*_{\mathcal{M}_H+\mathcal{E}^\Psi})$.

To generate such optimal balanced plans, we need to relax the goal requirement that the final plan is optimal in the human model. Instead we can incorporate the inexplicability penalty into the reasoning about the plan, by assigning the cost of a_∞ (the goal action) to be α times the cost difference between the optimal plan in the human model and the current plan. When α is set to zero the problem would just identify the cheapest plan in the original robot model that is executable in the human model. We can also use this formulation to generate plans that are still guaranteed to be optimal in the human model by setting α higher than a threshold κ, where κ is some upper bound on plan length for the robot (that includes explanatory actions).

18.4 Evaluation

Since the nature of our solution has already been validated in literature through human factors evaluation – model reconciliation explanation has been studied by Chakraborti *et al.* (2019e), balanced plans by Chakraborti *et al.* (2019f), explicable plans by Zhang *et al.* (2017); Kulkarni *et al.* (2019a), and the use of physical actions to communicate robot model information by Kwon *et al.* (2018) – we will focus on demonstrating the generality of our framework and studying empirically the performance of the compilation. The code can be found at `http://bit.ly/2Xb7OCp`.

18.4.1 Illustrative Example of Cost-tradeoff

We start by demonstrating how our approach can lead to different solution by altering various costs associated with agent actions. Consider again the USAR domain described earlier: the models for the robot and the user is provided in the appendix (the action for opening a door has an epistemic side effect that the observer would know that the door is unlocked). We start by assigning a cost of 10 to every robot action other than clear-rubble action (which is 50) and the move-through-door action (set to 20). We set the cost of communication action to 1 to start with. The solution produced corresponds to the blue plan in Figure 18.1.

explains_μ^{+}_{init}_clear_p2_p3 ->

explains_μ^{-}_{init}_clear_p16_p17 ->

move_p1_p2-> move_p2_p3-> move_p3_p4->

move_p4_p11->move_p11_p13-> move_p13_p14

-> move_p14_p18-> move_p18_p17

This plan includes the **optimal robot plan and corresponding MCE**. Now if we were to set the cost of communication actions to 100, we see the agent deviating to plans which on their own may not be optimal – e.g. a plan that involves opening the door at P8:

explains_μ^{-}_{init}_clear_p16_p17 ->

move_p1_p7-> move_p7_p8-> opendoor_p8_d1->

movethroughdoor_p8_p9_d1-> move_p9_p10->

move_p10_p13-> move_p13_p14-> move_p14_p18->

move_p18_p17

Here the robot does not have to explicitly provide a separate explanation for the status of the door, but still needs to explain that the path from P15 to P16 is blocked. Note that this plan is an example of **a balanced plan** that leverages **epistemic side effects**. Now we go one step further and relax the need to assure optimality of the plan in the human model by changing it from a hard constraint to just a penalty. This gets us the exact same plan as above but without the explanation about the blocked corridor from P15 to P16, thus allowing a notion of soft explicability.

18.4.2 Runtime Complexity

Next we establish how our approach compares in terms of runtime to previous work. In particular, we will use as reference the optimistic and approximate version of the balancing approach by Chakraborti *et al.* (2019f) that identifies only one optimal plan per search node and the search ends as soon as it finds a node where the optimal plan produced has the same cost as the robot plan and is executable in the robot model. This means all the solutions we generate are guaranteed to be better (in terms of cost) than that generated by the other. For comparison, we selected five IPC domains and for each domain, we created three unique models by introducing 10 random updates in the model, except in the case of Gripper and Driverlog where only 5 were removed. Each of these five domains were paired with five problem instances and then tested on each of the possible configurations. Each instance was run with a limit of one hour, all explanatory actions were restricted to the beginning of the plan and the cost of explanatory actions were set to be twice the cost of original action. Table 18.1 lists the time taken to solve each of these problems. For calculating the average runtime, we used 3600 secs as the stand in for the runtime of all the instances that timed out. We used h_max (admissible) as the heuristic for all the configurations.

The table shows that the new approach does better than the original method for generating balanced plans for most of the domains. Gripper seems to be the only domain, where model search seems to perform slightly better but this is also a domain that had the smallest number of model differences. This indicates that the ability to leverage planning heuristics can make a marked difference in domains with a large number of explanatory actions.

	New Compilation		Model Space Search	
	coverage	**runtime**	**coverage**	**runtime**
Blocksworld	13/15	**569.38**	13/15	2318.73
Elevator	**15**/15	**59.20**	1/15	3382.462
Gripper	5/15	2301.90	**6**/15	**2093.54**
Driverlog	**4**/15	**2740.38**	2/15	3158.59
Satellite	**2**/15	**3186.93**	0/15	3600

Table 18.1: Coverage and Average Runtime (Sec) for Explanations Generated for a Few Standard IPC Domains.

18.5 Related Work

We end with a review of existing literature and emphasize the key differentiators for our framework.

Epistemic Planning

It is well understood in social sciences that explanations must be generated while keeping in mind the beliefs of the agent receiving the explanation (Miller, 2017a). As such, epistemic planning makes for an excellent framework for studying the problem of generating these explanations. While the most general formulation of epistemic planning has been shown to be undecidable, many simpler fragments have been identified (Bolander *et al.*, 2015). In human-aware planning settings too, there is wide consensus that epistemic planning could be an extremely useful tool. Readers can refer to Miller (2017b) for an overview of works done in employing epistemic planning for “social planning”. Recently, there have been a lot of interest in developing efficient methods for planning in such settings (Muise *et al.*, 2015; Kominis and Geffner, 2015, 2017; Le *et al.*, 2018; Huang *et al.*, 2018). Cohen and Perrault (1979) have also investigated the use of speech acts in planning problems. Following the conventions of Cohen and Perrault (1979), the explanatory actions studied within this chapter can be viewed as INFORM acts.

Model Reconciliation

Among the works related to model reconciliation, the work that is most closely connected to this method is the one discussed in Chapter 17. Unfortunately, the actual algorithm study there is incomplete and is not guaranteed to produce the least expensive balanced plan. Even the complete version they hypothesize in their paper relies enumerating all the possible optimal plans for a given updated model, which can be extremely inefficient, particularly since it is expected to be performed for every possible model in the model space. As we see in the empirical evaluations, our method (which is also complete) is often faster against the optimistic approximate version. Moreover the methods discussed in that paper are unable to utilize task-level actions with epistemic side effect or take into account task level constraints for purely communicative actions and the effects of execution on an observer, as we illustrate through examples.

Communicative Actions

Our work also looks at the use of explanatory actions as a means of communicating information to the human observer. The most obvious types of such explanatory action includes purely communicative actions such as speech (Tellex *et al.*, 2014) or the use of mixed reality projections (Chakraborti *et al.*, 2018; Ganesan, 2017). Recent works have shown that physical agents could also use movements to relay information such as intention (MacNally *et al.*, 2018; Dragan *et al.*, 2013) and incapability (Kwon *et al.*, 2018). Our framework allows for a natural trade-off between these different types of communication.

Contrastive Explanations and Inferential Capabilities

Many recent works dealing with explanation generation for planning have looked at characterizing explanations in terms of the types of questions they answer (Fox *et al.*, 2017; Smith, 2012). This characterization is orthogonal to the question of what type of information constitutes valid explanations. Putting aside questions regarding observability, the reason why a user may request an explanation is either due to knowledge mismatch (incomplete or incorrect knowledge of the task) or due to limitations of their inferential capabilities. The answer to any of these questions would require correcting the human's model of the task and/or providing inferential assistance. Works that have looked at model reconciliation explanations have mostly focused on the former. Explanations discussed in this chapter can be viewed as an answer to the question *"Why this plan?"* (which can also be viewed as a contrastive question of the form *"Why this plan and not any other plan?"*). This is not to say that in complex scenarios just the model reconciliation information would suffice, but it would need to be supplemented with information internal to the model that can address the differences in inferential capabilities. Use of abstractions (Sreedharan *et al.*, 2018c), providing refutation of specific foils (Sreedharan *et al.*, 2018c; Valmeekam *et al.*, 2020) and providing causal explanations (Seegebarth *et al.*, 2012) could be used to augment model reconciliation.

Explicable Planning

Explicable planning looks at cases where the agent is incapable of updating the users' expectations and can choose to following the plan that best matches the user expectations and is valid in the robot model. Two representative works in this direction are the ones presented by Zhang *et al.* (2017) and Kulkarni *et al.* (2019a). Zhang *et al.* (2017) investigated scenarios where the human model may be unknown while Kulkarni *et al.* (2019a) proposed an iterative planning formalism that tries to find the most explicable plan by generating all possible valid solutions of a given cost threshold and then tries to find the most explicable plan from that set. Unlike the work presented here they look at more general distance measures for explicability, some of which are based on global plan properties. We can extend our current formulation to take into account such scores by turning these distances into the cost of an extra GOAL action (similar to the balancing formalism that allows for sub-optimality).

18.6 Concluding Remarks

The chapter presents a unifying formulation for the task of planning in the presence of users with incorrect mental models of the planning agent's capabilities. One of the features of the proposed method is the fact that it places expectation-aware planning in the realm of epistemic planning, thereby laying the groundwork to study more complex interaction scenarios including cases with more levels of nesting, uncertainty about mental models, more expressive models, incorporating non-deterministic effects, and so on. Appendix B contains some discussions on how the framework could be further extended to support multiple new novel explanatory features. It would also be worth investigating specific considerations for choosing heuristics or formulating new ones for such problems.

Chapter 19

A UNIFYING BAYESIAN FORMULATION FOR MEASURES OF INTERPRETABILITY

In the previous chapters, we looked at how the agent can account for and leverage communication during explicable behavior generation. However, it is worth noting that explicability is not the only interpretability measure that has been studied in the context of human-robot interaction. Chakraborti *et al.* (2019c) lists predictability and legibility as two other important interpretability measures that an agent should take into account to effectively interact and work with humans. Unfortunately, existing approaches for generating human-aware agent behaviors have considered different measures of interpretability in isolation. Further, these measures have been studied under differing assumptions, thus precluding the possibility of designing a single framework that captures these measures under the same assumptions. In this chapter, we present a unifying Bayesian framework that models a human observer's evolving beliefs about an agent and thereby define the problem of *Generalized Human-Aware Planning.* We will show that the definitions of interpretability measures like explicability, legibility and predictability from the prior literature fall out as special cases of our general framework. Through this framework, we also bring a previously ignored fact to light that the human-robot interactions are in effect open-world problems, with respect to the human's beliefs about the agent. The human may hold beliefs unknown to the agent and may also form new hypotheses about the agent when presented with novel or unexpected behaviors.

19.1 Background

In this chapter, we will be agnostic to specific planning formulations or representations when discussing the agent's model. Instead, we will use the term "model" in a general sense to not only include information about agent actions and transition functions, but also their reward/cost function, goals and initial state. We will assume that the model can be parameterized and use $\theta_i(\mathcal{M})$ to characterize the value of a parameter θ_i for the model $\mathcal{M}$.

Since we are interested in cases where a human is observing an agent acting in the world, we will mainly focus on agent behavioral traces (instead of plans or policies). A behavior trace τ in this context will consist of a sequence of state, action pairs. The likelihood of the sequence given a model will take the form $P_\ell : \mathbf{M} \times \mathcal{T} \rightarrow [0, 1]$, where $\mathbf{M}$ is the space of possible models and $\mathcal{T}$ is the set of behavioral traces that the agent can generate. While we will try to be agnostic to likelihood functions, a fairly common approach Fisac *et al.* (2018); Baker *et al.* (2007) is a noisy rational model based on the Boltzmann distribution: $P_\ell(M, \tau) \propto e^{-\beta \times C(\tau)}$. Where $C(\tau)$ is the cost of the behavior and $\beta \in \mathbb{R}^+$ is a parameter that reflects level of perceived determinism in the agent's choice of plans Baker *et al.* (2009). Note that in our case, a likelihood function captures both the human's expectations about the agent's computational capabilities and their own cognitive limitations. Thus noisy-rational models like the one mentioned above are particularly useful in our scenario. For example, by setting a low β value we could possibly capture the fact that the observer may not be able to correctly differentiate between strategies of relatively similar costs.

For the human-aware scenario, we are dealing with two different models Dragan (2017); Sreedharan *et al.* (2021a); Reddy *et al.* (2018): the model that is driving the agent behavior (denoted $\mathcal{M}^R$) and the human's belief $\mathcal{M}^R_h$ about it. We make no assumptions about whether these two models are represented using equivalent representational schemes or use the same likelihood functions. *This setup assumes that while the human may have expectations about the agent's model, she may have no expectation about its ability to model her. Thus she isn't actively expecting the agent to mold its behavior to what she thinks the agent knows about her, thereby avoiding additional nesting of beliefs.*

19.2 Running Example

In our running example, we will consider a robotic office assistant (Figure 19.1), that can perform various repetitive tasks in the office, including picking up and delivering various objects to employees, emptying trash cans, and so on. Further, we will assume it can only move in three directions: down, left and right; and that it can not revisit a cell. These restrictions allow us to control the set of possible completions of a given plan prefix. You, as the floor manager, are tasked with observing the agent and making sure it is working properly. Given your previous experience, you have come to form expectations about its capabilities and its tasks: e.g. you may think that the goal of the agent is to either deliver coffee or to deliver mail to a room (represented by the door), though you know that there may be other possible goals that you have not considered. Unbeknownst to you, the agent is trying to deliver coffee and it needs to do this while keeping in mind your beliefs about it. This scenario is particularly designed to accommodate the considerations made by prior works on interpretable behaviors. Throughout this chapter, we will revisit this example to show different behaviors.

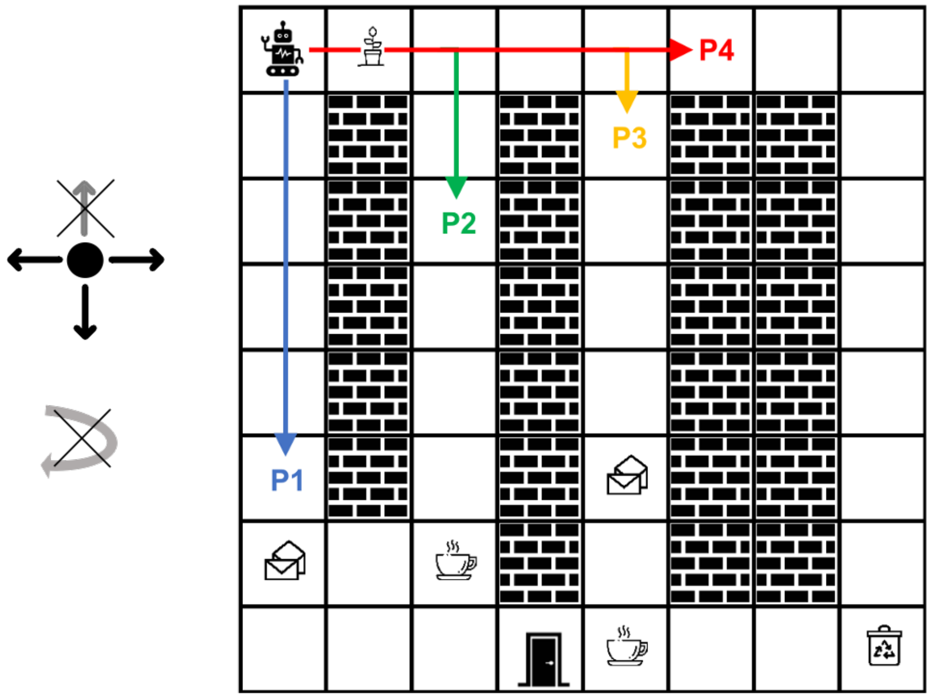


Figure 19.1: An Illustration to Show Different Interpretable Behaviors. Here the Agent Only Moves in Three Directions: Down, Left and Right; And It Does Not Revisit a Cell.

19.3 A Unified Framework

The ability to anticipate and shape a human's beliefs about the agent, is a central requirement for any successful human-aware agent. So we start with a framework to capture the human's reasoning about their beliefs about the agent. In particular, we will adopt a Bayesian model of the human's reasoning process (Figure 19.2). This is motivated by both the popularity of such models in previous works in observer modeling and existing evidence to suggest that people do engage in Bayesian reasoning L Griffiths *et al.* (2008). The node $\mathbb{M}_h^R$ represents possible models the human thinks the agent can have, τ_{pre} corresponds to the behavior prefix that they observed (in this chapter we will assume full observability), and τ_{post} corresponds to possible completions of the prefix.[1]

In addition to explicit models that the human thinks are possible for the agent, we also allow for the possibility that the human may realize that she in fact doesn't know the exact agent model. That is, her previously held beliefs about the agent may not be sufficient to explain or justify the observed behavior. We incorporate this assumption by adding a special model $\mathcal{M}^0$ to the set of models in $\mathbb{M}_h^R$, that corresponds to the hypothesis that the agent model is not one of the models that the human expects.This allows for open-world reasoning since the human can form additional hypotheses about the robot and is not limited by the explicit set she originally has. This strategy of introducing a specific hypothesis that corresponds to a previously unexpected entity has been commonly used to model scenarios where there is a possibility of a novel or previously unknown event happening (c.f. Zabell (1992)). We represent $\mathcal{M}^0$ using a high entropy model: i.e. the likelihood function of this model assigns a small but equal likelihood to any of the possible behaviors, including the ones facilitated by other models. This can be viewed as a model belonging to a random agent. We assume that the human, by default, assigns smaller priors to $\mathcal{M}^0$ than other models. We can now define the following problem:

Definition 47. *A* Generalized Human-Aware Planning Problem *(G-HAP) is a tuple $\Pi_{\mathcal{H}} = \langle \mathcal{M}^R, \mathbb{M}_h^R, P_h^0, P_\ell, C_{\mathcal{H}} \rangle$, where P_h^0 is the human's initial prior over the models in the hypothesis set $\mathbb{M}_h^R$ and $C_{\mathcal{H}}$ is a generalized cost function that depends on the exact objective of the agent.*

A solution to G-HAP consists of a behavior that is valid in $\mathcal{M}^R$ and minimizes $C_{\mathcal{H}}$. In the most general setting, $C_{\mathcal{H}}$ would be a mapping from entire behavior to a cost. Though internally $C_{\mathcal{H}}$ may be a function that takes into account each of

[1]In this chapter, we focus on quantifying these measures for one shot or episodic interactions only, rather than longitudinal ones. In Section 19.4, we discuss more about longitudinal interactions.

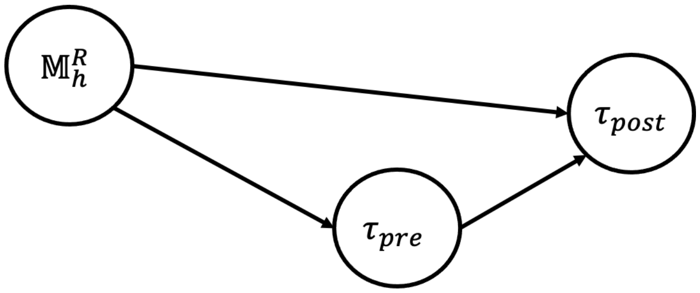


Figure 19.2: Graphical Representation of the Human's Model.

the intermediate steps (not just in $\mathcal{M}^R$ but also the other models in $\mathbb{M}_h^R$). While the exact form would depend on the specific agent objectives, in general the cost function may need to consider (1) costs of the action in the sequence in $\mathcal{M}^R$ and their counterparts in each of the models in $\mathbb{M}_h^R$ (2) the state induced by the action in each model (3) possible completions at each intermediate step and their relation to the actual behavior and (4) the beliefs over $\mathbb{M}_h^R$ it may induce. Rather than investigate the space of all possible cost functions, we will ground the discussion by focusing on scenarios and objectives previously studied in the literature. We will see how this specialization of the framework, naturally gives rise to the specific interpretability measures. Throughout the discussion we will use the notations τ_{pre}^i and τ_{post}^i for a complete behavior τ to represent the behavior prefix that would have been observed and the behavior postfix remaining to be executed for a timestep i respectively. The overall framework presented in the chapter is summarized in Figure 19.3. It illustrates that a human that could hold multiple hypotheses about the agent and show how the various existing measures could be extended to this more general setting. Explicability, in this case, becomes the human's confidence that they can explain the robot behavior with one of the explicit hypothesis they have regarding the robot, while legibility maps to their specific confidence that the robot model actually includes some parameter (which is, in fact, present) and predictability

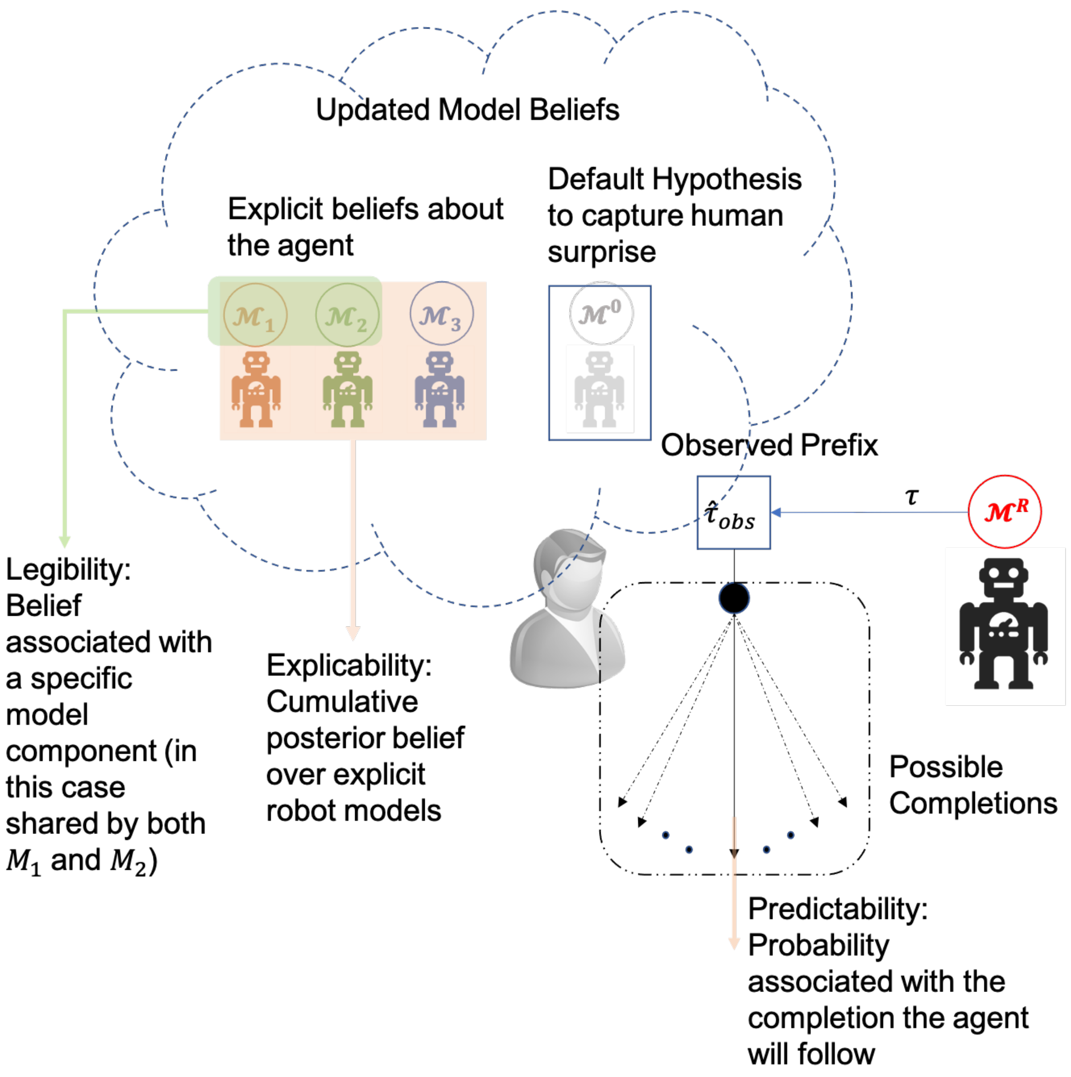

Updated Model Beliefs
Explicit beliefs about the agent
Default Hypothesis to capture human surprise
$\mathcal{M}_1$
$\mathcal{M}_2$
$\mathcal{M}_3$
$\mathcal{M}^0$
Observed Prefix
$\hat{\tau}_{obs}$
τ
$\mathcal{M}^R$
Legibility: Belief associated with a specific model component (in this case shared by both M_1 and M_2)
Explicability: Cumulative posterior belief over explicit robot models
Possible Completions
Predictability: Probability associated with the completion the agent will follow

Figure 19.3: An Overview of Our Unifying Framework. The Human Holds Multiple Hypotheses about the Agent and She Uses the Observed Behavioral Prefix to Update Her Beliefs about the Model. Each of the Interpretability Measure Optimizes for Specific Inferential Outcomes in This Framework.

turns into a measure of confidence they assign to the actual future behavior the robot is going to generate. Below we will look at each of these individual measures in more detail and see how they arise from G-HAP.

19.3.1 Explicability

We will start by looking at cases where the agent wants to avoid behaviors that may confuse the observer about the agent model. That is the human should be able to explain the observed behavior with the explicit models they hold. We will refer to such behaviors as *explicable behaviors*. We can capture the generation of such behavior within our framework by using a cost function that is proportional to the posterior probability associated with model $\mathcal{M}^0$, i.e.,

$$C_{\mathcal{H}}(\tau) \propto \sum_i \alpha_i P(\mathcal{M}^0|\tau^i_{pre}) \tag{19.1}$$

Where $\alpha_i \geq 0$ is the weight associated with each timestep i. This means the formulation would prefer behavior with high likelihood in the explicit models for timesteps with non-zero weight. We will define the explicability score ($\mathcal{E}$) associated with a behavior prefix (τ_{pre}) to be directly proportional to one minus this probability, i.e.,

$$\mathcal{E}(\tau^i_{pre}) \propto \sum_{\mathcal{M} \in \mathbb{M}^R_h \setminus \{\mathcal{M}^0\}} P(\mathcal{M}|\tau^i_{pre}) \tag{19.2}$$

Likelihood functions that assign high probabilities to optimal (or low cost traces), give rise to traces like P1 and P2 in Figure 19.1, since they correspond to optimal plans in the explicit models considered in the example (i.e. the model for delivering coffee or delivering mail).

Reduction to Previous Explicability Definitions

Previous works generally identify a behavior to be explicable if it meets the human's expectation from the agent for the given task Zhang *et al.* (2017). In the binary form this is usually taken to mean that a plan is explicable if it is one of the plans that the human expects from the agent Chakraborti *et al.* (2019f). In the more general continuous form, this expectation is taken to be proportional to the distance between the observed trace and the closest expected behavior Kulkarni *et al.* (2019a); Zhang *et al.* (2017):

$$\tau^*_{\mathcal{E}} = \arg\min_{\tau} \delta(\tau, \tau^E_{\mathcal{M}^R_h}) \tag{19.3}$$

where δ is some distance function between two plans and $\tau^E_{\mathcal{M}^R_h}$ is the closest expected behavior for the model $\mathcal{M}^R$. While there is no consensus on the distance function or expected behavior, a reasonable possibility for the expected set is the set of optimal plans Chakraborti *et al.* (2017) and the distance can be the cost difference Kulkarni *et al.* (2020).

To see how our framework subsumes earlier works, lets start by plugging in the two assumptions made by the original works, namely (1) the human only has one explicit model about the agent (i.e. $\mathbb{M}^R_h = \{\mathcal{M}^R_h, \mathcal{M}^0\}$) and (2) the explicability is measured over the entire plan (i.e. $\alpha_i = 0$ for all i other than the last step). Thus the cost function is dependent only on the explicability of the entire behavior

$$\mathcal{E}(\tau) \propto P(\mathcal{M}^R_h|\tau) \propto P(\tau|\mathcal{M}^R_h) * P(\mathcal{M}^R_h) \tag{19.4}$$

Since the observed prefix is the entire plan, we can directly use the likelihood function

$$\mathcal{E}(\tau) \propto P_\ell(\mathcal{M}^R_h, \tau) * P(\mathcal{M}^R_h) \tag{19.5}$$

Let us consider two plausible likelihood models. First, for a normative model where the agent is expected to be optimal, $P_\ell(\mathcal{M}^R_h, \tau_{pre})$ assigns high but equal probability to all the optimal plans and 0 probabilities for the others. This is the original binary explicability formulation used by Chakraborti *et al.* (2019f,e).

Another possible likelihood function is a noisy rational model Fisac *et al.* (2020) given by:

$$P_\ell(\mathcal{M}_h^R, \tau) \propto e^{-\beta \times C(\tau)} \propto e^{\beta \times C(\tau^*) - C(\tau)} \tag{19.6}$$

where τ^* is an optimal behavior in $\mathcal{M}_h^R$, $C(\tau) \geq C(\tau^*) \geq 0$ for $\mathcal{M}_h^R$. This maps the formulation to the distance based definition as in Kulkarni *et al.* (2020) where a cost-based distance is defined. We can also recover the earlier normative model by setting $\beta \to \infty$ and model $\mathcal{M}^0$ by setting $\beta = 0$.

Going back to the original motivation of explicability, it was meant to capture the human's understanding of the agent's behavior generation process (which includes both its perceived model and computational component). Earlier formulations rely on using the space of expected plans as a proxy of this process. This is further supported by the fact that the works that have looked at updating the human's perceived explicability value of a plan do so by providing information about the model and not by directly telling the human what plans to expect Chakraborti *et al.* (2017, 2019f); Sreedharan *et al.* (2020a). Thus our formulation of explicability directly in terms of the human's beliefs about the agent's model connects to the original motivation of explicability definitions.

Novel Properties of Generalized Explicability

An interesting side-effect of a probability-based explicability formulation is that, the probability of behavior and hence the explicability score can now be affected by the presence or absence of other plans. For example, consider two scenarios, one where $\mathbb{M}_h^R$ contains $\mathcal{M}_1$ and $\mathcal{M}^0$ and another where it contains $\mathcal{M}_2$ and $\mathcal{M}^0$. Now consider a behavior trace τ that is equidistant from an optimal plan in both models $\mathcal{M}_1$ and $\mathcal{M}_2$. Even though they are at the same distance, the trace may be more explicable in the first scenario than in the second, if the second scenario allows for more traces that are closer. Assuming the probability of choosing optimal plans isn't reduced, introducing new plans into the sample space better than the current trace would cause more probability to be assigned to those and thus less to the trace in question. We argue that this makes intuitive sense for explicability since the user should be more surprised in the second scenario as the agent would have ignored many more behaviors that the observer would have considered desirable. Figure 19.4 illustrates such an example.

Proposition 40. *Explicability of a trace is dependent not only on the distance from the expected plans but also on the presence or absence of plans close to the expected plans.*

Here the plan remains explicable whether or not the observation leads to all the probability being assigned to a single model versus being distributed across multiple models. This means that the formulation doesn't require the human to have a single explanation for the behavior, rather it allows their belief to be distributed across multiple hypotheses. While the exact values would depend on the likelihood function, in the office robot scenario our formulation would assign high explicability scores (need not be the same) to both P1 and P2. For P1, the probability mass would be distributed across the two possible hypotheses corresponding to the two goals, while for P2 the probability mass is centered around the model corresponding to the goal to fetch coffee.

