Improving Transparency and Intelligibility of Multi-Objective Probabilistic Planning

Roykrong Sukkerd

CMU-ISR-22-104

May 10, 2022

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
David Garlan, Co-Chair
Reid Simmons, Co-Chair
Manuela Veloso
Bogdan Vasilescu
Amel Bennaceur (The Open University, UK)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Keywords: Explainable AI Planning, Markov Decision Process Planning, Multi-Objective Planning, Multi-Objective Markov Decision Processes, Policy Explanations, Contrastive Explanations, Consequence-Oriented Contrastive Explanations

Abstract

Sequential decision-making problems with multiple objectives are natural to many application domains of AI-enabled systems. As these systems are increasingly used to work with people or to make decisions that impact people, it is important that their reasoning is intelligible to the end-users and stakeholders, to foster trust and effective human-agent collaborations. However, understanding the reasoning behind solving sequential decision problems is difficult for end-users even when white-box decision models such as Markov decision processes (MDPs) are used. Such intelligibility challenge is due to the combinatorial explosion of possible strategies for solving long-horizon problems. The multi-objective optimization aspect further complicates the problem as different objectives may conflict and reasoning about tradeoffs is required. These complexities pose a barrier for end-users to know whether the agent has made the right decisions for a given context, and may prohibit them from intervening if the agent was wrong. The goal of this thesis is to develop an explainability framework that enables the agent making sequential decisions to communicate its goals and rationale for its behavior to the end-users.

We present an explainable planning framework for MDP, particularly to support problem domains with multiple optimization objectives. We propose *consequence-oriented contrastive explanations*, in which an argument for an agent's policy is in terms of its expected consequences on the task objectives, put in context of the selected viable alternatives to demonstrate the optimization and tradeoff reasoning of the agent. Our modeling framework supports reward decomposition, and augments MDP representation to ground the components of the reward or cost function in the domain-level concepts and semantics, to facilitate explanation generation. Our explanation generation method computes policy-level contrastive foils that describe the inflection points in the agent's decision making in terms of optimization and tradeoff reasoning of the decomposed task objectives. We demonstrate the applicability of our explainable planning framework by applying it to three planning problem domains: waypoint-based navigation, UAV mission planning, and clinic scheduling.

We design and conduct a human subjects experiment to evaluate the effectiveness of explanations based on measurable task performance. We design the users' task in the experiment to be: assessing the agent's planning decisions to determine whether they are the best decisions for a given problem context. Our experimental results show that our proposed consequence-oriented contrastive explanation approach significantly improves the users' ability to correctly assess the agent's planning decisions, as well as the users' confidence in their assessment.

Lastly, we investigate the feasibility of a user-guided approach to our consequence-oriented contrastive explanation paradigm. We propose a theoretical framework and approaches to formulate *Why Not* behavioral questions as state-action constraints and linear temporal logic constraints on the planning problem, and to solve for satisfying policies in order to explain the full impact that the queried behavior has on the subsequent decisions and on the task objectives.

Acknowledgments

I would like to thank my PhD advisors, David Garlan and Reid Simmons, for their tremendous support, advice, mentorship, and patience throughout my time as a PhD student. I thank my thesis committee members, David Garlan, Reid Simmons, Bogdan Vasilescu, Manuela Veloso, and Amel Bennaceur, for their time, input, and invaluable advice and feedback that have bettered my work. I also thank my research group, ABLE, and my academic program and department, Software Engineering and ISR, for providing me the environment and opportunities to grow as an independent researcher.

I would like to thank the funding agencies that have supported my research throughout my PhD. My research is supported in part by award N00014172899 and award N000141612961 from the Office of Naval Research (ONR); by award H9823018D0008 from the National Security Agency (NSA); and by the US Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8750-16-2-0042. Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the agencies.

I would like to thank my family: my parents, my brother, and my aunt Nid who have loved and supported me throughout both happy and difficult times. Lastly, I would like to thank my partner Jens, who have supported me tremendously throughout my journey from thesis proposal to thesis defense and beyond. In all the difficult and trying times, Jens have stood by me and have made everything all better. I am grateful for your support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automated decisions made by systems with Artificial Intelligence (AI) components are often opaque and difficult to understand for the end-users, due to algorithmic complexity of the AI reasoning and information they use that are hidden from the users. In particular, sequential decision making processes, in which multiple dependent actions are taken sequentially over time, pose a significant challenge for end-user understanding due to the combinatorial explosion of possible strategies, especially when involving reasoning under uncertainty. As AI-enabled systems are increasingly used in many domains to work with people or to make decisions that impact people, it is important that these systems make their reasoning intelligible to the end-users and stakeholders. Understanding the systems' goals and behavior, and having confidence that the systems make the right decisions for their tasks, or knowing when they do not, are crucial to allow people to trust and to effectively use or collaborate with these AI-enabled systems. The goal of this thesis is to develop an explainability framework that enables the agent making sequential decisions to communicate its goals and rationale for its behavior to the end-users.

For many sequential decision-making or planning application domains, oftentimes there are multiple ways of achieving a goal that may differ in some important qualities, such as execution time and various measures of performance and costs. Such sequential decision problems involve optimization of the domain-specific quantitative properties [78]. An important intelligibility type [61] for the end-users or stakeholders of such application domains is: Why are the agent's decisions optimal? This question has several aspects. The sequential decision framework being used, the domain models, the task objectives (i.e., goals and optimization objectives), and the algorithm to solve the problem all play the roles in selecting optimal decisions. Therefore, multiple aspects of the problem can be explained. In this thesis, we particularly focus on the aspects of the domain models and the task objectives, with Markov decision process (MDP) as the sequential decision framework. We investigate algorithm-agnostic approaches to explain to the users why MDP planning agents' policies are optimal, with respect to the agents' decision models. That is, our approaches explain the agents' optimal policies with respect to their MDP models, independent of the specific algorithms that solve the MDPs. We aim to explain to the users the reasoning that gives rise to planning solutions that may not be apparent to them, based on what the agent's decision making model does rather than how the underlying planning algorithm achieves it.

We argue that *contrastive explanations* are essential for answering why a plan or a policy is optimal, as optimality is fundamentally a comparative property. The idea of contrastive explanations is rooted in the social phenomenon that, when people ask for an explanation of an event, i.e., fact ("Why did P happen?"), they often are asking for an explanation relative to some contrast case, i.e., foil ("Why did P happen rather than Q?") [69, 70].

Existing works in explainable AI planning that employ contrastive explanations for inference reconciliation (meaning to help the inference process of the users to better understand the planner's decisions) [18] largely focus on a contrastive foils at an individual action level. These works typically aim to answer the questions "Why is this action in this plan?" and "Why not this other action?" [36]. However, to answer the question "Why is this policy optimal?", we argue that framing contrastive foils at an individual action level is not sufficient. For the users to understand why a policy is optimal, knowing that choosing an alternate action at a state would have a lower value (i.e., $Q(s, a')$ value, or its decomposed components, for an alternate action a' at at state s) may not be sufficient. The lower value can be the result of the subsequent states and actions that are not explained to the user. Moreover, there are many possible sets of subsequent states and actions that the agent could choose that would result in different policy values. These alternate decisions are not addressed if the explanation frames contrastive foils as individual actions. Instead, we argue that combinations of actions that lead to lower policy values should be explored, in order to provide the user more insights into why a policy is optimal. The key challenge is to identify the appropriate combinations of actions as contrastive foils from an exponentially large space.

In this thesis, we investigate the questions of what constitutes effective contrastive explanations for why a policy is optimal, and how to automatically generate such explanations. We focus on explaining Markov decision process planning, for which, even though the planning model is a white-box, the long-horizon sequential and probabilistic nature of the decision making still makes the agent's reasoning difficult for the user to understand. Our work is particularly motivated to target problem domains in which multiple reward or cost objectives are involved, as they reflect many real world applications of planning [78]. The multi-objective optimization nature of the problem adds to the complexity of the agent's reasoning that the user needs to understand.

The primary goal of explanations we investigate in this thesis is to enable the user to assess whether the planning agent's decisions are optimal with respect to the user's objectives and preferences, which may differ from those of the agent. The reason we focus on this goal is two-fold. First, aligning the agent's and the user's objectives and preferences is a difficult problem in designing a reward or cost function for the planning problem [37, 59]. Particularly, when there are multiple components to the planning objectives, misalignment may occur. Explanations that can assist the user with detecting such (mis)alignment can be impactful. Second, establishing a task-oriented goal for explanations allows for a more objective evaluation method for measuring the effectiveness of proposed explanation approaches.

We investigate how to use contrastive explanations as a mechanism to describe how the agent's planning objective steers its decision making to a certain policy and not others. Unlike the commonly used definitions of contrastive explanations in explainable AI literature, we do not focus on causal difference conditions or non-causal differences in properties associated with different circumstances. Instead, our main idea is to focus on the inflection points in the agent's decision making that make it chooses a certain policy over some other reasonable alternatives. That is, we focus on the points at which the agent changes its optimization trajectory in order to balance competing task objectives. To identify a manageable set of informative contrastive foils – at a combination-of-actions level – from an exponentially large space, we leverage the multi-objective nature of the planning problems to the explanations' advantage. Via reward decomposition, we identify a subset of Pareto efficient policies as candidate contrastive foils. The main idea in our contrastive explanation method is to describe how the components of the reward or cost values drive the inflection points in the optimization decision in the Pareto efficient subspace, and how tradeoffs are made to reconcile competing task objectives. Our argument for this type of contrastive explanations is two-fold. First, it is faithful to the underlying mathematical approach of the planning algorithm that computes an optimal value function for a multi-objective reward or cost function. Second, it supports the goal of explanations motivated in this thesis, namely, to assist the user in recognizing the alignment or misalignment between the agent's objectives and preferences and theirs.

1.1 Thesis Statement

The thesis statement is:

We can improve transparency and intelligibility of Markov decision process (MDP) planning agents for end-users, in terms of how they reason about finding optimal policies, by means of consequence-oriented contrastive explanations*. Our proposed approaches allow the users to understand the planning rationale from the perspective of consequences of different sequential decisions on the various task objectives, and tradeoffs among competing objectives. Our proposed explanation mechanisms enable end-users to evaluate the alignment between planning-based agents' objectives and those of their own, and to potentially address unexpected agents' decisions from the users' point of view.*

Next, we elaborate on the thesis statement.

This thesis focuses on the problem of inference reconciliation (i.e., to help the inference process of the users to better understand the planner's decisions [18]) for Markov decision process (MDP) planning for end-users, particularly for problem domains in which multiple optimization objectives are involved. That is, we aim to generate explanations for why an agent's policy is optimal with respect to the task objectives and *a priori* preferences. We design our approach to be algorithm-agnostic and model-based. That is, our approach explains the agent's optimal policy with respect to its MDP model, independent of the specific algorithm that solves the MDP. We propose *consequence-oriented contrastive explanations*, in which an argument for a policy is in terms of its expected consequences on the task objectives, put in context of the selected viable alternatives to demonstrate the optimization and tradeoff reasoning of the agent.

Leveraging reward decomposition, our approach computes a small set of alternative policies as contrastive foils by replanning with constraints on the value functions of the individual components of the reward or cost function, which correspond to the task objectives of the problem. The value constraints are set up to solve for alternative policies at the inflection points on the Pareto front that indicate the agent's tradeoff decisions to reconcile competing task objectives. Our approach formulates explanations that describe how the different contrastive foils at a combination-of-actions level affect the task objectives compared to the agent's policy.

This thesis demonstrates that using consequence-oriented contrastive explanations improves end-users' understanding of the planning rationale, as measured by their ability to assess whether the agent's decisions align well with the users' own objectives and preferences for the task, which may differ from those of the agent [59]. We motivate in this thesis that such explanations provide the end-users with actionable insights about the agent's decisions. Namely, the users can have confidence that the agent has made the correct decisions, or they can identify what the agent has potentially gotten wrong and intervene accordingly.

Furthermore, we investigate an extension to our approach that allows the users to directly query on unexpected decisions in the agent's policy. We explore a user-guided approach to consequence-oriented contrastive explanations, to use user inputs in the form of "Why-Not" queries to guide the search for contrastive foils that are informative to the users' specific questions. To this end, we propose a theoretical framework and approaches to formulate "Why-Not" queries as state-action constraints and linear temporal logic (LTL) constraints on the planning problem, and to solve for satisfying policies in order to explain the full impact that the queried behavior has on the subsequent decisions and on the task objectives. The important consideration in using this explanation framework is: the computed policies for contrastive foils that satisfy the queried temporal patterns may be non-Markovian with respect to the state abstraction in the original MDP problem. The difference between the Markovian reward assumption in the agent's original reasoning, and the non-Markovian approach used to generate user-queried contrastive foils, must be acknowledged. Nonetheless, we posit that allowing contrastive foils to challenge the assumption in the agent's formal planning framework can be informative for the users to understand and assess the optimality criteria they desire for the agent's task. This investigation is a proof of concept. Its main goal is to explore the feasibility of going beyond soliloquy in our consequence-oriented contrastive explanation paradigm, and incorporating complex user queries to identify more relevant contrastive foils for better inference reconciliation.

1.2 Contributions

In this thesis, we make the following contributions:

1. We design an algorithm-agnostic, model-based explainable planning approach for Markov decision process (MDP) planning [94]. Our explanation goal is to provide inference reconciliation for why a policy is optimal with respect to the task objectives and *a prior* preferences of the planning agent. We propose an approach to generate *consequence-oriented contrastive explanations*. Our approach consists of two parts:
   (a) A modeling framework that supports reward decomposition and augments MDP representation to ground the components of the reward or cost function in the domain-level concepts and semantics, to facilitate explanation generation.
   (b) A method for computing policy-level contrastive foils that describe the inflection points in the agent's decision making in terms of optimization and tradeoff reasoning of the decomposed task objectives. Our method computes the contrastive foils by replanning with hard and soft constraints on the value functions of the decomposed task objectives. We formulate the constrained planning problem as a mixed-integer linear programming (MILP) problem. In particular, we formulate MILP problems to solve for deterministic policies [28] that satisfy the hard constraints on the value

functions. We use a penalty method [23] and a piecewise linear approximation [25] of non-linear penalty functions in MILP formulation for soft constraints.

2. We demonstrate the applicability of our proposed explainable planning framework by applying our approach to three planning problem domains: waypoint-based indoor robot navigation, Unmanned Aerial Vehicles (UAV) mission planning, and outpatient clinic scheduling. We demonstrate how each problem domain can be modeled in our explainable planning representation, and we discuss examples of explanations generated by our approach for different problem instances.

3. We design and conduct a human subjects experiment to evaluate the effectiveness of explanations based on measurable task performance. In particular, we design the users' task to be: assessing the agent's planning decisions to determine whether they are the best decisions for a given problem context. Our experimental results show that our proposed consequence-oriented contrastive explanation approach significantly improves the users' ability to correctly assess the agent's planning decisions, as well as the users' confidence in their assessment.