Proposition 41. *Explicability is agnostic to whether it is supported by multiple models or by a single one.*

Further, the explicability of a trace is now controlled by the priors on the models. E.g., a trace that is only possible in a model with low prior will not have high

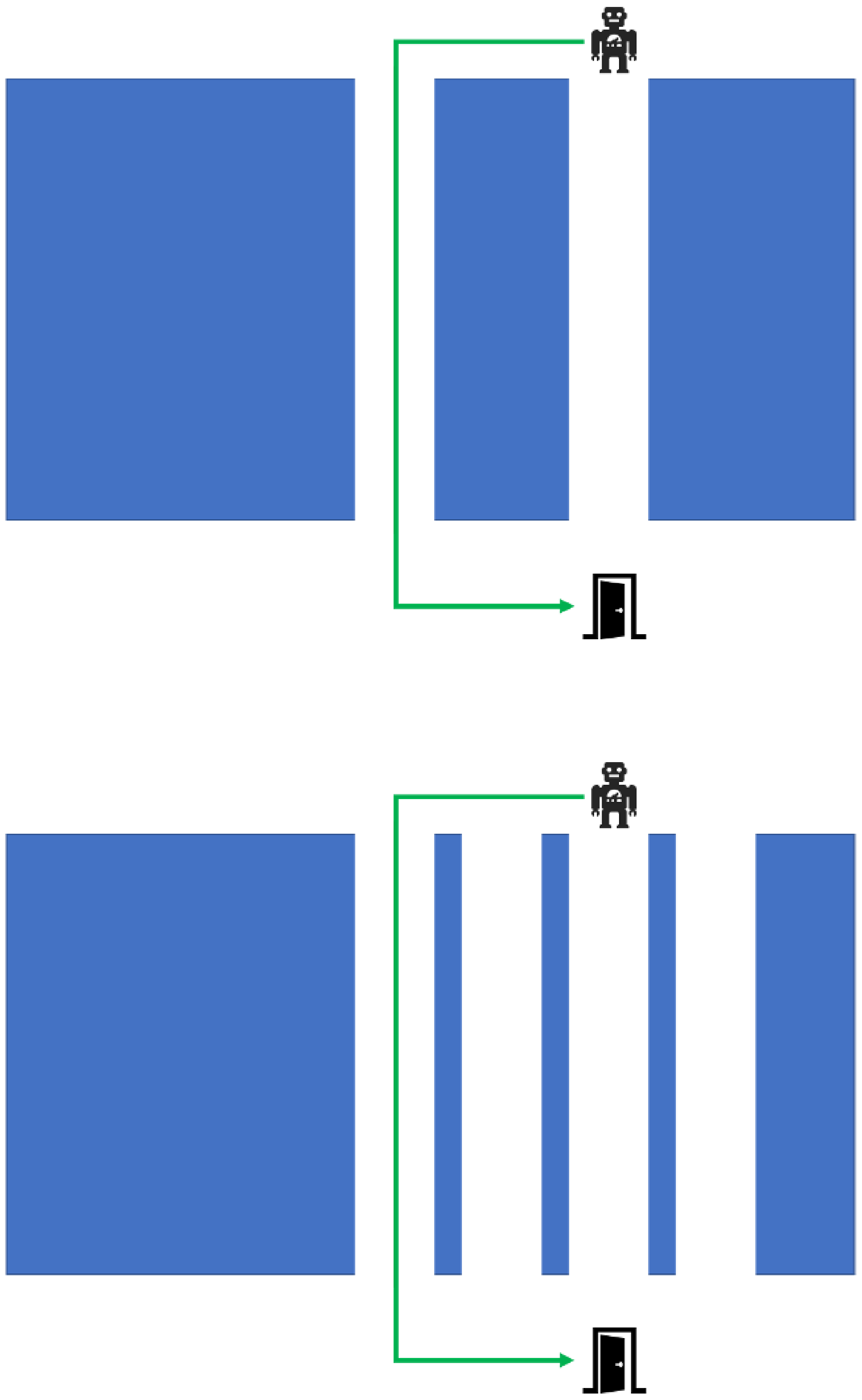


Figure 19.4: A Possible Scenario, Where the Introduction of New Plans Could Cause the Explicability to Drop.

explicability score even if it is highly likely in that model.

19.3.2 Legibility

The next class of behavior is the one where the agent is trying to choose behavior that increases the agent's belief about some component (captured by the parameter θ) of the agent model. Such behavior could be especially important when the achievement of the human's desired outcome is tied directly to the model possessing a certain parameter value. An obvious example would be establishing if the end-goal itself is what the human desires, but this could also be in relation to other model parameters. Thus inducing high confidence in relation to such model parameters in the human's mind could be tied intimately with engendering trust in the human that the agent can achieve the desired objectives.

$$C_{\mathcal{H}}(\tau) \propto \sum_i \alpha_i * (1 - P(\theta = \theta(\mathcal{M}^R)|\tau^i_{pre})) \tag{19.7}$$

That is the cost here becomes the weighted sum of the probability associated with the target parameter having the incorrect value (i.e. different from what is true in the robot model) at each step. Keeping with the existing literature, we will refer to such behaviors as *legible behavior*, with the actual legibility score of a behavior prefix being proportional to the probability of the parameter being the true value

$$\mathcal{L}^\theta(\tau_{pre}) \propto P(\theta = \theta(\mathcal{M}^R)|\tau_{pre}) \tag{19.8}$$

$$\propto \Sigma_{\mathcal{M}\in\mathbb{M}^R_h\setminus\{\mathcal{M}^0\} \text{ Where } \theta(\mathcal{M}^R)=\theta(\mathcal{M})} P(\mathcal{M}|\tau_{pre}) \tag{19.9}$$

We skip $\mathcal{M}^0$ since it doesn't correspond to an explicit model in the human's mind. In the context of Figure 19.1, a plan prefix with high legibility score for the goal of deliver coffee would be P2 as compared to the other options illustrated. Since P1, allows for an optimal completion for both objectives and P3's completions in both models are equally bad. As we will see, while original formulations might assign P4 as a more legible option given the fact that it would assign zero probability to delivering mail, our formulation allows for the possibility that P4 may lead to more probability getting assigned to $\mathcal{M}^0$.

Reduction to Previous Legibility Definitions

Legibility was originally formalized Dragan *et al.* (2013) as the ability of a behavior to reveal its underlying objective. This involves a human who is considering a set of possible goals ($\mathbb{G}$) of the agent and is trying to identify the real goal by observing its behavior. Legibility is, thus, the maximization of the probability of the real goal through behavior:

$$\hat{\tau}^*_{\mathcal{L}} = \underset{\tau_{pre}}{\arg\max}\, P(G^R|\tau_{pre}) \tag{19.10}$$

where G^R is the agent's true goal. While originally introduced in the context of motion planning, this was later adapted to task planning by MacNally *et al.* (2018), and generalized to implicit communication of beliefs when the human has partial observability by Kulkarni *et al.* (2019b) as well as to implicit communication of any model parameter by Miura and Zilberstein (2020).

To keep the discussion in line with previous works, we will focus our attention on communicating end-goals (over arbitrary parameters). Some central assumptions made by earlier works is that the model only differs in terms of the end goal and the actual model is part of the set ($\mathcal{M}^R \in \mathbb{M}^R_h$). Also, the agent is expected to communicate its information as early as possible, so earlier α_i terms are given higher weights than the latter ones. They also assume that at no point would the human consider goals outside the explicit ones she had in mind. That is the possibility that she may be wrong about the original model and that the agent may be possibly trying to achieve something she didn't consider before would never cross her mind. In our framework, this would correspond to assigning a zero prior to $\mathcal{M}^0$. Thus the legibility score here would be

$$\mathcal{L}^\theta(\tau_{pre}) \propto P(M^R|\tau_{pre}) \tag{19.11}$$

A zero prior on $\mathcal{M}^0$ means the agent can create extremely circuitous routes as legible behavior provided the behavior is more likely in the agent model than others. This means that regardless of how suboptimal the plan is in the agent model (or ones with the parameter value), given its even lower probability in other models (or for other parameter values) the agent model will get assigned higher posterior probability and thus higher legibility score. For example in Figure 19.1, the restricted formulation would select the prefix P4 highlighted in red in order to reveal the goal of delivering coffee, eventhough that corresponds to an extremely sub-optimal plan given the set of possible plans.

Novel Properties of Generalized Legibility

A core assumption relaxed by the general formulation is that we now allow for the possibility that the human could be surprised by unexpected behavior and they may form new hypotheses about the agent. If you assume a non-zero prior for $\mathcal{M}^0$, then in cases where the agent presents an extremely suboptimal behavior they have a new hypothesis they can consider. That is they can now shift some of their belief to the fact that they may have been originally wrong about the agent model. Going back to the case of route P4 in Figure 19.1, given how far it is from the optimal, any completion of that prefix would have extremely low likelihood in the model for delivering coffee as opposed to $\mathcal{M}^0$ where that path is as likely as any other. This means our formulation now assigns more weight to $\mathcal{M}^0$ and thus capturing the fact that, when presented with highly unlikely behavior, the observer may question their beliefs about the agent. This brings us to the property

Proposition 42. *Inexplicable plans are also illegible.*

We believe allowing for such uncertainty is essential to capture more realistic human-robot interaction as it is rare for people to have absolute certainty about the agent models (and even discard the possibility that something might have just gone wrong with the agent). Also if we wish to move to a more longitudinal setting, indicating that the human no longer believes in one of the possible hypotheses in the set may not be enough, but we may need to explicitly try to identify what the newly formed hypothesis might be.

19.3.3 Predictability

The final case is one where the agent is interested in communicating to the human the future behavior it will be selecting. In this case, the agent would be required to choose behavior prefixes that allow the human to correctly guess the rest of the behavior the agent will follow with high confidence. This may be useful in cases where the agent may be sharing a workspace with the observer and may want to allow the observer to take into account future agent actions when coming up with their plans.

$$C_{\mathcal{H}}(\tau) \propto \sum_{i} \alpha_i * (1 - P(\tau^i_{post}|\hat{\tau}^i_{pre})) \tag{19.12}$$

This gives us *predictable behavior*. Further, $P(\tau^i_{post}|\hat{\tau}^i_{pre})$ denotes the predictability score for the prefix τ^i_{pre} (with respect to the completion τ^i_{post})

$$\begin{aligned} \mathcal{P}^{\tau^i_{post}}(\tau^i_{pre}) &\propto P(\tau^i_{post}|\tau^i_{pre}) \\ &\propto \sum_{\mathcal{M}\in\mathbb{M}^R_h} P(\tau^i_{post}|\tau^i_{pre}, \mathcal{M}) \times P(\mathcal{M}) \end{aligned} \tag{19.13}$$

From Figure 19.1, a plan prefix with high predictability would be P3. Given the prefix P3, the completion of going down the corridor has the highest likelihood for both the goals. So after marginalizing across all possible models that completion will have high probability and therefore the prefix has high predictability.

Reduction to Previous Predictability Definitions

We need to incorporate two main assumptions into the framework to reduce it to existing definitions of predictability: (1) the human observer only has a single explicit model about the agent and this is equal to the actual agent model $\mathbb{M}_h^R = \{\mathcal{M}^R, \mathcal{M}^0\}$ and (2) the user will not form new hypothesis about the agent regardless of how unexpected the behavior is (i.e. the $\mathcal{M}^0$ prior is zero). Thus we get:

$$\mathcal{P}^\tau(\tau_{pre}) \propto P(\tau' = \tau | \tau_{pre}, \mathcal{M}^R) \tag{19.14}$$

This directly maps to the predictability measure as defined in earlier works Fisac *et al.* (2020). Previous works have also looked at the possibility of generating k step predictable plans, i.e., plans that try to guarantee predictability only after k steps. This allows for the system to choose unlikely prefixes for cases where the agent is only required to achieve required levels of predictability after the first k steps. We can capture such optimization preferences by setting α_i to zero for all but $i = k$. Going back to the example, prefix P3 optimizes for predictability for $k = 5$.

Generalized Predictability

Our generalization introduces two new aspects to the predictability formulation. The fact that the human now considers potential models and we also introduce the new hypothesis $\mathcal{M}^0$. However, the formulation marginalizes out the model and thus, effectively, for a given prefix, the human observer has to consider all the possible completions of the prefix in each of the individual models. Thus even if the trace is perfectly predictable in an individual model, the fact that the human has uncertainty over the models means the prefix may not be predictable. On the other hand, the fact that $\mathcal{M}^0$ assigns equal probability to all the possible completions would mean that the introduction of this new hypothesis would have less of an influence on the resulting predictability score.

19.3.4 Deception and Interpretability

The interpretability measures being discussed involve leveraging reasoning processes at the human's end to allow them to reach specific conclusions. At least for legibility and predictability, the behavior is said to exhibit a particular interpretability property only when the conclusion lines up with the ground truth at the agent's end. But as far as the human is concerned, they would not be able to distinguish between cases where the behavior is driving them to true conclusions or not. This means that the mechanisms used for interpretability could be easily leveraged to perform behaviors that may be adversarial Chakraborti *et al.* (2019c). Two common classes of such behaviors are deception and obfuscation. Deceptive behavior corresponds to behavior meant to convince the user of incorrect information about the agent model or its future plans Masters and Sardina (2017):

$$\mathcal{D}^{\mathbb{M}_h^R}(\tau_{pre}) \propto -1 * P(\mathcal{M}^R|\tau_{pre}) \tag{19.15}$$

Adversarial behaviors meant to confuse the user are either inexplicable plans that increase the posterior on $\mathcal{M}^0$ or, plans that actively obfuscate Keren *et al.* (2016); Kulkarni *et al.* (2019b):

$$\mathcal{O}^{\mathbb{M}_h^R}(\tau_{pre}) \propto H(\mathbb{M}_h^R|\tau_{pre}) \tag{19.16}$$

This is proportional to the conditional entropy of the model distribution given the observed behavior.

With explicability, the question of deceptive behavior becomes interesting, since explicable plan generation is relevant when the actual agent model may not be part of the human's expected set of models (else the agent could just follow its optimal behavior). By choosing to generate plans that align with a non-true model, explicability can be seen as deceptive behavior as it is reinforcing incorrect notions about the agent's model. Such plans would have a high deceptive score per the formulation above (since $P(\mathcal{M}^R|\tau) = 0$). One can argue that explicable behaviors are white lies in such scenarios as the goal here is just to ease the interaction and the behavior is not driven by any malicious intent. One could even further restrict the explicability formulation to a version that only lies by omission by restricting the agent to just behavior optimal in the original agent model. The agent chooses from this set the one that best aligns with the human's expectation. It is a lie by omission in the sense that while the agent has not explicitly been deceptive, by choosing behavior that aligns with the human's expectations, it is maintaining the human's incorrect beliefs.

19.4 Implications of the Framework

Below we briefly discuss several implications of our unifying framework.

Legibility, Explicability. These notions are related to the human's desire to recognize the model Aineto *et al.* (2019)). Our formulation shows that outside limited cases, legibility, and explicability are closely connected. Earlier works have been separating these measures by assuming away either legibility, like in existing explicability works with the human's hypothesis consisting of a single model Zhang *et al.* (2017); Kulkarni *et al.* (2019a), or by assuming away explicability by assigning zero prior on

$\mathcal{M}^0$ for legibility Dragan *et al.* (2013); Dragan and Srinivasa (2013); MacNally *et al.* (2018); Kulkarni *et al.* (2019b); Miura and Zilberstein (2020). Interestingly, in cases where the human is aware that the agent is trying to be legible or more generally they know the agent is trying to model the observer, the human may be more open to suboptimal behavior from the agent as they might attribute it to the agent trying to communicate. However, this does not eliminate $\mathcal{M}^0$ but instead introduces a new level of nesting for reasoning. This comes with all the known complexities and pitfalls of reasoning with nested beliefs Fagin *et al.* (2003). Though studying a limited amount of additional nesting could be important especially in cases where the agent plans to leverage communication. Since communication strategies make the most sense when the human is expecting the agent to model them.

Longitudinal Interactions. Our formulation currently looks at interpretability metrics for one-off interactions only. In cases where a human interacts with the agent for a long period, we can expect the user to start with a uniform distribution over models and a low probability for $\mathcal{M}^0$. In order to take a more long-term view of the human's interaction with the same agent (say, over a time horizon), legibility and predictability measures can be handled by directly carrying over the posterior from each interaction to the next one. However, for explicability more care needs to be taken. For example, Kulkarni *et al.* (2020) hypothesize a possible discounting of inexplicable behavior. The paper argues that after the first inexplicability, a human would be less surprised when similar inexplicable behavior is again presented to her. Part of this discounting can be explained by the human forming new hypothesis that explain the unexpected behavior and using that to analyze future agent behavior. As mentioned earlier, going to a longitudinal setting may require introducing new mechanisms to identify such newly formed hypothesis.

Planning and Environment Design. One of the next logical steps would be to facilitate the generation of plans that maximize the measures described in the chapter. In particular, we could build on the work done in Chapter 18 to encode the human's belief about the task into the planning space. Though in this case, given the possible multiple hypotheses held by the human, we would have to consider a belief space formulation, where the state includes information about the various models and each one is associated with a probability. Now each action of the robot has an effect on the likelihood of each hypothesis. Also unlike the earlier formulations, we can't just have a cost associated with each action or even one that is state-dependent. In fact for measures like predictability, not only does the plan cost for an intermediate step depend on the current state but also the eventual path that will be followed by the robot. As such these costs can only be computed at each goal node, where a cost will be assigned to each step of the path to the goal. These formulations could also be used to design the environment to facilitate easy generation of behavior that naturally aligns with these objectives (similar to Kulkarni *et al.* (2020)).

Goal/Plan/Model Recognition and Interpretability. This chapter focuses on scenarios where the agent is acting in the world, with the knowledge that it is being observed. Though there could well be scenarios where the agent may be the observer and trying to reason about the human's model. In these settings, the agent may be engaged in similar reasoning to what is expected of the human in this chaptter. Chief among them is the case of model recognition Aineto *et al.* (2019) and it's more popular special case of goal recognition Baker *et al.* (2007); Ramírez and Geffner (2009). In a way this could be viewed as the inverse of the legibility as studied in this chapter and is also associated with explicability. Though in most cases these papers assume away the possibility of the agent being surprised by assuming that the candidate hypothesis set contains the target model/goal that generated the behavior. But as the community starts shifting to more open-world cases or allow for the possibility of novel behavior Senator (2019), we will need to allow for the possibility that our agents may come across truly novel and inexplicable behavior (as per previous beliefs) and enable them to detect and update their beliefs from such behaviors. The next related class of abductive reasoning problems that have been studied in the literature is that of plan recognition Kautz and Allen (1986), wherein the agent tries to identify the full plan/behavior from some observations. One could consider this to be the inverse of the predictability problem studied in this chapter.

Generalized Collaborative Behavior. One of the goals of the generalized human-aware planning problem is to establish the fact that specific interpretability behaviors could naturally be generated by an agent capable of reasoning about the human's belief and the impact of its actions on these beliefs. Though studying individual measures are still helpful not only in creating more specialized algorithms for generating them but also for understanding general strategies the agent may engage in. In this vein, we could further generalize most of the interpretability strategies the agent could engage in, to two broad categories, namely, a *model-communication* strategy or a *model-following* strategy. Model-communication involves molding the human's expectation through implicit or explicit communication, to allow the agent to achieve their objectives. On the other hand, the model following strategy involves taking the current understanding of the human and generating behavior that conforms to current human expectations. One could see the agent engaging cyclically in model-communication and model-following behaviors or possibly a combination of the two (for example involving actions that may have epistemic side-effects), wherein the agent may choose to mold the user's expectations to a point where their behavior may be better received by the observer. Legible behavior and explanations Sreedharan *et al.* (2020a) could be understood as specific instances of model communication strategies, while explicability can be seen as an example of a model following behavior. Predictability is a bit harder to place, as at any point the agent is following the most likely plan as per the human's beliefs. Though the agent may have previously engaged in behavior meant to limit its future behaviors (including using techniques like projection Chakraborti *et al.* (2018)). One could definitely argue that these earlier efforts are in fact communicative in so far as they are trying to inform the human about the agent's intentions.

Part V

POLICY SUMMARIZATION

Chapter 20

PART-V OVERVIEW

In the previous chapters of this thesis, we looked at the question of how one could explain the decisions and plans that are to be explained. However, before the explanatory dialogue can start the robot needs to communicate its plan to the user. This is especially important if the explanatory dialogue needs to occur before the robot can execute the plan. This communication becomes even more challenging when one considers more general planning formalisms like stochastic planning, where the solution takes the form of policy which needs to account for the various contingencies that may arise. In this part, we will look at the use of abstraction as a way to effectively communicate policies in such cases.

20.1 Structure for Part III and Technical Contributions

This part will only contain the description of one method

1. Chapter 21: In this chapter, we investigate the utility of temporal abstractions derived through analytically computed landmarks and their relative ordering to build a summarization of policies for *Stochastic Shortest Path Problems.* We formalize the concept of policy landmarks and show how it can be used to provide a high level overview of a given policy. Additionally, we establish the connections between the type of hierarchy we generate and previous works in temporal abstractions, specifically MaxQ hierarchies. The approach is evaluated through user studies as well as empirical metrics that establish that people tend to choose landmarks facts as subgoals to summarize policies and demonstrates

the performance of our approach on standard benchmarks.

20.2 Important Takeaways

I would argue that the problem of summarizing decisions is a problem that is specifically tied to sequential decision-making. After all, in many one-shot decision-making settings like classification or regression the problem of communicating the decision is relatively straightforward. The idea of using abstractions as a way to summarize policies has been considered by other works, particularly in the context of reinforcement learning (cf. (Topin and Veloso, 2019; Zahavy *et al.*, 2016)). However, many of these works tend to characterize these summaries as explanations in themselves. In this thesis, all the explanations presented could be understood as answers to *Why* questions. This is a stance that also aligns with the consensus from social sciences on the nature of everyday explanations (Miller, 2017a). I have thus chosen to separate policy summaries into a distinct category, as this information may be best understood as an answer to a *What* question (specifically *what is the agent trying to do?*). Landmarks have also been used to generate summaries for classical planning problems (Grover *et al.*, 2020a; Sreedharan *et al.*, 2021b). However, these works tend to use problem-level landmarks, which are not plan-dependent. A more interesting parallel between policy landmarks and methods used in classical planning is the one established by (Sreedharan *et al.*, 2022b). This paper shows that policy landmarks and their ordering include information about causal links and one could extract causal links from policy landmarks for deterministic plans.

Chapter 21

TLDR: POLICY SUMMARIZATION FOR FACTORED SSP PROBLEMS USING TEMPORAL ABSTRACTIONS

In this chapter, we will look at ways to effectively present the policy at a high-level of abstraction, while still allowing the users to delve into details as required. In this chapter, we will look at ways to effectively present a summary of the policy at a high-level of abstraction, while still allowing the users to delve into details as required. Our choice of summarization techniques were motivated by three main factors (1) People tend to decompose long horizon planning into sequences of subgoals. This is a well-known fact in psychological literature and has been validated by number of studies (c.f. (Donnarumma *et al.*, 2016; Cooper and Shallice, 2006; Simon and Newell, 1971)); (2) The approach shouldn't assume that the user is an expert in the domain or has prior knowledge about the dynamics of the domain, since the method is a preceding step for explanations; and (3) The approach shouldn't assume it is summarizing optimal policies.

With these design principles in mind we introduce our approach: **T**emporal Abstraction Through **L**an**d**mark **R**ecognition or **TLdR**. TLdR hypothesizes that one way to extract useful temporal abstractions for a given policy is to identify sets of bottleneck or *landmarks facts* and their relative ordering that needs to be satisfied by any valid execution of the given policy. We believe our work represents the first formalization of landmarks for stochastic domains. We also introduce the idea of policy landmarks along with compilation-based methods to generate these landmarks with formal guarantees. This chapter will focus on Stochastic Shortest Path problems (SSPs) since their goal-directed nature is more natural for everyday users (Newell *et al.*, 1972) as well as being more general than infinite horizon discounted MDPs. Once identified, end users can use these landmarks as the basis for generating their explanatory queries (of the form discussed in (Miller, 2017a)) or can further drill down by focusing on specific landmarks to get more details.

The rest of the chapter is structured as follows: Section 21.1 starts with a brief overview of related works in this space and then Section 21.2 introduces the setting and some of the formalisms we will be using throughout the chapter. Section 21.3 introduces a simple illustrative scenario that will act as our running example and Section 21.4 will delve into the details of our methods. Through Section 21.5 we will further investigate the specifics of the hierarchy we are generating and we discuss the evaluations we performed in Section 21.6.

21.1 Related Work

In recent years there has been increasing interest in the problem of explainable AI in general and also specifically for explaining/summarizing policies. While many of these works focus on policies learned through neural networks (cf. (Greydanus *et al.*, 2018)), there are a few works that have considered factored MDP settings as well (cf. (Khan *et al.*, 2009)).

In the context of policy summarization, Lage *et al.* (2019) presents an approach that tries to identify a subset of state action tuples that can help users guess the rest of the policy. The tuple is selected under some assumptions about the computational model the user may be making. Unfortunately, such works that aims to generate summaries optimized for policy completion assume the user has some prior knowledge about the task. Our motivation in this work is to provide summaries that could act as a first step before providing more explanations about the task. Consequently, we address scenarios where the user may be unaware of task details or may misunderstand the the task (similar to explanatory settings studied by Chakraborti *et al.* (2017) and Sreedharan *et al.* (2018c)).

Topin and Veloso (2019), utilize state abstractions to simplify policies. This is completely complementary to our approach and we can use methods described in that paper along with ours to identify landmarks in abstract state spaces (cf. (Sreedharan *et al.*, 2019b) for similar strategies applied to classical planning setting). Hayes and Shah (2017) have also looked at producing summaries specific to user questions.

In MDP literature, people have previously used landmarks in different contexts, for example Ramesh *et al.* (2019) uses the term landmark as a way to denote prototypical state and Kaelbling (1993) uses landmarks to refer to centers of a region of the environment. Options learning literature (cf. (McGovern and Barto, 2001; Stolle and Precup, 2002)) have also used the term bottleneck states to refer to states that appear frequently in valid execution traces. They utilize such states as a basis for learning options and as we will see this is closely related to the techniques discussed in the chapter. The idea of bottleneck states are also related to landmarks as discussed in classical planning literature (Hoffmann *et al.*, 2004).

21.2 Background

We will focus on cases where the planning problem corresponds to an SSP_{s_0}, i.e a stochastic shortest path problem with a single initial state (Kolobov *et al.*, 2012). In the rest of the chapter when we refer to MDPs we will be in fact referring to an SSP_{s_0}. Similar to Section 2.3, the model will be formally defined by the tuple $\mathcal{M} = \langle S, A, T, C, G, s_0 \rangle$, where S is the set of states, A is the set of actions, $T : S \times A \rightarrow [0,1]$ defines the transition function corresponding to the MDP, while C captures the cost function, G the set of goal states, and s_0 the initial state. For this setting, we are generally interested in policies that guarantee that any history sampled from the policy will eventually lead to a goal state with probability one, such policies are generally referred to as proper policies. Moreover, the fact that we are given an initial state means that we only need to identify actions for states that are reachable from from s_0 under π_0, hence we can restrict our attention to partial proper policies (Kolobov *et al.*, 2012).

As with Section 2.3, we will assume a factored representation is provided, such that the state space S can be specified by a set of propositional fluents F. Additionally, we assume the set of goal states G can be concisely described by a subset of propositions $\mathcal{G}$. For example, if we are considering a travel planning task, $\mathcal{G}$ might just include the proposition onboard_plane and the corresponding goal state set G consists of all states where the fluent onboard_plane will be true.

The expected cost of a state is defined here similarly to the standard undiscounted indefinite horizon SSPs and the Bellman optimality equation is provided as

$$J(s) = C(s,a) + \Sigma_{s' \in S} T(s,a,s') * J(s')$$

and all goal states are absorbing states.

A partial policy π_0^* is defined to be optimal if there exists no other partial policy whose expected cost for the initial state is smaller than π_0^* (i.e. $J^{\pi_0^*} \leq J^{\pi}$ for all proper policies π).

We will consider a setting where the underlying MDP $\mathcal{M}$ is known to the explainer and it is tasked with explaining a given policy π. Note that we don't assume π to be optimal, but rather it can be any partial proper policy. To be succinct, in the rest of the chapter we will use policy in place of partial proper policies and we will specifically note any exceptions to the case. Before we delve further into the problem, let us take a quick look at a travel planning domain that will act as our illustrative example for the rest of the chapter.

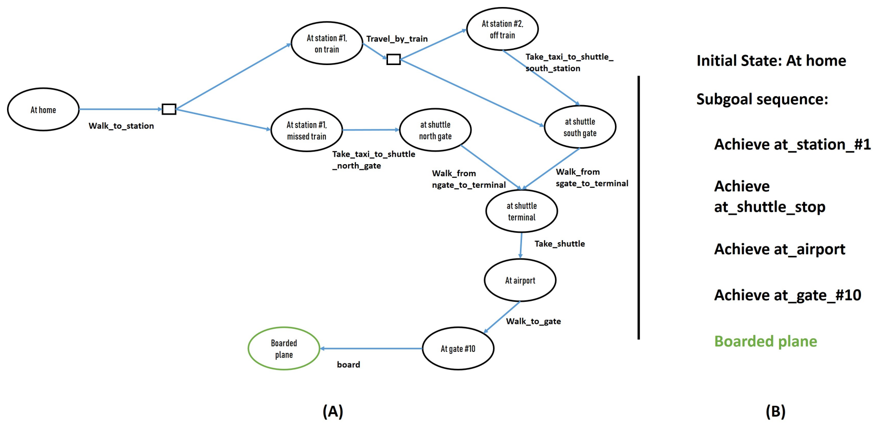


Figure 21.1: The Subfigure (a) Presents Policy Graph Corresponding to Example Detailed in the Illustrative Domain. Here the Edges with Boxes Represents Actions with Stochastic Effects and the Goal Is to Board the Plane. Subfigure (B) Presents One Possible Summarization That Describes a Sequence of Subgoals That Needs to Achieved under the given Policy.

21.3 Motivating Example: Travel Planning Domain

Consider an intelligent personal assistant that is being used to track and plan various daily activities of its users. The personal assistant is capable of gathering information from multiple sources and generating probabilistic models for various events like weather, vehicle delays, traffic, etc.. Many of these models may be too complex for the user to easily understand or the information sources too rapidly changing for the user to keep track off. Now if the user was to ask the agent to come up with a plan that allows them to get to their flight from their home, the agent could easily use the models it generated for various sources to create an MDP and compute a policy that is guaranteed to take the user from their home to the designated flight.

Even in this setting, challenges will arise when the user wants to actually make sense of the policy suggested by the system. The policy could be extremely large, with many branches to handle various contingencies (Figure 21.1 (A) presents a simplified version of such a policy). The system could hardly expect the user to make sense of the policy if it merely dumped the entire policy-graph. Given the fact that the intelligent assistant is almost always available to the user on various devices, the agent could decide to give the user the policy one step at a time, always coming back to the user with the next action to perform given where the user is. This approach could also be extremely unsatisfying as the user would want to know if the policy as a whole aligns with their preferences; being fed the decisions one at a time, prevents them from getting an overall idea about the policy.

A more reasonable strategy would involve first presenting the user with a summary of the policy that highlights some of the important waypoints on the way to the airport, along with the order in which they should be crossed. Figure 21.1 (B) presents one possible set of such waypoints and their order in which they are to be achieved. If the agent were to follow this summarization scheme, the system could report that: you would first need to get to "the station #1" and then get to "airport shuttle station", then get to the "airport", then to "gate 10" and finally "board the flight".

Now given these abstract subgoals, the user can the possibly raise contrastive questions (cf. (Miller, 2017a)) in terms of these subgoals, for example, "What if you try to avoid the shuttle staion?" or the user could further drill down further to get more information. In the next sections, we will discuss how for a given model and policy, we can automatically generate such subgoals and their relative ordering to summarize the policy.

21.4 Our Approach: TLdR

As hinted in earlier sections, we will use partially ordered landmarks as our summary. Since we are not aware of any earlier works that have formalized landmarks for MDPs, we will start by introducing and formalizing the notion of landmarks in this setting. With the basic notions of landmarks in place, we can define policy landmarks and establish some basic properties. To generate these policy landmarks, we will propose compilation based methods with soundness and completeness guarantees.

21.4.1 Landmarks in MDPs

In the simplest terms, a landmark can be understood as formulas over propositional fluents that need to be satisfied by all paths from the initial state to goal states. More formally we can define landmarks as as

Definition 48. For a given model $\mathcal{M}$ whose state space is defined over a set of factors F, let's define a tuple $\mathcal{L} = \langle \Phi, \prec \rangle$, where Φ is a set of formulas specified over F and $\prec$ defines some ordering over them. Now $\mathcal{L}$ is said to be a landmark set for $\mathcal{M}$, if all transition sequences from start state to goal state of non-zero probability in $\mathcal{M}$ satisfy the formulas in Φ and their relative ordering, i.e.,
Let $\tau = \langle s_0, a_0, ..., s_g \rangle$ be a transition sequence from s_0 to $s_g \in G$, if $\mathcal{L}$ specifies landmarks for $\mathcal{M}$ then for all $\varphi \in \Phi$ there exists $s_j \in \tau$ such that $s_j \models \varphi$ and for all other formulas φ_1, φ_2 in Φ such that $\varphi_1 \prec \varphi$ and $\varphi \prec \varphi_2$ there must exist $s_i, s_k \in \tau$ $(i < j < k)$ such that $s_i \models \varphi_1$ and $s_k \models \varphi_2$.

So when identifying landmarks, we expect not only to identify a single formula, but rather a partially ordered set of formulas. For example the set

$$\{(\mathsf{at\_airport}) \wedge (\mathsf{has\_ticket}), (\mathsf{onboard\_plane})\}$$

along with the ordering

$$(\mathsf{at\_airport}) \wedge (\mathsf{has\_ticket}) \prec (\mathsf{onboard\_plane})$$

would constitute landmarks for the aforementioned travel task. This definition of landmarks parallels their usage in classical planning literature (cf. (Hoffmann *et al.*, 2004)) where they have been identified as an extremely useful information for guiding the planner during plan generation. Throughout the text we will use 'landmark' in singular to denote individual formulas while we will reserve the use of 'landmarks' to denote the partially ordered set of formulas.

While such landmarks on their own could still act as useful subgoals for policy summarization, they may be too few and far between and may not capture the specifics of the policy being pursued (unless the user choose to focus on specific intermediate states). Instead, we will consider landmarks that are specific to the policy at hand, which would lead us to define *policy landmarks*

Definition 49. For a given model $\mathcal{M}$, fluent set F and a policy π, a tuple $\mathcal{L}^{\pi}_{s_0,G} = \langle \Phi, \prec \rangle$ is a policy landmarks from s_0 to the goal set G if and only if, the formulas in Φ and their corresponding ordering can be satisfied by every execution trace $\tau \sim \pi(s_0)$ from s_0 to G.

Note that while any landmarks for the model as a whole will also be a policy landmark, but the reverse is not true. For example, in the travel scenario in_taxi might be a landmark for some policy but may not be a landmark for the task as a whole since it may be possible to get to the flight without ever boarding a taxi. We include the initial state and goal state set in the definition to allow for the possibility of recursively generating landmarks from any two reachable states for the given policy.

Landmarks as defined above are quite expressive and extensible. By leveraging disjunctive formulas we can even generate subgoals in cases where there are no state facts that are shared by all the paths. For example, in the case of the travel domain described in Figure 21.1 at_shuttle_north_gate $\vee$ at_shuttle_north_gate is a landmark. In fact, we can show that there exists a landmark set that can capture the entirety of any given policy.

Proposition 43. For a given model $\mathcal{M}$, fluent set F and a policy π, there exists a policy landmark $\mathcal{L}^{\pi}_{s_0,G} = \langle \Phi, \prec \rangle$ such that for every reachable state s from s_0 under π, there exists a non-trivial formula $\phi \in \Phi$ such that $s \models \phi$.

Proof Sketch. We will show that the property is true by constructing such a landmark set. Let's start with the policy graph corresponding to the given policy and partition the states in the graph to levels based on number of hops from the initial state (in case of policies with loops we assign each state the level corresponding to the shortest number of steps in which it can be reached by an execution of the policy). Now we will generate a DNF formula ϕ for each level, such that formula for level l contains a clause corresponding to each state reachable within l steps. Now we can create landmarks by ordering these DNF formulas according to the levels (and limit ordering to $\preceq$ starting from the level where the first goal state is reached). Such a landmark set should satisfy the requirements based on their construction. $\square$

The above property merely demonstrates the extensibility of landmarks as a concept and is not an endorsement for using more complex landmark formulas for generating summaries. In fact, in this chapter we will focus on *fact landmarks*, where each formula corresponding to a landmark consists of a single proposition. So when viewing these landmarks as subgoals, they can be thought of as achieving the fact corresponding to that proposition.

Now that we have defined partially ordered landmarks for a policy, our next step would be to identify methods that allow us to generate such landmarks.

21.4.2 Generating Landmarks

Our proposed algorithm for generating landmarks would rely on compilation into a corresponding classical planning problem, so we will start with a quick definition of classical planning problem (Geffner and Bonet, 2013b). Classical planning problems are generally describe by a tuple of the form $\mathcal{M}^c = \langle F^c, A^c, \mathcal{I}^c, \mathcal{G}^c \rangle$, where F^c provides the set of propositional fluents, A^c the list of deterministic actions, $\mathcal{I}$ the initial state and $\mathcal{G}^c$ is the goal specification (similar to how we defined $\mathcal{G}$ for the MDP). Each action a is defined by a tuple $\langle pre_+(a), pre_-(a), add(a), del(a) \rangle$, where $pre_+(a)$ and $pre_-(a)$ respectively provides the set of facts that must be true and false for the applicability of actions, $add(a)$ specifies the fluents it will set true on execution and $del(a)$ specifies the fluents that it will set false. For our purposes we will assume all the individual components of the action definition can be represented as sets.

It is well known that one way we can approximate MDPs is to determinize the MDP to an equivalent classical planning problem (Yoon *et al.*, 2007). One of the popular forms of determinization is what's called an all outcome determinization (Yoon *et al.*, 2007) where for every possible effect of the original MDP action, the classical planning problem would include a separate action with that specific effect, i.e. if for a given state s executing an action a could either result in s_1 or in s_2, then the determinization should produce two actions, one which will result in s_1 and another action that result in s_2. Such determinization procedures become simpler when the original MDP is specified in a factored form such as PPDDL (Younes and Littman, 2004). While a naive all outcome determinization could result in an exponential sized planning model, we can get more concise models by relying on more expressive representations (cf. (Keller and Eyerich, 2011)). As we will see in our particular setting if we relax the need for generating all landmarks we can still rely on simple models.

Now given such an all outcome determinization of our MDP $\mathcal{M}$ we can see that planning landmarks in the determinized model should be valid landmarks for $\mathcal{M}$. This property holds since landmarks for classical planning models are defined over all possible plans and plans in the determinized model have a one to one correspondence with traces in the original model. Unfortunately, we aren't interested in just generating landmarks for the model as a whole, but rather for the specific policy. One way to get there would be to restrict the model only to produce plans that correspond to traces we can sample from the given policy. We can create a compiled version of the classical planning problem which meets the above requirement if the policy is specified as a finite state machine (see Sreedharan *et al.* (2019b) or Baier *et al.* (2007) for possible compilation). More often than not, many of the popular offline planners for MDPs generate policies in tabular form, i.e., they explicitly enumerate the reachable states and their corresponding actions. So in this chapter we will focus on cases where the policy is provided in this form. Instead of relying on a compilation of policies to FSAs and then to planning problem, we will develop a method to create a compiled classical planning problems from the given policy, such that, the landmarks for that model aligns with policy landmarks of the given tabular policy.

Specifically, given our original MDP $\mathcal{M}$ and a policy π with a reachable state set $\mathcal{R}(s_0, \pi)$ let's consider the following classical planning model, $\mathcal{D}(\mathcal{M}) = \langle F, A^{\mathcal{D}}, I^{\mathcal{D}}, \mathcal{G}\rangle$. Here the new classical planning model makes use of the same fluent set, initial state ($I^{\mathcal{D}} = s_0$) and goal as the MDP $\mathcal{M}$. The new model contains one action a^s for each reachable state s, such that the precondition of the action is $pre_{+}(a^s) = s$ (assuming set representation for the state), the negative preconditions and delete effects is empty (i.e $pre_{-}(a^s) = del(a^s) = \{\}$), and finally the add effect is the set of all new fluents that can be set true by the corresponding policy action, i.e.,

$$add(a^s) = \Big(\bigcup_{s' \in \{s' | s \in S \wedge T(s, \pi(s), s') > 0\}} s'\Big) \setminus s$$

The above model is polynomial in the size of the given policy even for factored model representation, and the reason why we do not need to worry about delete effects is because most established methods for landmark generation in classical planning rely on approximations that ignore deletes. We will restrict ourselves to landmark generation methods that focus on generating *causal landmarks* (Zhu and Givan, 2003), where causal landmarks are defined to be facts that are required as preconditions for every possible plan. All that remains to be shown is that the causal landmarks derived from this model are sound policy landmarks for original model.

Theorem 14. *If f is a causal fact landmark extracted from $\mathcal{D}(M)$ then f must be a policy landmark for $\mathcal{M}$, π, initial state s_0 and goal set G. Also for another landmark f_2 (also causal in $\mathcal{D}(M)$), if the precedence $f \prec f_2$ holds for $\mathcal{D}(M)$ then it must also hold for the original MDP policy.*

Proof Sketch. We will basically establish this fact by showing that for every trace possible in the policy there must be a corresponding trace in $\mathcal{D}(M)$ (it may contain more). This means that causal landmarks for $\mathcal{D}(M)$ must satisfy all traces possible in the original policy. We will prove this through contradiction. First off, it is easy to see that for any trace $\tau \sim \pi$, where τ is of the form $\langle s_0, a_0, ..., s_k \rangle$, there exists a corresponding executable plan trace in $\mathcal{D}(M)$, $\hat{\pi} = \langle s_0, a_0^{s_0}, ..., \hat{s}_k \rangle$. Since for any prefix of the trace, the result of executing the actions in the sequence in $\mathcal{D}(M)$ must be a superset of the resulting state in the trace, i.e., $\langle a_0, ...a_k \rangle(s_0) = \hat{s}_k \supseteq s_k$. This means the action $a_k^{s_k}$ in $\mathcal{D}(M)$ whose precondition was s_k must now be executable in $\hat{s}_k$. This is because $\mathcal{D}(M)$ is a delete relaxed model and any fact established by any action in the original trace is conserved through the plan in $\mathcal{D}(M)$. If the causal landmark f in $\mathcal{D}(M)$ was not a policy landmark for the original MDP, then there must exist at least a trace from s_0 to some state in G we can sample from π where f never appears in any of the states. This means that in the corresponding valid plan for $\mathcal{D}(M)$ (where it goes from I to some state that satisfy $\mathcal{G}$), f can't appear in the precondition of any of the actions which contradicts the definition of causal landmark. Thereby proving our initial assertion. The soundness of ordering can be established following a similar line of reasoning. □

Another interesting property with this setting is the fact that we can exhaustively generate all policy landmarks from delete relaxation of the determinized model. Though this requires us to move away from the relatively concise representation used for $\mathcal{D}(M)$ to a more traditional all outcome determinization where we have a separate action for each possible transition in $\mathcal{M}$. That is, for a reachable state s, we add an action $a^{s,s'}$ for every s' such that $T(s, \pi(s), s') > 0$, where the action is defined as

$$a^{s,s'} = \langle pre_{+}(a^{s,s'}) = s, pre_{-}(a^{s,s'}) = F \setminus s, add(a^{s,s'}) = s' \setminus s, del(a^{s,s'}) = s \setminus s' \rangle$$

We will call this encoding $\mathcal{D}(M)^c$

Theorem 15. *If f is a policy landmark for $\mathcal{M}$, policy π, initial state s_0 and goal set G then f must be a causal fact landmark for $\mathcal{D}(M)^c$.*

Proof Sketch. We can again show this through contradiction. Since $\mathcal{D}(M)^c$ is a standard all outcome determinization there should be a one to one correspondence between all possible traces from π and plans in $\mathcal{D}(M)^c$. Assume that f is a policy landmark that is not a causal landmark for $\mathcal{D}(M)^c$. This means that there must be a valid plan for $\mathcal{D}(M)^c$, where f won't appear in any of the preconditions. This means that there must be a trace from s_0 to a state in G that doesn't result in the generation of the fact f. This means that f can't be a policy landmark, hence resulting in a contradiction. This proves our assertion. □

Since we are not as focused on ensuring completeness for our evaluation we will just use the more concise compilation. While our formulation was based on proper policies to simplify formulation, the methods proposed in this chapter do not require the policy itself to be proper. In fact, all we require is that there exists at least one possible trace from initial state to goal to generate the possible landmarks.

21.5 Formal Semantics for Hierarchies Generated Using TLdR

While in the earlier sections we alluded to the fact that the landmarks can be viewed as providing a form of temporal abstraction, we have yet to discuss specifics of the abstraction hierarchy induced by the landmarks. One way to understand the abstraction induced is to map it to a MaxQ hierarchy and a related hierarchical policy (Dietterich, 1998) (we consider a slightly modified version to allow application to SSP as opposed to just an infinite horizon discounted MDPs).

MaxQ is a particular method for specifying temporal abstractions for MDPs that involve specifying a task hierarchy for the given MDP. Usual MaxQ task hierarchies start with a root task and each task is recursively composed of subtasks. While the general MaxQ framework allows for parameterized tasks, we will just focus on a non-parameterized version where each subtask can be defined by the tuple $\langle T, A_i \rangle$ where T specifies the termination predicate or condition and A_i specifies the set of subtasks (including primitive actions) that can be performed as part of the execution of the subtask. Since we do not concern ourselves with learning the policies for these subtasks, we can easily skip the pseudo reward component that is usually also included in the subtask definition.

One of the main challenges of mapping landmarks we generate to a MaxQ task hierarchy is the fact that they may be partially ordered. So let us start from a partially ordered set of landmarks $\mathcal{L} = \langle \Phi, \prec \rangle$ (where partial ordering are not reflecting conjunctive landmarks) and construct a totally ordered set $\mathcal{L}^{tot} = \langle \Phi^{tot}, \prec \rangle$ such that Φ^{tot} is the maximal subset of Φ that allows for total ordering and still contains the goal.

The root task itself would be a task corresponding to achieving the goal, and we can add a subtask corresponding to each fact in Φ^{tot} to the root task. The termination condition of each subtask is specified by the landmark formula (which for our case is just the individual fact), the action set consists of all the individual atomic actions. Now for each formula dropped (i.e formulas in $\Phi \setminus \Phi^{tot}$) we will add a subtask node to:

1. that are it's closest remaining successor, i.e., add a subtask to the node corresponding to ϕ', such that $\phi' \in \Phi^{tot}$, $\phi \prec \phi'$ and $\nexists \hat{\phi} \in \Phi^{tot}$ and $\phi \prec \hat{\phi} \prec \phi'$.
2. to any node in $\mathcal{L}^{tot}$ that is not comparable (as per $\mathcal{L}$) with the dropped node.

Addition of these new subtasks follows from the fact that when there are partial ordering among landmarks, each of their possible linearizations could occur under different traces. To formally describe this, let's introduce the concept of a completion of a landmark. We can define the completions of a landmark ϕ (denoted as $\mathcal{C}(\phi)$) as the set of states where the formula is satisfied for the first time ,i.e.,
$\mathcal{C}(\phi) = \{s_k | s_k \in \mathcal{R}(s_0, \pi) \land \ s_k \models \phi \land \ \exists \tau \sim \pi(s_0), s_k \in \tau \land \forall s_j \in \tau, (s_j \models \phi \implies k \leq j)\}$.
Now given this concept, we will assert

Proposition 44. *Let f_0, f_1, f_2 and f_3 be landmarks such that $f_0 \prec f_1$, $f_0 \prec f_2$, $f_1 \prec f_3$ and $f_2 \prec f_3$. If $|\mathcal{C}(f_1) \cap \mathcal{C}(f_2)| > 0$ and the ordering is complete, then for any state $s_{f_2} \in \mathcal{C}(f_2)$ (where $s_{f_2} \not\models f_1$), there either exists a states $s_{f_0} \in \mathcal{C}(f_0)$ such that f_1 is a landmark for paths from s_{f_0} to s_{f_2} or there exists a completion for f_3 such that f_1 is a landmark for paths from the completion of f_2 to f_3.*

The above proposition states that if f_1 and $f2$ are not fact landmark representations of a larger conjunctive landmark (thus $|\mathcal{C}(f_1) \cap \mathcal{C}(f_2)| > 0$) and the ordering

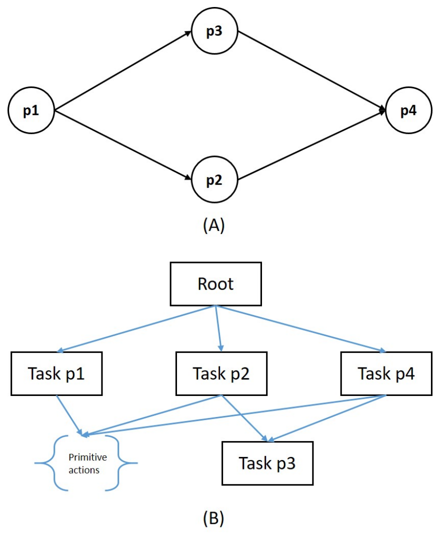


Figure 21.2: (A) Graph (with Arrow Directions Presenting the Ordering) Corresponding to a Partially Ordered Landmark Set and (B) an Equivalent MaxQ Hierarchy.

is complete (i.e partial ordering is not due to missing ordering), then for any state corresponding to achievement of f_2, the landmark f_1 needs to be achieved before reaching the current state or needs to be achieved before the successor subgoal (after crossing the current state). This proposition trivially follows from the nature of ordering between landmarks and shows the landmarks excluded from the original total-ordering can still be used as a subtask to achieving the landmarks listed in $\mathcal{L}^{tot}$. For conjunctive landmarks, the dropping of one of the facts shouldn't result in any information loss since the completion set of one fact contains all the states where the other fact is also established. Moreover such partial ordering should disappear once we start generating conjunctive landmarks.

The landmarks also allow us to convert the current policy into a hierarchical one that can be executed in the context of MaxQ (we will denote this policy as $\pi^{\mathcal{L}}$). We can use the following rules to recursively define the hierarchical policy

1. For root node: The initial state gets mapped to the subtask corresponding to

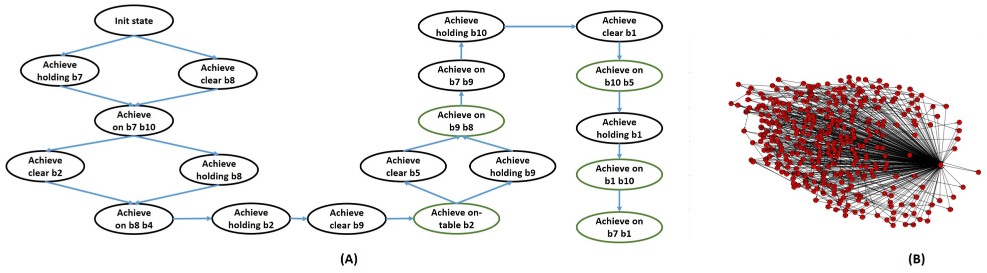


Figure 21.3: (A) The Diagram Corresponding to the Policy Landmarks Generated for the Fifth Exploding Blocksworld Problem Presented as Part of IPC 2006. (B) A Simple Visualization of the Entire Policy.

the first landmark and all completions of each landmark in Φ^{tot} gets mapped to the next subtask in the ordering. For completeness, we will map the remaining states to the primitive actions specified in the policy, though those should never get executed.

2. For other subtasks: If the state corresponds to the completion of the last subtask and is known to lead to one of the child subtask (as per Proposition 21.5) then map it to the corresponding child node. For any other state execute action specified by the current policy for that state.

Proposition 45. *The hierarchical cost of the policy $\pi^{\mathcal{L}}$ ($J^{\pi^{\mathcal{L}}} = \langle s_0, nil \rangle$) is equal to the cost of the original $J^{\pi}(S)$*

If we follow the hierarchical policy execution procedure specified by Dietterich (1998), it should be easy to see that the hierarchical cost is the same as the original policy cost, since the hierarchical policy execution would only be executing the primitive actions specified in the original policy.

21.6 Evaluation

21.6.1 User Studies

Concerning evaluation, our first priority was to perform an assessment of our central hypothesis, namely, that landmarks constitute useful subgoals. So one of the questions, that can be raised is whether people would choose landmarks when they use subgoals to summarize policies. The specific hypothesis we were interested in evaluating was

Hypothesis: *When presented with a policy for a task with non-deterministic / stochastic dynamics, people will choose landmark facts as high-level subgoals over non-landmark facts (where the landmarks are as defined in earlier sections).*

The hypothesis was tested by presenting different planning scenarios to participants from Amazon Mechanical Turk. Each participant was then asked to choose from a set of possible facts they believe would be the most appropriate subgoal to be included as part of a summary. In particular, we used a modified version of the travel scenario discussed earlier and a logistics planning scenario that dealt with the problem of delivering a package to a pre-specified location. We used 30 participants per scenario, and the scenario description included a description of the policy graph and no details on the actual transition probabilities or the rewards. After reading the scenario, the participants were provided a list of facts about reachable states under the given policy. They are then asked to select four facts from the list they believe can be provided as subgoals as part of a summary of the given policy. The list consisted of 13 facts in total, six of which were landmarks in the travel domain questionnaire, and five of them were landmarks for the logistics domain questionnaire. The users were presented the facts in a randomized order to make sure the results are counterbalanced. We also filtered the answers from the participants based on their answers on some factual questions about the policy. For filtering the response, each participant was presented two questions and their entire response was filtered out if they got any of the answers incorrect. We also changed the questions for every 15 participants to make sure the questions didn't introduce any additional bias in the participants' reply. After filtering we were left with 41 participants and 164 subgoal selections. Out of this 164 selection, 125 were landmark selections, which puts the number of landmarks selected at 76.2%. While non-landmark facts were selected by participants, the results show that the majority did in fact choose landmark facts to summarize the give policy. To test the statistical significance of our result we ran a paired two tailed t-test with the fact type (landmark or not) as the independent variable and the number of responses per group as the measure. We were able to establish that there exists a statistically significant difference between the groups with a level of significance set at 0.001 (the p-value was 0.0000093 for travel domain and 0.000533 for logistics). A PDF copy of the exact surveys has been included can be found at `http://bit.ly/39cxiV8`.

21.6.2 Empirical Evaluation

As for the empirical evaluation, we were interested in understanding whether policy landmarks provided any advantage over problem landmarks in terms of identifying more subgoals. Since model landmarks are always a subset of policy landmarks, any additional facts part of the policy landmark should hopefully capture policy specific characteristics. Moreover, we were interested in seeing how many fact landmarks were, in fact, extracted for some prototypical problems. So we selected four standard PPDDL domains from some earlier probabilistic tracks of IPC competitions (International Planning Competition, 2011) and five problem instances per domain (except triangle tireworld for which we only used four). We used compilation methods dis-

Domain	$\mid \mathcal{R}^{\pi}(I) \mid$	$\mid \Psi^{\pi} \mid$	$\mid \Psi^{\mathcal{M}} \mid$
Ex-Blocksworld	104.8	8.8	5.2
Elevator	13.2	7	4
Tireworld	13.6	1.8	0.8
Tri-Tireworld	3973	6.25	0

Figure 21.4: Table Showing the Reachable State Set and Landmark Sizes for Benchmark Domains. The Second Column of the Table Presents the Number of Reachable States, While the Third and Fourth Columns List the Average Number of Policy and Model Landmarks (Excluding Goals). All Domain and Problem Instances Were Taken from IPC 2006 and 2008.

cussed in earlier section to prepare the deterministic domain for policy landmark extraction. Table 21.4 presents results from this analysis and presents the average policy size, the number of non-initial and non-goal causal landmarks extracted (i.e landmark facts aren't part of the initial state or the goal state) for the policy and for the model as a whole. All policies were generated using an $LAO^{\star}$ (Hansen and Zilberstein, 2001) implementation and because some of these domains we considered included dead ends, we didn't enforce the proper policy requirement during the evaluation. The landmarks were generated using FastDownward (Helmert, 2006) implementation of Keyder *et al.* (2010). For all the domains we considered, we found that policy landmarks do lead to identifying more fact as compared to model landmarks. In fact, for triangle tireworld problems there were no non-init, non-goal landmarks, while our method was able to identify multiple policy landmarks. In all the domains, we see that the landmark sets does lead to more concise policy summary when compared to the size of the reachable state set.

As an illustration of the kind of summaries we can generate from such PPDDL domain, consider one of the problems from the exploding blocksworld domain. The domain is quite similar to the deterministic blocks world domain, except that now putting blocks on top of another or on the table could potentially lead to it exploding. Once a block explodes you can't place another block on top of it. The specific instance we looked at contained, 10 blocks and had a goal consisting of five facts. Figure 21.3 (A) presents the policy landmarks and their ordering for the specific policy we considered which allowed for 362 reachable states (Figure 21.3 (B)). In addition to showing the various intermediate facts and the order in which to achieve them, one interesting fact the landmarks highlight is that it presents the order in which the policy expects the various goal facts to be achieved (highlighted in green).

21.7 Concluding Remarks

In summary, our work presents a policy summarization technique that tries to automatically identify subgoals for a given policy by identifying landmark facts. Towards this end, we formalized the concept of landmarks for MDPs and proposed the idea of policy landmarks. We introduced compilation based procedures for extracting these landmarks and established the relationship between the hierarchy induced by the landmarks with previously studied methods for reasoning with task hierarchies in MDPs. Our user studies suggest that in the absence of task details, people do tend to choose landmarks when asked to select intermediate objectives. Additionally, we also found that in many of the domains choosing to extract policy landmarks does provide us with more information.

One point hinted at but not expanded upon in the current chapter is how the summary could be expanded based on user response. For example, from the given set of subgoals, the user might want to know how exactly the policy plays out. Interestingly, we can leverage the exact methods discussed in the chapter to generate these lower-level subgoals. When the user wants to drill down, they can be asked to choose from the list of completions for source subgoal. Then we can use the compilation methods developed in the chapter to generate new landmarks from the new initial state to the destination subgoal.

One of the issues that we didn't quite cover in the chapter is the fact that users may not be interested in all landmarks, but rather ones related to a subset of fluents. For example, an engineer trying to view the policy of an extra-planetary rover may be more interested in seeing landmarks related to fuel-levels and engine performance, rather than ones related to samples collected by the robot. On the other hand, a geologist may be completely oblivious to the details about the robot's engines or batteries. To allow for such differing views, we would need to marry our landmark extraction methods with methods for state abstraction.

Another point to note is that as the underlying model becomes more deterministic, it should start introducing more policy landmarks, with the entire policy turning into a single sequence of states and actions in the extreme case. This would mean that each state in the sequence would be technically a policy landmark. In cases where the model is deterministic or nearly deterministic, it may be worth focusing on a subset of policy landmarks that actually appear in the preconditions of actions. In fact, Sreedharan *et al.* (2022b) have investigated the close relationship between such policy landmarks and causal links. If one were to limit oneself to such policy landmarks, then the summary being presented here would be equivalent to returning just the causal links that are part of a given plan/policy.

Part VI

CURRENT LANDSCAPE OF XAIP AND FUTURE WORK

Chapter 22

CATEGORIZATION AND ANALYSIS OF EXISTING WORKS IN XAIP

In this chapter, I will present an overview of the various works done in the context of XAIP and explainable reinforcement learning (henceforth referred to as XRL). The works will be primarily discussed with in their relation to the three dimensions of explanation as laid out in the introductory chapter. The goal of this chapter is not just to present a snapshot of the field as it exists right now, but also to demonstrate the ability of these three dimensions to provide insights into the extant literature. In regards to work coverage, I will strive to be exhaustive in the works related to explainable planning, but with regards to works from XRL I will focus on discussing some representative works. We will specifically focus on works that introduce a new explanation generation method as opposed to systems that leverage existing methods.

22.1 Analyses of Current Works

In this section, I will start by providing a brief introduction to each work that I will be looking at. In particular, I will focus on the following factors, (a) whether the work focuses on explanations or plan/policy summarization, (b) whether the explanation could be understood as an answer to a specific question, (c) whether the explanation is a preference account or a procedural one and (d) the kind of problem formalisms expected by the method.

Tables 22.1,22.2 and 22.3, present this overview. Note that I have chosen to classify methods aimed at determining the features in the current state that led to the selection of the current action as summarization techniques. Such methods are particularly popular in the context of reinforcement learning problems where the policy itself is a non-interpretable artifact, best represented as a black box function mapping states to actions. My rationale behind classifying these methods as summarization techniques is the fact that these methods help the user better understand the current policy, but aren't explicitly designed to help the user understand why the current action is an appropriate action to follow in the current state. It is worth noting that most of the time these methods directly leverage explanatory techniques for single-shot decision making settings (particularly for classification) to describe the learned policies. However, the focus of this thesis has been on explaining the robot or the agent's rationale for selecting a particular policy or plan and the works in this category fail to meet that requirement. Putting aside such summarization works, we can see that preference based accounts are the most popular types of explanations. A lot of current works focus on MDPs (probably given the popularity of RL), and there are only a few works that consider the problem of generating explanations for more expressive planning formalism like numeric planning or problems with durative actions.

22.1.1 Asymmetry in Knowledge

Tables 22.4, 22.5, 22.6, 22.7 and 22.8, present the properties of the works with respect the first dimension, i.e., the assymetry between the robot and the human in regards to their knowledge about the task. In particular, we will talk about the assumptions the works make about human and agent models, and whether the works try to resolve any asymmetry.