4. We investigate the feasibility of a user-guided approach to our consequence-oriented contrastive explanation paradigm. We propose a theoretical framework and approaches to formulate "Why-Not" behavioral questions as state-action constraints and linear temporal logic (LTL) constraints on the planning problem, and to solve for satisfying policies in order to explain the full impact that the queried behavior has on the subsequent decisions and on the task objectives. For solving MDPs with LTL constraints, we leverage an existing problem formulation approach [82] for constructing a product MDP that incorporates a deterministic Rabin automaton (DRA) generated from a LTL property, and the reward or cost function of the product MDP is defined based on the acceptance condition of the DRA. Policies for contrastive foils computed by this approach may be non-Markovian with respect to the state abstraction in the original MDP problem. Namely, the contrastive foils are *finite-memory policies* for the original MDP problem.

Chapter 2

Related Work

Research in the field of explainable AI [40] at large is significant and prevalent, with many different disciplines and objectives [1, 7, 71]. Notably, much of explainable AI research focuses on machine learning interpretability, particularly of large and complex deep neural network models [14, 19, 106]. Interpretability of such black-box neural network models is exceptionally challenging. However, AI agents that perform complex tasks such as autonomous cars and health care robots must have the capabilities to reason at task-level, abstracting over multiple components, some of which may be learned (e.g., perception and low-level control policies) and some may be specified (e.g., mechanisms of the systems). The field of *Explainable Agency* aims to enable autonomous agents (e.g., robots) performing such complex tasks to be able to explain their decisions and the reasoning that produced their choices [6, 58, 79].

There are various objectives of explanations in the existing explainable agency research and related fields such as Human-Robot Interaction (HRI). For instance, [87] proposes a categorization of explanation types: *Teaching* explanations convey to the human concepts and knowledge that the agent has gathered. *Introspective tracing* explanations are based on introspection into the underlying models and provide enough information to trace the decision making process. *Introspective informative* explanations convey less information than Tracing explanations, but still able to shed light on the agent's underlying models and decision making process to allow the human to know whether the agent's decisions are correct or where errors may have occurred. *Post-Hoc* explanations are constructed by the agent to explain a decision without accurately representing the underlying decision making process. These are typically used for explaining learned black-box models such as neural networks, where full-fidelity introspective explanations can be infeasible. *Execution* explanations report the traces of operations the agent has taken. The goal of this thesis aligns most with that of *introspective informative* explanations, but we also leverage *post-hoc* explanations to support introspection from the agent's perspective [86]. We ultimately aim to provide the users with sufficient insights into the agent's reasoning to enable them to have confidence that the agent is correct, and to detect and understand discrepancies between the agent's decisions and those expected by the users.

Explanations of sequential decision making processes, or planning problems, are studied in many different flavors in the field of Explainable AI Planning (XAIP) [36]. The recent survey on the emerging landscape of explainable AI planning [18] outlines three broad categories of approaches:

Algorithm-based explanations aim to explain the underlying planning algorithms. These explanation methods are common for explaining decisions generated by deep reinforcement learning. For instance, [39] generates saliency maps to understand how a deep reinforcement learning agent learns and executes a policy. [53] learns a finite-state representation of a recurrent neural network policy with continuous memory and continuous observations to improve interpretability. Work in explaining classical planning algorithms such as visualizing the search tree of a planning problem [65] supports debugging heuristic functions and semantic errors in planning domain specifications.

Model-based explanations are typically algorithm-agnostic, and focus on explaining properties of a solution that can be evaluated independently of the algorithms used to solve it given the decision making model. These explanations are in general more appropriate for end-users than algorithm-based explanations. Model-based explanations aim at addressing the users' limited computational power compared to that of the planning agent (referred to as *inference reconciliation*), or at addressing the differences between the users' mental models of the task and the agent's decision making models (referred to as *model reconciliation*). Inference reconciliation approaches generally aim to answer questions such as why a particular action is in a plan [2, 11, 17, 45, 46, 84, 99, 102, 103], why a different plan is not used [15, 30, 54, 73, 97], why a plan is optimal [27, 44, 50, 94], or why a problem is not solvable [31, 32, 33, 38, 91]. Model reconciliation paradigm [16, 89, 90, 91, 104], first proposed by [16], argues that explanations must not only be a "soliloquy" in the agent's own model. Instead, the process of explanations must consider the differences between the users' mental models and the agent's model, and reconcile them in order for both to agree on some property of the decisions made (e.g., that the decisions are optimal).

Plan-based explanations are concerned with a variety of problems related to interpreting or describing plans and policies. For instance, [68, 88] have explored approaches for interpreting plans, or observed behaviors of the agent, to infer the goals, beliefs, and intentions of the agent. [80, 92, 95] have contributed approaches for plan or policy summarization using various abstraction schemes. [41] enables the agent to synthesize policy descriptions and responses to both general and targeted queries by the human collaborators. Works such as [47, 85] infer specifications that describe temporal properties of plan traces, and temporal differences between different sets of plan traces [52]. Inferring properties of plans or policies can help the users in various ways, such as analyzing the correctness of plans, categorizing plans, or detecting anomalies. Plan-based explanations are complementary to our main goal and contributions in this thesis.

Algorithm-based, model-based, and plan-based explanations are complementary approaches that focus on different facets of explainable AI planning, and target different use cases of explanations. In this thesis, we investigate approaches for model-based explanations, primarily for the purpose of inference reconciliation. We aim to explain to the users the reasoning that gives rise to planning solutions that may not be apparent to them, based on *what* the agent's decision making model does rather than *how* the underlying planning algorithm achieves it. Many works in inference reconciliation, as well as in the broader XAIP, include some form of *contrastive* property in the explanations. The idea of contrastive explanations is rooted in the social phenomenon that, when people ask for an explanation of an event, i.e., fact ("Why did P happen?"), they often are asking for an explanation relative to some contrast case, i.e., foil ("Why did P happen rather than Q?") [69, 70]. Different paradigms of contrastive explanations have been explored in philosophy and psychology – and those have inspired approaches for explainable AI at large [93]. Notably, *causal contrastive explanations* use the notion of the "difference condition": a causal difference between P and not-Q, consisting of a cause of P and the absence of a corresponding event in the history of not-Q [62]. On the other hand, *non-causal contrastive explanations* address the physical nature of a modeled system. They can be used to explain the properties and relations inherent to such systems. For instance, [20] answers the questions such as "What dispositions of p are relevant to circumstances x as opposed to y?", where p is the object whose traits require an explanation and x and y are the circumstances determined by the question dependent context. Hybrid approaches of causal and non-causal contrastive explanations have also been proposed. For instance, [21] and [22] provide a unified theoretical framework for causal and non-causal contrastive explanation for category learning.

Many existing works in contrastive explanations for inference reconciliation focus on a contrastive foil at an action level, and they are often causally oriented. Among these, contrastive explanation techniques have been used to answer the question *"Why is this action in the plan?"* as well as *"Why not this other action?"* [36]. To answer the former question, a contrastive foil is determined by the explanation approach itself. For instance, works such as [102] explain why an agent chooses a particular action in its plan in the context of a specific execution trace. The explanation is causal contrastive in that its answer is a prior point in the execution trace where there was a choice, and making a different choice (i.e., taking an alternative action at a prior state) would not have led to the queried action being chosen. On the other hand, some works such as [15, 54] aim to answer the latter question. They consider an explicit contrastive foil specified by the user as constraints on the plan the user is expecting. Particularly, these constraints are a specific action or sequence of actions to be included in the plan. The explanation approach proposed in these works is generally to identify an exemplary plan that contains the specified actions to contrast with the computed plan.

Another important type of questions that inference reconciliation should answer is why the agent's chosen plan or policy is optimal. For many planning application domains, often times there are multiple ways of achieving a goal that may differ in some important qualities, such as execution time and various measures of performance and costs. In this thesis, we argue that contrastive explanations are essential for answering why a plan or a policy is optimal, as optimality is fundamentally a comparative property. Although, the requirement for contrastive explanations in this setting is different from those for answering why-this-action and why-not-this-other-action questions. The former question requires a contrastive foil to be the exclusion of the particular action from the agent's plan. The latter question requires a contrastive foil to be the queried alternative action. To explain a policy's optimality, it is unclear what the appropriate contrastive foil(s) should be. As there can potentially be an infinite number of suboptimal policies in a given planning problem, the key question is how to identify contrastive foils that can provide sufficient insights for the user to understand why a policy is optimal.

For Markov decision process planning and reinforcement learning, there are some existing works in inference reconciliation for explaining why a policy is optimal. Some are causally oriented in terms of showing how an action in an optimal policy allows for the execution of more desirable actions later [27]. However, such approaches lack a contrative property in the explanations to address why the policy is in fact optimal. Some other works in this area do have contrastive aspects in their explanation approaches, where the contrastive foils are framed as individual alternative actions in each state. For instance, [50] explains an optimal policy in terms of the frequency with which the action would lead the agent to high-value states, and that the particular action has the highest expected frequency among all applicable actions. Similarly, [44] explains why an action in an optimal policy is preferred over another action based on how the actions affect the total value in terms of various human-understandable components of the reward function.

In this thesis, we argue that there are shortcomings in framing contrastive foils at an individual action level. For the user to understand why a policy is optimal, knowing that choosing an alternate action at a state would have a lower value (i.e., $Q(s, a')$ value, or its decomposed components, for an alternate action a' at at state s) may not be sufficient. The lower value can be the result of the subsequent states and actions that are not explained to the user. Moreover, there are many possible sets of subsequent states and actions that the agent could choose that would result in different policy values. These alternate decisions are not addressed if the explanation frames contrastive foils as individual actions. Instead, we argue that combinations of actions that lead to lower policy values should be explored, in order to provide the user more insights into why a policy is optimal. The key challenge addressed in this thesis is to identify the appropriate combinations of actions as contrastive foils from an exponentially large space.

Chapter 3

Explainable Planning for Multi-Objective Markov Decision Processes

As automated decision-making and autonomous systems are increasingly used in many aspects of people's lives, it is essential that people have trust in them – that the systems are making the right decisions and doing the right things. Gaining such trust requires people to understand, at the appropriate levels of abstractions, why the automated agents or systems made the decisions that they did [36, 40].

Real-world decision making and planning often involves multiple objectives and uncertainty [66, 78, 101]. Since competing objectives are possible, or even inherent in some domains, an essential part of the reasoning behind multi-objective planning decisions is the tradeoff rationale. For the end-users to understand why an automated agent makes certain decisions in a multi-objective domain, they would need to be able to recognize when a situation involves tradeoffs, understand how the available actions can impact the different objectives, and understand the specific tradeoffs made by the agent. This is clearly challenging for end-users when there are a large number of possible, sequential decision choices to consider. Particularly, it can often be difficult for the users to evaluate the full, long-term consequences of each action choice in terms of the domain-specific concerns of the problem, and to be aware of the interactions among the different concerns. As a result, the users may not understand why the agent made the decisions that it did. This may potentially undermine the users' trust in the agent.

In this work, we address the challenge of helping end-users understand multi-objective planning rationale by focusing on one of its essential parts: the tradeoff rationale. We propose an approach for explaining multi-objective Markov decision process (MDP) planning that enables the planning agent to explain its decisions to the user, in a way that communicates the agent's tradeoff reasoning in terms of domain-level concepts. The main idea of our approach is to detect and explain whether a situation involves competing objectives, and to explain why the agent selected its planning solution based on the objective values. In instances where there are competing objectives, our approach explains the agent's decisions by explaining how it makes tradeoffs: by contrasting the (expected) consequences of its solution to those of a selected set of rational (i.e., non-dominated) alternatives with varying preferences.

In this chapter, our contributions are the following:

(1) We design an explainable planning approach for multi-objective planning, formalized as a Markov decision process (MDP) with linear scalarization of multiple cost functions. Our approach consists of two parts. (i) A modeling approach that extends MDP planning representation to ground the cost functions in the domain-specific human-interpretable concepts, to facilitate explanation generation. (ii) A method for generating *consequence-oriented contrastive explanations* for a MDP policy, which explain the planning agent's reasoning in terms of the policy's expected consequences on the task objectives, put in the context of other rational decisions that the agent did not make, due to its (a priori) preference.

(2) We demonstrate the applicability of our proposed explainable planning framework by applying our approach to three planning problem domains: indoor mobile robot navigation, Unmanned Aerial Vehicles (UAV) mission planning, and outpatient clinic scheduling. We demonstrate how each problem domain can be modeled in our explainable planning representation, and we discuss examples of explanations generated by our approach for different problem instances.

3.1 Overview of Approach

The goal of our approach is to enable automatic explanation of multi-objective planning decisions, with the focus on explaining the tradeoff rationale. In this paper, we refer to the optimization concerns of a given planning problem as *quality attributes (QAs)* [48] (e.g., execution time, energy consumption, etc.). We focus on MDP planning with linear scalarization of multiple objective functions. Our approach supports both the expected total-cost and the long-run average cost criteria of MDP planning.

The main idea of our approach is to explain the agent's planning solution (i.e., an optimal policy with respect to the agent's overall cost function) in the context of other rational alternative solutions that the agent did not select. Particularly, our approach detects if there are competing objectives in a given problem instance. If so, our explanation approach aims to *quantitatively illustrate how the agent makes tradeoffs*, by contrasting the (expected) consequences of the agent's solution to those of selected Pareto-optimal alternatives with varying preferences for the competing objectives. Otherwise, the explanation indicates that the agent's planning solution is optimal with respect to all objectives.

3.1.1 Motivating Example

We will use the following example to discuss our approach. Figure 3.1 shows a mobile robot whose task is to drive from its current location to a goal location in a building [3]. The robot has to arrive at the goal as soon as possible, while trying to avoid collisions for its own safety and to avoid driving intrusively through human-occupied areas. The robot has access to the building's map (locations, and connections and distances between them), placement and density of obstacles (sparse or dense obstacles), and the kinds of areas in the environment (public, private, and semi-private areas). The robot can determine its current location, knows its current driving-speed setting, and can detect when it bumps into obstacles. The robot can also navigate between two adjacent locations, and change its driving speed between two settings: full- and half-speed.