Under assumptions made about human and agent models, I have listed both the explicit and implicit assumptions made by the works. For those works that make no explicit assumption about the human model, there are two common categories of implicit assumptions being made, ones that require the human model to be the same

Paper	Type of Information	Associated Question	Process or Preference Account	Problem Class
(Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2020a)	Explanation	Why is this plan optimal?	Preference	PDDL
(Seegebarth *et al.*, 2012; Bercher *et al.*, 2014)	Explanation	Why this action?	Preference	HTN
(Sreedharan *et al.*, 2018c)	Explanation	Why not this set of plan?	Preference	PDDL
(Sreedharan *et al.*, 2019b)	Explanation	Why can't you satisfy this constraint or why is this problem unsolvable?	Preference	PDDL
(Eifler *et al.*, 2020a,b)	Explanation	Why this set of objectives not another set?	Preference	Planning for multiple objectives (in particular oversubscription planning)
(Göbelbecker *et al.*, 2010)	Explanation	Why is this problem not solvable?	Preference	PDDL
(Khan *et al.*, 2009)	Explanation	Why did you do action a in state s?	Preference	Discounted infinite horizon MDP
(Dodson *et al.*, 2013)	Explanation	Why did you do action a in state s?	Preference	Discounted infinite horizon MDP
(Madumal *et al.*, 2020b)	Explanation	Why action a and not b in state s?	Preference	Discounted infinite horizon MDP
(Greydanus *et al.*, 2018; Anderson *et al.*, 2019)	Summary			Discounted infinite horizon MDP
(Zahavy *et al.*, 2016)	Summary			Discounted infinite horizon MDP
(Magnaguagno *et al.*, 2017)	Explanation			PDDL
(Krarup *et al.*, 2021, 2019)	Explanation	Contrastive Query	Process	PDDL 2.1
(Koul *et al.*, 2019)	Summary			Discounted infinite horizon MDP
(Kim *et al.*, 2019)	Summary			Sets of traces over propositional states

Table 22.1: An Overview of the Works That Will Covered in This Discussion (Part 1)

Paper	Type of Information	Associated Question	Process or Preference Account	Problem Class
(Vasileiou *et al.*, 2021)	Explanation	Why is this formula entailed by the knowledge base?	Preference	Propositional knowledge bases
(Lin *et al.*, 2021)	Explanation	Why was action a chosen in state s instead of b?	Process	Discounted infinite horizon MDP
(Juozapaitis *et al.*, 2019)	Explanation	Why was action a chosen in state s instead of b?	Preference	Discounted infinite horizon MDP with multiple reward components
(Madumal *et al.*, 2020a)	Explanation	Why was action a chosen in state s instead of b?	Preference	Discounted infinite horizon MDP
(Waa *et al.*, 2018)	Explanation	Why this set of actions instead of others? (a form of partial foils as applied to MDP)	Preference	Discounted infinite horizon MDP (annotated with action outcomes)
(Valmeekam *et al.*, 2020)	Explanation	Why not this other strategy (where strategy specified as a partial plan)	Preference	PDDL
(Sukkerd *et al.*, 2020)	Explanation	why this policy	Preference	Multi objective MDP (discounted and inifinite horizon)
(Lage *et al.*, 2019; Amir and Amir, 2018)	Summary			Discounted infinite horizon MDP
(Kasenberg *et al.*, 2020)	Explanation	Why was the formula ϕ satisfied? (ϕ being a temporal logic formula)	Preference	Deterministic discounted infinite horizon MDP with a set of preferences expressed as LTL formulas with a lexicographic ordering
(Ferrer-Mestres *et al.*, 2020)	Summary			Discounted infinite horizon MDP

Table 22.2: An Overview of the Works That Will Covered in This Discussion (Part 2)

as the agent model (for example those related to excuse generation (Göbelbecker *et al.*, 2010)), versus one that might work if the user's model is empty, as in they have no prior knowledge about the domain. There are a few ways this setting could be grounded, one would be to assume optimism under ignorance, in that humans think goals are reachable from all states and all transitions have minimal costs. In the context of PDDL (assuming only positive goals) this translates into a model with empty preconditions and an effect set that is equivalent to the fluent set. In the

Paper	Type of Information	Associated Question	Process or Preference Account	Problem Class
(Sreedharan *et al.*, 2018a)	Explanation	Why is the plan π optimal?	Preference	PDDL
(Sreedharan *et al.*, 2022c)	Explanation	Why plan π and not plan π'?	Preference	A deterministic planning problem expressible as a PDDL model over some set of fluents
(Hayes and Shah, 2017)	Summary			Discounted infinited horizon MDP
(Sreedharan *et al.*, 2020c)	Summary			SSP
(Topin and Veloso, 2019)	Summary			Discounted infinited horizon MDP
(Brandao *et al.*, 2021)	Explanation	Why is path p1 optimal rather than p2?	Preference	NavMesh Path Planning Problems
(Olson *et al.*, 2019)	Summary			MDP (where states can be visually represented)
(Huber and André, 2019)	Summary			MDP (where states can be visually represented)
(Coppens *et al.*, 2019)	Summary			MDP (where states can be visually represented)
(Soni *et al.*, 2021)	Explanation	Why is this behavior part of an optimal policy?	Preference	MDP
(Kumar *et al.*, 2021)	Explanation	Why is this plan optimal?	Preference	PDDL
(Vasileiou *et al.*, 2022)	Explanation	Why is this formula entailed by the knowledge base?	Preference	Propositional knowledge bases and SMT (supports PDDL+)
(Sreedharan *et al.*, 2019a)		Why is this behavior part of an optimal policy?	Preference	MDP

Table 22.3: An Overview of the Works Continued from Table (Part 3)

context of MDPs, I have taken this to mean, actions have an equal likelihood of taking you to any state, and all transition costs are minimal. Additionally, I assume the human would accept all the information provided by the system (for example they may accept information about the). Many explanation schemes where the explanation provides information about execution traces of the current strategy or alternate ones specified by the human would still be valid in these settings. This includes works like the one presented by Waa *et al.* (2018) or the ones that use learned causal models to contrast the outcomes of the current action with another (cf. (Madumal *et al.*, 2020b)). A more interesting group of works that I would argue can support this category are the ones like the method presented by Eifler *et al.* (2020a), where the explanation provides the various constraints between the different possible objectives. In a way, we could see these works as performing a kind of model simplification, where they effectively convert a multi-objective planning problem into a constraint satisfaction problem where there are specified constraints among various objectives. If the human is ready to accept that such constraints exist, and do not ask why such constraints may exist then the generated explanations should completely satisfy the human. This is in contrast with works like the ones presented by Göbelbecker *et al.* (2010), that could fail if the human model is not the same as the agent model.

For summarization works, I have mostly left the assumptions related to human models blank, since most of those works do not directly use the human's mental models to create summaries. The exception being works like Lage *et al.* (2019); Amir and Amir (2018), which expects the human to reconstruct the rest of the policy given a few state action samples. Depending on the work, they either assume the human's model is the same as the agent or the agent has access to some representation of the human's beliefs about the task.

In regards to addressing this model mismatch, I have listed two possible category works. The first corresponds to methods that try to directly engage in model reconciliation as part of addressing the explanatory queries, and methods that expose model information as part of explanation even though they may not have explicitly assumed a model difference. The latter was a hard category to narrow down since many works indirectly expose model information through explanatory witnesses like possible constraints between various objectives, value functions, and plan traces, etc. Here I have only included works that are directly providing model information to the human.

Paper	Assumptions regarding models		Model Mismatch	
	Explicit	Implicit	Sufficiently reconciles	Exposes Model Information
(Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2020a)	Human model known and could be different along multiple dimensions	No additional assumptions identified	Yes	Yes
(Seegebarth *et al.*, 2012; Bercher *et al.*, 2014)	None mentioned	Is guaranteed to work if the model of the human is empty, or doesn't have any deletes.	Not in the general case	Yes (preconditions and effects)
(Sreedharan *et al.*, 2018c)	Human model is expected to be an abstraction of the true robot model	Will be able to resolve the foil even if the assumption is not met, provided the lattice considered is complete and defined over all propositions and the model is disjunction free	Yes	Yes
(Sreedharan *et al.*, 2019b)	Human model is expected to be an abstraction of the true robot model	Will be able to resolve the foil even if the assumption is not met, provided the lattice considered is complete and defined over all propositions and the model is disjunction free	Yes	Yes

Table 22.4: Assumptions Made by the Works in Regards to Dimension One and Whether the Works Can Address the Asymmetry (Part 1)

Paper	Assumptions regarding models		Model Mismatch	
	Explicit	Implicit	Sufficiently reconciles	Exposes Model Information
(Eifler *et al.*, 2020a,b)	None mentioned	Will work if the human model is empty, provided the human don't ask why the plan property entailments holds	Not in the general case	No – though it could be argued that the global explanation they provide is a simplified representation of the problem
(Göbelbecker *et al.*, 2010)	None mentioned	Has to be the same model	No	By the virtue of providing a counterfactual value of a static proposition, it exposes some initial state values
(Khan *et al.*, 2009)	None mentioned	Has to be the same model	No	No – though the explanation here consists of reachability
(Dodson *et al.*, 2013)	None mentioned	Will work if the human model is the same	No	No
(Madumal *et al.*, 2020b)	None mentioned	Will work if the human model is empty	No	The explanation here consists of providing a causal chain which includes information about the transition and even information about reward function/goal is provided at a high level
(Greydanus *et al.*, 2018; Anderson *et al.*, 2019)	None mentioned	N/A	No	No
(Zahavy *et al.*, 2016)	None mentioned	N/A	No	As part of the abstracted representation of the policy they also reveal the transitions
(Magnaguagno *et al.*, 2017)	None mentioned	Will work if the human model is empty	Yes (visualizing the entire search tree)	Yes
(Krarup *et al.*, 2021, 2019)	The journal version mentions they make no explicit assumptions	Same model/-Constrained model	No	No

Table 22.5: Assumptions Made by the Works in Regards to Dimension One and Whether the Works Can Address the Asymmetry (Part 2)

Paper	Assumptions regarding models		Model Mismatch	
	Explicit	Implicit	Sufficiently reconciles	Exposes Model Information
(Koul *et al.*, 2019)	None mentioned	N/A	No	No
(Kim *et al.*, 2019)	None mentioned	N/A	No	No
(Vasileiou *et al.*, 2021)	Human model is given and could differ from the robot model in multiple ways	None identified	Yes	Yes
(Lin *et al.*, 2021)	None mentioned	MSX requires the model to be the same or the human to be pessimistic	No	No
(Juozapaitis *et al.*, 2019)	None mentioned	MSX requires the model to be the same or the human to be pessimistic	No	No
(Madumal *et al.*, 2020a)	None mentioned	Will work if the human model is empty	No	Yes - The explanation here consists of providing a causal chain and distal action. So it does reveal information about transition and even preconditions of actions
(Waa *et al.*, 2018)	None mentioned	Will work if the human model is empty	No	The explanation involves traces and positive and negative outcomes of actions
(Valmeekam *et al.*, 2020)	Human model expected to be given	N/A	Yes	Yes
(Sukkerd *et al.*, 2020)	None mentioned	Will work if the human model is empty	No	No – but one could argue since by presenting the tradeoff between the various metrics they are in fact presenting the model information at an abstract level

Table 22.6: Assumptions Made by the Works in Regards to Dimension One and Whether the Works Can Address the Asymmetry (Part 3)

Paper	Assumptions regarding models		Model Mismatch	
	Explicit	Implicit	Sufficiently reconciles	Exposes Model Information
(Lage *et al.*, 2019; Amir and Amir, 2018)	None mentioned	Requires knowledge about human model	No	No
(Kasenberg *et al.*, 2020)	None mentioned	Will work if the human model is empty	No	No – But part of the explanation would present relationship between achieving ϕ and other LTL preferences
(Ferrer-Mestres *et al.*, 2020)	None mentioned	Will work if the human model is empty	Not specified, but one could theoretically convey the abstract model generated	Not specified
(Sreedharan *et al.*, 2018a)	A partial specification of human model provided	Will theoretically work if the human model is completely unknown, provided they share the same action and fluent set	Yes	Yes
(Sreedharan *et al.*, 2022c)	Human model is empty	Will be able to resolve the foil even if the assumption is not met, provided the precondition for the failing action in the human model is disjunction free	Yes	Yes
(Hayes and Shah, 2017)	None mentioned	N/A	No	No

Table 22.7: Assumptions Made by the Works in Regards to Dimension One and Whether the Works Can Address the Asymmetry (Part 4)

Paper	Assumptions regarding models		Model Mismatch	
	Explicit	Implicit	Sufficiently reconciles	Exposes Model Information
(Sreedharan *et al.*, 2020c)	None mentioned	N/A	No	No
(Topin and Veloso, 2019)	None mentioned	N/A	No	No (They do expose some information about transition probabilities at an abstract level)
(Brandao *et al.*, 2021)	Human model known	No additional assumptions identified	Yes	Yes
(Olson *et al.*, 2019)	None mentioned	N/A	No	No
(Huber and André, 2019)	None mentioned	N/A	No	No
(Coppens *et al.*, 2019)	None mentioned	N/A	No	No
(Soni *et al.*, 2021)	Human model belongs to one of a set of finite types. Specific models of each type not known	No additional assumptions identified	Yes	Yes
(Kumar *et al.*, 2021)	Human Model given	No additional assumptions identified	Yes	Yes
(Vasileiou *et al.*, 2022)	Human Model given	No additional assumptions identified	Yes	Yes
(Sreedharan *et al.*, 2019a)	Human model is not known, but can interact with people with the same background knowlede	No additional assumptions identified	Yes	Yes

Table 22.8: Assumptions Made by the Works in Regards to Dimension One and Whether the Works Can Address the Asymmetry (Part 5)

22.1.2 Asymmetry in Inferential Capabilities

Table 22.9 and 22.10 focuses on aspects related to addressing inferential mismatch. Since Table 12.3 and 12.4 in Chapter 12 provides a comprehensive list of explanatory witnesses used in previous works, I have skipped them in this table. The focus on this table would be on whether the works use model simplification strategies and as evident from the table the vast majority of current works do. In this section, I would like to specifically highlight two sets of works. First being the method presented by Lage *et al.* (2019), which to the best of my knowledge is the only work that incorporates an explicit model of the inferential process used by the user. While the method doesn't use model simplification, it uses the human's inferential model to figure out the set of examples to show. The method expects the human to employ their inferential algorithm to reconstruct the rest of the policy from the specified examples. Secondly, consider the works presented by Sreedharan *et al.* (2019a) and Soni *et al.* (2021). While these works don't explicitly try to address inferential difference, they can in theory account for it. This is because the labeling models are directly learned from the data collected from humans and thus the label would be negative if the explanation places too much inferential burden on the human. Finally, the explanatory messages used by these methods could also be derived from a simplified version of the model.

22.1.3 Asymmetry in Vocabulary

Finally Tables 22.11 and 22.12 discusses how the methods relate to the third dimension. As evident from the table, there are very few works that explicitly bring up the possibility of vocabulary mismatch. However, there are quite a few works that can operate succesfully in the presence of such asymmetry. A prominent group of such works are the ones that use various visualization methods. In these cases,

Paper	Inferential Mismatch	
	Does it try to account for it?	Performs Model Simplification
(Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2020a)	No	No
(Seegebarth *et al.*, 2012; Bercher *et al.*, 2014)	Yes	No
(Sreedharan *et al.*, 2018c)	Yes	Yes (Abstraction)
(Sreedharan *et al.*, 2019b)	Yes	Yes (Abstraction)
(Eifler *et al.*, 2020a,b)	Yes	Yes (Only presents the constraints among the different objectives)
(Göbelbecker *et al.*, 2010)	Yes	No
(Khan *et al.*, 2009)	Yes	Yes(A form of state abstraction, particularly the ones that use factored subsets)
(Dodson *et al.*, 2013)	Yes	No
(Madumal *et al.*, 2020b)	Yes	Yes (Instead of presenting transition over entire state they present effect of each action specific state factors)
(Greydanus *et al.*, 2018; Anderson *et al.*, 2019)	Yes	No
(Zahavy *et al.*, 2016)	Yes	Yes (Provide SAMDP which uses both temporal and state abstraction)
(Magnaguagno *et al.*, 2017)	Yes	No
(Krarup *et al.*, 2021, 2019)	Yes	No
(Koul *et al.*, 2019)	Yes	Yes
(Kim *et al.*, 2019)	Yes	No
(Vasileiou *et al.*, 2021)	No	No
(Lin *et al.*, 2021)	Yes	Yes
(Juozapaitis *et al.*, 2019)	Yes	No

Table 22.9: A Summary of Whether the Different Methods Contribute to Addressing Inferential Differences Between the Robot and the Human. (Part 1)

	Inferential Mismatch	
Paper	Does it try to account for it?	Performs Model Simplification
(Madumal *et al.*, 2020a)	Yes	Yes (Instead of presenting transition over entire state they present effect of each action specific state factors)
(Waa *et al.*, 2018)	Yes	Maps the reward function into positive and negative outcomes
(Valmeekam *et al.*, 2020)	Yes	No
(Sukkerd *et al.*, 2020)	Yes	Yes
(Lage *et al.*, 2019; Amir and Amir, 2018)	Yes	No
(Kasenberg *et al.*, 2020)	Yes	Yes
(Ferrer-Mestres *et al.*, 2020)	Yes	Yes
(Sreedharan *et al.*, 2018a)	No	No
(Sreedharan *et al.*, 2022c)	No	No
(Hayes and Shah, 2017)	Yes	No
(Sreedharan *et al.*, 2020c)	Yes	Yes
(Topin and Veloso, 2019)	Yes	Yes
(Brandao *et al.*, 2021)	No	No
(Olson *et al.*, 2019)	Yes	No
(Huber and André, 2019)	Yes	No
(Coppens *et al.*, 2019)	Yes	No
(Soni *et al.*, 2021)	Can support it	Can support it
(Kumar *et al.*, 2021)	No	No
(Vasileiou *et al.*, 2022)	No	No
(Sreedharan *et al.*, 2019a)	Can support it	Can support it

Table 22.10: A Summary of Whether the Different Methods Contribute to Addressing Inferential Differences Between the Robot and the Human. (Part 2)

regardless of how the human may reason about the task, they would still be able to make sense of the information. Another group of works that still can operate correctly are the summarization works that rely on providing the users with various behavioral demonstration (cf. (Lage *et al.*, 2019; Amir and Amir, 2018)). Finally, there are works that can support the use of features that are not necessarily the same as the ones used in the original decision-making process (Madumal *et al.*, 2020b,a).

22.2 Other Considerations for XAIP

While the focus of this chapter has been to characterize the works using the three dimensions of human-aware explanations highlighted in this thesis, it would be instructive to take a quick look at some of the other salient features that could be used to characterize the works.

End Users of the Explanations One question we haven't quite discussed in this thesis is, who is meant to use the explanation. In the planning setting, one could imagine three broad user classes

- *End user:* This is the person who interacts with the system in the form of a user. For a planning system, this may be the human teammate in a human-robot team (Chakraborti *et al.*, 2019f) who is impacted by or is a direct stakeholder in the plans of the robot, or a user collaborating with an automated planner in a decision support setting (Grover *et al.*, 2020a).

- *Domain Designer:* This is the person involved in the acquisition of the model that the system works with: e.g. the designer of goal-oriented conversation systems (Sreedharan *et al.*, 2020b).

Paper	Vocabulary Mismatch	
	Does it explicitly try to account for it?	Is the method applicable under vocabulary mismatch?
(Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2020a)	No	No
(Seegebarth *et al.*, 2012; Bercher *et al.*, 2014)	No	No
(Sreedharan *et al.*, 2018c)	No	No
(Sreedharan *et al.*, 2019b)	No	No
(Eifler *et al.*, 2020a,b)	No	No
(Göbelbecker *et al.*, 2010)	No	No
(Khan *et al.*, 2009)	No	No
(Dodson *et al.*, 2013)	No	No
(Madumal *et al.*, 2020b)	No	Yes
(Greydanus *et al.*, 2018; Anderson *et al.*, 2019)	No	Yes
(Zahavy *et al.*, 2016)	No	Yes
(Magnaguagno *et al.*, 2017)	No	No
(Krarup *et al.*, 2021, 2019)	No	No
(Koul *et al.*, 2019)	No	Yes
(Kim *et al.*, 2019)	No	No
(Vasileiou *et al.*, 2021)	No	No
(Lin *et al.*, 2021)	No	No
(Juozapaitis *et al.*, 2019)	No	No

Table 22.11: The Table Summarizes How the Different Methods Relate to the Third Dimension, I.E., Asymmetry in Vocabulary (Part 1)

Paper	Vocabulary Mismatch	
	Does it explicitly try to account for it?	Is the method applicable under vocabulary mismatch?
(Madumal *et al.*, 2020a)	No	Yes
(Waa *et al.*, 2018)	Yes	Yes
(Valmeekam *et al.*, 2020)	No	No
(Sukkerd *et al.*, 2020)	No	No
(Lage *et al.*, 2019; Amir and Amir, 2018)	No	Yes
(Kasenberg *et al.*, 2020)	No	No
(Ferrer-Mestres *et al.*, 2020)	No	No
(Sreedharan *et al.*, 2018a)	No	No
(Sreedharan *et al.*, 2022c)	Yes	Yes
(Hayes and Shah, 2017)	Yes	Yes
(Sreedharan *et al.*, 2020c)	No	No
(Topin and Veloso, 2019)	No	No
(Brandao *et al.*, 2021)	No	No
(Olson *et al.*, 2019)	No	Yes
(Huber and André, 2019)	No	Yes
(Coppens *et al.*, 2019)	No	Yes
(Soni *et al.*, 2021)	No	Yes
(Kumar *et al.*, 2021)	No	Yes
(Vasileiou *et al.*, 2022)	No	No
(Sreedharan *et al.*, 2019a)	Yes	Yes

Table 22.12: The Table Summarizes How the Different Methods Relate to the Third Dimension, I.E., Asymmetry in Vocabulary (Part 2)

- *Algorithm Designer:* The final persona is that of the developer of the algorithms themselves: e.g. in the context of automated planning systems, this could be someone working on informed search.

A vast majority of explanation work in XAIP tends to be designed from the end-user perspective (characterized by the popularity of the preference account explanations). Though there exists works (Sreedharan *et al.*, 2020b; Lin and Bercher, 2021) designed for the domain designer and works (Magnaguagno *et al.*, 2017) better suited for the algorithm designer.

Local versus Global Explanations. Another consideration is whether an explanation is geared towards a particular decision (local), e.g. LIME (Ribeiro *et al.*, 2016), or for the entire model (global), e.g. TCAV (Kim *et al.*, 2018) – for a planning problem this distinction can manifest in many ways: whether the explanation is for a given plan versus if it is for the model in general. From this perspective, most works in XAIP may be best understood as local explanations. However, some examples of global explanation include works like those presented by Sreedharan *et al.* (2019b); Göbelbecker *et al.* (2010).

Other Forms of Explanations Another thread of explanation generation work that doesn't quite fit into the groups described here is the one that focuses on abductive reasoning, wherein the objective is to identify a hypothesis that best explains the given set of observations. Some prominent examples of these include the method presented by Sohrabi *et al.* (2011a), where they try to identify the most preferred trajectory constraints that best fit the set of observations or the work on explanations for open-world planning (Hanheide *et al.*, 2017), which tries to identify the specific assumption (made as part of the planning process) whose failure may best explain the unexpected observations. I have chosen not to list them as their use case and motivation is usually different from the types of explanation listed here where the system is trying to explain the rationale for choosing certain actions. Thus, making it harder to put them into a single category. Though I don't want to imply that they are independent or orthogonal works, particularly because many works that use the analysis of black box/inscrutable models to generate explanations could be argued to be performing similar kinds of reasoning.

Chapter 23

CONCLUSION

This dissertation looked at the 'Human-Aware Explanation' framework that frames explanation as a process of reconciling the difference between the expectation of the human and the robot in regards to the most preferred solution. Within this framework, we looked at three salient dimensions of asymmetry between the robot and the human that need to be addressed by any process hoping to bridge such expectation mismatch. Specifically, we looked at the dimensions (a) asymmetry in knowledge, (b) asymmetry in inferential capabilities, and (c) asymmetry in vocabulary. The first two dimensions are the potential reason for the mismatch in expectation, and the third dimension controls how effectively the robot can communicate with the human to resolve the expectation mismatch.

Throughout the first three parts of the dissertation, we have looked at various explanation generation methods that can address and cope with these three dimensions. In part IV, we looked at specialized planning algorithms that can take into account the overhead of explaining a given plan when choosing the robot's behavior. In part V, we looked at methods for summarizing agent plans or policies. Finally, in Chapter 22, we used the framework of 'Human-Aware Explanation' itself as a lens to analyze the current landscape of explainable planning works.

23.1 Next Step: Leveraging the Three Dimensions in the Larger Context of Human-AI Interaction

We have already covered specific next steps for the individual explanation techniques, particularly in the overview chapters. As such, I would like to use this section to instead highlight how the use of the dimensions of asymmetry extends beyond explanation generation. In particular, I would like to make the case that these three points of asymmetry are important factors that come into play in every interaction between an automated AI system and a human. One of the dimensions, namely vocabulary mismatch, has recently been getting some attention in regards to its role in the larger picture of human-AI interaction (cf. (Kim, 2022; Kambhampati *et al.*, 2022)). However, to create agents that can truly collaborate with humans, we need to consider all three dimensions simultaneously.

To see the need for considering the other two dimensions, let us look at the problem of achieving value alignment between the human and the robot (Hadfield-Menell *et al.*, 2016). Value alignment is widely considered as being central to addressing many of the existential risks posed by superhuman AI. Even if we were to discount such risks, it is easy to see that we would want to work with AI systems that align with our values and works to maximize our inherent rewards. One of the oft-cited challenges to achieving such alignment is the fact that humans are not good at specifying underlying objectives. Sometimes referred to as the Midas problem, it points out how simple misspecification of rewards provided to a powerful optimizer could lead to unforeseen outcomes. However, once we apply the lens of human-aware AI, it is easy to see that the root cause of such misalignment still lies within the three dimensions we have seen so often in this dissertation.

Figure 23.1 visualizes the scenario where a human is trying to provide some advice or instructions to a robot. Here we see a human with a previously unspecified preference on the robot's behavior. This could be the result of some human-specific reward function or some underlying objective that they hope the robot would maximize or achieve as part of its operation. Based on their current belief about the robot model ($\mathcal{M}_h^R$), the human identifies some model updates whose inclusion in the robot model, they hope, will result in the desired behavior. Among many others, the update could take the form of some change to the robot's objective or some constraint placed on the robot's behavior. This update depends on both the human's current knowledge about the model and their inferential ability. The latter is due to the fact that the human relies on their inferential ability to verify that the updated model will result in the desired behavior. As we have seen through the chapters, humans could be off in both dimensions and thus blindly following human instructions could lead to not just suboptimal behaviors but ones that may be detrimental to the actual human objective.

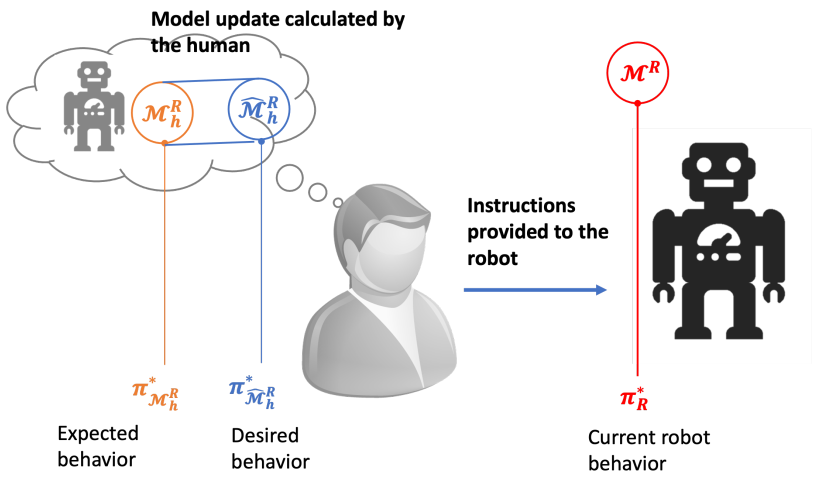

Model update calculated by the human
$\mathcal{M}_h^R$
$\widehat{\mathcal{M}}_h^R$
$\mathcal{M}^R$
Instructions provided to the robot
$\pi^*_{\mathcal{M}_h^R}$
Expected behavior
$\pi^*_{\widehat{\mathcal{M}}_h^R}$
Desired behavior
π^*_R
Current robot behavior

Figure 23.1: A mental-model centric visualization of the case where the human is trying to come up with instructions or advice that will be provided to the robot.

To concretize this setting, consider a scenario consisting of a robot, working with a human commander to find and rescue survivors from a collapsed building. Let us assume that at some point, the commander who is supervising and guiding the robot asks it to clear the north hallway of the building. The human's true objective is to look for potential victims in a specific room and may be under the incorrect belief that the north hallway is the shortest route to the room. However, unbeknownst to the human the robot has easier ways of getting in the room and following the human instruction exactly would result in the team wasting valuable time.

By the robot being cognizant of these asymmetries and having models of the human's actions and goals – thus mirroring and inverting the modeling we discussed in Chapter 1, the robot can infer the underlying intention behind the human instructions and offer alternate suggestions that may result in higher team utilities.

Abel, D., “simple_rl: Reproducible Reinforcement Learning in Python”, ICLR Workshop on Reproducibility in Machine Learning (2019).

Aeronautiques, C., A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, “Pddl— the planning domain definition language”, Tech. rep., Technical Report (1998).

Aineto, D., S. Jiménez, E. Onaindia and M. Ramírez, “Model Recognition as Planning”, in “ICAPS”, (2019).

Albore, A., H. Palacios and H. Geffner, “A Translation-Based Approach to Contingent Planning.”, in “IJCAI”, pp. 1623–1628 (2009).

Alharin, A., T.-N. Doan and M. Sartipi, “Reinforcement learning interpretation methods: A survey”, IEEE Access **8**, 171058–171077 (2020).

Alkhazraji, Y., M. Frorath, M. Grützner, T. Liebetraut, M. Ortlieb, J. Seipp, T. Springenberg, P. Stahl, J. Wülfing, M. Helmert and R. Mattmüller, “Pyperplan”, https://bitbucket.org/malte/pyperplan (2016).

Allan, K., “Common ground – aka “common knowledge”, “mutual knowledge*”, “shared knowledge”, “assumed familiarity”, “presumed background information””, Handbook of Pragmatics (2013).

Amir, D. and O. Amir, “Highlights: Summarizing agent behavior to people”, in “Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems”, pp. 1168–1176 (2018).

Anderson, A., J. Dodge, A. Sadarangani, Z. Juozapaitis, E. Newman, J. Irvine, S. Chattopadhyay, A. Fern and M. Burnett, “Explaining Reinforcement Learning to Mere Mortals: An Empirical Study”, in “IJCAI”, (2019).

Arora, S. and B. Barak, *Computational Complexity - A Modern Approach* (Cambridge University Press, 2009).

Asai, M. and A. Fukunaga, “Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary”, in “AAAI”, (2018).

Atrey, A., K. Clary and D. Jensen, “Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning”, arXiv preprint arXiv:1912.05743 (2019).

Bacchus, F. and F. Kabanza, “Using temporal logics to express search control knowledge for planning”, Artificial Intelligence **116**, 1-2, 123–191 (2000).

Bäckström, C. and P. Jonsson, “Bridging the gap between refinement and heuristics in abstraction”, in “IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013”, pp. 2261–2267 (IJCAI/AAAI, 2013).

Baier, J. A., C. Fritz and S. A. McIlraith, "Exploiting procedural domain control knowledge in state-of-the-art planners.", in "ICAPS", (2007).

Baier, J. A. and S. A. McIlraith, "Planning with first-order temporally extended goals using heuristic search", in "AAAI", pp. 788–795 (2006).

Baker, C. L., R. Saxe and J. B. Tenenbaum, "Action Understanding as Inverse Planning", Cognition (2009).

Baker, C. L., J. B. Tenenbaum and R. R. Saxe, "Goal Inference as Inverse Planning", in "Proceedings of the Annual Meeting of the Cognitive Science Society", (2007).

Bansal, G., B. Nushi, E. Kamar, E. Horvitz and D. S. Weld, "Is the most accurate ai the best teammate? optimizing ai for teamwork", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 35, pp. 11405–11414 (2021).

Bartlett, C. E., "Communication between Teammates in Urban Search and Rescue", Thesis Arizona State University (2015).

Bau, D., B. Zhou, A. Khosla, A. Oliva and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 6541–6549 (2017).

Bayani, D. and S. Mitsch, "Fanoos: Multi-resolution, multi-strength, interactive explanations for learned systems", in "IJCAI XAI Workshop", (2020).

Bechlivanidis, C., D. A. Lagnado, J. C. Zemla and S. Sloman, "Concreteness and abstraction in everyday explanation", Psychonomic bulletin & review **24**, 5, 1451–1464 (2017).

Benton, J., N. Lipovetzky, E. Onaindia, D. E. Smith and S. Srivastava, eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019* (AAAI Press, 2019), URL `https://aaai.org/ojs/index.php/ICAPS/issue/view/239`.

Bercher, P., S. Biundo, T. Geier, T. Hoernle, F. Nothdurft, F. Richter and B. Schattenberg, "Plan, repair, execute, explain - how planning helps to assemble your home theater", in "Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014", (AAAI, 2014).

Bernhard, K. and J. Vygen, "Combinatorial optimization: Theory and algorithms", Springer, Third Edition, 2005. (2008).

Bertsekas, D. P., *Dynamic programming and optimal control, 3rd Edition* (Athena Scientific, 2005), URL `https://www.worldcat.org/oclc/314894080`.

Biundo, S., M. Do, R. Goldman, M. Katz, Q. Yang and H. H. Zhuo, eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China (virtual), August 2-13, 2021* (AAAI Press, 2021).

Bolander, T., M. H. Jensen and F. Schwarzentruber, "Complexity Results in Epistemic Planning", in "IJCAI", (2015).

Bonet, B. and H. Geffner, "Planning as heuristic search", Artificial Intelligence **129**, 1-2, 5–33 (2001).

Bonet, B. and H. Geffner, "An Algorithm Better Than AO*?", in "AAAI", pp. 1343–1348 (2005).

Bonet, B. and H. Geffner, "Learning first-order symbolic representations for planning from the structure of the state space", in "ECAI", (2019).

Boonzaier, A., J. McClure and R. M. Sutton, "Distinguishing the effects of beliefs and preconditions: The folk psychology of goals and actions", European journal of social psychology **35**, 6, 725–740 (2005).

Brandao, M., A. Coles and D. Magazzeni, "Explaining path plan optimality: Fast explanation methods for navigation meshes using full and incremental inverse optimization", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 31, pp. 56–64 (2021).

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "Openai gym", CoRR **abs/1606.01540**, URL `http://arxiv.org/abs/1606.01540` (2016).

Broekens, J., M. Harbers, K. Hindriks, K. Van Den Bosch, C. Jonker and J.-J. Meyer, "Do you get it? user-evaluated explainable bdi agents", in "German Conference on Multiagent System Technologies", pp. 28–39 (Springer, 2010).

Bryce, D., J. Benton and M. W. Boldt, "Maintaining evolving domain models", in "Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016", pp. 3053–3059 (IJCAI/AAAI Press, 2016).

Bryce, D. and O. Buffet, "6th international planning competition: Uncertainty part", Proceedings of the 6th International Planning Competition (IPC'08) (2008).

Bylander, T., "The computational complexity of propositional STRIPS planning", Artif. Intell. **69**, 1-2, 165–204 (1994).

Byrne, R. M., "Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning.", in "IJCAI", pp. 6276–6282 (2019).

Cai, C. J., E. Reif, N. Hegde, J. Hipp, B. Kim, D. Smilkov, M. Wattenberg, F. Viegas, G. S. Corrado, M. C. Stumpe *et al.*, "Human-centered tools for coping with imperfect algorithms during medical decision-making", in "Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems", pp. 1–14 (2019).

Carbonell, J. G. and Y. Gil, "Learning by experimentation: The operator refinement method", in "Machine learning", pp. 191–213 (Elsevier, 1990).

Cashmore, M., A. Collins, B. Krarup, S. Krivic, D. Magazzeni and D. Smith, "Towards Explainable AI Planning as a Service", in "XAIP Workshop", (2019).

Chakraborti, T., K. P. Fadnis, K. Talamadupula, M. Dholakia, B. Srivastava, J. O. Kephart and R. K. Bellamy, "Planning and Visualization for a Smart Meeting Room Assistant – A Case Study in the Cognitive Environments Laboratory at IBM T.J. Watson Research Center, Yorktown", AI Communications (2019a).

Chakraborti, T. and S. Kambhampati, "(How) Can AI Bots Lie?", in "ICAPS Workshop on Explainable AI Planning (XAIP)", (2019a).

Chakraborti, T. and S. Kambhampati, "(when) can ai bots lie?", in "Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society", pp. 53–59 (2019b).

Chakraborti, T. and S. Kambhampati, "(when) can ai bots lie?", in "Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society", pp. 53–59 (2019c).

Chakraborti, T. and Y. Khazaeni, "D3BA: A Tool for Optimizing Business Processes Using Non-Deterministic Planning", in "AAAI Workshop on Intelligent Process Automation (IPA-20)", (2020).

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 Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019", pp. 86–96 (AAAI Press, 2019b).

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 "ICAPS", (2019c).

Chakraborti, T., C. Muise, S. Agarwal and L. Lastras, "D3WA: An Intelligent Model Acquisition Interface for Interactive Specification of Dialog Agents", in "AAAI & ICAPS System Demonstration Tracks", (2019d).

Chakraborti, T., S. Sreedharan, S. Grover and S. Kambhampati, "Plan Explanations as Model Reconciliation – An Empirical Study", in "HRI", (2019e).

Chakraborti, T., S. Sreedharan and S. Kambhampati, "Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots in a Mixed-Reality Workspace ", in "IROS", (2018).

Chakraborti, T., S. Sreedharan and S. Kambhampati, "Balancing explicability and explanations in human-aware planning", in "IJCAI 2019", pp. 1335–1343 (IJCAI Organization, 2019f).

Chakraborti, T., S. Sreedharan and S. Kambhampati, "The emerging landscape of explainable ai planning and decision making", in "IJCAI", (2020).

Chakraborti, T., S. Sreedharan, Y. Zhang and S. Kambhampati, "Plan explanations as model reconciliation: Moving beyond explanation as soliloquy", in "IJCAI 2017", pp. 156–163 (ijcai.org, 2017).

Chandrasekaran, B., M. C. Tanner and J. R. Josephson, "Explaining control strategies in problem solving", IEEE Annals of the History of Computing **4**, 01, 9–15 (1989).

Cimatti, A., M. Pistore, M. Roveri and P. Traverso, "Weak, strong, and strong cyclic planning via symbolic model checking", Artificial Intelligence **147**, 1-2, 35–84 (2003).

Clarke, E., O. Grumberg, S. Jha, Y. Lu and H. Veith, "Counterexample-guided abstraction refinement", in "International Conference on Computer Aided Verification", pp. 154–169 (Springer, 2000).

Cohen, P. R. and C. R. Perrault, "Elements of a Plan-Based Theory of Speech Acts", Cognitive science (1979).

Cooke, N. J., J. C. Gorman, C. W. Myers and J. L. Duran, "Interactive Team Cognition", Cognitive Science (2013).

Cooper, R. P. and T. Shallice, "Hierarchical schemas and goals in the control of sequential behavior.", Psychological Review (2006).

Coppens, Y., K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber and D. Magazzeni, "Distilling deep reinforcement learning policies in soft decision trees", in "Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence", pp. 1–6 (2019).

Culberson, J. C. and J. Schaeffer, "Pattern databases", Computational Intelligence **14**, 3, 318–334 (1998).

De Giacomo, G., L. Iocchi, M. Favorito and F. Patrizi, "Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications", in "ICAPS", (2019).

De Giacomo, G. and M. Y. Vardi, "Synthesis for ltl and ldl on finite traces.", in "IJCAI", vol. 15, pp. 1558–1564 (2015).

Dietterich, T. G., "The MAXQ method for hierarchical reinforcement learning", in "Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998", pp. 118–126 (Morgan Kaufmann, 1998).

Dodson, T., N. Mattei, J. T. Guerin and J. Goldsmith, "An English-Language Argumentation Interface for Explanation Generation with Markov Decision Processes in the Domain of Academic Advising", TiiS (2013).

Donnarumma, F., D. Maisto and G. Pezzulo, "Problem solving as probabilistic inference with subgoaling: explaining human successes and pitfalls in the tower of hanoi", PLoS computational biology **12**, 4, e1004864 (2016).

Dragan, A. and S. Srinivasa, "Generating Legible Motion", in "RSS", (2013).

Dragan, A. D., "Robot Planning with Mathematical Models of Human State and Action", arXiv:1705.04226 (2017).

Dragan, A. D., K. C. Lee and S. S. Srinivasa, "Legibility and predictability of robot motion", in "HRI", (2013).

Edelkamp, S., "Planning with pattern databases", in "ECP", (2000).

Eifler, R., M. Cashmore, J. Hoffmann, D. Magazzeni and M. Steinmetz, "A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning", in "AAAI 2020", pp. 9818–9826 (AAAI Press, 2020a).

Eifler, R., M. Steinmetz, A. Torralba and J. Hoffmann, "Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties.", in "IJCAI", pp. 4091–4097 (ijcai.org, 2020b).

Eiter, T., E. Erdem, M. Fink and J. Senko, "Updating Action Domain Descriptions", Artificial Intelligence Journal (2010).

Fagin, R., Y. Moses, J. Y. Halpern and M. Y. Vardi, *Reasoning About Knowledge* (MIT press, 2003).

Ferrer-Mestres, J., T. G. Dietterich, O. Buffet and I. Chades, "Solving k-mdps", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 30, pp. 110–118 (2020).

Finucane, C., G. Jing and H. Kress-Gazit, "Ltlmop: Experimenting with language, temporal logic and robot control", in "IROS", pp. 1988–1993 (IEEE, 2010).

Fisac, J. F., A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin and A. D. Dragan, "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions", in "RSS", (2018).

Fisac, J. F., C. Liu, J. B. Hamrick, S. Sastry, J. K. Hedrick, T. L. Griffiths and A. D. Dragan, "Generating Plans that Predict Themselves", in "Algorithmic Foundations of Robotics", (Springer, 2020).

Fox, M., R. Howey and D. Long, "Validating Plans in the Context of Processes and Exogenous Events", in "AAAI", (2005).

Fox, M., D. Long and D. Magazzeni, "Explainable Planning", in "IJCAI Workshop on Explainable AI (XAI)", (2017).

Freund, Y., R. Schapire and N. Abe, "A short introduction to boosting", Journal-Japanese Society For Artificial Intelligence **14**, 771-780, 1612 (1999).

Fritz, C. and S. A. McIlraith, "Monitoring plan optimality during execution.", in "ICAPS", pp. 144–151 (2007).

Ganesan, R. K., *Mediating Human-Robot Collaboration through Mixed Reality Cues*, Ph.D. thesis, Arizona State University (2017).

Garfinkel, A., *Forms of explanation: Rethinking the questions in social theory* (Yale University Press, 1982).

Geffner, H. and B. Bonet, *A Concise Introduction to Models and Methods for Automated Planning*, Synthesis Lectures on Artificial Intelligence and Machine Learning (Morgan & Claypool Publishers, 2013a).

Geffner, H. and B. Bonet, "A concise introduction to models and methods for automated planning", Synthesis Lectures on Artificial Intelligence and Machine Learning **8**, 1, 1–141 (2013b).

Geißer, F., T. Keller and R. Mattmüller, "Abstractions for planning with state-dependent action costs", ICAPS (2016).

Ghorbani, A., J. Wexler, J. Y. Zou and B. Kim, "Towards automatic concept-based explanations", in "Advances in Neural Information Processing Systems 32", pp. 9277–9286 (2019).

Givan, R., S. Leach and T. Dean, "Bounded-parameter markov decision processes", Artificial Intelligence **122**, 1-2, 71–109 (2000).

Göbelbecker, M., T. Keller, P. Eyerich, M. Brenner and B. Nebel, "Coming up with good excuses: What to do when no plan can be found", in "Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010", pp. 81–88 (AAAI, 2010).

Greydanus, S., A. Koul, J. Dodge and A. Fern, "Visualizing and understanding atari agents", in "Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018", vol. 80 of *Proceedings of Machine Learning Research*, pp. 1787–1796 (PMLR, 2018).

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

Grover, S., S. Sengupta, T. Chakraborti, A. P. Mishra and S. Kambhampati, "RADAR: Automated Task Planning for Proactive Decision Support", in "HCI Journal", (2020a).

Grover, S., S. Sengupta, T. Chakraborti, A. P. Mishra and S. Kambhampati, "Radar: automated task planning for proactive decision support", Human–Computer Interaction **35**, 5-6, 387–412 (2020b).

Grumberg, O. and H. Veith, *25 years of model checking: history, achievements, perspectives*, vol. 5000 (Springer, 2008).

Gunning, D., "Explainable artificial intelligence (xai)", Defense Advanced Research Projects Agency (DARPA), nd Web **2**, 2 (2017).

Hadfield-Menell, D., S. J. Russell, P. Abbeel and A. Dragan, "Cooperative Inverse Reinforcement Learning", in "NIPS", (2016).

Hamidi-Haines, M., Z. Qi, A. Fern, F. Li and P. Tadepalli, "Interactive naming for explaining deep neural networks: a formative study", arXiv (2018).

Hanheide, M., M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janícek, H. Zender, G. M. Kruijff, N. Hawes and J. L. Wyatt, "Robot task planning and explanation in open and uncertain worlds", Artif. Intell. **247**, 119–150 (2017).

Hankinson, R. J., *Cause and explanation in ancient Greek thought* (Oxford University Press, 2001).

Hansen, E. A. and S. Zilberstein, "Lao*: A heuristic search algorithm that finds solutions with loops", Artificial Intelligence **129**, 1-2, 35–62 (2001).

Hart, P. E., N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", IEEE transactions on Systems Science and Cybernetics **4**, 2, 100–107 (1968).

Hart, S. G. and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research", in "Advances in psychology", vol. 52, pp. 139–183 (Elsevier, 1988).

Haslum, P., J. Slaney, S. Thiébaux *et al.*, "Incremental lower bounds for additive cost planning problems.", in "ICAPS", vol. 12, pp. 74–82 (2012).

Hayes, B. and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation", in "2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)", pp. 303–312 (IEEE, 2017).

Helmert, M., "The Fast Downward Planning System", JAIR (2006).

Helmert, M. and C. Domshlak, "Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?", in "ICAPS", (2009).

Hempel, C. G. and P. Oppenheim, "Studies in the logic of explanation", Philosophy of science **15**, 2, 135–175 (1948).

Herzig, A., V. Menezes, L. N. de Barros and R. Wassermann, "On the Revision of Planning Tasks", in "ECAI", (2014).

Hitchcock, C. and J. Woodward, "Explanatory generalizations, part ii: Plumbing explanatory depth", Noûs **37**, 2, 181–199 (2003).

Hoffman, R. R., S. T. Mueller, G. Klein and J. Litman, "Metrics for explainable ai: Challenges and prospects", arXiv preprint arXiv:1812.04608 (2018).

Hoffmann, J., "The metric-ff planning system: Translating"ignoring delete lists"to numeric state variables", Journal of artificial intelligence research **20**, 291–341 (2003).

Hoffmann, J., P. Kissmann and Á. Torralba, “”distance”? who cares? tailoring merge-and-shrink heuristics to detect unsolvability”, in “ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)”, vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 441–446 (IOS Press, 2014).

Hoffmann, J. and D. Magazzeni, “Explainable AI Planning (XAIP): Overview and the Case of Contrastive Explanation”, in “Reasoning Web. Explainable Artificial Intelligence”, (2019), extended Abstract.

Hoffmann, J. and B. Nebel, “The ff planning system: Fast plan generation through heuristic search”, Journal of Artificial Intelligence Research **14**, 253–302 (2001).

Hoffmann, J., J. Porteous and L. Sebastia, “Ordered landmarks in planning”, JAIR **22**, 215–278 (2004).

Howey, R., D. Long and M. Fox, “VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL”, in “ICTAI”, (2004).

Huang, X., B. Fang, H. Wan and Y. Liu, “A General Multi-Agent Epistemic Planner Based on Higher-Order Belief Change”, in “IJCAI”, (2018).

Huber, T. and E. André, “Introducing Selective Layer-Wise Relevance Propagation to Dueling Deep Q-learning”, in “XAI Workshop”, (2019).

Illanes, L., X. Yan, R. T. Icarte and S. A. McIlraith, “Symbolic plans as high-level instructions for reinforcement learning”, in “ICAPS”, (2020).

International Planning Competition, “IPC Competition Domains”, `https://goo.gl/i35bxc` (2011).

Isaac, A. and W. Bridewell, “White Lies on Silver Tongues: Why Robots Need to Deceive (and How)”, Journal of Robot Ethics (2017).

Iyer, R., Y. Li, H. Li, M. Lewis, R. Sundar and K. Sycara, “Transparency and explanation in deep reinforcement learning neural networks”, in “Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society”, pp. 144–150 (2018).

Juozapaitis, Z., A. Koul, A. Fern, M. Erwig and F. Doshi-Velez, “Explainable Reinforcement Learning via Reward Decomposition”, in “IJCAI XAI Workshop”, pp. 47–53 (2019).

Kaelbling, L. P., “Hierarchical learning in stochastic domains: Preliminary results”, in “Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993”, pp. 167–173 (1993).

Kahneman, D. and A. Tversky, “Prospect theory: An analysis of decision under risk”, in “Handbook of the fundamentals of financial decision making: Part I”, pp. 99–127 (World Scientific, 2013).

Kambhampati, S., "A Classification of Plan Modification Strategies Based on Coverage and Information Requirements", in "AAAI Spring Symposium on Case Based Reasoning", (1990).

Kambhampati, S., C. A. Knoblock and Q. Yang, "Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning", Artificial Intelligence **76**, 1, 167–238 (1995).

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", in "AAAI", (2022).

Kasenberg, D., R. Thielstrom and M. Scheutz, "Generating explanations for temporal logic planner decisions", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 30, pp. 449–458 (2020).

Kautz, H. A. and J. F. Allen, "Generalized plan recognition.", in "AAAI", vol. 86, p. 5 (1986).

Kautz, H. A., D. A. McAllester and B. Selman, "Encoding plans in propositional logic", in "KR 96", pp. 374–384 (Morgan Kaufmann, 1996).

Keane, M. T. and B. Smyth, "Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI)", in "Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings", vol. 12311 of *Lecture Notes in Computer Science*, pp. 163–178 (Springer, 2020).

Keller, T. and P. Eyerich, "A polynomial all outcome determinization for probabilistic planning", in "Twenty-First International Conference on Automated Planning and Scheduling", (2011).

Kelley, H. H., "Attribution theory in social psychology.", in "Nebraska symposium on motivation", (University of Nebraska Press, 1967).

Keren, S., A. Gal and E. Karpas, "Privacy Preserving Plans in Partially Observable Environments", in "IJCAI", (2016).

Keren, S., L. Pineda, A. Gal, E. Karpas and S. Zilberstein, "Equi-reward Utility Maximizing Design in Stochastic Environments", in "IJCAI", (2017).

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

Keyder, E. R., J. Hoffmann, P. Haslum *et al.*, "Semi-relaxed plan heuristics.", in "ICAPS", (2012).

Khan, O. Z., P. Poupart and J. P. Black, "Minimal sufficient explanations for factored markov decision processes", in "Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009", pp. 194–200 (AAAI, 2009).

Kim, B., “Beyond interpretability: developing a language to shape our relationships with ai”, URL `https://www.youtube.com/watch?v=Ub45cGEcTB0&t=6s`, iCLR (2022).

Kim, B., W. M., J. Gilmer, C. C., W. J., , F. Viegas and R. Sayres, “ Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) ”, ICML (2018).

Kim, J., C. Muise, A. Shah, S. Agarwal and J. Shah, “Bayesian inference of linear temporal logic specifications for contrastive explanations”, in “Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019”, edited by S. Kraus, pp. 5591–5598 (ijcai.org, 2019).

Klößner, T., J. Hoffmann, M. Steinmetz and A. Torralba, “Pattern databases for goal-probability maximization in probabilistic planning”, in “Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China, August 2-13, 2021”, vol. 31, pp. 201–209 (2021).

Koh, P. W., T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim and P. Liang, “Concept bottleneck models”, in “ICML”, (2020).

Kolobov, A., Mausam and D. S. Weld, “A theory of goal-oriented mdps with dead ends”, in “Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012”, pp. 438–447 (AUAI Press, 2012), URL `https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2305&proceeding_id=28`.

Kolobov, A., Mausam, D. S. Weld and H. Geffner, “Heuristic search for generalized stochastic shortest path mdps”, in “Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany, June 11-16, 2011”, pp. 130–137 (AAAI, 2011).

Kolobov, A., D. Weld *et al.*, “Sixthsense: Fast and reliable recognition of dead ends in mdps”, in “AAAI”, (2010).

Kominis, F. and H. Geffner, “Beliefs In Multiagent Planning: From One Agent to Many”, in “ICAPS”, (2015).

Kominis, F. and H. Geffner, “Multiagent Online Planning with Nested Beliefs and Dialogue”, in “ICAPS”, (2017).

Konidaris, G., L. P. Kaelbling and T. Lozano-Perez, “From skills to symbols: Learning symbolic representations for abstract high-level planning”, Journal of Artificial Intelligence Research **61**, 215–289 (2018).

Koul, A., A. Fern and S. Greydanus, “Learning finite state representations of recurrent policy networks”, in “7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019”, (OpenReview.net, 2019).

Krarup, B., M. Cashmore, D. Magazzeni and T. Miller, "Model-Based Contrastive Explanations for Explainable Planning", in "XAIP Workshop", (2019).

Krarup, B., S. Krivic, D. Magazzeni, D. Long, M. Cashmore and D. E. Smith, "Contrastive explanations of plans through model restrictions", arXiv preprint arXiv:2103.15575 (2021).

Kulkarni, A., T. Chakraborti, Y. Zha, S. G. Vadlamudi, Y. Zhang and S. Kambhampati, "Explicable Robot Planning as Minimizing Distance from Expected Behavior", in "AAMAS Extended Abstract", (2019a).

Kulkarni, A., S. Sreedharan, S. Keren, T. Chakraborti, D. Smith and S. Kambhampati, "Designing environments conducive to interpretable robot behavior", IROS (2020).

Kulkarni, A., S. Srivastava and S. Kambhampati, "A Unified Framework for Planning in Adversarial and Cooperative Environments", in "AAAI", (2019b).

Kumar, A., S. L. Vasileiou, M. Bancilhon, A. Ottley and W. Yeoh, "Vizxp: A visualization framework for conveying explanations to users in model reconciliation problems", in "ICAPS 2021 Workshop on Explainable AI Planning", (2021).

Kwon, M., S. Huang and A. Dragan, "Expressing Robot Incapability", in "HRI", (2018).

L Griffiths, T., C. Kemp and J. B Tenenbaum, "Bayesian Models of Cognition", The Cambridge Handbook of Computational Psychology (2008).

Lage, I., D. Lifschitz, F. Doshi-Velez and O. Amir, "Exploring Computational User Models for Agent Policy Summarization", in "IJCAI", (2019).

Lakkaraju, H., J. Adebayo and S. Singh, "Explaining Machine Learning Predictions: State-of-the-art, Challenges, and Opportunities", in "NeurIPS Tutorial", (2020), `https://explainml-tutorial.github.io/`.

Langley, P., "Varieties of explainable agency", in "ICAPS Workshop on Explainable AI Planning (XAIP)", vol. 1365 (2019).

Langley, P., B. Meadows, M. Sridharan and D. Choi, "Explainable Agency for Intelligent Autonomous Systems", in "AAAI/IAAI", (2017).

Le, T., F. Fabiano, T. C. Son and E. Pontelli, "EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains", in "ICAPS", (2018).

Li, L., T. J. Walsh and M. L. Littman, "Towards a unified theory of state abstraction for mdps.", ISAIM **4**, 5 (2006).

Lin, S. and P. Bercher, "Change the world – how hard can that be? on the computational complexity of fixing planning models", in "IJCAI 2021", pp. 4152–4159 (IJCAI Organization, 2021).

Lin, Z., K. Lam and A. Fern, "Contrastive explanations for reinforcement learning via embedded self predictions", in "9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021", (OpenReview.net, 2021).

Linux, *urandom(4) - Linux man page* (2013).

Lombrozo, T., "The Structure and Function of Explanations", Trends in Cognitive Sciences (2006).

Lombrozo, T., "Explanation and Abductive Inference", Oxford Handbook of Thinking and Reasoning (2012).

Luss, R., P.-Y. Chen, A. Dhurandhar, P. Sattigeri, Y. Zhang, K. Shanmugam and C.-C. Tu, "Generating contrastive explanations with monotonic attribute functions", arXiv preprint arXiv:1905.12698 (2019).

MacNally, A. M., N. Lipovetzky, M. Ramirez and A. R. Pearce, "Action Selection for Transparent Planning", in "AAMAS", (2018).

Madumal, P., T. Miller, L. Sonenberg and F. Vetere, "Distal explanations for explainable reinforcement learning agents", arXiv preprint arXiv:2001.10284 (2020a).

Madumal, P., T. Miller, L. Sonenberg and F. Vetere, "Explainable reinforcement learning through a causal lens", in "The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020", pp. 2493–2500 (AAAI Press, 2020b).

Madumal, P., T. Miller, L. Sonenberg and F. Vetere, "Explainable reinforcement learning through a causal lens", in "AAAI", (2020c).

Magnaguagno, M. C., R. F. Pereira, M. D. Móre and F. Meneguzzi, "Web Planner: A Tool to Develop Classical Planning Domains and Visualize Heuristic State-Space Search", in "ICAPS Workshop on User Interfaces in Scheduling and Planning", (2017).

Marthi, B., S. J. Russell and J. A. Wolfe, "Angelic semantics for high-level actions.", in "ICAPS", pp. 232–239 (2007).

Martin, K., A. Liret, N. Wiratunga, G. Owusu and M. Kern, "Developing a catalogue of explainability methods to support expert and non-expert users", in "International Conference on Innovative Techniques and Applications of Artificial Intelligence", pp. 309–324 (Springer, 2019).

Masters, P. and S. Sardina, "Deceptive path-planning", in "IJCAI", (2017).

McGovern, A. and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density", in "Proceedings of the Eighteenth International Conference on Machine Learning", (2001).

Meadows, B. L., P. Langley and M. J. Emery, “Seeing beyond shadows: Incremental abductive reasoning for plan understanding.”, in “AAAI Workshop: Plan, Activity, and Intent Recognition”, vol. 13, p. 13 (2013).

Miller, T., “Explanation in Artificial Intelligence: Insights from the Social Sciences”, Artificial Intelligence Journal (2017a).

Miller, T., “Social Planning – Reasoning with and about others”, Invited Talk at IJCAI Workshop on Impedance Matching in Cognitive Partnerships (2017b).

Miller, T., “Contrastive Explanation: A Structural-Model Approach”, arXiv:1811.03163 (2018).

Miura, S. and S. Zilberstein, “Maximizing plan legibility in stochastic environments”, in “AAMAS”, (2020).

Muise, C., *Exploiting Relevance to Improve Robustness and Flexibility in Plan Generation and Execution* (Doctoral Dissertation, University of Toronto, 2014).

Muise, C., T. Chakraborti, S. Agarwal, O. Bajgar, A. Chaudhary, L. A. Lastras-Montano, J. Ondrej, M. Vodolan and C. Wiecha, “Planning for Goal-Oriented Dialogue Systems”, arXiv:1910.08137 (2019a).

Muise, C., M. Vodolan, S. Agarwal, O. Bajgar and L. Lastras, “Executing Contingent Plans: Challenges in Deploying Artificial Agents”, in “AAAI Fall Symposium”, (2019b).

Muise, C. J., V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce and L. Sonenberg, “Planning Over Multi-Agent Epistemic States: A Classical Planning Approach.”, in “AAAI”, (2015).

Muise, C. J., S. A. McIlraith and J. C. Beck, “Improved non-deterministic planning by exploiting state relevance”, in “Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012”, (AAAI, 2012).

Murphy, R. R., S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset and A. M. Erkmen, “Search and rescue robotics”, in “Springer Handbook of Robotics”, edited by B. Siciliano and O. Khatib, pp. 1151–1173 (Springer, 2008).

Myers, K. L., “Advisable planning systems”, Advanced Planning Technology pp. 206–209 (1996).

Nau, D., Y. Cao, A. Lotem and H. Munoz-Avila, “The shop planning system”, AI Magazine **22**, 3, 91 (2001).

Newell, A., H. A. Simon *et al.*, *Human problem solving*, vol. 104 (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., M. Do, A. E. Gerevini, I. Serina, B. Srivastava and S. Kambhampati, “Generating Diverse Plans to Handle Unknown and Partially Known User Preferences”, Artificial Intelligence Journal (2012).

Nikolaidis, S., A. Kuznetsov, D. Hsu and S. S. Srinivasa, “Formalizing human-robot mutual adaptation: A bounded memory model”, in “The Eleventh ACM/IEEE International Conference on Human Robot Interation, HRI 2016, Christchurch, New Zealand, March 7-10, 2016”, edited by C. Bartneck, Y. Nagai, A. Paiva and S. Sabanovic, pp. 75–82 (IEEE/ACM, 2016).

Nilsson, N. J., *Principles of Artificial Intelligence* (Morgan Kaufmann, 1980).

Olson, M. L., L. Neal, F. Li and W.-K. Wong, “Counterfactual states for atari agents via generative deep learning”, arXiv preprint arXiv:1909.12969 (2019).

Palacios, H. and H. Geffner, “Compiling uncertainty away in conformant planning problems with bounded width”, Journal of Artificial Intelligence Research **35**, 623–675 (2009).

Palmieri, J. J. and T. A. Stern, “Lies in the doctor-patient relationship”, Primary care companion to the Journal of clinical psychiatry **11**, 4, 163 (2009).

Papadimitriou, C. H., *Computational complexity* (Addison-Wesley, 1994).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, “Scikit-learn: Machine learning in Python”, Journal of Machine Learning Research **12**, 2825–2830 (2011).

Porteous, J., A. Lindsay, J. Read, M. Truran and M. Cavazza, “Automated Extension of Narrative Planning Domains with Antonymic Operators”, in “AAMAS”, (2015).

Premack, D. and G. Woodruff, “Does the chimpanzee have a theory of mind?”, Behavioral and brain sciences **1**, 4, 515–526 (1978).

Prolific, UK, “Prolific”, (2021).

Puri, N., S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy and S. Singh, “Explain your move: Understanding agent actions using specific and relevant feature attribution”, arXiv preprint arXiv:1912.12191 (2019).

Puterman, M. L., *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).

Raman, V. and H. Kress-Gazit, “Towards minimal explanations of unsynthesizability for high-level robot behaviors”, in “IROS”, pp. 757–762 (IEEE, 2013).

Ramesh, R., M. Tomar and B. Ravindran, “Successor options: An option discovery framework for reinforcement learning”, in “IJCAI”, (2019).

Ramírez, M. and H. Geffner, “Plan recognition as planning”, in “Twenty-First International Joint Conference on Artificial Intelligence”, (2009).

Ramırez, M. and H. Geffner, “Probabilistic plan recognition using off-the-shelf classical planners”, in “AAAI”, pp. 1121–1126 (2010).

Reddy, S., A. D. Dragan and S. Levine, “Where do you think you’re going?: Inferring beliefs about dynamics from behavior”, in “Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada”, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, pp. 1461–1472 (2018).