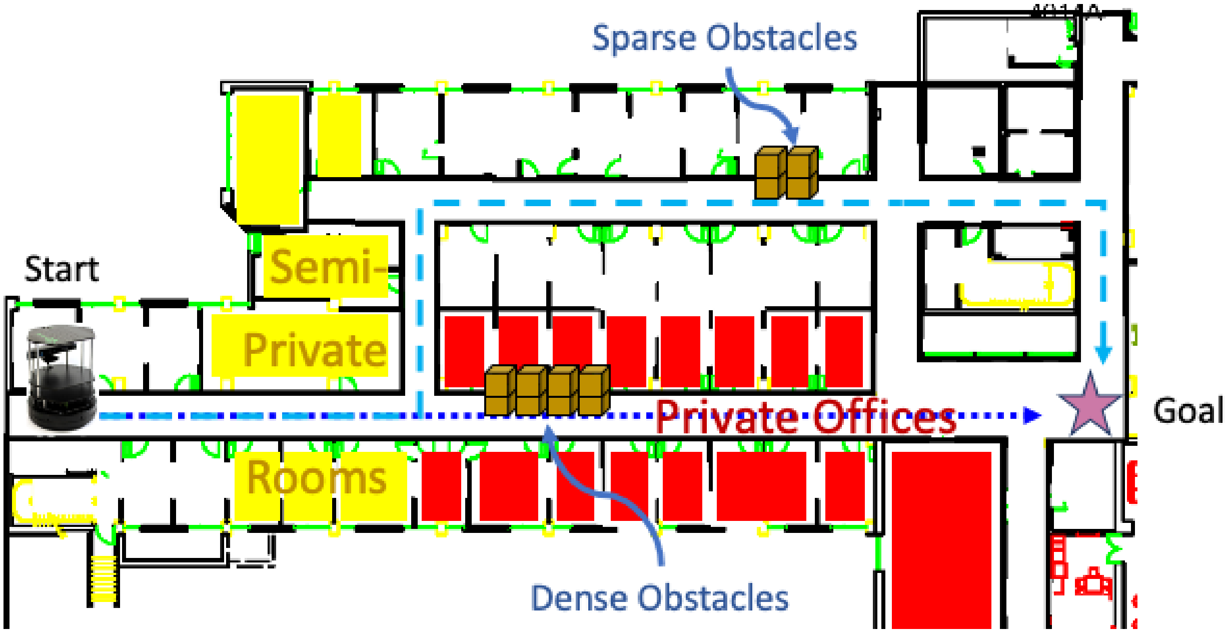


Figure 3.1: Indoor robot navigation example: The dotted line and the dashed line indicate 2 potential navigation paths. The dotted path has shorter travel time, while the dashed path has lower intrusiveness and lower chance of obstacle collision.

When the robot drives at its full speed through a path segment that has obstacles, it has some probability of colliding, depending on the density of the obstacles: dense obstacles yield higher probability of colliding than sparse obstacles. Furthermore, the intrusiveness of the robot is based on the kinds of areas the robot travels through. The robot is non-intrusive, somewhat intrusive, or very intrusive when it is in a public, semi-private, or private area, respectively.

Figure 3.1 shows an example of a tradeoff reasoning the robot might have to make. Suppose the robot computes the dashed path as its solution. The robot could provide its rationale by showing the alternative dotted path that is more direct, and explaining that while the dashed path takes longer travel time than the direct dotted path, it has lower intrusiveness and lower chance of collision. This type of explanations allow the users to better understand the tradeoffs that the robot makes, and to determine whether such tradeoffs align well with their own preferences.

3.1.2 Explainable Planning Framework

We create a framework for explainable MDP-based planning. Our framework allows the users (i.e., developers of planning models) to specify their planning domain models, including the quality attribute models, in our explainable planning representation, which we refer to as *eXplainable MDP* (XMDP) representation. The framework also allows the users to create a custom visualization for their planning domain, which includes how various features of a problem instance are visualized and how a solution policy is presented.

Given a planning problem specified in our XMDP representation, our framework translates it into an equivalent MDP problem in a standard MDP representation. This standard MDP prob-

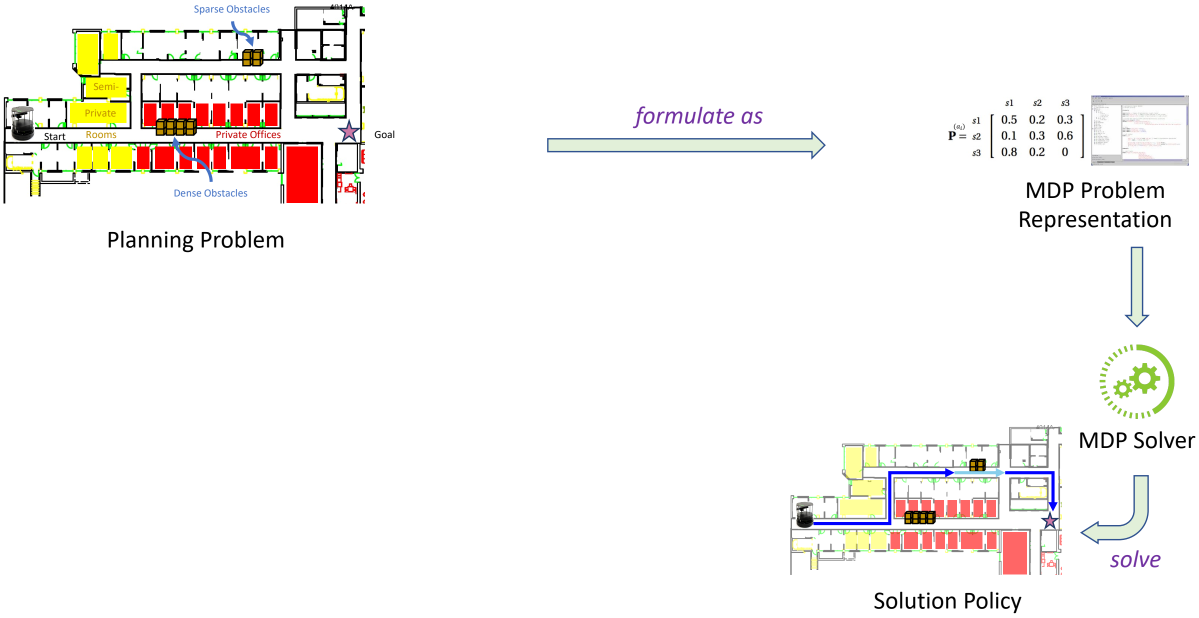

Sparse Obstacles
Semi-Private Rooms
Private Offices
Start
Goal
Dense Obstacles
Planning Problem
formulate as
$\mathbf{P}^{(a_i)} = \begin{array}{c|ccc} & s1 & s2 & s3 \\ s1 & 0.5 & 0.2 & 0.3 \\ s2 & 0.1 & 0.3 & 0.6 \\ s3 & 0.8 & 0.2 & 0 \end{array}$
MDP Problem Representation
MDP Solver
solve
Solution Policy

Figure 3.2: Traditional planning workflow: Domain designer formulates a planning problem in a standard MDP representation and uses an MDP solver to compute an optimal solution policy.

lem can be solved by any optimal MDP algorithm. Once an optimal policy is generated by the planning algorithm, our framework generates verbal and visual explanations for that policy, leveraging the XMDP representation and the custom visualization. The explanations describe the consequences of the solution policy in terms of the quality attributes, and explain whether the planning decisions had to reconcile any competing objectives and how. The latter is in the form of contrastive explanations.

See Figures 3.2 and 3.3 for a comparison between the traditional planning workflow and our proposed explainable planning workflow. In the traditional planning workflow, the domain designer formulates a planning problem in a standard MDP representation, and uses any existing MDP solution method (e.g., value iteration, policy iteration, linear programming) to compute an optimal policy. In our proposed explainable planning workflow, the domain designer formulates a planning problem in our XMDP representation. This intermediate representation is automatically translated to the standard MDP representation, which is then solved by the MDP solver to compute an optimal policy. Then, our explanation generator takes as inputs the XMDP representation and the computed optimal policy to generate consequence-oriented contrastive explanations.

Next, we discuss the details of our explainable planning approach. Our approach consists of two parts. First is XMDP representation, the extended planning representation of MDPs that grounds the cost functions in the quality-attribute semantics of the problem domain, which enables human-interpretable explanation (Section 3.3). Second is the method for generating consequence-oriented contrastive explanations for optimal MDP policies (Section 3.4 through Section 3.5).

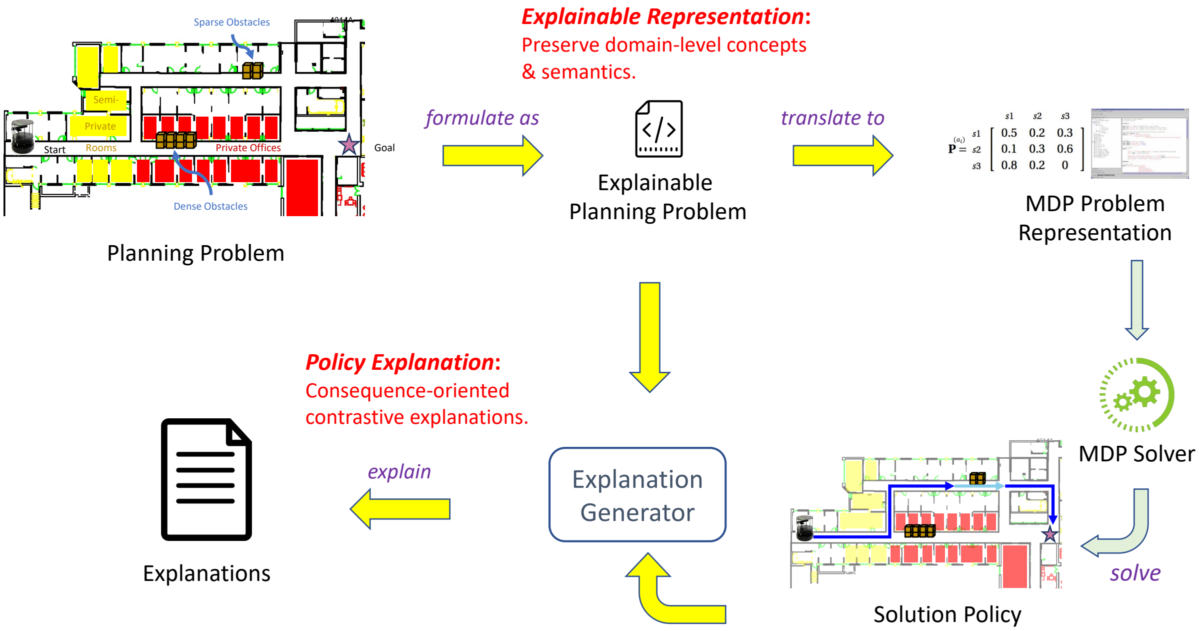


Figure 3.3: Our proposed explainable planning workflow: Domain designer formulates a planning problem in our augmented MDP representation. This intermediate representation is automatically translated to the standard MDP representation, which is then solved by the MDP solver to compute an optimal policy. Then, our explanation generator takes as inputs the augmented MDP representation and the computed optimal policy to generate consequence-oriented contrastive explanations.

3.2 Background

3.2.1 Markov Decision Processes

A *Markov decision process* (MDP) [76] is a tuple $\langle S, A, P, R\rangle$, where S and A are a finite set of states and actions, respectively; $P : S \times A \times S \rightarrow [0,1]$ is the transition probability function; and $R : S \times A \rightarrow \mathbb{R}$ is the immediate reward function.

A *deterministic stationary policy* $\pi : S \rightarrow A$ defines the action that the agent takes in each state. The goal of solving a MDP is to find a policy π that maximizes the *value function* $V_\pi(s)$, which is defined using the discounted reward criterion with an infinite horizon:

$$V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r_t | s_0 = s]$$

with γ as the discount factor $0 \leq \gamma < 1$, r_t as the reward obtained at time t, and s_0 as the initial state. The action-value function $Q_\pi(s,a)$ is defined as:

$$Q_\pi(s,a) = \sum_{s' \in S} P(s'|s,a)(R(s',a) + \gamma V_\pi(s'))$$

The optimal value function V^* and an optimal policy π^* are defined by $V^* = V_{\pi^*}$ and for any policy π and any state s: $V^*(s) \geq V_\pi(s)$. Given known reward function R and transition function P, solution methods based on Dynamic Programming, such as *Value Iteration* and *Policy Iteration*, can be used to solve for an optimal policy.

In a major part of this thesis, we use MDP with *costs* as a planning framework: $\langle S, A, P, C\rangle$, where $C : S \times A \rightarrow \mathbb{R}_{\geq 0}$ is the immediate cost function. We use costs instead of rewards because the application domains investigated in this thesis mainly involve minimization objectives. For a MDP with costs, we use the $J_\cdot(s)$ to denote the value function. To solve a MDP with costs, we consider two classical types of cost criteria: the expected total *cost-to-go* (cumulative cost until reaching an absorbing goal state), and the *long-run average cost* (average cost per unit time over the infinite time horizon) [76]. Both cost criteria capture the fundamental objectives of the broad set of planning applications. The value function for the cost-to-go criterion is:

$$J_\pi^{\text{to-go}}(s) = C(s, \pi(s)) + \sum_{s'} P(s'|s, \pi(s)) J_\pi^{\text{to-go}}(s') \tag{3.1}$$

The value function for the long-run average cost criterion is:

$$J_\pi^{\text{average}}(s) = \lim_{N \rightarrow \infty} \frac{1}{N} E[\sum_{t=0}^{N} C(s_t, \pi(s_t)) | s_0 = s] \tag{3.2}$$

Solving a MDP with costs is to find a policy π that minimizes the value function $J_\pi(s)$. The same solution methods for solving MDPs with rewards apply.

3.3 Explainable Planning Representation

We augment the representation constructs of a factored MDP to preserve the underlying semantics of the quality attributes (QAs) of a problem domain. In this work, we refer to it as an *eXplainable* representation, or XMDP. Our approach requires that the designer of an explainable agent specify the planning problem in this representation.

A standard factored MDP is a tuple of the form: $\langle S, A, P, C \rangle$, where S denotes the set of states described via a set of variables $S_1, ..., S_n$; A denotes the set of actions; P denotes the probabilistic transition function; and C denotes the cost function, which is a composition of the sub-costs of the QAs. Our approach extends the standard factored MDP constructs to explicitly represent: (i) the *attributes* of state variables and actions that impact the values of the QAs of a planning problem, and (ii) the *analytic models* of those QAs. For instance, in the mobile robot navigation example in Section 3.1.1, an area type, which can be a public, semi-private, or private space, is an attribute of a state variable representing the robot's current location. The area type of each location visited by the robot's navigation policy determines the intrusiveness level. The domain designer of the planning problem must specify the QA analytic models using XMDP representation.

In addition, our approach requires a mapping from the user-defined types of QAs (e.g., travel time, intrusiveness) and their determining factors (e.g., distance, area type) in the explainable representation to the domain-level vocabulary of the corresponding concepts. This vocabulary will be used to generate explanations.

3.3.1 Quality-Attribute Determining Factors

State variables and actions in our explainable representation are typed, and each type defines a set of attributes associated with instances of that type. For instance, the example in Section 3.1.1 has a state-variable type Location, which has a value ID representing the unique ID of a particular location, and an attribute Area representing the type of area that location is in (e.g., public, semi-private, or private area). Another state-variable type RobotSpeed has a floating point value from a finite set of possible speed settings (e.g., full speed at 0.7 m/s and half speed at 0.35 m/s).

An action type MoveTo, parameterized by a Location argument denoting the destination, has two attributes: Distance and Obstacle, representing the distance and obstacle density (e.g., none, sparse obstacles, or dense obstacles) of the path between the source and the target locations of the action. The attribute Area of a Location state variable has an impact on the intrusiveness of the robot, and the attributes Distance and Obstacle of a MoveTo action, together with the state variable RobotSpeed, have an impact on the travel time and the safety concerns of the robot, respectively.

In Section 3.6, we discuss in further details the explainable representation constructs via modeling examples of planning application domains.