Ribeiro, M. T., S. Singh and C. Guestrin, “”why should I trust you?”: Explaining the predictions of any classifier”, in “Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016”, edited by B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen and R. Rastogi, pp. 1135–1144 (ACM, 2016).

Richter, S., M. Helmert and M. Westphal, “Landmarks revisited.”, in “AAAI”, vol. 8, pp. 975–982 (2008).

Richter, S. and M. Westphal, “The lama planner: Guiding cost-based anytime planning with landmarks”, Journal of Artificial Intelligence Research **39**, 127–177 (2010).

Sacerdoti, E. D., “Planning in a hierarchy of abstraction spaces”, Artificial intelligence **5**, 2, 115–135 (1974).

Schaul, T., J. Quan, I. Antonoglou and D. Silver, “Prioritized experience replay”, arXiv preprint arXiv:1511.05952 (2015).

Schrader, M.-P. B., “gym-sokoban”, `https://github.com/mpSchrader/gym-sokoban` (2018).

Seegebarth, B., F. Müller, B. Schattenberg and S. Biundo, “Making hybrid plans more clear to human users - A formal approach for generating sound explanations”, in “Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012”, (AAAI, 2012).

Seipp, J. and M. Helmert, “Diverse and additive cartesian abstraction heuristics”, in “Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014”, (AAAI, 2014).

Seipp, J. and M. Helmert, “Counterexample-guided cartesian abstraction refinement for classical planning”, J. Artif. Intell. Res. **62**, 535–577 (2018).

Senator, T., "Science of artificial intelligence and learning for open-world novelty (sail-on)", `https://www.darpa.mil/program/science-of-artificial-intelligence-and-learning-for-open-world-novelty`, accessed: 2021-06-07 (2019).

Sengupta, S., T. Chakraborti and S. Kambhampati, "Ma-radar–a mixed-reality interface for collaborative decision making", ICAPS UISP (2018).

Shah, N., P. Verma, T. Angle and S. Srivastava, "Jedai: A system for skill-aligned explainable robot planning", in "ICAPS 2022 Workshop on Explainable AI Planning", (2022).

Shvo, M., T. Q. Klassen and S. A. McIlraith, "Towards the role of theory of mind in explanation", in "Explainable, Transparent Autonomous Agents and Multi-Agent Systems - Second International Workshop, EXTRAAMAS 2020, Auckland, New Zealand, May 9-13, 2020, Revised Selected Papers", vol. 12175 of *Lecture Notes in Computer Science*, pp. 75–93 (Springer, 2020).

Silver, D., S. Singh, D. Precup and R. S. Sutton, "Reward is enough", Artificial Intelligence p. 103535 (2021).

Simon, H. A., "A behavioral model of rational choice", Models of man, social and rational: Mathematical essays on rational human behavior in a social setting pp. 241–260 (1957).

Simon, H. A. and A. Newell, "Human problem solving: The state of the theory in 1970.", American Psychologist **26**, 2, 145 (1971).

Smith, D. E., "Planning as an Iterative Process", in "AAAI", (2012).

Sohrabi, S., J. A. Baier and S. A. McIlraith, "Preferred explanations: Theory and generation via planning", in "Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011", edited by W. Burgard and D. Roth (AAAI Press, 2011a).

Sohrabi, S., J. A. Baier and S. A. McIlraith, "Preferred explanations: Theory and generation via planning.", in "AAAI", (2011b).

Soni, U., S. Sreedharan and S. Kambhampati, "Not all users are the same: Providing personalized explanations for sequential decision making problems", in "2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 6240–6247 (IEEE, 2021).

Sreedharan, S., T. Chakraborti and S. Kambhampati, "Handling model uncertainty and multiplicity in explanations via model reconciliation", in "ICAPS 2018", pp. 518–526 (AAAI Press, 2018a).

Sreedharan, S., T. Chakraborti and S. Kambhampati, "Foundations of explanations as model reconciliation", Artif. Intell. **301**, 103558 (2021a).

Sreedharan, S., T. Chakraborti, C. Muise and S. Kambhampati, “Expectation-aware planning: A unifying framework for synthesizing and executing self-explaining plans for human-aware planning”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 34, pp. 2518–2526 (2020a).

Sreedharan, S., T. Chakraborti, C. Muise, Y. Khazaeni and S. Kambhampati, “D3WA+: A Case Study of XAIP in a Model Acquisition Task”, in “ICAPS”, (2020b).

Sreedharan, S., T. Chakraborti, Y. Rizk and Y. Khazaeni, “Explainable composition of aggregated assistants”, in “ICAPS 2021 Workshop on Explainable AI Planning”, (2021b).

Sreedharan, S., A. O. Hernandez, A. P. Mishra and S. Kambhampati, “Model-free model reconciliation”, in “IJCAI 2019”, pp. 587–594 (IJCAI Organization, 2019a).

Sreedharan, S. and S. Kambhampati, “Leveraging pddl to make inscrutable agents interpretable: A case for post hoc symbolic explanations for sequential-decision making problems”, in “ICAPS 2021 Workshop on Explainable AI Planning”, (2021).

Sreedharan, S., A. Kulkarni and S. Kambhampati, “Explainable human–ai interaction: A planning perspective”, Synthesis Lectures on Artificial Intelligence and Machine Learning **16**, 1, 1–184 (2022a).

Sreedharan, S., A. Kulkarni, D. E. Smith and S. Kambhampati, “A unifying bayesian formulation of measures of interpretability in human-ai interaction”, in “Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021”, pp. 4602–4610 (ijcai.org, 2021c).

Sreedharan, S., M. P. Madhusoodanan, S. Srivastava and S. Kambhampati, “Plan Explanation Through Search in an Abstract Model Space: Extended Results”, in “ICAPS XAIP Workshop”, (2018b).

Sreedharan, S., C. Muise and S. Kambhampati, “Why did you do that? generalizing causal link explanations to fully observable non-deterministic planning problems”, in “ICAPS 2022 Workshop on Explainable AI Planning”, (2022b).

Sreedharan, S., U. Soni, M. Verma, S. Srivastava and S. Kambhampati, “Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with black box simulators”, in “ICLR”, (2022c).

Sreedharan, S., S. Srivastava and S. Kambhampati, “Hierarchical expertise level modeling for user specific contrastive explanations”, in “Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden”, pp. 4829–4836 (ijcai.org, 2018c).

Sreedharan, S., S. Srivastava and S. Kambhampati, “Tldr: Policy summarization for factored SSP problems using temporal abstractions”, in “Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020”, pp. 272–280 (AAAI Press, 2020c).

Sreedharan, S., S. Srivastava and S. Kambhampati, “Using state abstractions to compute personalized contrastive explanations for AI agent behavior”, Artif. Intell. **301**, 103570 (2021d).

Sreedharan, S., S. Srivastava, D. E. Smith and S. Kambhampati, “Why can’t you do that hal? explaining unsolvability of planning tasks”, in “IJCAI 2019”, pp. 1422–1430 (ijcai.org, 2019b).

Srivastava, B., T. A. Nguyen, A. Gerevini, S. Kambhampati, M. B. Do and I. Serina, “Domain Independent Approaches for Finding Diverse Plans”, in “IJCAI”, (2007).

Srivastava, S., S. J. Russell and A. Pinto, “Metaphysics of planning domain descriptions”, in “Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA”, edited by D. Schuurmans and M. P. Wellman, pp. 1074–1080 (AAAI Press, 2016).

Steinmetz, M. and J. Hoffmann, “Towards clause-learning state space search: Learning to recognize dead-ends.”, in “AAAI”, pp. 760–768 (2016).

Steinmetz, M. and J. Hoffmann, “Search and learn: On dead-end detectors, the traps they set, and trap learning.”, in “IJCAI”, pp. 4398–4404 (2017).

Stern, R. and B. Juba, “Efficient, safe, and probably approximately complete learning of action models”, in “IJCAI”, (2017).

Stockmeyer, L. J., “The polynomial-time hierarchy”, Theoretical Computer Science **3**, 1, 1–22 (1976).

Stolle, M. and D. Precup, “Learning options in reinforcement learning”, in “International Symposium on abstraction, reformulation, and approximation”, pp. 212–223 (Springer, 2002).

Sukkerd, R., R. Simmons and D. Garlan, “Toward Explainable Multi-Objective Probabilistic Planning”, in “ICSE Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)”, (2018).

Sukkerd, R., R. Simmons and D. Garlan, “Tradeoff-Focused Contrastive Explanation for MDP Planning”, arXiv:2004.12960 (2020).

Sutton, R., “The bitter lesson”, Incomplete Ideas (blog) **405** (2019).

Sutton, R. S., D. Precup and S. Singh, “Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning”, Artificial intelligence **112**, 1-2, 181–211 (1999).

Swartout, W. R. and J. D. Moore, “Explanation in second generation expert systems”, in “Second generation expert systems”, pp. 543–585 (Springer, 1993).

Talamadupula, K., G. Briggs, T. Chakraborti, M. Scheutz and S. Kambhampati, “Coordination in Human-Robot Teams Using Mental Modeling and Plan Recognition”, in “IROS”, (2014).

Tellex, S., R. Knepper, A. Li, D. Rus and N. Roy, "Asking for Help Using Inverse Semantics", in "RSS", (2014).

Topin, N. and M. Veloso, "Generation of policy-level explanations for reinforcement learning.", in "AAAI", pp. 1074–1080 (2019).

Unsolvability International Planning Competition, "IPC Competition Domains", `https://unsolve-ipc.eng.unimelb.edu.au/` (2016).

Valmeekam, K., S. Sreedharan, S. Sengupta and S. Kambhampati, "Radar-x: An interactive interface pairing contrastive explanations with revised plan suggestions", in "Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2022", (AAAI Press, 2020).

Van Roy, B., "Performance loss bounds for approximate value iteration with state aggregation", Mathematics of Operations Research **31**, 2, 234–244 (2006).

Vasileiou, S. L., A. Previti and W. Yeoh, "On exploiting hitting sets for model reconciliation", in "Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021", pp. 6514–6521 (AAAI Press, 2021).

Vasileiou, S. L., W. Yeoh, T. C. Son, A. Kumar, M. Cashmore and D. Magazzeni, "A logic-based explanation generation framework for classical and hybrid planning problems", J. Artif. Intell. Res. **73**, 1473–1534 (2022).

Waa, J., J. v. Diggelen, K. Bosch and M. Neerincx, "Contrastive Explanations for Reinforcement Learning in Terms of Expected Consequences", in "IJCAI Workshop on explainable AI (XAI)", (2018).

Wayllace, C., P. Hou, W. Yeoh and T. C. Son, "Goal Recognition Design with Stochastic Agent Action Outcomes", in "IJCAI", (2016).

Weld, D. S. and G. Bansal, "The challenge of crafting intelligible intelligence", Communications of the ACM **62**, 6, 70–79 (2019).

Wikipedia contributors, "Montezuma's revenge (video game) — Wikipedia, the free encyclopedia", [Online; accessed 14-January-2020] (2019).

Winikoff, M., "Debugging agent programs with why? questions", in "Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems", pp. 251–259 (2017).

Wright, J. R. and K. Leyton-Brown, "Predicting human behavior in unrepeated, simultaneous-move games", Games and Economic Behavior **106**, 16–37 (2017).

Wu, K., Q. Yang and Y. Jiang, "Arms: An automatic knowledge engineering tool for learning action models for ai planning", The Knowledge Engineering Review **22**, 2, 135–152 (2007).

Yeh, C.-K., B. Kim, S. Arik, C.-L. Li, T. Pfister and P. Ravikumar, "On completeness-aware concept-based explanations in deep neural networks", Advances in Neural Information Processing Systems **33** (2020).

Yoon, S. W., A. Fern and R. Givan, "Ff-replan: A baseline for probabilistic planning", in "Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007", p. 352 (AAAI, 2007).

Yoon, S. W., A. Fern, R. Givan and S. Kambhampati, "Probabilistic planning via determinization in hindsight.", in "AAAI", pp. 1010–1016 (2008).

Younes, H. L. and M. L. Littman, "Ppddl1. 0: The language for the probabilistic part of ipc-4", in "Proc. International Planning Competition", (2004).

Young, N. E., "Greedy set-cover algorithms", in "Encyclopedia of algorithms", pp. 1–99 (Springer, 2008).

Zabell, S. L., "Predicting the unpredictable", Synthese **90**, 2, 205–232 (1992).

Zahavy, T., N. Ben-Zrihem and S. Mannor, "Graying the Black Box: Understanding DQNs", in "ICML", (2016).

Zahedi, Z., A. Olmo, T. Chakraborti, S. Sreedharan and S. Kambhampati, "Towards Understanding User Preferences for Explanation Types in Explanation as Model Reconciliation", in "HRI", (2019), late Breaking Report.

Zahedi, Z., M. Verma, S. Sreedharan and S. Kambhampati, "Trust-aware planning: Modeling trust evolution in longitudinal human-robot interaction", arXiv preprint arXiv:2105.01220 (2021).

Zakershahrak, M., Z. Gong, N. Sadassivam and Y. Zhang, "Online Explanation Generation for Human-Robot Teaming", ICAPS Workshop on Explainable AI Planning (XAIP) (2019).

Zhang, R., S. Guo, B. Liu, Y. Zhu, M. Hayhoe, D. Ballard and P. Stone, "Machine versus human attention in deep reinforcement learning tasks", arXiv preprint arXiv:2010.15942 (2020).

Zhang, Y., V. Narayanan, T. Chakraborti and S. Kambhampati, "A Human Factors Analysis of Proactive Assistance in Human-Robot Teaming", in "IROS", (2015).

Zhang, Y., S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo and S. Kambhampati, "Plan Explicability for Robot Task Planning", in "RSS Workshop on Planning for Human-Robot Interaction: Shared Autonomy and Collaborative Robotics", (2016).

Zhang, Y., S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo and S. Kambhampati, "Plan explicability and predictability for robot task planning", in "ICRA", pp. 1313–1320 (IEEE, 2017).

Zhi-Xuan, T., J. Mann, T. Silver, J. Tenenbaum and V. Mansinghka, “Online bayesian goal inference for boundedly rational planning agents”, in “Advances in Neural Information Processing Systems”, vol. 33, pp. 19238–19250 (Curran Associates, Inc., 2020).

Zhu, L. and R. Givan, “Landmark extraction via planning graph propagation”, ICAPS Doctoral Consortium pp. 156–160 (2003).

APPENDIX A

APPENDIX FOR CHAPTER 15

Overview

This appendix contains the following information; (1) the assumptions and theoretical results related to the representational choices made by the method, (2) the pseudo-code for the probabilistic version of the algorithms which were used in the evaluation, (3) the derivation of the formulas for confidence calculation (4) the derivation of the formulas for using a noisy classifier and finally (5) the details on the experiment that are not included in the main chapter (6) analysis of assumptions made and (7) extended discussion of various future works and possible limitations of the current method and (8) screenshots of the various interfaces.

Sufficiency of the Representational Choice

The central representational assumption we are making is that it is possible to approximate the applicability of actions and cost functions in terms of high-level concepts. Apart from the intuitive appeal of such models (many of these models have their origin in models from folk psychology), these representation schemes have been widely used to model real-world sequential decision-making problems from a variety of domains and have a clear real-world utility (Benton *et al.*, 2019).

Revisiting the definition of local approximations for a given internal model $\mathcal{M} = \langle S, A, T, \mathcal{C}\rangle$ and a set of states $\hat{S} \subseteq S$, we define a local approximation as

Definition 50. *A symbolic model $\mathcal{M}_{\mathcal{S}}^{\mathbb{C}} = \langle \mathbb{C}, A_{\mathcal{S}}^{\mathbb{C}}, \mathbb{C}(I), \mathbb{C}(\mathbb{G}), \mathcal{C}_{\mathcal{S}}^{\mathbb{C}}\rangle$. is said to be a* ***local symbolic approximation*** *for the problem $\Pi^R = \langle \mathcal{M}^R, I, \mathcal{G}\rangle$ (where $\mathcal{M}^R = \langle S, A, T, \mathcal{C}\rangle$) for regions of interest $\hat{S} \subseteq S$ if $\forall s \in \hat{S}$ and $\forall a \in A$, we have an equivalent action $a^{\mathbb{C}} \in A_{\mathcal{S}}^{\mathbb{C}}$, such that (a) $a^{\mathbb{C}}(\mathbb{C}(s)) = \mathbb{C}(T(s,a))$ (assuming $\mathbb{C}(\bot) = \bot$) and (b) $\mathcal{C}_{\mathcal{S}}^{\mathbb{C}}(\mathbb{C}(s), a) = \mathcal{C}(s,a)$ and (c) $\mathbb{C}(\mathbb{G}) = \bigcap_{s_g \in \mathbb{G} \cap \hat{S}} \mathbb{C}(s_g)$.*

We can show that

Proposition 46. *If $\hat{S}$ and A are finite, there exists a symbolic model*

$$\mathcal{M}_{\mathcal{S}}^{\mathbb{C}} = \langle \mathbb{C}, A_{\mathcal{S}}^{\mathbb{C}}, \mathbb{C}(I), \mathbb{C}(\mathbb{G}), \mathcal{C}_{\mathcal{S}}^{\mathbb{C}}\rangle$$

such that it's a local symbolic approximation.

We can show this trivially by construction. We introduce a set of concepts whose size is equal to the number of state in $\hat{S}$ (for $s_i \in \hat{S}$, we introduce a concept c_{s_i}). and we define a conditional effect and conditional cost function for every viable transition in for an action a in $\mathcal{M}^R$, and the precondition of a becomes a disjunction over the negation of concepts corresponding to the states where a fails. In Proposition 48, we will further discuss how these disjunctive precondition turns into new conjunctive preconditions.

Proposition 47. *If π_i, π_2 are two plans such that $\pi_1 \preceq \pi_2$ for Π^R, then precedence is conserved in a symbolic model $\mathcal{M}_{\mathcal{S}}^{\mathbb{C}}$ that is local approximation that covers all states appear in π_i or π_2.*

a Algorithm for Finding Missing Precondition

```
1: procedure PRECONDITION-SEARCH
2:     Input: $s_{\text{fail}}, a_{\text{fail}}$, Sampler, $\mathcal{M}^R, \mathbb{C}, \ell, \kappa$
3:     Output: Missing precondition $C_{\text{prec}}$
4:     Procedure:
5:     $\mathbb{P} \leftarrow$ Missing concepts in $s_{fail}$
6:     $P_{\mathbb{P}} \leftarrow$ Initialize priors
7:     sample_count = 0
8:     while sample_count < $\ell$ do
9:         $s \sim$ Sampler
10:        if $T(s, a_{\text{fail}}) \neq \perp$ then
11:            Update $P_{\mathbb{P}}$
12:            Eliminate any $c_i \in \mathbb{P}$ , such that $P_{\mathbb{P}}(c_i) < \kappa$
13:        if $|\mathbb{P}| = 0$ then return Signal that concept list is incomplete
14:        sample_count += 1
    return $C_i \in$poss_prec_set, with highest probability
```

b Algorithm for Finding Cost Function

```
1: procedure COST-FUNCTION-SEARCH
2:     Input: $\pi_f, \mathcal{C}_\pi$, Sampler, $\mathcal{M}^R, \mathbb{C}, \ell$
3:     Output: $\mathbb{C}_{\pi_f}$
4:     Procedure:
5:     for conc_limit in 1 to $|\mathbb{C}|$ do
6:         current_foil_cost = 0
7:         Prob_list = []
8:         conc_list = []
9:         for i in 1 to k (the length of the foil) do
10:            $\hat{\mathbb{C}}_i$, min_cost, $P_{(\hat{\mathbb{C}}_i, a_i) \geq min\_cost}$ = find_min_conc_set($\mathbb{C}(T(s, \langle a_1, ...a_{i-1} \rangle)), a_i$, conc_limit, $\ell$)
11:            current_foil_cost += min_cost
12:            Prob_list.push($P_{(\hat{\mathbb{C}}_i, a_i) \geq min\_cost}$)
13:            conc_list.push($\hat{\mathbb{C}}_i$, min_cost)
14:        if current_foil_cost > $\mathcal{C}_\pi$ then return conc_list, Prob_list
    return Signal that the concept list is incomplete
```

Figure A.1: The Extension of the Two Algorithms to Use the Confidence Values, Subfigure (A) Presents the Algorithm for Missing Precondition and (B) the One for Cost Function

This trivially follows from the fact that per definition of local approximation both the invalidity of plans and cost of plans are conserved. Which means any preference over plans that can be established in the complete model can be established in the approximate model.

Apart from this basic assumption, we make one additional representational assumption, namely, that the precondition can be expressed as a conjunction of positive concepts. Which brings us to the next proposition

Proposition 48. *A precondition of an action $prec_{a_i}$, which is represented as an arbitrary logical formula over a set of propositions P can be mapped to a conjunction over positive literals from a different set P' (which can be generated from P).*

To see why this holds, consider a case where the precondition of action a is expressed as an arbitrary propositional formula, $\phi(\mathbb{C})$. In this case, we can express it in its conjunctive normal form $\phi'(\mathbb{C})$. Now each clause in $\phi'(\mathbb{C})$ can be treated as a new compound positive concept. Thus we can cover such arbitrary propositional formulas by expanding our concept list with compound concepts (including negations and disjuncts) whose value is determined from the classifiers for the corresponding atomic concepts. Note that the set of all possible compound concepts could be precomputed and add no additional overhead on the human's end. Also note, this is completely compatible with relational concepts as relational concepts defined over a finite set of objects can be compiled into a set of finite propositional concepts. Proposition 48 also extends to cost functions, which we assume to be defined over conjunction of concepts.

Now the objective of the methods become not only identify an explanation but one that has the likelihood of being true. This means for precondition the objective becomes. Given the failing state s_{fail} and action a_{fail} and a set of states $\mathbb{S}$ where a_{fail} is executable, explanation identification for a failing precondition involves finding a concept $c_i \in \mathbb{C} \setminus \mathbb{C}(s_{\text{fail}})$, such that $c_i = \arg\max_c P(c \in p_{a_{\text{fail}}}|\mathbb{S})$.

In terms of the algorithm for finding precondition, the main difference is the fact that instead of eliminating the hypothesis directly from their presence or absence in the given state, we will instead use the observation to update the posterior over the hypothesis being true. We can also further improve the efficiency of search by removing possible hypothesis for final precondition, when its probability dips below a low threshold κ. Note that when $\kappa = 0$ and the observation model is perfect ($P(O^s_{c_i}|c_i \in s) = 1$ and $P(O^s_{c_i}|c_i \not\in s) = 0$), then the reasoning reflects the elimination based model learning method used by many of the previous approaches like those presented by Carbonell and Gil (1990); Stern and Juba (2017).

Now for the cost function, the objective becomes

Definition 51. *Given a foil $\pi^f = \langle a_1, ..., a_f\rangle$ with $m \leq f$ unique actions (where $A(a_i)$ returns the unique action label), a set sets of states $\{\mathbb{S}_1, ..., \mathbb{S}_m\}$ where each $\mathbb{S}_{A(a_i)}$ corresponds to a set of states where an action $A(a_i)$ is executable, the explanation for suboptimality correpond to finding an abstract cost function, that tries to minimize for the cost and maximize the probabilities.*

$$min_{\hat{\mathbb{C}}_1,...,\hat{\mathbb{C}}_f}\Big(\sum_{i=1..f} \|\hat{\mathbb{C}}_i\|, -1 \times P(\mathcal{C}(\hat{\mathbb{C}}_1, a_1) \geq k_1|\mathbb{S}_{A(a_1)}), ..., -1 \times P(\mathcal{C}(\hat{\mathbb{C}}_f, a_f) \geq k_f|\mathbb{S}_{A(a_k)})$$

$$\textit{subject to } \mathcal{C}^{abs}_s(\mathbb{C}_{\pi_f}, \pi_f) > \mathbb{C}(I, \pi)$$

Rather than solve this full multi-objective optimization problem, we will use the cost as the primary optimization criteria and use probabilities as a secondary one. Figure A.1(b), present a modified version of the greedy algorithm to find such cost abstractions. The procedure find_min_conc_set, takes the current concept representation of state i in the foil and searches for the subset $\hat{\mathbb{C}}_i$ (and its probabilistic confidence) of the state with the maximum value for $\mathcal{C}^{abs}_{\mathcal{S}}(\hat{\mathbb{C}}_i, a_i)$, where the value is again approximated through sampling (with budget ℓ), and the subset size is upperbounded by conc_limit. If there are multiple abstract cost function with the same max cost it selects the one with the highest probability. Similar to the algorithm described in the chapter, here we incrementally increase the max concept size till we find an abstract cost function that meets the requirement.

Confidence Calculation

For confidence calculation, we will be relying on the relationship between the random variables as captured by Figure A.2 (A) for precondition identification and Figure A.2 (B) for cost calculation. Where the various random variables captures the following facts: O^s_a - indicates that action a can be executed in state s, $c_i \in p_a$ - concept c_i is a precondition of a, $O^s_{c_i}$ - the concept c_i is present in state s, $\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k$ - the abstract cost function is guaranteed to be higher than or equal

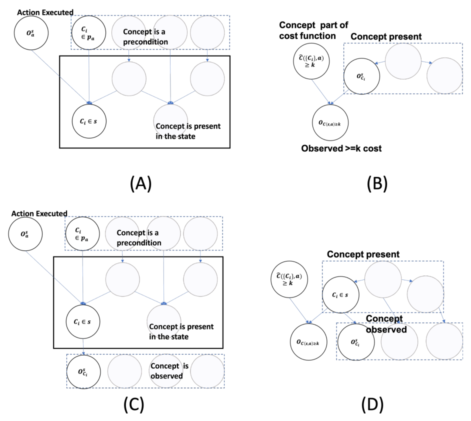


Figure A.2: A Simplified Probabilistic Graphical Models for Explanation Inference, Subfigure (A) and (B) Assumes Classifiers to Be Completely Correct, While (C) and (D) Presents Cases Where the Classifier May Be Noisy.

to k and finally $O_{\mathcal{C}(s,a)>k}$ - stands for the fact that the action execution in the state resulted in cost higher than or equal to k. Since we don't know the exact relationship between the current concept in question is, for notational convenience we will use the symbol $O_{\mathbb{C}(s)\setminus c_i}$ to stand for the other concepts observed in the given state.

We will allow for inference over these models, by relying on the following simplifying assumptions - (1) the distribution of all non-precondition concepts in states where the action is executable is the same as their overall distribution across the problem states (which can be empirically estimated), (2) cost distribution of an action over states corresponding to a concept that does not affect the cost function is identical to the overall distribution of cost for the action (which can again be empirically estimated). The first assumption implies that the likelihood of seeing a non-precondition concept in a sampled state is equal to the likelihood of it appearing in any sampled state. In the most general case this distribution can be given as $P(c_i|O_{\mathbb{C}(s)\setminus c_i})$, i.e. the likelihood of seeing this concept given the other concepts in the state. This distribution can be empirically estimated per user (or shared vocabulary) independent of the specific explanatory query. While the second one implies that for a concept that has no bearing on the cost function for an action, the likelihood that executing the action in a state where the concept is present will result in a cost greater than k will be the same as that of the action execution resulting in a cost greater than k for a randomly sampled state ($p_{\mathcal{C}(.,a)\geq k}$). This assumption can be further relaxed by considering the distribution of the action cost under the observed set of concepts (though this would clearly require more samples to learn).

For a single sample, the posterior probability of explanations for each case can be expressed as follows: For precondition estimation, updated posterior probability for a positive observation can be computed as $P(c_i \in p_a|O_{c_i}^s \wedge O_a^s \wedge O_{\mathbb{C}(s)\setminus c_i}) = (1 - P(c_i \notin p_a|O_{c_i}^s \wedge O_a^s \wedge O_{\mathbb{C}(s)\setminus c_i}))$, where

$$P(c_i \notin p_a | O^s_{c_i} \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$$
$$= \frac{P(O^s_{c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a | O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})}{P(O^s_{c_i} | O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})}$$

Given $c_i \notin p_a$ is independent of O^s_a and $O_{\mathbb{C}(s)\setminus c_i}$ also expanding the denominator we get

$$= \frac{P(O^s_{c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a)}{P(O^s_{c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a) + P(O^s_{c_i} | c_i \in p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \in p_a)}$$

From our assumption, we know $P(O^s_{c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$ is same as the distribution c_i over the problem states $(p(c_i | O_{\mathbb{C}(s)\setminus c_i}))$ and $P(O^s_{c_i} | c_i \in p_a \wedge O^s_a)$ must be one.

$$= \frac{p(c_i | O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a)}{p(c_i | O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a) + P(c_i \in p_a)}$$

For cost calculation, we can ignore $O_{\mathbb{C}(s)\setminus c_i}$, since according to the graphical model once the concept c_i is observed, $O_{\mathcal{C}(s,a)\geq k}$ is independent of the other concepts.

$$P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k | O^s_{c_i} \wedge O_{\mathcal{C}(s,a)\geq k}) = \frac{P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i} \wedge \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) * P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k | O^s_{c_i})}{P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i})}$$

Where $P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i}, \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)$ should be 1 and $\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k$ independent of $O^s_{c_i}$. Which gives

$$= \frac{P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)}{P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i})}$$

$$= \frac{P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)}{P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i}, \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)) * P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)) + P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i} \wedge \neg\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)) \times P(\neg\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k))}$$

From our assumptions, we have $P(O_{\mathcal{C}(s,a)\geq k} | O^s_{c_i} \wedge \neg\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)) = p_{\mathcal{C}(.,a)\geq k}$

$$= \frac{P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)}{P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)) + p_{\mathcal{C}(.,a)\geq k} * P(\neg\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k))}$$

Using Noisy Concept Classifiers

Note that in previous sections, we made no distinction between the concept being part of the state and actually observing the concept. Now we will differentiate between the classifier saying that a concept is present $(O^s_{c_i})$ is a state from the fact that the concept is part of the state $(c_i \in \mathbb{C}(S))$. Here we note that the learned relationship is over the actual concepts in the state rather than the observation (and thus we would need to learn it from states with true concept labels). The relationship between the random variables can be found in Figure A.2 (C) and (D). We will assume that the probability of the classifier returning the concept being present is given by the probabilistic confidence provided by the classifier. Of course, this still assumes the classifier's model of its prediction is accurate. However, since it is the only measure we have access to, we will treat it as being correct. Now we can use this updated model for calculating the confidence. For the precondition estimation, we can update the posterior of a concept being a precondition given a negative observation ($O^s_{\neg c_i}$) using the formula

$$P(c_i \notin p_a | O^s_{\neg c_i} \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) = \frac{P(O^s_{\neg c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) * P(c_i \notin p_a | O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})}{P(O^s_{\neg c_i} | O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})}$$

Where $P(c_i \notin p_a | O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) = P(c_i \notin p_a)$ and we can expand $P(O^s_{\neg C_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$ as follows

$$\begin{aligned}
&P(O_{\neg c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) = \\
&P(O_{\neg c_i} | c_i \in \mathbb{C}(s)) * P(C_i \in \mathbb{C}(s) | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) + \\
&P(O_{\neg c_i} | c_i \notin \mathbb{C}(s)) * P(c_i \notin \mathbb{C}(s) | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})
\end{aligned}$$

Where as defined earlier $P(c_i \notin \mathbb{C}(s) | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$ and $P(c_i \in \mathbb{C}(s) | C_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})$ would get expanded to the learned relationship between the concepts and their corresponding observation model. The denominator also needs to be marginalized over $c_i \notin \mathbb{C}(s)$.

Similarly for posterior calculation for positive observations, we have

$$\begin{aligned}
&P(O^s_{c_i} | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) = \\
&P(O_{c_i} | c_i \in \mathbb{C}(s)) * P(c_i \in \mathbb{C}(s) | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i}) + \\
&P(O_{c_i} | c_i \notin \mathbb{C}(s)) * P(c_i \notin \mathbb{C}(s) | c_i \notin p_a \wedge O^s_a \wedge O_{\mathbb{C}(s)\setminus c_i})
\end{aligned}$$

Now for the cost, we can similarly incorporate the observation model as follows.

$$P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k | O^s_{c_i} \wedge O_{\mathcal{C}(s,a) >= k}) = \frac{\begin{array}{r} P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i} | \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) \\ * P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) \end{array}}{P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i})}$$

$$= \frac{\begin{array}{l} (P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i} | c_i \in \mathbb{C}(s), \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) * P(c_i \in \mathbb{C}(s)) + \\ \quad P(O_{\mathcal{C}(s,a) > k}, O^s_{c_i} | c_i \notin \mathbb{C}(s), \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) * P(c_i \notin \mathbb{C}(s))) * P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) \end{array}}{P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i})}$$

Given their parents, $O_{\mathcal{C}(s,a)\geq k}$ and $O^s_{c_i}$ are conditionally independent, and given its parent $O^s_{c_i}$ is independent of $\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k)$, there by giving

$$= \frac{\begin{array}{l} (P(O_{\mathcal{C}(s,a)\geq k} | c_i \in \mathbb{C}(s), \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) * P(O^s_{c_i} | c_i \in \mathbb{C}(s)) * P(c_i \in \mathbb{C}(s)) + \\ \quad P(O_{\mathcal{C}(s,a)\geq k} | c_i \notin \mathbb{C}(s), \mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) * P(O^s_{c_i} | c_i \notin \mathbb{C}(s)) * P(c_i \notin \mathbb{C}(s))) * P(\mathcal{C}^{abs}_s(\{c_i\}, a) \geq k) \end{array}}{P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i})}$$

Now $P(O_{\mathcal{C}(s,a)\geq k} | c_i \in \mathbb{C}(s), \mathcal{C}_s^{abs}(\{c_i\}, a) \geq k) = 1$ and $P(O_{\mathcal{C}(s,a)\geq k} | c_i \notin \mathbb{C}(s), \mathcal{C}_s^{abs}(\{c_i\}, a) \geq k)$ can either be empirically estimated from true labels or we can make the assumption that is equal to $p_{\mathcal{C}(.,a)\geq k}$ (which we made use of in our experiments), which would take us to

$$= \frac{\begin{array}{l}(P(O^s_{c_i} | c_i \in \mathbb{C}(s))P(c_i \in \mathbb{C}(s)) + \\ p_{\mathcal{C}(.,a)\geq k} * P(O^s_{c_i} | c_i \notin \mathbb{C}(s)) * P(c_i \notin \mathbb{C}(s))) * P(\mathcal{C}_s^{abs}(\{c_i\}, a) \geq k)\end{array}}{P(O_{\mathcal{C}(s,a)\geq k}, O^s_{c_i})}$$

Experiment Domains

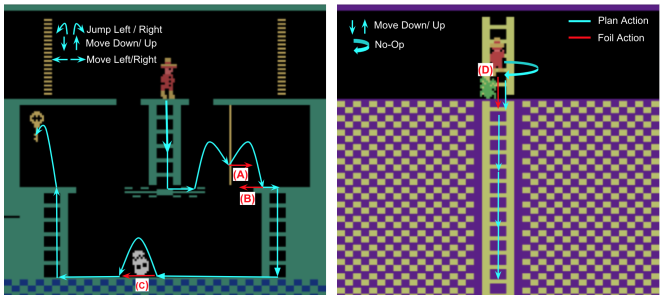
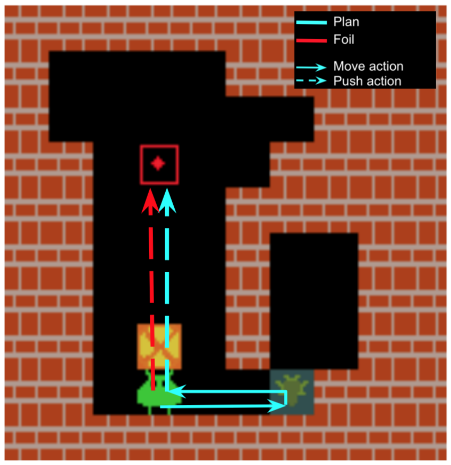


Figure A.3: Montezuma Foils: Left Image Shows Foils for Level 1, (A) Move Right Instead of Jump Right (B) Go Left over the Edge Instead of Using Ladder (C) Go Left Instead of Jumping over the Skull. Right Image Shows Foil for Level 4, (D) Move down Instead of Waiting.

For validating the soundness of the methods discussed before, we tested the approach on the open-AI gym implementation of Montezuma's Revenge (Brockman *et al.*, 2016) and variants of Sokoban (Schrader, 2018). Most of the search experiments were run on an Ubuntu 14.0.4 machine with 64 GB RAM.

For Montezuma, we used the deterministic version of the game with the RAM-based state representation (the game state is represented by the RAM value of the game controller, represented by 256-byte array). We considered executing an action in the simulator that leads to the agent's death (falling down a ledge, running into an enemy) or a non-NOOP (NOOP action is a specific agent action that is designed to leave agent's state unchanged) action that doesn't alter the agent position (trying to move left on a ladder) as action failures. We selected four possible foils for the game (illustrated in Figure A.3), three from level 1 and one from level 4. The base plan in level 1 involves the agent reaching the key, while level 4 required the agent to cross the level.

For Sokoban, we considered two variants with a single box and a single target. Both variants allow for 8 actions and a NOOP action. Four of those actions are related to the agent's movements in the four directions and four to pushing in specific directions. We restricted the move actions only to be able to move the agent if the cell is empty, i.e., it won't move if there is a box or a wall in that direction. The push action also moves the agent in the direction of push if there is a box in that direction,



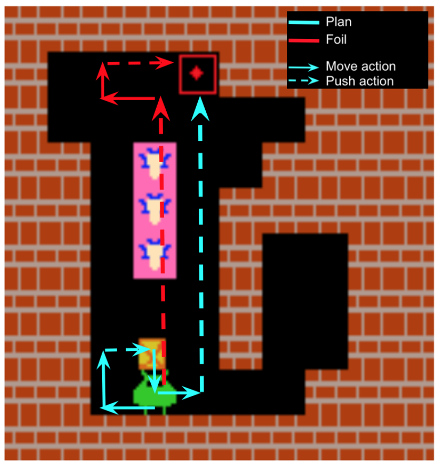


Figure A.4: Sokoban Foils: Left Image Shows Foils for Sokoban-switch, Note That the Green Cell Will Turn Pink Once the Agent Passes It. Right Image Shows Foil for Sokoban-cell.

and the agent will occupy the cell previously occupied by the box (provided there are no walls to prevent the box from moving). Similar to Montezuma, we consider any action that doesn't change the agent position to be a failure. The first version of Sokoban included a switch the player could turn on to push the box (we will refer to this version as Sokoban-switch), and the second version (Sokoban-cells) included particular cells from which it is costlier to push the box. We considered two versions of Sokoban-switch, one in which turning on the switch only affected the cost of pushing the box and another one in which it was a precondition. For the cost case of Sokoban-switch, while the switch is on (i.e. the cell is pink), all actions have unit cost, while when the switch is off, the cost of pushing actions is 10. The cell can be switched on by visiting it, and any future visit will cause it to turn off. For the precondition-version, pushing of the box while the switch is off causes the episode to end with high cost (100). Since we also trained an RL agent for this version for generating the saliency map. We also added a small penalty for not pushing the switch and a penalty proportional to the distance between the box and the target. In Sokoban-cells, the cost of all actions except pushing boxes in the pink region is one, while that of pushing boxes in the pink region is 10. We selected one foil per variation, and the original plan and the foil are shown in Figure A.4.

Concept Learning For Montezuma, we came up with ten base concepts for each level that approximates the problem dynamics at the level, including the foil failure. We additionally created ten more concepts by considering the negations of them. All state samples (used to generate the samples for the classifier and the algorithm) were created by randomly selecting one of the states from the original plan and then performing random walks from the selected state. For the classifiers, we used game-specific logic and RAM byte values to identify each positive instance and then randomly selected a set of negative examples. We used around 600 positive examples (except for the concepts *skull_on_right* and *skull_on_left* in level 1, which had 563 and 546 examples, respectively) and twice as many negative examples for each concept. These RAM state examples were fed to a binary AdaBoost Classifier (Freund *et al.*, 1999) (Scikit-learn implementation (Pedregosa *et al.*, 2011) version 0.22.1, with default parameters), with 70% of samples used as train set and the rest as the test set, for each concept. Finally, we obtained a test accuracy range of 98.57% to 100%, with an average of 99.72% overall concepts of both the levels. All the samples used for the classifier were collected from 5000 sampling episodes for level 1 and 4000 sampling episodes for level 4. During the search, We used a threshold of 0.55 on classifiers for concepts of level 1, such that a given state has a given concept when the classifier probability is greater than 0.55, to reduce false positives. The code for sampling and training the classifiers can be found in the directory PRECOND_BLACKBOX/sampler_and_conceptTrain inside the code directory.

Concept Collection

For the Sokoban variants, we wanted to collect at least the list of concepts from people. We used surveys to collect the set of concepts. The survey allowed participants to interact with the game through a web interface, and at the end, they were asked to specify game concepts that they thought were relevant for particular actions. Each user was asked to specify a set of concepts that they thought were relevant for four actions in the game. They were introduced to the idea of concepts and their effect on the action by using PACMAN as an example and presenting three example concepts. For Sokoban-switch, we collected data from six participants, four of whom were asked to specify concepts for push actions and two people for move actions. For Sokoban-cell, we collected data from seven participants, six of whom were asked to specify concepts for push actions, and one was asked to specify concepts for move action. We went through the submitted concepts and clustered them into unique concepts using their description. We skipped ones where they just listed strategies rather than concepts describing the state. We removed two concepts from the Sokoban-cell list and two from Sokoban-switch because we couldn't make sense of the concept being described there. For Sokoban-switch, we received 25 unique concepts, and for Sokoban-cell, we collected 38 unique concepts. We wrote scripts for each of the concepts and used it to sample example states. We ran the sampler for 1000 episodes to collect the examples for the concepts. We trained classifiers for each of the concepts that generated more than 10 positive examples for the concepts. For sokoban-switch, we removed two additional concepts because their training set didn't contain any positive examples. We had, on average, 178.46 positive examples for Sokoban-cell per concept and 215.55 for Sokoban-switch. We used all the other samples as negative examples. We again used 70% of samples for training and the remaining for testing. We used Convolutional Neural Networks (CNNs) based classifiers for the Sokoban variants. The CNN architecture involved four convolutional layers, followed by three fully connected layers that give a binary classification output. The average accuracy of the Sokoban-switch was 99.46%, and Sokoban-cell was 99.34%. The code used for sampling and training for each domain can be found under the folder COST_TRAINER (inside the directory BLACKBOX_CODE). The classifier network is specified in the file CNNnetwork.py. The details on how to run them are provided in the README file in the root code directory.

Explanation Identification

For Montezuma, the concept distribution was generated using 4000 episodes, and the probability distribution of concepts ranged from 0.0005 to 0.206. For some of the less accurate models, we did observe false negatives resulting in the elimination of the accurate preconditions and empty possible precondition set. So we made use of the probabilistic version of the search with observation probabilities calculated from the test set. We applied a concept cutoff probability of 0.01, and in all cases, the precondition set reduced to one element (which was the expected precondition) in under the 500 step sampling budget (with the mean probability of 0.5044 for foils A, B & C. Foil D, in level 4, gave a confidence value of 0.8604). The ones in level 1 had lower probabilities since they were based on more common concepts, and thus, their presence in the executable states was not strong evidence for them being a precondition.

For each Sokoban variant, we ran another 1000 episode sampler, which used random restarts from the foil states to collect samples that we used for the explanation generation. During the generation of the samples, we used the previously learned classifiers to precompute the mapping from concepts to states.

We used a variant of Figure A.1(b), where we sped up the search by allowing for memoization. Specifically, when sampling is done for an action and a specific conc_limit for the first time, then we precompute the min-cost for all possible concept subset of that size. Then for every step that uses that action, we look up the min value for the subset that appears in the state. The search was run with a sampling budget of 750. For calculating the confidence, all required distributions were calculated only on the states where the action was executable. Again the search was able to find the expected explanation. We had average confidence of 0.9996 for the Sokoban-switch and 0.998 for the Sokoban-cell. The exact observation models values used can be found in constant.py under the directory COST_BLACKBOX/src, and the file cost_explainer.py in the same directory contains the code for the exact search we used.

In terms of time taken to generate the sample, creating 100 samples from montezuma for the first screen (averaged across the foils) took 59.754 seconds, montezuma second screen it took 97.673 seconds (the additional time is due to the overhead of the agent being moved to the new level before the start of the episode) and the Sokoban variants took 40.660 secs.

User Study

With the basic explanation generation method in place, we were interested in evaluating if users would find such an explanation helpful. Specifically, the hypotheses we tested were

H1*: People would prefer explanations that establish the corresponding model component over ones that directly presents the foil information (i.e. the failing action and per-step cost)*

H2*: Concept-based precondition explanations help users understand the task better than saliency map based ones.*

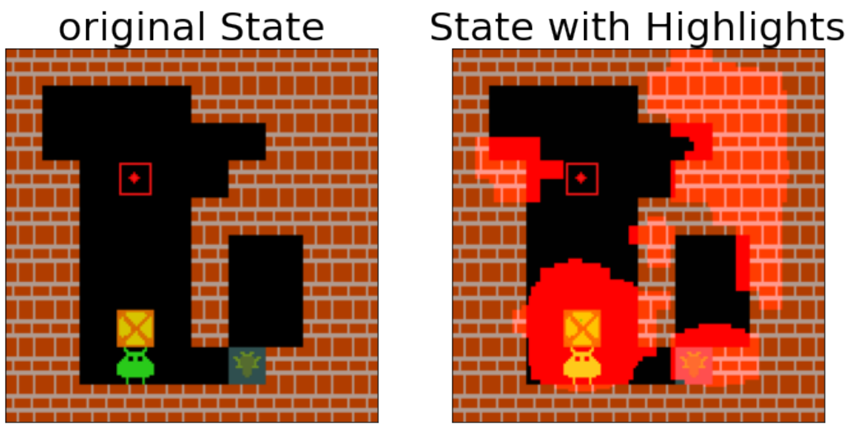


Figure A.5: Saliency Map Based Explanation Shown to Users as Part of H2

To evaluate this, we performed a user study with all the foils used along with the generated explanation and a simple baseline. In the study, each participant was presented with a random subset of the concept we used for the study (around five) and then was shown the plan and a possible foil. Then the participant was shown two possible explanations for the foil (the generated one and the baseline) and asked to choose between them. There were additional questions at the end asking them to specify what they believed was the completeness of the selected explanation, on a Likert scale from 1 to 5 (1 being least complete and 5 being the most). They were also provided a free text field to provide any additional information they felt would be useful.

For precondition explanation, the options showed to the user includes one that showed the state at which the foil failed along with the information that the action cannot be executed in that state and the other one reported that the action failed because a specific concept was missing (the order of the options was randomized). In total, we collected data from 20 participants, where 7 were women, the average age was 25, and 10 people had taken an AI class. We found that 19 out of the 20 participants selected precondition based explanation as a choice. On the question of whether the explanation was complete, we had an average score of 3.476 out of 5 on the Likert scale (1 being not at all complete and 5 being fully complete). The results seem to suggest that information about missing precondition are useful explanations though these may not be complete. While not a lot of participants provided information on what other information would have been useful, the few examples we had generally pointed to providing more information about the model (for example, information like what actions would have been possible in the failed state).

For the cost function explanation, the baseline involved pointed out the exact cost of executing each action in the foil, and concept-explanation showed how certain concepts affected the action costs. In the second case, all action costs were expressed using the abstract cost function semantics in that they were expressed as the 'action costing at least X' even though in our case, the cost was the same as the abstract cost. For the cost condition, again, we collected 20 replies in total (ten per foil) and found 14 out of 20 participants selected the concept-based explanation over the simple one.

The concept explanations had, on average, a completeness score of 3.214 out of 5. The average age of the participants was 24.15, 10 had AI knowledge out of 20 people, 11 were masters students (rest were undergrad), and we had 14 males, 5 females, and one participant did not specify.

For H2, as mentioned the baseline was a saliency map based explanation. For generating the saliency map, we trained the RL agent using DQN with prioritized experience replay (Schaul *et al.*, 2015)[1]. The agent was trained for 420k epochs. The Saliency map itself was generated for four states, with the agent placed on the four sides of the box. The saliency map itself was generated using (Greydanus *et al.*, 2018), where we used only the procedure for generating the map for the critic network[2]. Figure A.5 shows the saliency map generated for one of the images. This was shown when the user tried push up action and fails. At the beginning of the study, both groups of the users were made to familiarize themselves with five concepts that were randomly ordered (the concepts themselves remained the same) and had to take a quiz matching new states to those concepts, before moving on to play the game. Out of the 60 responses we considered, 16 identified as female and 41 identified as men. 23 of the participants reported they had some previous knowledge of AI, but only three participant reported having any planning knowledge. *The participants who got concept based explanations took 43.78 ± 12.59 secs (95% percentile confidence) and 35.87 ± 9.69 steps on average to complete the game. On the otherhand participants of the other group took 134.24 ± 61.72 secs and 52.64 ± 11.11 steps on average.*

Below, we have included some screenshots of the interface

Analysis of Assumptions Made for Confidence Calculations

In this section we present results from additional tests we ran to verify some of the assumptions made in the confidence calculations on the Sokoban variants. We mainly focused on Sokoban variants since the concepts were collected directly from users and we tested the assumptions - (1) the distribution of all non-precondition concepts in states where the action is executable is the same as their overall distribution across the problem states (which can be empirically estimated) and (2) cost distribution of an action over states corresponding to a concept that does not affect the cost function is identical to the overall distribution of cost for the action (which can again be empirically estimated). We didn't run a separate test on the independence of concepts as we saw that many of the concepts listed by the users were in fact correlated and were denoting similar or even the same phenomena. All concepts were assumed to be distributed according to a Bernoulli distribution, whose MLE estimates were calculated by running the sampler for ten thousand episode, where we used states from the original successful/optimal plan as the initial state for the random walk (ensuring the distributions are generated from state space local to the plan of interest). For first assumption, we compared the distribution of concept for states where the action was executed against the distribution of the concept over all the sampled states. For the second assumption, we compared the distribution of the

[1]For exact agent, we followed the approach described in `https://github.com/higgsfield/RL-Adventure/blob/master/4.prioritized%20dqn.ipynb`

[2]We made use of the code available at `https://github.com/greydanus/visualize_atari` which had a GPL licence

Action	Concept with Max difference	Max Absolute Difference in Estimates	Average Difference in Estimates
push up	empty_above	0.018	0.004
push down	empty_below	0.0154	0.0033
push left	empty_below	0.0163	0.0037
push right	empty_below	0.0172	0.0046
move up	empty_left	0.002	0.0009
move down	wall_left	0.0017	0.0007
move left	empty_right	0.0032	0.0009
move right	empty_right	0.0045	0.0008

Figure A.6: Results from Sokoban-switch on the Distribution of Non-precondition Concepts for Each Action. For Each Action the Table Reports the Concept with the Maximum Difference Between the Distribution of Concept for That Specific Version, Versus the Overall Distribution and the Average Difference in Estimates Across Concepts.

Domain	Action	Concept with Max difference	Max Absolute Difference in Estimates	Average Difference in Estimates
Sokoban-Switch	push up	wall_left_below_of_box	0.1132	0.0461
	push down	wall_left_below_of_box	0.1107	0.0473
	push left	above_switch	0.1135	0.0461
	push right	wall_left_below_of_box	0.111	0.0479
Sokoban-Cell	push up	box_on_right	0.0956	0.0411
	push down	box_on_right	0.1098	0.0476
	push left	box_on_right	0.1012	0.0486
	push right	wall_on_left	0.0889	0.0433

Table A.1: Results from Sokoban-switch and Sokoban-cell on the Distribution of Action Cost Across Different Concepts. Here We Report Only the Cost for Push Actions, since Only Those Actions Result in Higher Cost.

states with the high cost (>=10) where the concept is present versus the distribution of high cost for the action.

Table A.6, summarizes the results from testing the first assumption for Sokoban-switch. For each action the table reports the non-precondition concept which had the maximum difference in estimates (the reported difference in the table). In this domain, the only precondition concept is switch_on for the push actions. As we can see for the domain, the differences are pretty small and we expect the differences to further reduce once we start accounting the correlation between concepts.

Table A.1 presents the results for the second assumption. In the cases of Sokoban-switch, we again skipped the switch_on concept and for Sokoban-Cell we skipped the concepts related to the pink cells since they are all highly correlated to central concept controlling the cost function (on_pink_cell).

Extended Discussion

Collecting Concepts In most of the current text, we expect the set of concepts and classifiers to be given. As such before the system is actually used in practice we would have a stage where the initial set of concepts are collected. Collecting all the concepts from the same user may be taxing, even if we made use of more straightforward methods to learn the classifiers. Instead, a more scalable method may be to set up domain-specific databases for each task that includes the set of commonly used concepts for the task. This was also the strategy used by Cai *et al.* (2019), who created a medical concept database to be used along with TCAV explanations. One could also use off-the-shelf object recognition and scene graph generation methods to identify concepts for everyday settings.

Describing Plans: While contrastive explanations are answers to questions of the form "Why P and not Q?", we have mostly focused on refuting the foil (the "not Q?" part). For the "Why P" part the agent could start by demonstrating its plan to show why it is valid and report the total cost it may accumulate. We can further augment such traces with the various concepts that are valid at each step of the trace. We can also report a cost abstraction for the cost function corresponding to each step taken as part of the plan. Here the process for finding the cost abstraction stays the same as what is described in Section 15.4.

Stochastic Domains: While most of the discussions in the chapter have been focused on deterministic domains, *these ideas also carry over to stochastic domains.* The way we are identifying preconditions and estimating cost functions remain the same in stochastic domains. The only difference would be the estimation of the value or the failure point of the foil. One way to easily adapt them to our method would be to compare against the worst case, or best case execution cost of the foil or the failure point under one of the execution traces corresponding to the foil. So the only change we would need to make to the entire method is before the actual search starts. Namely, when the agent is trying to simulate the foil in its own model, rather than doing it once, it would need to simulate it multiple times and look for the worst-case or best-case trace. Another possibility is rather than looking for the worst-case execution trace, would be to consider the most likely traces and perform the same process but now over a set of possible foils.

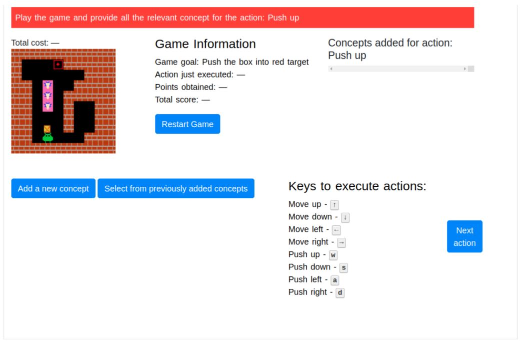
Partial Observability and Non-Markovian Concepts: The chapter focuses on cases where the mapping is from each state to concept. Though there may be cases where the concept the user is interested in corresponds to a specific trajectory. Or there may be cases where there may be some disparity between the internal state of the agent and what the human can observe, or the problem itself is partially observable. In such cases, rather than learning a concept that is completely identified by the current state, the system may need to learn mappings from state action trajectories (or observation histories) to concepts. Then the rest of the process stays the same (except the search process now needs to keep track of history). We can still use the same class of symbolic models as before because while these concepts are non-markovian with regards to the underlying model, the fact that the concept symbols are part of the symbolic model state means any action transition or cost function that depends on these concepts can be defined purely with respect to the current state.

Temporally Extended Actions: Another assumption we have made through the chapter is that the action space stays the same across the symbolic and internal models. However, this may not always be true. For domains like robotics, the human may be reasoning about the actions at a higher level of abstractions than what the agent may be considering. Such actions may be modeled as temporal abstractions (for example, options (Sutton *et al.*, 1999)) defined over the agent's atomic actions. If such abstract actions are not pre-specified to the agent, it could try to learn it by a process similar to how it learned concepts. Once learned, the agent can either directly simulate these temporally abstract actions to identify its preconditions (for options, a characterization of its initiation set) and cost function (the expected cost over the holding time) or from the sampled traces which correspond to the execution of these abstract actions (if the system could only learn a mapping from trajectories to action labels).

Acquiring New Concepts: In the main text, we discussed how the original vocabulary set may be incomplete and how our search algorithms could identify scenarios when the concept list is incomplete and insufficient to generate the required explanation. In such cases, the system would need to gather more concepts from the user. While the system could ask the user for any previously unspecified concepts, this would just result in the user wasting time specifying potentially irrelevant concepts. One would preferably want to perform a more directed concept acquisition. One possibility might be to use the system's internal model to provide hints to the user as to what concepts may be useful. For example, the system could generate low-level explanations like saliency maps to highlight the parts of the state that may be important and ask for concepts that are relevant to those states. Once the new concepts are collected, we can again make use of our current approach to see if any of the concepts could be used to build a satisfactory explanation. One could also wrap the entire interaction into a meta-sequential decision-making framework, where the framework captures the repeated interaction between the user and the system. The actions that are available to the meta decision-maker would include the ability to query the user for more concepts and the objective would be to reduce the overall burden placed on the user. While one could follow the basic interaction pattern laid out in this chapter, namely, querying the user only when the current set of concepts proves to be insufficient, the use of such a reasoning framework would allow the system to pre-emptively collect concepts from the user that may be useful for future interactions.

Confidence Threshold Our system currently expects to be provided a confidence threshold that decides the minimum confidence that must be satisfied by any explanation provided to the user. Such a threshold may be provided by the stakeholders of the domain/system or the user. One could also determine when to provide explanations based on decision-theoretic principles. Since the confidence values are just probabilities, if the system has access to penalty values attached to getting an explanation wrong, then it can associate an expected value to an explanation (since after all the confidence is just the probability of the fact being true) and use it drive its decisions. This penalty value could be associated with both the user potentially making a mistake because of incorrect information but also could be related to a loss of trust from the system providing an incorrect explanation. One could also use our explanatory methods along with the explanatory confidence in the context of trust-aware decision-making systems like that presented by Zahedi *et al.* (2021).

Ethical Implications The use of confidence values makes sure that the system is not giving explanations to humans that it doesn't have high confidence in, thereby



Play the game and provide all the relevant concept for the action: Push up

Total cost: —

Game Information

Game goal: Push the box into red target
Action just executed: —
Points obtained: —
Total score: —

Concepts added for action:
Push up

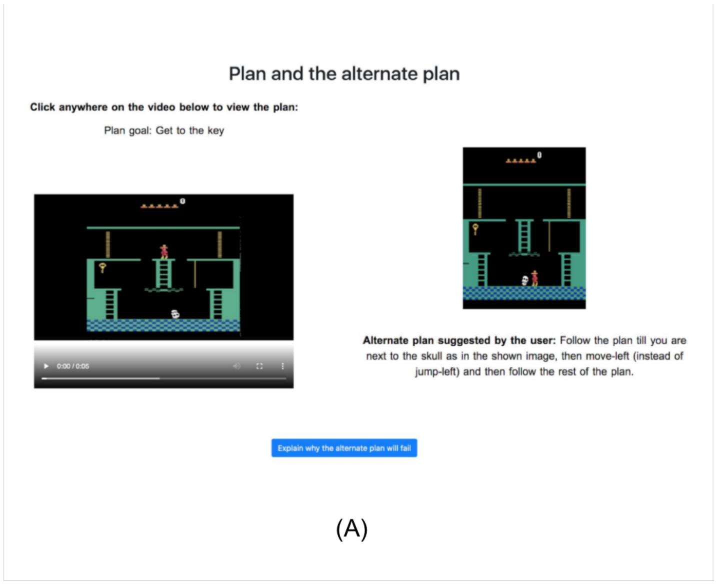
Keys to execute actions:

Move up - ↑
Move down - ↓
Move left - ←
Move right - →
Push up - w
Push down - s
Push left - a
Push right - d

Figure A.7: Screenshot from the Survey Done to Collect Sokoban Concepts.

sidestepping many of the core issues related to post hoc explanations. One of the assumptions we have made in the chapter that allows us to do this is that the agent can correctly reason over its internal model. So if we can learn an equivalent representation for this internal model, then one could guarantee that the explanations are sound, in so far that the agent would only generate behaviors that align with the stated model components. However, this assumption may not always hold and the agent could generate behavior that may not conform to its internal models. Unfortunately, in such cases, we would need additional mechanisms to test whether the agent conforms to the identified model component. Otherwise, the explanation could lead to the human assigning undeserved trust to the agent. One scenario where the soundness of the agent's reasoning process doesn't matter are cases where the mechanism described in the chapter is being used purely as a way to generate preference accounts (Langley, 2019), i.e., the system is purely trying to explain why one decision may be better than other in terms of its model of the task, regardless of how the agent generated the original decision. Such explanations could be helpful in cases where the agent and the user are collaborating to generate better solutions.

Study Interface Screenshots



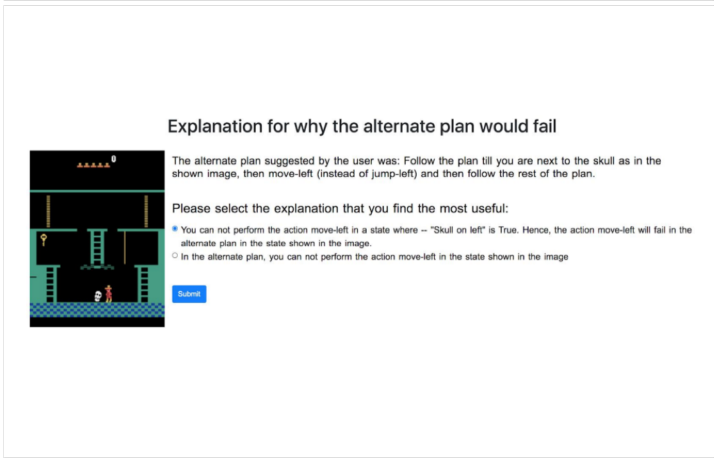
Plan and the alternate plan

Click anywhere on the video below to view the plan:

Plan goal: Get to the key

Alternate plan suggested by the user: Follow the plan till you are next to the skull as in the shown image, then move-left (instead of jump-left) and then follow the rest of the plan.