3.3.2 Quality-Attribute Analytic Models

A QA i is characterized by a function $QA_i : S \times A \rightarrow \mathbb{R}_{\geq 0}$ that calculates the expected quality attribute value for a single decision epoch, that is, the expected value incurs by taking a particular action in a particular state. Since the state variables and actions are typed, each QA_i is also a function of the attributes associated with the state variables and actions. This maintains explicit measurement models of the QAs in a compact representation.

We consider two types of cost criteria for MDPs: the expected total cost-to-go criterion (when there are absorbing goal states), and the long-run average cost criterion. For the expected total cost-to-go criterion, the total value of QA i of a policy π starting from a state s is characterized by the value function:

$$J_{\pi,i}^{\text{to-go}}(s) = QA_i(s, \pi(s)) + \sum_{s'} P(s'|s, \pi(s)) J_{\pi,i}^{\text{to-go}}(s'). \tag{3.3}$$

For the long-run average cost criterion, the average value of QA i of a policy π starting from a state s is characterized by the value function:

$$J_{\pi,i}^{\text{average}}(s) = \lim_{N \rightarrow \infty} \frac{1}{N} \mathbb{E} \left\{ \sum_{t=1}^{N} QA_i(s_t, \pi(s_t)) \middle| s \right\}. \tag{3.4}$$

For either type of the cost criteria, we consider a multi-objective cost function C of an MDP as a linear scalarization of all QA_i functions:

$$C(s, a) = \sum_i k_i QA_i(s, a), \tag{3.5}$$

where $k_i > 0$ parameters capture the preference over multiple concerns.

There are different kinds of QAs based on how they are quantified. Thus, they shall be explained differently. Our explainable representation has explicit constructs for defining the following types of QAs:

Counts of Events QAs of this type are quantified as expected numbers of occurrences of events of interest, where the objectives are to minimize occurrences of "bad" events. The robot colliding with obstacles is an example of such event, where the safety objective of planning is to minimize the expected number of collisions.

We consider a general form of an event to be a transition (s, a, s'). An event e is defined by a boolean predicate over the source- and sink-state variables and the action of a transition, including the attributes associated with those state variables and action. For instance, a collision event $e_{collide}$ is defined by a predicate over the speed setting of the robot in the source state, the bump sensor state in the sink state, and a constant representing a threshold of safe speed. A collision event occurs when the robot is driving (MoveTo action) faster than the safe speed threshold when it bumps into obstacles: $rSpeed > \theta_{\text{safeSpeed}} \wedge rBumped' = True$.

To define a count-of-events QA in the explainable representation, the specification writer must provide a definition of an event e as discussed above. Our framework then creates the corresponding QA function QA_e that calculates the probability of the event e occurring from taking a particular action in a particular state, according to the domain's transition model. For instance, suppose that the robot has a 0.0, 0.2, and 0.4 probability of bumping into obstacles when it drives at a speed over $\theta_{\text{safeSpeed}}$ through a segment classified as obstacle free, sparse obstacles, and dense obstacles, respectively. The $QA_{collide}$ computes the probability of collision for a given (s, a) according to the obstacle density between the source and the sink locations of the MoveTo action.

The value function $J^{\pi}_{e}(s)$ characterizes the expected total occurrences (or the long-run average occurrences, depending on the problem's cost criterion) of event e of a policy π starting in a state s. Currently, we exclude events that have duration (i.e., span across multiple transitions) from the definition.

Standard Measurements QAs of this type can be measured in a standard, scientific way, either directly or by deriving from other measurements. Such QAs are quantified by their magnitudes (e.g., duration, length, rate of change, utilization, etc.), typically measured in real values. The robot's travel time is an example of such measurement. Note that these values must be additive, i.e., the total value of multiple steps in a policy equals the sum of values of the individual steps, to be appropriate for the MDP cost criteria we use.

A standard-measurement QA model characterizes the value incurs over a single transition (s, a, s'). To define a standard-measurement QA, the specification writer provides an evaluation function f, which is a function of the source- and sink-state variables and the action of a transition, including their attributes. Our framework then creates the corresponding QA function QA_f that calculates the expected value of f from taking a particular action in a particular state, according to the domain's transition model. For instance, the robot's travel time is a function of its speed and the distance between the source and the sink locations of the MoveTo action, with a potential additional delay depending on the obstacle density along the path. Namely, the travel time is $\mathsf{Distance}/\mathit{rSpeed} \cdot \mathit{delayRate}(\mathsf{Obstacle})$.

The value function $J^{\pi}_{f}(s)$ characterizes the expected total magnitude (or the long-run average magnitude, depending on the problem's cost criterion) of value f (e.g., total travel time) of a policy π starting in a state s.

Non-Standard Measurements Some properties do not have associated standard measurements, or their exact standard measurements may not be available to the modelers. Nonetheless, given the domain knowledge, a carefully-defined arbitrary measurement can be used to characterize policies with respect to a particular property of concern. The robot's intrusiveness to the building occupants is an example of such measurement, where penalties can be assigned to indicate the degrees of intrusiveness.

A non-standard-measurement QA model in our explainable representation characterizes the level or degree of a particular property of a single transition (s, a, s'). To define a non-standard-measurement QA, the specification writer first defines a set of events E_m that characterize the different levels or degrees of a property m of concern. Then, the specification writer provides a mapping that assigns numerical values to the events in E_m, representing the costs or penalties of the events. Our framework then creates the corresponding QA function QA_m that calculates the expected "m-penalty" from taking a particular action in a particular state, according to the domain's transition model. For instance, the intrusiveness concern can be characterized by the types of areas the robot enters. The events in the set $E_{intrusive}$ are: the robot moving into the public, semi-private, and private areas of the building. The intrusiveness penalties can be $0, 1,$ and 3 for these events, respectively.

An expected sum of non-standard measurement values representing an abstract property does not have a well-defined meaning. Unlike for standard measurements, it is not intelligible to communicate a value such as $J^{\pi}_{intrusive}(s)$ in an explanation. Instead, we communicate the value $J^{\pi}_{m}(s)$ of a policy π by communicating the distribution of different non-standard measurement values throughout the policy. To this end, we calculate the expected count of each event in E_m occurring in the policy. We use qualitative terms (e.g., *"not intrusive"*, *"somewhat intrusive"*, and *"very intrusive"*) to describe how often different severity levels of m incur in the policy.

The next section discusses our approach to generate policy explanations, leveraging the explainable planning representation.

3.4 Policy Explanation

To explain an optimal MDP policy, we propose using *quality-attribute-based contrastive explanations*. For a given planning problem, the explanations: (i) describe the QA objectives and the expected consequences of the policy in terms of the QA values, and how those values contribute to the overall expected cost of the policy, and (ii) explains whether (a) the policy achieves the best possible values for all QAs simultaneously, or (b) there are competing objectives in this problem instance and explains the tradeoffs made to reconcile them, by contrasting the optimal policy to selected Pareto-optimal alternatives. In our explanations, we present the MDP policies – the optimal policy generated by the planner and the alternative policies discussed in the contrastive explanations – to the user via a custom graphical illustration.[1]

3.4.1 Policy Consequences

The first part of explaining an optimal MDP policy is informing the user of the QA objectives of the planning and the expected consequences of the policy with respect to those QAs. We present an optimal MDP policy, along with the factors that determine its QA values (see Section 3.3.1), via a graphical illustration that employs domain-specific visual components. We compute the expected QA consequences of the policy as discussed in Section 3.3.2. Specifically, to compute the expected consequence of a policy π on the QA i, with an initial state s_0, our approach computes the expected total value-to-go $J^{\text{to-go}}_{\pi,i}(s_0)$, or the long-run average value $J^{\text{average}}_{\pi,i}(s_0)$, of the QA i by computing the value functions in the Equations 3.1 and 3.2, respectively. Both value functions are computed by solving a system of linear equations [76]. In addition, our approach also computes, in a similar manner as above, the expected total (or long-run average) *costs* the policy π incurs for the particular QA values. These costs are the components $k_i QA_i$ for all quality attributes i that make up the multi-objective cost incurred each decision epoch – see Equation 3.5.

[1] A general approach for policy summarization is outside the scope of this work.

Type of Quality Attribute	Optimization Objective	Quality Attribute Value
Count of events	"*minimize the expected number of* collisions"	"*the expected number of* collisions *is* 0.8"
Standard measurement	"*minimize the expected* travel time"	"*the expected* travel time *is* 10 minutes"
Non-standard measurement	"*minimize the expected* intrusiveness"	"*the robot is expected to be* non-intrusive *at* 5 locations *and* somewhat intrusive *at* 2 locations"

Table 3.1: Examples of explanations of quality-attribute objectives and their values.

To generate textual explanations of the planning objectives and the QA values of the solution policy, we use predefined natural-language templates. Recall that different types of QAs should be explained differently. Table 3.1 shows examples of explanations of QA objectives and values. The non-italicized terms are originally placeholders in the templates. These placeholders are then filled with either terms from the domain-specific vocabulary provided in a planning problem definition, or the QA values of the solution policy (i.e., $J_{\pi,i}(s_0)$ for each QA i and initial state s_0) computed by our framework. The domain designer must instantiate the vocabulary in our framework, by specifying, for each quality attribute QA i, the associated noun (e.g., "travel time", "intrusiveness"), verb (e.g., "take", "be"), categorical values if applicable (e.g, "non-intrusive", "somewhat-intrusive", "very-intrusive"), preposition if applicable (e.g., "at"), and unit (e.g., "seconds", "locations").

3.4.2 Contrastive Explanations of Tradeoffs

The second part of explaining an optimal MDP policy is providing the tradeoff rationale of the decisions, if any, via contrastive explanations. Our contrastive explanation approach uses a selected subset of Pareto-optimal policies as rational alternatives to the solution policy, since they reflect the possible tradeoffs among competing optimization objectives. By contrasting the solution policy to each selected Pareto-optimal alternative, we illustrate the gains and losses with respect to the QAs by choosing one policy over the other. This quantitatively explains the QA-tradeoff that the solution policy makes.

We use a predefined natural-language template for generating a verbal explanation:

"I could [(1) improve these QAs by these amounts], *by* [(2) carrying out this alternative policy] *instead. However, this would* [(3) worsen these other QAs by these amounts]. *I decided not to do that because* [(4) the improvement in these QAs] *is not worth* [(5) the deterioration in these other QAs].

The statement fragments (1) and (3) contrast the QA values of the solution policy to those of an alternative policy, by describing the gains and the losses quantitatively. (2) describes the alternative policy using a graphical illustration, as discussed in Section 3.4.1. (4) and (5) restate the gains and the losses qualitatively, as part of the agent's reasoning to reject the alternative. For instance, the agent rejects an alternative navigation that is 5 minutes shorter than its solution because its expected number of collisions would have been 0.4 higher.

On the other hand, if there are QA objectives that are not in conflict with any other objectives, the explanation simply indicates that those QA values of the solution policy are already the best possible values. For instance, the agent's solution is already the least intrusive navigation route.

In addition, our explanation framework provides an option to relate the QA values of each policy to their contributions to the overall cost of the policy. Essentially, it decomposes the overall cost of the policy into sub-costs of different QA concerns. For instance, suppose the mobile robot's navigation policy has a 5-minute expected travel time and 0.2 expected collision, and suppose the overall cost of the policy is 100. An explanation indicates, for instance, that the 5-minute expected travel time, and the 0.2 expected collision, contributes to 70/100, and 30/100, of the overall cost of the policy, respectively. Section 3.4.1 discusses how we compute these sub-costs of a policy. This additional information can further explains how the planning agent values each QA objective, and it may potentially assist the reward design problem.

The next section discusses the specific approach to generate alternative policies.

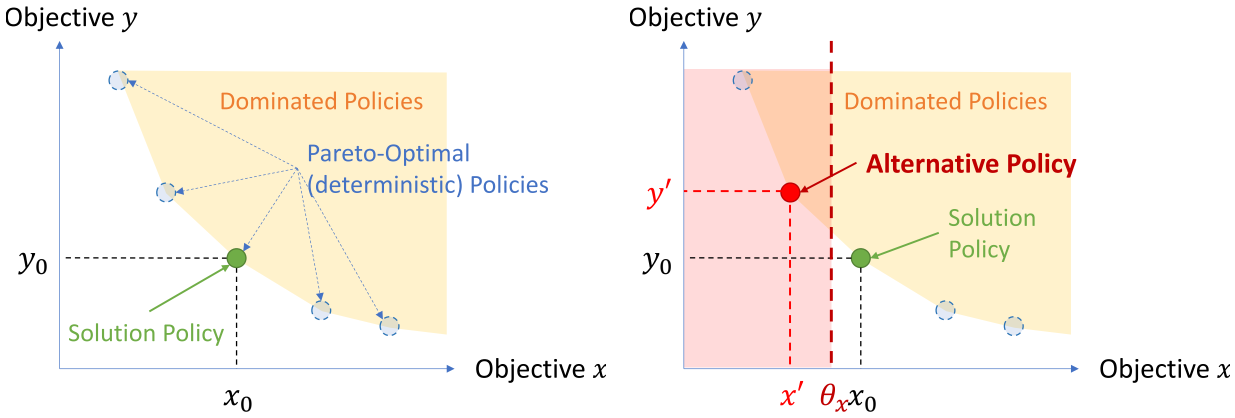
3.5 Alternatives Selection for Contrastive Explanations

To generate contrastive explanations, our approach obtains alternative policies by finding a Pareto optimal policy that improves each of the QAs of the solution policy, if one exists. Let θ_i be a value of the QA i that has δ_i magnitude improvement from the value of the solution policy (we will discuss more on δ_i in Section 3.5.2). We use θ_i as an upper-bound constraint on the QA i value for finding a Pareto-optimal, i-improving alternative policy. Intuitively, such a Pareto-optimal, i-improving alternative policy illustrates an *inflection point* in the planning agent's optimization and tradeoff decisions. Without loss of generality, the agent is trying to minimize the objective i as much as possible. However, there will be an inflection point where the agent will stop doing, as the low value of the object i will cause the value of the other objective $j \neq i$ to be too high for the agent's preference.

Specifically, let $\mathcal{M} = \langle S, A, P, C \rangle$ denotes the original MDP problem, where $C(\cdot, \cdot) = \sum_j k_j C_j(\cdot, \cdot)$ is a multi-objective cost function, and each C_j denotes a linear transformation of QA j function for the purpose of normalization. To compute each Pareto-optimal, i-improving alternative to the solution policy of $\mathcal{M}$, we solve another MDP $\mathcal{M}^{(i)} = \langle S, A, P, C^{(i)} \rangle$, where $C^{(i)}(\cdot, \cdot) = \sum_{j \neq i} k_j C_j(\cdot, \cdot) + k'_i C_i(\cdot, \cdot)$, where k'_i is relatively small, with the constraint that the expected QA i value of a solution of $\mathcal{M}^{(i)}$ must be $\leq \theta_i$. Note that keeping $C_i(\cdot, \cdot)$ in the objective function ensures that a solution will be Pareto-optimal with respect to all quality attributes including QA i. A relatively small value of k'_i puts an emphasis on optimizing the rest of the quality attributes QA $j \neq i$. Figure 3.4 illustrates the main idea of the approach. The specific constraint formulation depends on the cost criterion; as described below.

3.5.1 Deterministic Alternative Policies

A standard way of solving constrained MDPs via linear programming [4] will yield randomized stationary policies. However, in this work, we use only deterministic stationary policies as alternatives to explain the original solution policy, since: (a) they are more suitable for goal-oriented



(a) Solution policy and Pareto-optimal policies in the 2-dimensional space of objective values.

(b) Policies in the shaded region have better Objective x values than that of the solution policy.

Figure 3.4: Pareto-optimal policies in the shaded region have better Objective x values but worse Objective y values than those of the solution policy. Therefore, a Pareto-optimal policy in the shaded region that locates near the solution policy describes an inflection point in the tradeoff decisions between the competing Objectives x and y.

problems with total-cost criterion, in which decisions would be executed only once, and (b) they are simpler to describe than randomized policies.

To ensure that our approach produces deterministic alternative policies for the explanations, we use the mixed-integer linear program (MILP) formulation adapted from [28], for solving a constrained MDP with the expected total *cost-to-go* and the *long-run average cost* criteria.

Cost-to-Go Criterion

Our approach formulates a constrained MDP with the expected total cost-to-go criterion in Equation 3.6. We combine: (i) the standard approach for formulating a constrained MDP with the expected total cost-to-go criterion (i.e., the expected undiscounted total-cost criterion with goals) as a dual linear program (LP) [10, 96], and (ii) the MILP formulation for constrained MDPs from [28] that ensures deterministic policies.

$$
\begin{array}{rrcll}
\min\limits_{x} & \sum\limits_{s}\sum\limits_{a} x_{sa} C^{(i)}(s,a) & & & \\
\text{s.t.} & out(s) - in(s) & = & 0 & \forall s \in S \setminus (G \cup \{s_0\}) \\
& out(s_0) - in(s_0) & = & 1 & \\
& \sum\limits_{s_g \in G} in(s_g) & = & 1 & \\
& x_{sa} & \geq & 0 & \forall s \in S, a \in A(s) \\
& \sum\limits_{a} \Delta_{sa} & \leq & 1 & \forall s \in S \\
& x_{sa}/X & \leq & \Delta_{sa} & \forall s \in S, a \in A(s) \\
& \sum\limits_{s}\sum\limits_{a} x_{sa} QA_i(s,a) & \leq & \theta_i, &
\end{array}
\tag{3.6}
$$

where:

$$
\begin{array}{rcll}
out(s) & = & \sum\limits_{a} x_{s,a} & \forall s \in S \setminus G \\
in(s) & = & \sum\limits_{s',a} x_{s'a} P(s|s',a) & \forall s \in S
\end{array}
$$

and where: (i) the optimization variables x_{sa} are *occupation measure* of a policy, where x_{sa} represents the expected number of times action a is executed in state s; (ii) $\Delta_{sa} \in \{0,1\}$ are binary variables, and (iii) X is a constant such that $X \geq x_{sa} \forall s, a$, which can be computed in polynomial time using the approach such as in [28].

$in(s)$ and $out(s)$ can be interpreted as a flow entering and leaving of the state s, respectively [96]. The first constraint $out(s) - in(s) = 0$ is the flow conservation principle. Namely, all flows reaching a state s must leave s for all states that are neither the source s_0 nor a sink $s_g \in G$. The second constraint $out(s_0) - in(s_0) = 1$ defines the out-going flow of the source state s_0. The third constraint $\sum_{s_g \in G} in(s_g) = 1$ defines the in-coming flow of the sink states $s_g \in G$. The objective function $\min_x \sum_s \sum_a x_{sa} C^{(i)}(s,a)$ captures the minimization of the total expected cost $C^{(i)}$ to reach the goal (sink) from the initial state (source). The 7th constraint $\sum_s \sum_a x_{sa} QA_i(s,a) \leq \theta_i$ defines the maximum value of the total expected value of QA i.

The 4th, 5th, and 6th contraints in this MILP formulation ensures that for each $s \in S$, $x_{sa} > 0$ for a single $a \in A(s)$ [28]. Once the model is solved, the corresponding deterministic solution policy can be recovered as: $\pi(s) = a$ if $x_{sa} > 0$ for all $s \in S$.

Long-run Average Cost Criterion

For the long-run average cost criterion, our approach formulates a constrained MDP in Equation 3.7. Similar to our approach for the cost-to-go criterion, we adapt the MILP formulation for ensuring deterministic policies from [28], and combine it with the standard approach for formulating MDPs with the long-run average cost criterion [76].

$$
\begin{array}{rrcll}
\min\limits_{x,y} & \multicolumn{4}{l}{\sum\limits_{s}\sum\limits_{a} x_{sa}C^{(i)}(s,a)} \\
\text{s.t.} & out_x(s) - in_x(s) & = & 0 & \forall s \in S \\
& out_x(s) + out_y(s) - in_y(s) & = & \alpha_s & \forall s \in S \\
& x_{sa} & \geq & 0 & \forall s \in S, \\
& & & & a \in A(s) \\
& y_{sa} & \geq & 0 & \forall s \in S, \\
& & & & a \in A(s) \\
& \sum\limits_{a} \Delta_{sa}^{(x)} & \leq & 1 & \forall s \in S \\
& x_{sa}/X & \leq & \Delta_{sa}^{(x)} & \forall s \in S, \\
& & & & a \in A(s) \\
& \sum\limits_{a} \Delta_{sa}^{(y)} & \leq & 1 & \forall s \in S \\
& y_{sa}/Y & \leq & \Delta_{sa}^{(y)} & \forall s \in S, \\
& & & & a \in A(s) \\
& \sum\limits_{s}\sum\limits_{a} x_{sa}QA_i(s,a) & \leq & \theta_i, &
\end{array}
\tag{3.7}
$$

where:

$$
\begin{array}{rcll}
out_x(s) & = & \sum\limits_{a} x_{s,a} & \forall s \in S \\
in_x(s) & = & \sum\limits_{s',a} x_{s'a}P(s|s',a) & \forall s \in S \\
out_y(s) & = & \sum\limits_{a} y_{s,a} & \forall s \in S \\
in_y(s) & = & \sum\limits_{s',a} y_{s'a}P(s|s',a) & \forall s \in S
\end{array}
$$

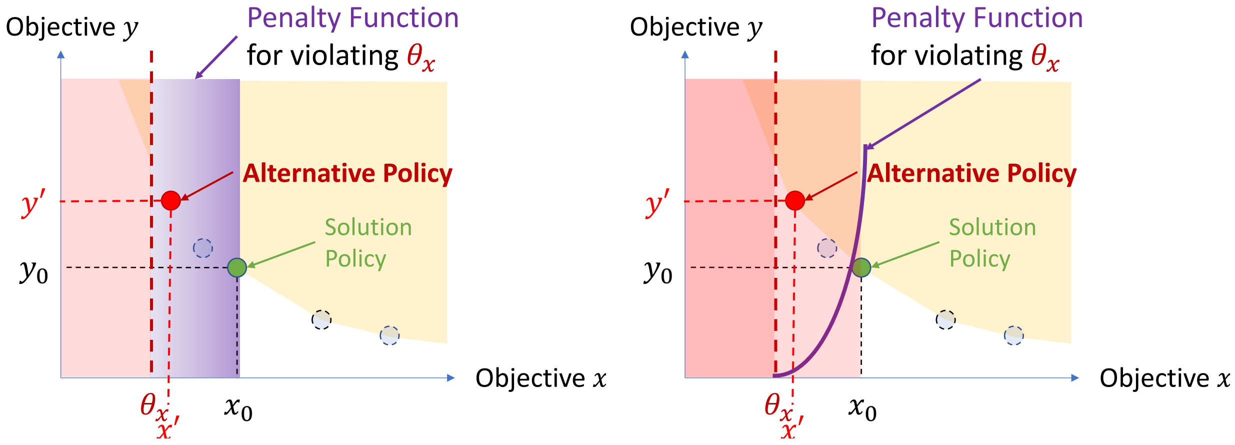
This formulation is applicable to any chain structure of MDPs (unichain or multichain) [76], where: (i) x_{sa} is the occupation measure as defined in Equation. 3.6, and y_{sa} is an additional optimization variables; (ii) α is a distribution of initial states; (iii) $\Delta_{sa}^{(x)}$ and $\Delta_{sa}^{(y)}$ are binary variables corresponding to x and y, respectively; and (iv) X and Y are constants such that $X \geq x_{sa}$ and $Y \geq y_{sa}\, \forall s, a$. We use $X = 1$ and $Y = 1$.

The optimization objective and the first four constraints in Equation 3.7 come from the standard LP formulation for a long-run average cost MDP. Theorems and proofs of the LP formulation can be found in [76] Chapter 9.3. The last constraint defines the maximum value of the total expected value of QA i.

Once the model is solved, the corresponding deterministic solution policy can be recovered as:

$$
\pi(s) = \begin{cases} a & \text{if } x_{sa} > 0 \text{ and } s \in S_x \\ a' & \text{if } y_{sa'} > 0 \text{ and } s \in S \setminus S_x, \end{cases}
$$

where $S_x = \{s \in S : \sum_a x_{sa} > 0\}$ is the set of *recurrent states*, and $S \setminus S_x$ is the set of *transient states* of the Markov chain generated by π [76]. Recurrent and transient states are



(a) Linear penalty function: The darker shade of purple illustrates the higher penalty value for a larger violation of θ_x.

(b) Non-linear penalty function: A separable convex function penalizes a larger violation of θ_x more aggressively.

Figure 3.5: Computing an alternative policy with a soft constraint: An alternative policy must satisfy the hard constraint that its Objective x value must be less than that of the solution policy, x_0. The soft constraint is that its Objective x value *should* be no greater than θ_x, but is allowed to with a penalty function for the violation. The selected alternative policy shown in red has its Objective x value greater than θ_x, but it receives a low penalty as the violation is small.

visited infinitely and finitely often in the Markov chain, respectively. The 5th, 6th, 7th, and 8th constraints in Equation 3.7 ensures that for each recurrent state $s \in S_x$, $x_{sa} > 0$ for a single $a \in A(s)$, and for each transient state $s \in S \setminus S_x$, $y_{sa} > 0$ for a single $a \in A(s)$.

3.5.2 Planning with Soft Constraints

As discussed in Section 3.5, to determine an upper-bound constraint θ_i on the QA i value to solve for a Pareto-optimal, i-improving alternative policy, we need to determine a minimum magnitude δ_i of improvement from the value of the solution policy. The simplest option would be to choose a relatively very small δ_i, so that the resulting alternative policy has the next possible improved value of QA i. However, depending on the planning problem structure, using such approach can yield alternative policies that have QA values that *are perceived by the users to be too similar to those of the solution policy*. This is due to the phenomena that numerical processing in people, as with basic sensory modalities, obeys Weber's law such that the discriminability of two numbers decreases as the magnitude of the numbers increase [64]. A model for this phenomena describe the mental number line as logarithmically compressive, such that as numbers get larger they lie closer to each other [26]. For instance, the users may not perceive a navigation route that has the expected travel time of 1 hour to have a different consequence than an alternative route that has the expected travel time of 1 hour and 2 minutes. Therefore, contrasting policies that have very similar consequences on the QAs may not be effective in helping the users understand the tradeoffs.

To avoid this issue, one may choose a sufficiently large δ_i, determined for a specific quality attribute type, to obtain an alternative policy that is sufficiently different from the solution policy in terms of its QA i value, if one exists. For instance, we may use Weber-Fechner's law to determine a noticeably different value of the QA i. Weber-Fechner's law is described as [77]:

$$P = k \cdot ln \frac{S}{S_0}, \tag{3.8}$$

where P is the magnitude of perception, S is the current magnitude of the stimulus, and S_0 and k are stimulus-specific constants, where S_0 can be interpreted as a stimulus threshold of minimum perceivable magnitude. From this law, given the current QA i value of the solution policy (e.g., the expected total travel time of the navigation route) as S, we can determine the users' perception P of that value. We can then determine the new value S' that yields a significantly different perception P'. Such S' value can be used as a threshold θ_i for the QA i value to obtain an alternative policy.

Due to the logarithmic relationship between the perception and the magnitude of values, θ_i can be much smaller than the current value of the solution policy. There may be no alternative policy that satisfies the threshold. Thus, it is appropriate to use θ_i as a *soft constraint* instead of a hard constraint. For the purpose of communicating tradeoffs, it is not necessary that the QA i value of the resulting alternative policy is strictly less than θ_i, as long as it is *sufficiently less* than that of the solution policy, with θ_i as a guideline. Using the method for handling soft constraints in linear programming presented in [23], we can re-formulate the MILP problems in Equation. 3.6 and Equation. 3.7 to use θ_i as a soft constraint by:

- Adding a penalty term $\phi_i(v_i)$ to the objective of MILP in Equation 3.6 or 3.7, where v_i is a variable for the amount of violation of θ_i and $\phi_i(\cdot)$ is a linear penalty function (Figure 3.5a). v_i is an additional optimization variable in the MILP formulation. For the cost-to-go criterion, the new objective is:

$$\min_{x,v_i} \sum_s \sum_a x_{sa} C^{(i)}(s,a) + \phi_i(v_i) \tag{3.9}$$

For the long-run average cost criterion, the new objective is:

$$\min_{x,y,v_i} \sum_s \sum_a x_{sa} C^{(i)}(s,a) + \phi_i(v_i) \tag{3.10}$$

- Replacing $\sum_s \sum_a x_{sa} QA_i(s,a) \leq \theta_i$ in Equation 3.6 or 3.7 with the following constraints:

$$\begin{array}{rrcl} \sum_s \sum_a x_{sa} QA_i(s,a) & & \leq & D_i \\ \sum_s \sum_a x_{sa} QA_i(s,a) & -v_i & \leq & \theta_i \\ & -v_i & \leq & 0, \end{array} \tag{3.11}$$

where D_i denotes the solution policy's QA i value.

Non-Linear Penalty Function

With a linear penalty function $\phi_i(\cdot)$, the parameter of the function needs to be tuned to balance minimizing the violation of the soft constraint θ_i and minimizing the original objective $\sum_s \sum_a x_{sa} C^{(i)}(s,a)$. Alternatively, we may use a nonlinear convex penalty function such as quadratic or log barrier function, or any separable convex function, to penalize higher amounts of violation v_i more aggressively relative to minimizing $\sum_s \sum_a x_{sa} C^{(i)}(s,a)$. Figure 3.5b illustrates a quadratic penalty function. We use the approach in [25] to handle a nonlinear penalty function $\phi_i(\cdot)$ using piecewise linear approximation.