(A)



Explanation for why the alternate plan would fail

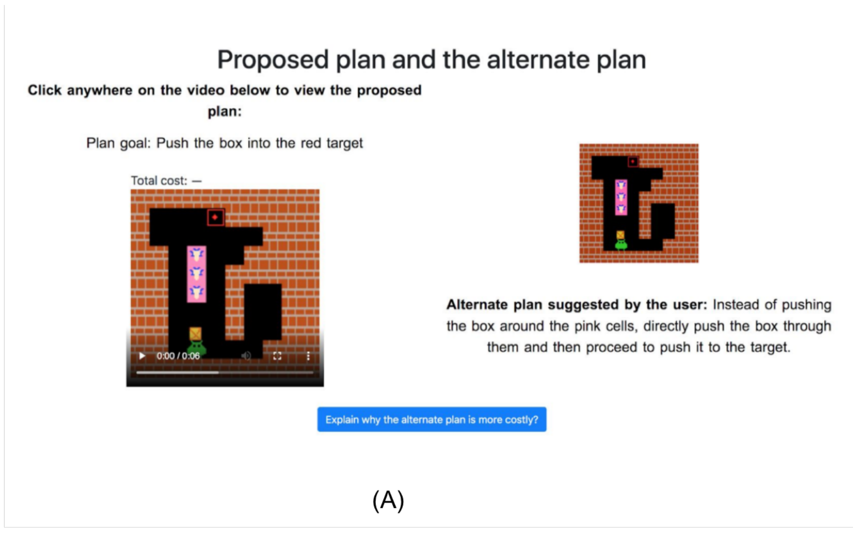
The alternate plan suggested by the user was: Follow the plan till you are next to the skull as in the shown image, then move-left (instead of jump-left) and then follow the rest of the plan.

Please select the explanation that you find the most useful:

- [✓] You can not perform the action move-left in a state where -- "Skull on left" is True. Hence, the action move-left will fail in the alternate plan in the state shown in the image.
- In the alternate plan, you can not perform the action move-left in the state shown in the image

(B)

Figure A.8: Screenshot from the Study Interface for H1 for Precondition.



Proposed plan and the alternate plan

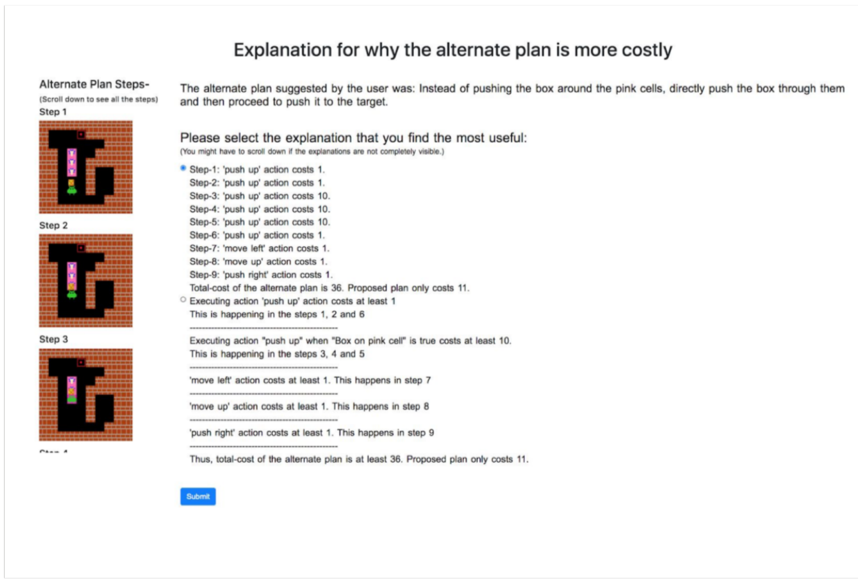
Click anywhere on the video below to view the proposed plan:

Plan goal: Push the box into the red target

Total cost: —

Alternate plan suggested by the user: Instead of pushing the box around the pink cells, directly push the box through them and then proceed to push it to the target.

(A)



Explanation for why the alternate plan is more costly

The alternate plan suggested by the user was: Instead of pushing the box around the pink cells, directly push the box through them and then proceed to push it to the target.

Please select the explanation that you find the most useful:
(You might have to scroll down if the explanations are not completely visible.)

● Step-1: 'push up' action costs 1.
Step-2: 'push up' action costs 1.
Step-3: 'push up' action costs 10.
Step-4: 'push up' action costs 10.
Step-5: 'push up' action costs 10.
Step-6: 'push up' action costs 1.
Step-7: 'move left' action costs 1.
Step-8: 'move up' action costs 1.
Step-9: 'push right' action costs 1.
Total-cost of the alternate plan is 36. Proposed plan only costs 11.

○ Executing action 'push up' action costs at least 1
This is happening in the steps 1, 2 and 6

Executing action "push up" when "Box on pink cell" is true costs at least 10.
This is happening in the steps 3, 4 and 5

'move left' action costs at least 1. This happens in step 7

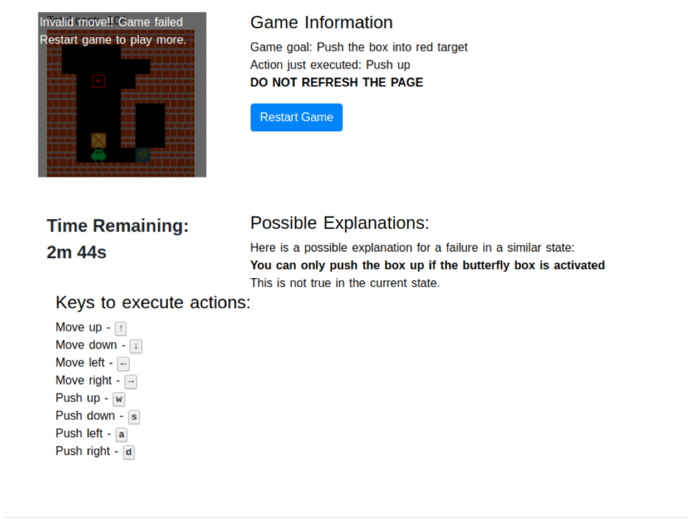
'move up' action costs at least 1. This happens in step 8

'push right' action costs at least 1. This happens in step 9

Thus, total-cost of the alternate plan is at least 36. Proposed plan only costs 11.

(B)

Figure A.9: Screenshot from the Study Interface for H1 for Cost Function Explanations.



Invalid move!! Game failed
Restart game to play more.

Game Information

Game goal: Push the box into red target
Action just executed: Push up
DO NOT REFRESH THE PAGE

Time Remaining:
2m 44s

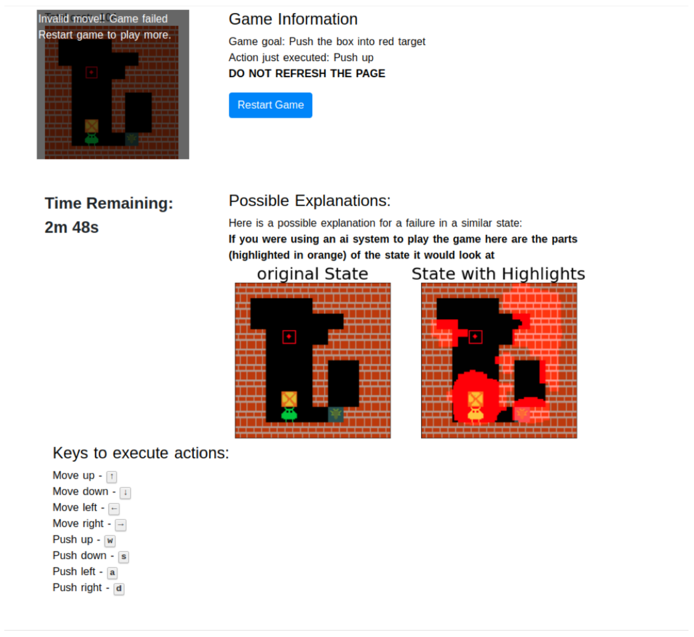
Possible Explanations:

Here is a possible explanation for a failure in a similar state:
You can only push the box up if the butterfly box is activated
This is not true in the current state.

Keys to execute actions:

Move up - ↑
Move down - ↓
Move left - ←
Move right - →
Push up - w
Push down - s
Push left - a
Push right - d

(A)



Invalid move!! Game failed
Restart game to play more.

Game Information

Game goal: Push the box into red target
Action just executed: Push up
DO NOT REFRESH THE PAGE

Time Remaining:
2m 48s

Possible Explanations:

Here is a possible explanation for a failure in a similar state:
If you were using an ai system to play the game here are the parts (highlighted in orange) of the state it would look at

Keys to execute actions:

Move up - ↑
Move down - ↓
Move left - ←
Move right - →
Push up - w
Push down - s
Push left - a
Push right - d

(B)

Figure A.10: Screenshot from the Study Interface for H2.

APPENDIX B

APPENDIX FOR CHAPTER 18

Proof for Theorem 1

Theorem 16. *Given a canonical augmented model $\mathcal{M}_{\mathcal{E}}$ for the human and robot models $\mathcal{M}_R$ and $\mathcal{M}_H$, if the sum of cost of all explanatory actions is less than or equal to $\Delta\pi_{\mathcal{M}_R}$ and if $\pi_{\mathcal{E}}$ is an optimal plan for $\mathcal{M}_{\mathcal{E}}$, then $\mathcal{D}(\pi_{\mathcal{E}})$ is optimal for $\mathcal{M}_R$ and $\mathcal{E}(\pi_{\mathcal{E}})$ is the MCE for $\mathcal{D}(\pi_{\mathcal{E}})$. Additionally, there exists no plan $\pi \in \Pi^*_H$ such that MCE for π is cheaper than $\mathcal{E}(\pi_{\mathcal{E}})$.*

Proof Sketch. We observe that there exists no valid plan π' for the augmented model ($\mathcal{M}_{\mathcal{E}}$) with a cost lower than that of $\pi_{\mathcal{E}}$ and where the ontic fragment ($\mathcal{D}(\pi')$) is optimal for the human model. Let's assume $\mathcal{D}(\pi_{\mathcal{E}}) \notin \Pi^*_{\mathcal{R}}$ (i.e current plan's ontic fragment is not optimal in robot model) and let $\hat{\pi} \in \Pi^*_{\mathcal{R}}$. Now let's consider the augmented plan corresponding to $\hat{\pi}$, $\hat{\pi}_{\mathcal{E}}$, i.e, $\mathcal{E}(\hat{\pi}_{\mathcal{E}}) = \mathcal{E}^{\mathcal{M}_R,\mathcal{M}_H}_{MCE}(\hat{\pi})$ (the MCE for the plan $\hat{\pi}$) and $\mathcal{D}(\hat{\pi}_{\mathcal{E}}) = \hat{\pi}$. Then the given augmented plan $\hat{\pi}_{\mathcal{E}}$ is a valid solution for our augmented planning problem $\mathcal{M}_{\mathcal{E}}$ (since the $\hat{\pi}_{\mathcal{E}}$ consists of the MCE for $\hat{\pi}$, the plan must be valid and optimal in the human model), moreover the cost of $\hat{\pi}_{\mathcal{E}}$ must be lower than $\pi_{\mathcal{E}}$. This contradicts our earlier assumption hence we can show that $\mathcal{D}(\pi_{\mathcal{E}})$ is in fact optimal for the robot model.

Using a similar logic we can also show that no cheaper explanation exists for $\pi_{\mathcal{E}}$ and there exists no other plan with a cheaper explanation. □

USAR Domain

The basic scenario consists of an autonomous agent that has been deployed to the disaster scene and an external commander who is monitoring the activities of the robot. Both agents start with the same model of the world (i.e the map of the building before the disaster) but the models diverge over time owing to the fact that robot has access to more accurate information about the current status of the building. In the specific setting we are looking at the robot start at position P1 and needs to reach position P14. The main difference between the human and robot maps is the fact that the human incorrectly believes that the path from P5 to P6 is clear while that from P8 to P12 is blocked. The robot could potentially clarify these two misconceptions through the explanatory actions –

```
(:action explain_obstructed_P6_P5
    :precondition
    (and
    )
    :effect
    (and
        (B(obstructed_P6_P5))
        (increase (total-cost) 10)
    )
)
(:action explain_away_obstructed_P8_P12
    :precondition
    (and
    )
    :effect
    (and
       (not (𝓑(obstructed_P6_P5)))
       (increase (total-cost) 10)
    )
)
```

The robot could also potentially explain the obstruction or (the fact that the passage is clear) by visiting it

```
(:action move_from_P7_P8
    :precondition
    (and
         (robot_at_P7) (connected_P7_P8)
         (𝓑(connected_P7_P8))
         (𝓑(robot_at_P7)))
    :effect
    (and
       (robot_at_P8) (𝓑(robot_at_P8))
       (𝓑(clear_P8_P12))
       (increase (total-cost) 1))
)
(:action move_from_P1_P6
    :precondition
    (and
         (robot_at_P1) (connected_P1_P6)
         (𝓑(connected_P1_P6))
         (𝓑(robot_at_P1)))
    :effect
    (and
       (robot_at_P6) (𝓑(robot_at_P6))
       (not 𝓑(clear_P6_P5))
       (increase (total-cost) 1))
)
```

A Unifying Framework for Multi-model Planning

Since the foundational paper on model reconciliation, the approach has been extended to support many new features. Some of the most relevant ones are the ability to support incomplete information about the human mental model (Sreedharan *et al.*, 2018a) and the ability to generate explanations when the human model is at a different level of abstraction (Sreedharan *et al.*, 2018c). It is easy to see that the proposed encoding can be easily extended to capture most of these extensions.

In the case of model uncertainty, we can capture this by assuming that the initial value of the meta-state fluents (F_μ) is unknown and then generating either conformant or contingent plans. For generating conformant plans, we can either use techniques discussed by Palacios and Geffner (2009) and Muise *et al.* (2015) and compile it into a classical planning problem or directly perform a search in the belief space. We can also use the intuition of min and max models that were introduced by Sreedharan *et al.* (2018a) to consider a much smaller belief space. When generating contingent plans, we will need to consider additional actions for collecting information about the human models.

In cases where the more abstract models of the task can be formed by simply projecting out fluents from the rules in the model, our encoding can be extended to handle them by introducing new meta-fluents that capture whether the human knows about certain state fluents. Now each effect or precondition in the augmented model related to these state fluents will be also conditioned on these new meta fluents. This captures the fact that you can ignore those rules if the human did not know about the fluents that appear in them. We can also add new explanatory actions that turn these meta fluents true. These new explanatory actions capture the ability to explain these state properties to the human observer.

Another aspect investigated by Sreedharan *et al.* (2018c) was the fact that we may not need to give the entirety of MCE, but rather just provide explanations corresponding to specific user queries. This queries take the form of user specifying a set of alternate plans (referred to as foils). In such, cases we can guarantee that there exist a subset of the MCE generated by the algorithm that can resolve the given set of foils. We can thus first generate the MCE and then look for appropriate subsets.

We can also extend the formulation to support more expressive models such as ones with disjunctive preconditions and cases where the models may differ in actions costs. For disjunctive preconditions, the easiest method would be to just normalize the preconditions to CNF and then use a different action for each precondition clause. For cases with differing costs, we will need to keep track of the cost of the plan according to the robot and the plan cost as perceived by the human. The augmented model should now try to minimize the weighted sum of these two costs (again we can use optimality gap to establish how these components should be weighed). We will also need to use state dependent costs to capture the fact that the human's perceived cost can change.

New Explanatory Capabilities

The formulation allows us to not only capture previously discussed approaches, but also allows us to generate explanatory behavior that has not been previously studied. We will look at these new capabilities by grounding the need for these capabilities in the USAR scenario.

Explanatory Actions With Ontic Effects

As soon as we start considering explanatory actions in our encoding it is inevitable that they would have other side effects. Consider an action that at first glance may seem purely communicative like, using natural language utterances or using some mixed-reality techniques to project some information, it may still have other effects on the task state like reducing the robot's power levels, blocking communication channels etc. Similarly action that may appear to only have task level effect may still end up updating the human's mental state. For example, the robot opening a door is enough to inform the human that the door was not locked in the first place and does not require a separate communication action. These *side effects* need to be taken into account when coming up with the plan and its explanations.

To illustrate the use of such explanatory actions in our encoding let us visit the USAR scenario and assume that the human thinks that the path from P8 to P12 is blocked and the one from P6 to P5 is free. Also in this setting, for explaining the status of passages (whether they are blocked or not) the robot can now use two actions, one a rather expensive explicit communication action, that sends the updated map information to the human or it can just visit the blocked passage and the human who is watching a video feed of robot actions will learn that the passage is blocked or clear. Thus the action descriptions for the move action will be –

```
(:action move_from_P7_P8
    :precondition (and (robot_at_P7)
          ...
          (𝓑(robot_at_P7)))
    :effect (and (robot_at_P8)
         ...
         (𝓑(clear_P8_P12)) (increase (total-cost) 1)))
```

With this new action the robot knows that as soon as it reaches P8 the human would know that the path from P8 to P12 is clear so it can continue on that path. So the new robot plan will be –

```
INIT_ACT→move_from_p1_p7→move_from_p7_p8→
move_from_p8_p12→move_from_p12_p15→
explain_obstructed_p6_p5→
move_from_p15_p13→move_from_p13_p14→GOAL_ACT
```

Balancing Explicability and Explanation

Previous literature has broadly identified two general strategies to handle inexplicability of actions arising from model divergence, namely, reconcile the differences through explanation or act in a manner that aligns with the human's expectations (referred to as explicable planning in the literature). It should be quite clear that these aren't necessarily mutually exclusive strategies and an explainable agent should be capable of employing both strategies and trade them off to generate the most desirable behavior. In this section, we will build on the work presented in Chapter 17 to show how such behavior can be generated using our new encoding.

Given the robot model $\mathcal{M}_R$ and the human mental model $\mathcal{M}_H$, we assume a utilitarian point of view that the quality of any given plan in each individual model can be completely characterized by the plan's cost in the model [1]. As mentioned earlier, in cooperative scenarios like the one studied in this chapter, the choice of a plan can no longer be made by their perceived quality in any one of these individual models, but rather we need to consider their impact on the entire team. Specifically, we need to consider the cost incurred by the actor (the robot) and the degree to which the plan conforms to the observer's expectations (and hence impact their future actions). This need for trade-off between the plan cost and the possible penalty

[1] We assume that the cost encapsulates both the value of soft-goals and actions costs and invalid plans have infinite cost.

Algorithm 11 Updated A^*-Search

```
1: procedure GOAL_CHECK(node, $\beta$)
2:     if node.state $\subseteq G_R \cup \{\mathcal{B}(p) | p \in G_H\}$ then
3:         if (under_budget) $\in$ node.state then
4:             node.cost $\leftarrow$ node.cost + $\beta * (C(\pi_H^*)$ - $C(\mathcal{D}$(node.plan)))    $\triangleright$ $\pi_H^*$ - Optimal human plan
5:     if node.plan.peek() == $a_\infty$ then
6:         return True
7:     return False
8: procedure FIND_SUCCESSOR_NODES(node, $\mathcal{K}$, $\eta$, $\mathcal{M}_\mathcal{E}, \alpha$)
9:     succ_list $\leftarrow$ {}
10:    $C_\mathcal{E}$ $\leftarrow$ Identify_Explanation_Cost(node.plan)
11:    if $C_\mathcal{E}$ + min_explanation_cost > $\mathcal{K}$ then
12:        new_node.state $\leftarrow$ node.state $\setminus$ {(under_budget)}
13:        new_node.plan = node.plan + {Stop_Checking_Budget}
14:        new_node.cost $\leftarrow$ node.cost + $\eta$
15:        succ_list.push(new_node)
16:    for a $\in$ applicable_actions(node, $\mathcal{M}_\mathcal{E}$) do
17:        new_node.state $\leftarrow$ a(node.state)
18:        new_node.plan = node.plan + {a}
19:        if a is an ontic action then
20:            node.cost $\leftarrow$ node.cost + $\alpha$ * C(a)
21:        else
22:            node.cost $\leftarrow$ node.cost + C(a)
23:        succ_list.push(new_node)
24:    return succ_list
```

incurred from producing inexplicable plan necessitates the development of planners capable of generating behavior of varying characteristics. A simple one, was the scenario discussed in Chapter 17 wherein an agent chooses to follow a costlier but easy to explain plan (in their case consisting of only ontic actions) to potentially save on the cost of communicating the required explanations. Similarly, there may be cases where the robot can't afford to either divert from the optimal plan or provide the required explanations. For example, consider a USAR scenario where sending the required explanatory message or deviating from the optimal path may result in delay of rescue of a victim in a critical state. In such cases, the planner needs to be aware of the fact that it is in fact incurring a penalty by not providing explanations, but the cost of providing the explanations outweigh any benefits it may provide. In this section, we will look at the planning formulation discussed earlier and look at the types of explanatory behavior it can produce and propose to generalizations of this formulation that allows us to provide a wide variety of additional behaviors.

1. **Current Formulation** - This corresponds to the planning formulation introduced in Section 18.2.1. The goal of this formulation can be captured as

$$\text{argmin}_\pi \ C(\mathcal{E}(\pi)) + C(\mathcal{D}(\pi))$$

$$\text{Provided } \mathcal{D}(\pi) \in \Pi^*_{\mathcal{M}_H + \mathcal{E}(\pi)} \text{ and } \pi(I_{\mathcal{M}_\mathcal{E}}) \models_{\mathcal{M}_\mathcal{E}} G_{\mathcal{M}_\mathcal{E}}$$

In this formulation, the search finds the plan in the augmented model with the minimal cost that ensures the optimality of the plan in the human model. As shown in Theorem 13, when the cost of the explanatory actions are scaled down, the formulation can generate the purely explanatory behavior discussed by Chakraborti *et al.* (2017). More interestingly this approach also allows us to capture the explanation explicability trade-off discussed by Chakraborti *et al.* (2019f), which is summarized as

$$\text{argmin}_{\pi,\mathcal{E}}\ C(\mathcal{E}) + \alpha \mid C(\pi) - C(\pi^*_R) \mid$$

$$\text{Provided } \pi \in \Pi^*_{\mathcal{M}_H+\mathcal{E}} \text{ and } \pi(I_{\mathcal{M}_R}) \models_{\mathcal{M}_R} G_{\mathcal{M}_R}$$

Where $C(\mathcal{E})$ is the cost of the explanation and $\mid C(\pi) - C(\pi^*_R) \mid$ the additional penalty accrued by the robot by deviating from optimality. α specifies a hyperparameter that controls the degree to which the robot is ready to deviate from the optimal, i.e. the higher the α, the more the robot would prefer to stick to the optimal plan and provide more explanations. In our formulation, α turns into a scaling factor over the cost of the ontic actions that can be used to generate the same behaviors.

2. **Allowing for Inexplicability** - This is the relaxation of the earlier formulation, where the robot now no longer requires the plan to be optimal in the human model, but presenting a suboptimal plan can induce a penalty. This is useful in cases where ensuring the optimality in human model can prove to be too expensive and is optional. This formulation can be summarized as –

$$\text{argmin}_{\pi}\ C(\mathcal{E}(\pi)) + \alpha * C(\pi) + \beta * (\mid C(\pi) - C(\pi^*_H) \mid)$$

$$\text{Given that } \pi(I_{\mathcal{M}_\mathcal{E}}) \models_{\mathcal{M}_\mathcal{E}} G_{\mathcal{M}_R}$$

We can model this penalty by setting the cost of a_∞ to be the difference between the optimal plan in the human model and the current plan. Note that we can use this formulation to generate all the behavior supported by earlier formulation by setting β sufficiently high.

3. **Budgeted Explanation Generation** - In most mission critical domains like USAR there may be prespecified constraints on how much effort the robot could spend on explanation, i.e, there may be a budget ($\mathcal{K}$) on explanation. In this formulation, as long as the explanations can be completed under budget the robot would choose to follow the objective function defined earlier and when the plan is over the budget the requirements to meet the human precondition or apply the penalty is relaxed –

$$\begin{aligned}&\text{argmin}_{\pi}\ C(\mathcal{E}(\pi)) + \alpha * C(\pi) + \\ &\gamma * \beta * (\mid C(\pi) - C(\pi^*_H) \mid) + (1-\gamma) * \eta\end{aligned}$$

$$\text{Given that } \pi(I_{\mathcal{M}_R}) \models_{\mathcal{M}_R} G_{\mathcal{M}_R} \text{ and}$$

$$\gamma = 1, \text{when}, C(\mathcal{E}(\pi)) \leq \mathcal{K}, \text{else}, \gamma = 0$$

where η is the penalty term for choosing not to explain. Satisfying this condition requires us to make the following changes to our formulation and search: a) introduce a new fluent under_budget and update the precondition of the actions such that preconditions on belief fluents ($\mathcal{B}(*)$) are now conditioned on this new fluent, i.e we will care about satisfying human preconditions only if there is still budget left for explanations; b) the penalty for the plan not being optimal is only added if we are under the explanation budget; c) we delete the fluent under_budget from the state when the current prefix has exhausted the budget or the cost of the cheapest explanation is less than the leftover budget; and finally d) deleting under_budget fluent adds η to the current search cost, this is to ensure that the search doesn't exhaust the explanation budget while there exists plans that can be explained. If we set the penalty to be higher than $\beta * C(\pi')$, where $\pi' \in \Pi^*_{\mathcal{M}_R}$ we can still guarantee all the earlier behavior. Algorithm 11 sketches the pseudo code for these new changes, in particular the procedure to generate the successors and the procedure to perform the goal check that also adds the penalty for inexplicability of the plan. The Find_Successor_Node procedure also inserts a new action called Stop_Checking_Budget when the explanatory budget is used up, and the cost of this action is set to η.

To illustrate the use of budgeted explanations and the need for sometimes generating inexplicable behavior, we go back to the urban search and rescue case. Here, the optimal path for the robot to follow would be to go through P7, P8 and P12. The human thinks the path should be the one through P6 and P5. Explaining the optimality of the path requires explaining that the path from P6 to P5 is blocked (which can be explained through the action explain_obstructed_P6_P5) and the path from P8 to P12 is clear (explained by explain_away_obstructed_P8_P12). Let us assume that the application of each of these explanatory actions increases the total cost by ten (please refer to the supplementary attachment for the PDDL description of each of these actions). If we try to solve this with the second formulation with $\beta = 1$ and a cost of 5 for the clear passage action, then the plan that would be generated would be –

init_act→move_from_p1_p9→
clear_passage_p9_p10→move_from_p9_p10→
move_from_p10_p13→move_from_p13_p14→GOAL_ACT

Now if we were to use a budget of five on the explanation, then no explanation action is possible and during the second iteration the plan the robot would come up with would be just the optimal plan.

INIT_ACT→move_from_p1_p7→
move_from_p7_p8→ Stop_Checking_Budget→
move_from_p8_p12→move_from_p12_p15→
move_from_p15_p13→move_from_p13_p14→GOAL_ACT

where Stop_Checking_Budget is the action that removes the under budget fluent from the search.

Another assumption made by many of the earlier work is the fact that the human observer is a perfect listener. Which means that once the explanation is provided she will definitely include it in her reasoning. Unfortunately, this is not true in most cases. The human's ability to understand the explanation may depend on factors like the hardness of the concept being explained is (for eg: explaining the robot's reachability vs the ability for the robot to pick up heavy objects) and the mode of explanation (a simple visualization vs natural language) and even the cognitive load of the listener. A more realistic approach would require us to model these uncertainties into the explanatory process. Our encoding can be extended to incorporate such uncertainty by making the effects of the explanatory actions non-deterministic or stochastic.

To see a simple example of how this would look, consider the USAR domain and look at the ability of the move action to inform the commander about the status of the passage from P8 to P12. The robot can not always guarantee that the commander would be looking at the screen and there is a chance that the commander won't be looking at the screen when the path is presented. Thus in the new definition of action (move_from_P7_P8) we will replace the effect that adds $\mathcal{B}$(clear_P8_P12) with the effect (one-off ($\mathcal{B}$(clear_P8_P12)) (and)) Which means that the action's ability to update the human model is a non-deterministic effect. In this case, we can look for a conformant plan by converting the earlier search into a search over belief space. A search node only passes the goal test if the goal condition are met in every state in the belief space. Thankfully, this particular problem does have a conformant solution –

explain_obstructed_p6_p5→explain_clear_p8_p12→
INIT_ACT→move_from_p1_p7→move_from_p7_p8→
move_from_p8_p12→move_from_p12_p15→
move_from_p15_p13→move_from_p13_p14→ GOAL_ACT

Reducing Search Effort During Secondary Search

We further improve the runtime efficiency of the proposed methods by allowing the search to reuse the search space from one secondary search (for the optimal plan in the updated human model) to another.

We do this by adopting the idea of sufficiency test for optimality introduced by Fritz and McIlraith (2007). The paper shows that a plan is guaranteed to be optimal if the value of the plan is better than fringe generated as part of the search for that plan. Where fringe is defined to be the current open list and the set of nodes that were found to be invalid during the search. The reason for keeping track of the invalid sequences is to allow for the possibility of those nodes being available later. This perfectly suits our requirements, since the change in the human model can render many of the previously infeasible search paths become feasible and previously feasible ones invalid. So in this setting, we will be carrying over the fringe and the last generated optimal plan from one instance of the secondary search to another. If the last generated optimal plan is still better than all the nodes in the fringe that will still be the optimal plan, else the node for the current plan is added back into the open list and the current best node is expanded. One point of interest is that if we want to carry over the fringe from one instance to another we would want the f value

	Config-1		config-2	
Domain Name	coverage	avg-runtime (s)	coverage	avg-runtime (s)
Blocksworld	8/10	527.64	8/10	463.745
Elevator	10/10	51.38	10/10	10.37
Gripper	5/10	1165.905	5/10	1105.926
Driverlog	3/10	1557.543	3/10	1575.099
Satellite	1/10	1722.991	1/10	1713.878

Figure B.1: Table Showing Runtime for Explanations Generated for Standard IPC Domains.

of the search nodes to stay steady. This means we need to use a heuristic value that stays steady across the various action models. We can achieve this by using the h_{max} (Bonet and Geffner, 2001) on a planning model $\mathcal{M}'$ such that for any action $a \in A'$, we have

$$pre(a) = pre(a_R) \cap pre(a_H)$$

and

$$add^{(}a) = add^{(}a_R) \cup add^{(}a_H)$$

This new model over-approximates the possibles add effects each action can achieve and under-approximates the precondition that needs to be satisfied for the applicability of an action. The h_{max} estimate produced in this model will be less than or equal to the estimate for the same state in any model that the human could potentially possess (assuming model updates happen only through valid explanations).

Table B.1 shows the improvement in time achieved by including this technique to our base search for a few IPC domains.

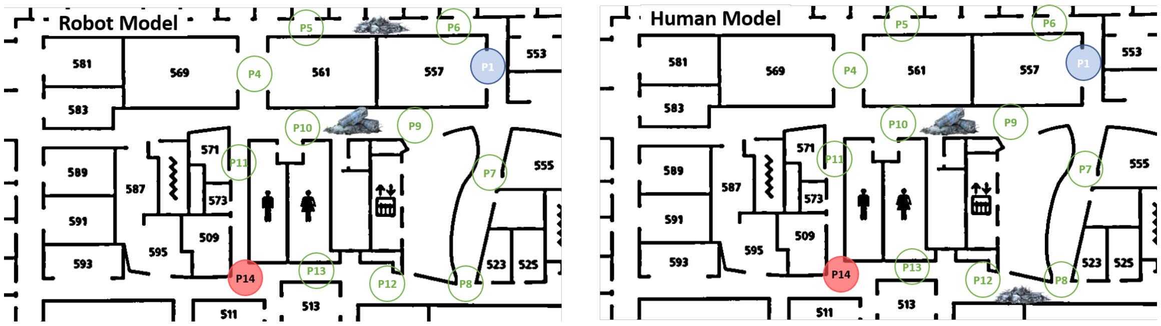


Figure B.2: The Basic Robot (Left) and Human (Right) Maps. The Robot Start at P1 and Needs to Go to P14. The Human Incorrectly Believes That the Path from P6 to P5 Is Clear and the One from P8 to P12 Is Blocked. Both Agents Know That There Are Some Movable Rubble Between P9 and P10 That Can Be Moved with the Help of a Costly Remove_rubble Action.