Suppose $\phi(\cdot)$ is a *separable convex* non-linear penalty function and v is a variable for the amount of violation of the threshold θ. Note that here we omit the subscript i for the QA i to constrain for simplicity in the notation. We obtain a piecewise linear approximation of $\phi(\cdot)$ by sampling m values (referred to as *breakpoints*) of violation $v_1, ..., v_m$, where v_1 and v_m are the lowest and highest values of v. The function is approximated by the linear segments $[(v_i, \phi(v_i)), (v_{i+1}, \phi(v_{i+1}))]$ for $i = 1...m-1$. An approximated penalty function is:

$$\tilde{\phi}(v) = \sum_{i=1}^{m} \alpha_i \phi(v_i) \tag{3.12}$$

where $v_1, ..., v_m$ are the sampled v values, and $\alpha_1, ...\alpha_m \in [0,1]$ are continuous values to be determined. To compute the approximate penalty value for a given amount of violation v, we let α_i be a continuous variable associated with each breakpoint i for $i = 1...m-1$. We let h_i be a binary variable associated with the ith interval $[v_i, v_{i+1}]$ for $i = 1...m-1$, with dummy values $h_0 = h_m = 0$. Then, we solve for h_i and α_i that satisfy the following set of constraints:

$$\begin{aligned} \sum_{i=1}^{m-1} h_i &= 1 \\ \alpha_i &\leq h_{i-1} + h_i \qquad i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i &= 1 \\ v &= \sum_{i=1}^{m} \alpha_i v_i \end{aligned} \tag{3.13}$$

Full details on this formulation are presented in [25].

The approximated penalty function $\tilde{\phi}(\cdot)$ in Equation 3.12 is linear and all of the constraints in Equation 3.13 are mixed-integer linear constraints. Therefore, we can add them to the MILP in Equation 3.6 or 3.7 in order to use an approximated non-linear penalty function for violating the QA threshold θ. Here, we introduce the following additional optimization variables to the MILP: the binary variables h_i and the continuous variables α_i for $i = 1...m$; and the continuous variable v denoting the amount of violation of θ. The re-formulation of the MILP is as follows (note that we again omit the index of the QA to constrain in the formulation for simplicity in the notation):

- We add the approximated penalty term to the objective function. For the cost-to-go criterion, the new objective is:

$$\min_{x,v,\alpha,h} \sum_{s} \sum_{a} x_{sa} C^{(\cdot)}(s,a) + \sum_{i=1}^{m} \alpha_i \phi(v_i) \tag{3.14}$$

  For the long-run average cost criterion, the new objective is:

$$\min_{x,y,v,\alpha,h} \sum_{s} \sum_{a} x_{sa} C^{(\cdot)}(s,a) + \sum_{i=1}^{m} \alpha_i \phi(v_i) \tag{3.15}$$

  Note that we again omit the index of the QA in the formulation for simplicity.

- In addition to the constraints in Equation 3.11, we add the constraints in Equation 3.13 to the set of constraints in Equation 3.6 or 3.7.

In this formulation, the variable v representing the amount of violation of θ is optimized, together with the variables h_i and α_i for $i = 1...m$, to balance the penalty from the violation with the total objective cost $\sum_s \sum_a x_{sa} C^{(\cdot)}(s,a)$ of the policy. The objective cost term is optimized by the variables x_{sa} (and additionally y_{sa} for the long-run averge cost criterion) for all $s \in S, a \in A$.

Selecting an appropriate penalty function $\phi_i(\cdot)$ and value θ_i for the threshold on the QA i can be challenging and is beyond the scope of this paper. Nonetheless, the framework presented here can be further investigated in more specific problem domains and contexts. For instance, θ_i may come from a user's query in a user-guided explanation setting, as the user asks for an explanation of a contrastive foil with a specific property: "Can the agent find a policy that has the QA i value no greater than θ_i?".

3.6 Example Applications of Explainable Multi-Objective MDPs

In this section, we discuss how to apply our explainable multi-objective MDP planning approach, illustrated via three example application domains. We discuss how to specify each problem domain in the XMDP representation, using our framework to define domain-specific types to be used as building blocks for modeling planning problem instances. We also discuss examples of explanations generated by our approach.

3.6.1 Indoor Mobile Robot Navigation

Our motivating example in Section 3.1.1 describes an indoor mobile robot navigation problem. This section demonstrates how to define the mobile robot planning problem domain in our XMDP representation.

State Variables The state space S of the mobile robot is defined by 2 variables: $\langle \mathit{rLoc}, \mathit{rSpeed} \rangle$, where rLoc denotes the current location and rSpeed denotes the current speed setting of the robot. Note that in this example domain, we choose not to include the state of the robot's bump sensor as a state variable for simplicity. However, the collision concern of the robot can still be captured via other XMDP constructs, as we will discuss.

We define a state-variable type Location as the type of $rLoc$. We define the following state-variable attributes associated with the Location type: ID representing the unique ID of a particular location, and Area representing the type of area in which the location is, which can be one of the following types: *public, semi-private* or *private*. Some examples of public areas include hallway and cafeteria, semi-private areas include conference rooms and student lounges, and private areas include faculty offices.

We define a state-variable type RobotSpeed as the type of $rSpeed$. This is a subtype of our built-in type for representing "floating-point" state variables. These are *not* continuous variables, but variables whose values come from a predefined finite set of floating-point values. Here, the robot has two discrete levels of speed settings: full-speed and half-speed, which are 0.7 m/s and 0.35 m/s, respectively. The floating-point value of $rSpeed$ variable is to be used directly in computation of QA values.

Actions The action space A of the mobile robot is defined by 2 types of actions: MoveTo denoting the robot moving from its current location to a given adjacent location, and SetSpeed denoting the robot changing its speed to a given setting.

The MoveTo action type is parameterized by a Location argument denoting the destination, denoted in the form MoveTo($rLocDest$: Location). We define the following action attributes associated with MoveTo type: Distance and Obstacle, denoting the distance and obstacle density of the path between the source and the destination locations of an action, respectively. Distance is a floating point and Obstacle density has the following discrete levels: *none, sparse* and *dense*.

The SetSpeed action type is parameterized by a RobotSpeed argument denoting the target speed setting, denoted in the form SetSpeed($rSpeedTarget$: RobotSpeed). The target speed setting must be from the predefined finite set: full-speed at 0.7 m/s and half-speed at 0.35 m/s.

Transition Model To represent the probabilistic transition function $P : S \times A \times S \to [0, 1]$, we use factored probabilistic STRIPS operators (factored PSOs) [12] to describe the preconditions and context-dependent, stochastic effects of each action type. A factored PSO consists of a set of mutually exclusive and exhaustive logical formulae, called *contexts*, and a *stochastic effect* associated with each context. A stochastic effect is a set of *change sets* – a list of variable values – with a probability attached to each change set.

The MoveTo($rLocDest$: Location) action type has a precondition that the robot's current location must be adjacent to the given destination $rLocDest$. Its effect is deterministic, namely, when the precondition is met, the robot's new location will be the given (adjacent) destination.

The SetSpeed($rSpeedTarget$: RobotSpeed) action type has a precondition that the robot's current speed setting must be different from the given target speed $rSpeedTarget$. Its effect is deterministic, namely, when the precondition is met, the robot's new speed setting will be the given target speed.

Note that the transition model for this problem domain is deterministic, as we do not model the state of the robot's bump sensor as a state variable. We made this modeling choice in order to not allow the planning agent to make different decisions depending on whether a collision has occurred. This leads to simpler navigation policies.

Quality Attribute Analytic Models We define the following quality attributes (QAs): travel time, expected collisions, and intrusiveness.

Travel Time We define $\mathsf{TravelTime}$ as a standard measurement QA. The $\mathsf{TravelTime}$ value of a single transition is a function of the state variable rSpeed and the action attributes $\mathsf{Distance}$ and $\mathsf{Obstacle}$ of a MoveTo action: the travel time is $\mathsf{Distance}/\mathit{rSpeed} \cdot \mathit{delayRate}(\mathsf{Obstacle})$. No travel time is incurred under other action types.

Collision We define $\mathsf{Collision}$ as an event for a count-of-events QA. As noted, we do not include the state of the robot's bump sensor as a state variable. Instead, we define probabilities of $\mathsf{Collision}$ event occurring given different levels of obstacle density of a path segment when the robot moves at a speed over $\theta_{\text{safeSpeed}}$. That is, the $\mathsf{Collision}$ probability of a single transition is a function of the action attribute $\mathsf{Obstacle}$ of a MoveTo action and the value of rSpeed state variable: the probability is $0.0, 0.2$, and 0.4 when $\mathit{rSpeed} > \theta_{\text{safeSpeed}}$ and the obstacle density is *none, sparse*, and *dense*, respectively. Probability of a $\mathsf{Collision}$ event is 0 under other action types, or when $\mathit{rSpeed} \leq \theta_{\text{safeSpeed}}$.

Intrusiveness We define $\mathsf{Intrusiveness}$ as a non-standard measurement QA. To this end, we define a set of events: *non-intrusive, somewhat-intrusive*, and *very-intrusive* to characterize the level of intrusiveness of the robot when it moves into a particular location. These events are defined by the state-variable attribute Area of the destination $\mathsf{Location}$ argument of a MoveTo action. Namely, *non-intrusive, somewhat-intrusive*, and *very-intrusive* events correspond to when the robot moves into *public, semi-private* and *private* areas, respectively. We assign arbitrary penalty values of $0, 1$, and 3 to these events, respectively, to quantify their relative severity.

Cost Function The cost function $C : S \times A \to \mathbb{R}_{\geq 0}$ is a composition sub-costs of the QAs. Specifically, C is a linear scalarization: $C(s,a) = \sum_i k_i C_i(s,a)$, where $k_i > 0$; $\sum_i k_i = 1$; and $C_i(\cdot,\cdot)$ is a normalization of QA i function: $C_i(s,a) = \dfrac{QA_i(s,a)}{\max\limits_{s',a'} QA_i(s',a')}$.

The cost criterion used in this problem domain is the expected total cost-to-go (Equation 3.1. The absorbing goal is a set of states where rLoc is at a given goal location.

Explanations For a given navigation planning XMDP problem, our framework computes an optimal navigation policy and generates explanations for that policy, which consist of the following:

First is the statement of the goal and quality-attribute objectives of the robot: *"I am planning to reach Location L32, while minimizing the travel time, expected number of collisions, and intrusiveness."*

Second is the presentation of the robot's chosen navigation policy in a visual form, and the expected consequences of that policy: *"I came up with this policy:* [visualization of the chosen navigation on the map]. *It is expected to take 10 minutes of travel time, have 0 expected collision, and be somewhat intrusive at 2 locations."*

Third is the explanations of the robot's tradeoff rationale, in the form of contrastive explanations via selected Patero optimal alternatives. For instance: *"I could reduce the travel time by 2 minutes by following this alternative policy:* [visualization of the alternative navigation]. *However, this would increase the expected number of collision by 0.2."* The number of constrastive explanations presented, which is the number of alternative policies explained, depends on the number of competing optimization objectives and whether the robot's chosen policy already has the minimal values for some of the objectives. For instance, in this example the number of expected collisions (0) is minimal, and thus there is no alternative policy generated that can reduce the value of that objective. The alternative policies may have different navigation routes and may use different speeds.

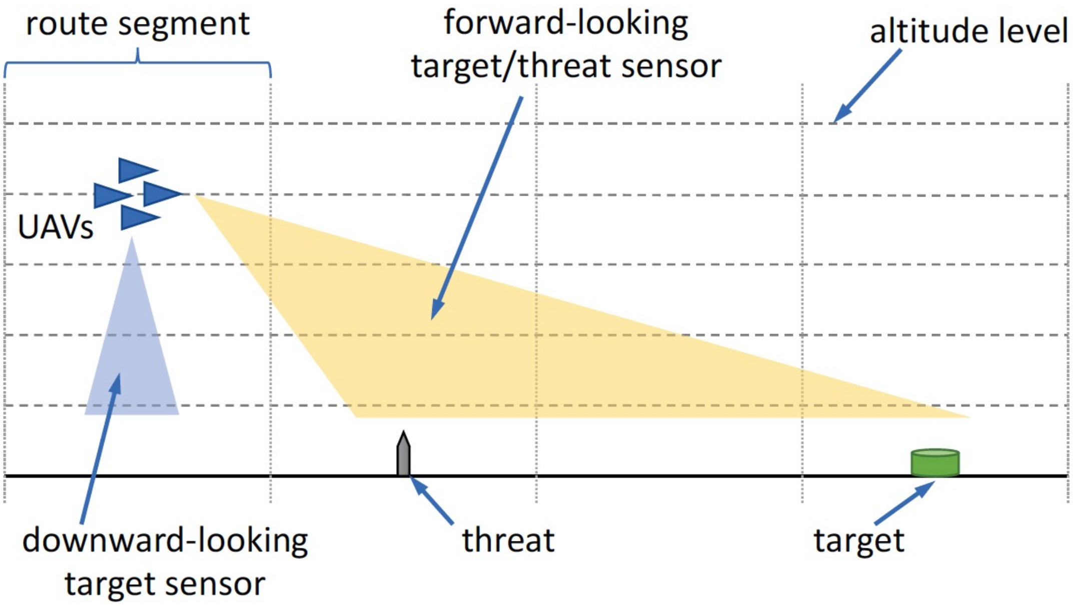
Our contrastive explanation approach makes the decision options that the robot has more transparent and comprehensible. The explanations offer arguments for the robot's chosen policy based not only on its direct consequences, but also on the counterfactual consequences.

3.6.2 Distributed Adaptive Real-Time System (DART)

The DART (Distributed Adaptive Real-Time) Systems project at the Carnegie Mellon Software Engineering Institute [42] implements a simulated team of unmanned aerial vehicles (UAVs) performing a reconnaissance mission in a hostile environment. We use the problem definition of a DART system simulation presented in [72] for this case study. The mission of the team is to fly a planned route at constant speed across an area to detect as many targets on the ground as possible. At the same time, the team must try to avoid being shot down by threats in the area, which would result in the failure of the mission. As the team flies during the mission, it discovers the environment (i.e., the location of targets and threats) with some uncertainty, and plans to adapt its altitude, formation, and other configurations for a finite horizon, accordingly. The team continues the cycle of planning and execution until it has reached the end of the route, or it is shot down by threats.

The team has two long-range forward-looking sensors to monitor the state of the environment for a fixed horizon ahead of the team: one for sensing targets and the other for sensing threats. The route is divided into segments of equal length. For each route segment in front of the sensor, it reports whether it detects a target/threat. Due to sensing errors, these reports can be false positive or false negative. The team therefore would get multiple observations to construct a probability distribution of target or threat presence in each route segment ahead. These probability distributions will be used in mission planning.

To detect targets on the ground, the team uses a downward-looking sensor as it flies over. The closer the UAVs fly to the ground, the more likely they are to detect the targets, but also the higher probability of being destroyed by a threat. The team can be in either loose or tight formation. Flying in tight formation reduces the probability of being shot down; however, it also reduces the target detection probability due to sensor occlusion or overlap. The team can use electronic countermeasures (ECM) to reduce the probability of being destroyed by threats, but



route segment
forward-looking target/threat sensor
altitude level
UAVs
downward-looking target sensor
threat
target

Figure 3.6: Illustration of a reconnaissance mission planning of a team of UAVs in a hostile and unknown environment, from DARTSim [72].

using ECM also reduce the probability of target detection.

Using the probability distributions of targets and threats ahead of the team constructed from observations, the team plans for any necessary adaptation to change the team's configuration or altitude to balance the objectives of maximizing the number of targets detected and minimizing the probability of being destroyed by threats.

This section demonstrates how to define the DART planning problem domain in our XMDP representation.

State Variables The state space S of the team of UAVs is defined by 5 variables: $\langle a, \phi, E, t, d \rangle$, where a denotes the current altitude of the team; ϕ denotes the current formation of the team: loose or tight; E denotes the status of ECM: on or off; t denotes the t^{th} route segment above which the team currently is; and d denotes whether the team has been destroyed by threats.

We define the following state-variable types:

- TeamAltitude as the type of a, representing the discrete altitude levels. This is a subtype of our built-in type for integer variables.
- TeamFormation as the type of ϕ, representing the loose ($\phi = 0$) and tight ($\phi = 1$) formations.
- TeamECM as the type of E, representing the ECM status: on ($E = 1$) and off ($E = 0$).
- RouteSegment as the type of t, representing a route segment. We define the following state-variable attributes associated with the RouteSegment type: TargetDistribution and ThreatDistribution representing the β-distribution of target and threat presence in a route

segment, respectively.

- TeamDestroyed as the type of d, representing whether the team has been destroyed by threats. This is a subtype of our built-in type for boolean variables. Using the explicit variable d allows the cost computation in planning to accrue the expected number of targets missed throughout the entire route, even after the team had been destroyed prior to reaching the end of the route. We discuss this in further details below.

Actions The action space A of the team of UAVs is defined by the following types of actions:

IncAlt, DecAlt and Fly denote the actions to increase altitude level, decrease altitude level, and fly in the same altitude level, respectively. All of these actions fly the team forward by 1 route segment. The IncAlt and DecAlt action types are parameterized by a TeamAltitude argument denoting the change in altitude level, denoted in the forms IncAlt(a_{change} : TeamAltitude) and DecAlt(a_{change} : TeamAltitude), respectively.

For specification convenience, we incorporate type hierarchy for action types to allow reuse of factored PSOs describing the actions' preconditions and effects on individual or groups of variables. For instance, IncAlt, DecAlt and Fly actions all have an effect of advancing the team forward by 1 route segment. We create a supertype DurativeAction with the factored PSO specifying the effect of advancing t's value by 1. When defining IncAlt, DecAlt and Fly as subtypes of DurativeAction, they automatically inherit its factored PSO of the effect on t. Although, subtypes are allowed to have additional preconditions, that is, the subtypes' preconditions can be stronger than (and subsume) its supertype's precondition.

We define a helper action type Tick, also as a subtype of DurativeAction, to represent the passage of time after the team has been destroyed by threats prior to reaching the end of the route. This helper action allows the cost computation in planning to accrue the expected number of targets missed throughout the entire route after the team has been destroyed.

ChangeForm denotes the action to change formation of the team. It is parameterized by a TeamFormation argument denoting the target formation: ChangeForm(ϕ_{target} : TeamFormation). SwitchECM denotes the action to switch on and off the team's ECM. It is parameterized by a TeamECM argument denoting the target ECM status: SwitchECM(E_{target} : TeamECM). Both of these action types are instantaneous relative to the others.

Transition Model As with the indoor mobile robot navigation domain in Section 3.6.1, we use factored PSOs to describe the preconditions and context-dependent, probabilistic effects of each action type.

Recall that all action subtypes inherit factored PSOs from their action supertype. The DurativeAction supertype has 2 factored PSOs associated with the effects on the variables t and d, respectively. The factored PSO of t has a precondition that the team has not yet reached the end of the planned route, and a deterministic effect that the team flies forward by 1 route segment – that is, t is incremented by 1. The factored PSO of d has the same precondition plus $d = \texttt{false}$, but its effect is probabilistic. Executing a DurativeAction has a probability of resulting in the team being shot down by threats. The probability of the team being destroyed in a single transition (i.e., the team flying over a route segment) is equal to the probability of a threat existing in a given route segment times the probability of the team being shot down by the threat given one exists:

$$Pr(d = \texttt{true}) = Pr(threat) \cdot Pr(d = \texttt{true} \,|\, threat).$$

Recall that the team has uncertain information about the threat presence in each route segment ahead, in the form of a β-distribution. The β-distribution of threat presence in each route segment is specified as the attribute ThreatDistribution of the RouteSegment type. For simplicity, we use the expected value of the target β-distribution as $Pr(threat)$. Recall also that the probability of the team being shot down by a threat, given one exists, depends on the team's altitude level, formation, and ECM status. Specifically, the probability of the team being destroyed is defined in [72] as:

$$Pr(d = \texttt{true} \,|\, threat) = \frac{\max(0, r_T - a)}{r_T} \Big((1-\phi) + \frac{\phi}{\psi}\Big) \Big((1-E) + \frac{E}{4}\Big),$$

where r_T is the threat range (i.e., at a altitude of r_T or higher, it is not possible for threats to shoot down the team); ψ is the factor by which the probability of the team being destroyed is reduced due to flying in tight formation; and when ECM is used, the probability of being destroyed is reduced by a factor of 4. We explicitly encode the above analytic model of $Pr(d = \texttt{true})$ in the corresponding factored PSO of DurativeAction.

The IncAlt, DecAlt and Fly action subtypes all inherit these DurativeAction's factored PSOs of the effects on t and d. Although, IncAlt, DecAlt and Fly override the factored PSO of the effect on t with an additional precondition: the team must not have been destroyed yet: $d = \texttt{false}$.

The IncAlt and DecAlt action types have their own additional factored PSOs describing the effect on the altitude variable a. The IncAlt(a_{change} : TeamAltitude) action type has a precondition that the team's current altitude is no less than a_{change} levels below the maximum altitude. Its effect is deterministic: incrementing a by a_{change}. Similarly, the DecAlt(a_{change} : TeamAltitude) action type has a precondition that the team's current altitude is no less than a_{change} levels above the minimum altitude. Its effect is deterministic: decrementing a by a_{change}.

The ChangeForm(ϕ_{target} : TeamFormation) and SwitchECM(E_{target} : TeamECM) action types are instantaneous actions (i.e., they do not impact t). The preconditions of these action types are: $d = \texttt{false}$ and that the team's current formation and ECM status is different from the target ones, respectively. Their effects are deterministic: changing the team's formation and ECM status accordingly.

Lastly, the helper action type Tick also inherits the factored PSOs from its supertype DutativeAction. However, unlike the action types IncAlt, DecAlt and Fly, it overrides the factored PSO of the effect on t with an additional precondition that the team must have already been destroyed: $d = \texttt{true}$. Tick is the only applicable action once the team has been destroyed. It only advances t to reflect the passage of time, and does not have any other effect. This allows the expected number of targets missed after the team has been destroyed to accrue over the remaining route segments ahead.

Quality Attribute Analytic Models We define the following quality attributes (QAs): the expected number of targets detected, and the probability of being destroyed by threats.

However, since our approach uses cost functions for XMDPs, we need all QA objectives to be minimization objectives. Therefore, for the team's objective to maximize the number of targets detected, we create an equivalent, dual objective to minimize the number of targets missed. We use the targets-missed minimization objective in the cost function C of the XMDP.

For the purpose of explanations, however, it is more comprehensible to communicate the expected number of targets detected to the users. Therefore, our approach maintains the dual QA functions of targets-missed and targets-detected, and uses the latter to compute and report the expected number of targets detected in explanations.

For brevity, here we only discuss the QA function for the expected number of targets missed, and we omit the dual function for the expected number of targets detected.

Number of Targets Missed We define MissTarget as an event for a count-of-events QA. This event is associated with a DurativeAction type. That is, the team can miss a target when flying over a route segment via a IncAlt, DecAlt, or Fly action, or it can miss a target in a remaining unexplored route segment after the team has been destroyed by threats, via a Tick action. The probability of MissTarget event occurring in a single transition (via a DurativeAction) is equal to the probability of a target existing in the corresponding route segment times the probability of the team failing to detect the target given one exists:

$$Pr(\textsf{MissTarget}) = Pr(target) \cdot Pr(\textsf{MissTarget} \,|\, target).$$

Recall that the team has uncertain information about the target presence in each route segment ahead, in the form of a β-distribution. The β-distribution of target presence in each route segment is specified as the attribute TargetDistribution of the RouteSegment type. For simplicity, we use the expected value of the target β-distribution as $Pr(target)$.

Recall also that the probability of the team successfully detecting a target, given one exists, depends on the team's altitude level, formation, and ECM status. Specifically, the probability of the team detecting a target is defined in [72] as:

$$Pr(\textsf{DetectTarget} \,|\, target) = \frac{\max(0, r_S - a)}{r_S}\Big((1-\phi) + \frac{\phi}{\sigma}\Big)\Big((1-E) + \frac{E}{4}\Big),$$

where r_S is the range of the sensor (i.e., at a altitude of r_S or higher, it is not possible to detect targets); σ is the factor by which the detection probability is reduced due to flying in tight formation; and when ECM is used, the detection probability is reduced by a factor of 4. Note that the above formula for $Pr(\textsf{DetectTarget} \,|\, target)$ is only for when the team is still surviving. If the team has already been destroyed, then $Pr(\textsf{DetectTarget} \,|\, target) = 0$.

We can obtain the $Pr(\textsf{MissTarget} \,|\, target)$ from $1 - Pr(\textsf{DetectTarget} \,|\, target)$. We explicitly encode the above analytic model of $Pr(\textsf{MissTarget})$ in the corresponding count-of-events QA model.

Probability of being Destroyed We define DestroyedProbability as a standard measurement QA, denoting the probability of a team being destroyed by threats. The DestroyedProbability value of a single transition, (s, a, s'), is simply 0 if $d = \texttt{true}$ in the destination state s', and 1 otherwise. However, since a QA function must be in the form $QA : S \times A \to R_{\geq 0}$, the DestroyedProbability is in effect a function of the ThreatDistribution attribute of the RouteSegment state variable t, the TeamAltitude state variable a, the TeamFormation state variable

ϕ, and the TeamECM state variable E. Our framework derives the analytic model of DestroyedProbability from the model of $Pr(d = \texttt{true})$ defined in the Transition Model. The total expected value of DestroyedProbability of a policy is the reachability probability of states where $d = \texttt{true}$.

Cost Function As with the Indoor Mobile Robot Navigation domain in Section 3.6.1, we define the cost function C in the form: $C(s, a) = \sum_i k_i C_i(s, a)$, where $k_i > 0$; $\sum_i k_i = 1$; and $C_i(\cdot, \cdot)$ is a normalization of QA i function. Since both QAs of this domain, MissTarget count of events and DestroyedProbability, have the maximum value of 1 for any state-action pair, we can use the normalization constant of 1 for each $C_i(\cdot, \cdot)$.

The cost criterion used in this problem domain is the expected total cost-to-go. The absorbing goal is a set of states where t indicates the end of the planned route. Note that a goal state is always reachable: either the team has survived the threats and flown to the end of the route, or that the team had been destroyed before completing the route, but afterwards the Tick action advances t until it reaches the end value.

Explanations The explanations for DART planning have the same structure as those for the mobile robot navigation planning in Section 3.6.1. The following is an example of explanations, assuming that the team has to fly a pre-planned route of length 10, and there are 2 targets and 1 threat along the route:

First is the statement of the goal and quality-attribute objectives of the team: *"I am planning to fly to Route Segment 10, while maximizing the number of targets detected and minimizing the probability of being destroyed by threats."*

Second is the presentation of the team's chosen adaptation policy in a visual form, and the expected consequences of that policy: *"I came up with this policy:* [visualization of the chosen adaptation policy on the 2D flight route]. *It has 1.375 expected number of targets detected and 0.625 probability of being destroyed."*

Third is the explanations of the team's tradeoff rationale, in the form of contrastive explanations: *"I could increase the expected number of targets detected by 0.125 by following this alternative policy:* [visualization of the alternative adaptation]. *However this would increase the probability of being destroyed by 0.125."* The alternative policies may have different flying altitude sequences, formation sequences, and uses of ECM. The explanations contrast the team's chosen policy with the viable alternatives, based on their QA values, to offer arguments for the team's chosen policy rooted not only in its direct consequences, but also the counterfactual consequences.

3.6.3 Outpatient Clinic Scheduling

[74] developed a MDP model for managing an efficient outpatient clinic scheduling. Outpatient clinic scheduling deals with the following problem scenario. Assuming that a clinic has a fixed capacity to service some number of patients per day, but it can operate overtime to service more patients in each day if needed. Before the clinic's service period begins on each day, new patients call in to request for appointments. The clinic has to make a decision on each day of how

many patients to schedule for that day (i.e., same-day booking), and how many to schedule for appointments in future days (i.e., advance booking). The clinic has an "advance booking policy" (ABP), which sets the maximum number of patients who can be booked in advance into each future day. A decision can be made to adjust the ABP limit. The clinic also has a fixed limit on the total number of patients who can be scheduled for appointment in future days (i.e., the maximum length of the service queue).

The objectives of the clinic scheduling problem are: to maximize the revenue gained from each patient serviced, and to minimize the overtime service, the clinic's idle time, the patients' appointment lead time, and the overhead of updating the ABP.

In this scheduling problem, the amount of revenue gained from patients, the overtime and idle time of the clinic can only be predicted with uncertainty. This is due to: (i) the prediction of daily request arrivals of patients, which is modeled as a Poisson distribution, and (ii) the chances of scheduled patients not showing up for their appointments. In particular, the probability that a same-day booking patient shows up for their appointment is constant at around 0.88. The probability that an advance-booking patient shows up for their appointment is decreasing as a function of the appointment lead time.

This section demonstrates how to define the outpatient clinic scheduling problem domain in our XMDP representation. We have adopted the MDP model for this problem domain as presented in [74].

State Variables The state space S of the clinic scheduling problem is defined by 3 variables: $\langle w, x, y \rangle$, where w denotes the current ABP limit, which is the set maximum number of patients who can be booked in advance in each future day; x denotes the number of patients who have been previously booked for appointments; and y denotes the number of new patient-requests arriving today.

We define the following state-variable types:

- ABP as the type of w, representing the advance-booking policy (ABP) limits. This is a subtype of our built-in type for integer variables.
- PatientCount as the type of x and y, representing the possible numbers of patients. This is a subtype of integer variables.

The clinic has a fixed capacity to service C patients per day, but with additional overtime available up to M patients per day, if necessary ($M > C$). This means the maximum value of w is M. Furthermore, the limit on the total number of patients who can be scheduled for future appointments is N. This is the maximum size of the queue for the advance-booking patients.

Actions The action space A of the clinic scheduling problem is defined by only one action type: Schedule action. This action combines 2 decisions: (i) adjusting the limit for ABP, and (ii) deciding how many of the new patients to service today vs. to schedule for appointments in future days.

The Schedule(a : ABP, b : PatientCount) action type is parameterized by an ABP argument a, representing the new ABP limit, and a PatientCount argument b, representing the number of new patients (i.e., today's request arrivals) to service today. This means: (i) the new ABP limit a will be in effect starting the next day, and (ii) out of all new patients whose requests arrived today, b patients will be service today ("same-day booking") and the remaining patients will be scheduled for future appointments ("advance booking").

Transition Model As with the indoor mobile robot navigation domain in Section 3.6.1 and the DART domain in Section 3.6.2, we use factored PSOs to describe the preconditions and context-dependent, probabilistic effects of each action type.

Note that in this scheduling domain, a single decision epoch corresponds to one day. That means, the effects of a Schedule action executed today will occur in the next day.

The Schedule(a : ABP, b : PatientCount) action type has the following preconditions:

- The number of new patients scheduled to service today ("same-day booking") cannot exceed the number of new patient-requests arriving today. That is, $b \leq y$.
- The scheduling has a constraint that the number of patients in the advance-booking queue cannot exceed the imposed limit. This constraint translates to a precondition of the Schedule action as: $x - min(x, w) + y - b \leq N$. Note that $min(x, w)$ is the number of patients who have previously been scheduled for services today, since x is the total number of patients in the advance-booking queue and w is effectively the number of advance-booking patients to be serviced each day.
- The new ABP cannot be more than 1 patient per day different from the previous ABP, and it cannot exceed the imposed limit. That is, $(|w - a| \leq 1) \wedge (a \leq M)$.

The effects of Schedule(a : ABP, b : PatientCount) can be broken down to the following 3 independent effects:

- The new value of w (i.e., the new ABP limit) for the next day is a.
- The new value of x (i.e., the resulting number of patients in the advance-booking queue) for the next day is: $x - min(x, w) + y - b$. Particularly, $x - min(x, w)$ is the number of patients previously in the advance-booking queue who still remain in the queue after today, and $y - b$ is the number of new patients from today who are added to the advance-booking queue.
- The new value of y for the next day (i.e., the number of new patient-requests arriving the next day) is probabilistic, represented by a Poisson distribution, with an assumed known average arrival rate λ. The probability mass function of patient-requests arrival Possion distribution is:

$$Pr(y = k \text{ new patients in a day}) = \frac{e^{-\lambda} \cdot \lambda^k}{k!}.$$

Note that the effect on y is not a direct result of a Schedule action, but of an exogenous event of patient-requests daily arrival, as a single decision epoch corresponds to a one-day period in this scheduling domain.

Quality Attribute Analytic Models We define the following quality attributes (QAs) for the clinic scheduling problem: the average revenue from the patients serviced per day, the average overtime of the clinic per day, the average idle time of the clinic per day, the average appointment lead time of the advance-booking patients (i.e., the average number of days between the request for service and the date of service), and the average change in the ABP limit between 2 consecutive days. The objectives of scheduling are to maximize the average daily revenue, and minimize the average daily overtime, idle time, lead time, and change in ABP limit.

Revenue We define Revenue as a standard measurement QA. The revenue of the clinic in each day depends on the number of patients whom the clinic service on that day. Specifically, the number of patients serviced on each day is the total of the advance-booking and the same-day-booking patients who show up to the clinic on that day. Note that there is uncertainty of whether these patients will show up or not.

The Revenue value of a single transition (corresponds to a single day) is a function of the state variables w, x, and y, and of the Schedule action parameter b. Specifically,

$$\text{Revenue} = f^R \cdot \big(p_s(w,x) \cdot min(w,x) + p_{sd} \cdot b\big),$$

where:

- f^R is the amount of revenue gained per patient serviced.
- $p_s(w,x)$ is the probability that an advance-booking patient shows up, given that there are x patients in the advance-booking queue and w is the current ABP limit, meaning that at most w of those patients are scheduled for service today, hence $min(w,x)$.
- p_{sd} is the probability that a same-day-booking patient shows up, which is around 0.88, and b is the number of new patient-requests arriving today that are scheduled for the same day.

The probability of an advance-booking patient showing up to their appointment is decreasing as a function of the appointment lead time:

$$p_s(w,x) = max\left(1 - \frac{\beta_1 + \beta_2 \cdot log(LT + 1)}{100}, \beta_3\right), \tag{3.16}$$

where:

- LT denotes the appointment lead time, which is a function of the current number of patients in the advance-booking queue, x, and the current ABP limit, w. See the Equation 3.17 for how LT is calculated.
- β_1, β_2, and β_3 are constants obtained from empirical data of patients. β_1 relates to the show probability of a same-day-booking patient. That is, when $LT = 0$, then the same-day show probability (denoted as p_{sd}) is $1 - \beta_1/100$. Given that the same-day show probability is around 0.88, the value of β_1 is 12. $\beta_2 = 36.54$ models the diminishing impact of an extra day's wait the further our a patient is booked. β_3 represents a lower bound on the show probability of a patient, which is around 0.5.

We explicitly encode the above analytic model of Revenue in the corresponding standard measurement QA model.

Overtime We define Overtime as a standard measurement QA. The unit of overtime is the expected number of patients serviced through overtime. The number of overtime patients in each day depends on the number of patients who the clinic service on that day and the capacity of the clinic. Any number of patients exceeding the clinic capacity will be serviced during the overtime.

The $\mathsf{Overtime}$ value of a single transition (corresponds to a single day) is a function of the state variables w, x, and y, and of the $\mathsf{Schedule}$ action parameter b. Specifically, the expected number of overtime patients in a day is calculated by:

$$\mathsf{Overtime} = \sum_{(i,j) \in W \times B} (i + j - C)^+ \cdot Pr(AB = i) \cdot Pr(SD = j),$$

where:

- W is the set of possible values for $min(w, x)$, that is, all possible numbers of advance-booking patients who are scheduled for services today. B is the set of all possible values for b, that is, all possible numbers of new patient-requests arriving today to schedule for services on the same day. Thus, all combinations (i, j) of W and B are all possible combinations of i advance-booking patients and j same-day-booking patients who are scheduled for services today.
- C denotes the capacity of the clinic, and $(i+j-C)^+$ denotes the number of patients (either advance booking or same-day booking) scheduled for today that exceeds the capacity.
- AB is a random variable denoting the number of advance-booking patients scheduled for today who show up. Recall that the probability of an advance-booking patient showing up for their appointment today is $p_s(w, x)$ as defined in Equation 3.16. Therefore, the probability that i advance-booking patients show up is modeled by a binomial distribution:

  $$Pr(AB = i) = \binom{N_{AB}}{i} \cdot p_s^i (1 - p_s)^{N_{AB} - i},$$

  where $N_{AB} = min(w, x)$ is the total number of advance-booking patients scheduled for today, and p_s is a short-hand for $p_s(w, x)$.
- SD is a random variable denoting the number of patients given same-day bookings who show up today. Recall that the probability of a same-day-booking patient showing up for their appointment is $p_{sd} = 0.88$. Therefore, the probability that j same-day-booking patients show up is modeled by a binomial distribution:

  $$Pr(SD = j) = \binom{N_{SD}}{j} \cdot p_{sd}^j (1 - p_{sd})^{N_{SD} - j},$$

  where $N_{SD} = b$ is the total number of same-day-booking patients scheduled for today.

We explicitly encode the above analytic model of $\mathsf{Overtime}$ in the corresponding standard measurement QA model.

Idle Time We define $\mathsf{IdleTime}$ as a standard measurement QA. The unit of idle time is the expected number of patients whom the clinic could have serviced during the idle time. The idle time in each day is defined as the difference between the clinic's capacity and the number of patients whom the clinic services on that day.

The $\mathsf{IdleTime}$ value of a single transition (corresponds to a single day) is a function of the state variables w, x, and y, and of the $\mathsf{Schedule}$ action parameter b. Specifically, the expected idle time (as the number of patients whom the clinic could have serviced) in a day is calculated by:

$$\mathsf{IdleTime} = \sum_{(i,j)\in W\times B} (C-i-j)^{+} \cdot Pr(AB=i) \cdot Pr(SD=j),$$

where all the symbols are defined in the same way as those of the $\mathsf{Overtime}$ function. The idle time calculation is almost equivalent to that of the overtime. The only difference is in the term $(C-i-j)^{+}$, which denotes the remaining capacity of the clinic today after scheduling patients (either through advance booking or same-day booking) for today.

Lead Time We define $\mathsf{LeadTime}$ as a standard measurement QA. The unit of lead time is the number of days patients have to wait for their appointments (i.e., the number of days between the time of request and the time of service). We consider lead time as a daily quantity. That means, in each day, some amount of lead time is assigned to a batch of new advance-booking patients whose requests arrive on that day.

The $\mathsf{LeadTime}$ value of a single transition (corresponds to a single day) is a function of the state variables w and x. Specifically, the lead time assigned to the new advance-booking patients of each day is calculated by:

$$\mathsf{LeadTime} = max\Big(1, \Big\lfloor \frac{x}{w} \Big\rfloor\Big), \tag{3.17}$$

where $\lfloor x/w \rfloor$ is the number of days the last patient in the advance-booking queue has to wait until their appointment. Note that we simplify the calculation by using the length of the queue at the time of service as a proxy for lead time, while in fact it is more accurate to use the length of the queue at the time of booking – but that would make the model more complicated. The maximum operator reflects the fact that even if the number of advance bookings is less than the ABP limit (i.e., $\lfloor x/w \rfloor < 1$), any patient booked in advance still waits a day.

Change in ABP Limit We define $\mathsf{ABPChange}$ as a standard measurement QA. This quantity measures the average change in the ABP limit between 2 consecutive days. The average change in ABP limit from day to day is a concern in the clinic scheduling problem because there is an associated overhead cost in changing the ABP.

The $\mathsf{ABPChange}$ value of a single transition (reflects a transition between days) is a function of the state variable w and of the $\mathsf{Schedule}$ action parameter a:

$$\mathsf{ABPChange} = |w - a|,$$

which is the difference between the current ABP limit, w, and the new ABP limit, a.

Cost Function In this clinic scheduling domain, all QAs of concern are translated to monetary costs.

Since the revenue objective is a maximization objective, we need to transform it to a minimization objective. To this end, we use an over-approximation of the loss of potential revenue in each day. The loss of potential revenue is the difference between the potential maximum revenue and the actual revenue gained from servicing patients. The potential maximum revenue is revenue from maximum number of patients per day. We assume that this is twice the average number of patient-requests per day.

The loss of potential revenue in a single-day period corresponding to the state-action (s, a) is calculated by:

$$2\lambda \cdot f^R - \textsf{Revenue}(s, a),$$

where λ is the average arrival rate of patient-requests per day, and f^R is the amount of revenue gained per patient serviced.

The cost function C is the sum of the monetary costs associated with the revenue, overtime, idle time, lead time, and overhead of changing ABP:

$$\begin{aligned} C(s, a) = 2\lambda{\cdot}f^R &- \textsf{Revenue}(s, a) \\ &+ f^{OT} \cdot \textsf{Overtime}(s, a) \\ &+ f^{IT} \cdot \textsf{IdleTime}(s, a) \\ &+ f^{LT} \cdot \textsf{LeadTime}(s, a) \\ &+ f^{S} \cdot \textsf{ABPChange}(s, a), \end{aligned}$$

where f^{OT} and f^{IT} are costs per overtime and idle time slot, respectively; f^{LT} is a cost per day that patients wait until their appointments; and f^S is an overhead cost of changing advance-booking policy. Note that since monetary costs are assigned to the QAs directly, there is no need to normalize the QA functions.

The cost criterion used in this problem domain is the long-run average cost. That is, an optimal clinic scheduling policy is one that has the lowest daily-average cost C in the long term.

Problem Scenarios We generate 9 outpatient clinic scheduling problem scenarios, as presented in [74]. The following constants are used in all problem scenarios:

- The capacity of the clinic C is 5 patients per day.
- The maximum ABP limit M is 5 advance-booking patients per day.
- the maximum size of the queue for the advance-booking patients (i.e., the total number of patients who can be scheduled for future appointments) N is 10 patients.
- The average arrival rate λ of the Poisson distribution of new patient-requests is 5 patients per day.

The initial state of the clinic scheduling problem $s_0 = \langle w_0, x_0, y_0 \rangle$ in all problem scenarios is:

- The initial ABP w_0 is set to 4 advance-booking patients per day.
- The initial number of patients who have been previously booked for appointments x_0 is 0 patient.
- The initial number of new patient-requests arriving in the first day y_0 is 5 patients.

Among the problem scenarios, we consider 3 different clinic types:

1. The first clinic type ignores idle time, and sets the revenue per-patient per-day to twice the cost of overtime per-slot per-day: $f^R = 20$, $f^{OT} = 10$, and $f^{IT} = 0$.

Scenario ID	Revenue and Costs Setup	Initial State	Constants
1	$f^R = 20, f^{OT} = 10, f^{IT} = 0, f^{LT} = 0, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
2	$f^R = 20, f^{OT} = 10, f^{IT} = 0, f^{LT} = 1, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
3	$f^R = 20, f^{OT} = 10, f^{IT} = 0, f^{LT} = 5, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
4	$f^R = 20, f^{OT} = 10, f^{IT} = 5, f^{LT} = 0, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
5	$f^R = 20, f^{OT} = 10, f^{IT} = 5, f^{LT} = 1, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
6	$f^R = 20, f^{OT} = 10, f^{IT} = 5, f^{LT} = 5, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
7	$f^R = 0, f^{OT} = 10, f^{IT} = 5, f^{LT} = 0, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
8	$f^R = 0, f^{OT} = 10, f^{IT} = 5, f^{LT} = 1, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$
9	$f^R = 0, f^{OT} = 10, f^{IT} = 5, f^{LT} = 5, f^S = 10$	$w_0 = 4, x_0 = 0, y_0 = 5$	$C = 5, M = 5, N = 10, \lambda = 5$

Table 3.2: The outpatient clinic scheduling problem scenarios.

2. The second clinic type adds a cost for idle time: $f^R = 20$, $f^{OT} = 10$, and $f^{IT} = 5$.
3. The third clinic type does not receive remuneration from each patient serviced (i.e., no revenue), but seeks to maximize the utilization of the available capacity to meet the demand: $f^R = 0, f^{OT} = 10$, and $f^{IT} = 5$.

For each clinic type, we consider 3 different lead time costs – the cost per day that patients wait until their appointments: $f^{LT} = \{0, 1, 5\}$. This results in the total of 9 problem scenarios. In all problem scenarios, we consider the overhead cost of changing ABP f^S to be equal to the cost of one over time slot: $10 per each time the policy is changed. Table 3.2 summarizes the settings in all 9 problem scenarios. The full details on the rationale for the revenue and costs setups are presented in [74].

Explanations The explanations for the clinic scheduling problem have the same structure as those for the mobile robot navigation planning in Section 3.6.1 and the DART planning in Section 3.6.2. The explanations state the optimization objectives of the scheduling problem: maximizing the average daily revenue from servicing patients, minimizing the average daily overtime and idle time of the clinic, minimizing the average lead time for the patients waiting for their appointments, and minimizing the average overhead of updating the clinic's advance-booking policy between any 2 consecutive days. The explanations present an optimal schedule and its expected consequences on the quality-attribute objectives. Then, the scheduling tradeoff rationale is offered via contrastive explanations of alternative Pareto-optimal schedules, to make the scheduling decision options and their consequences transparent. That is, the explanations demonstrate how some QA objectives are competing and how different scheduling choices lead to different compromises among those objectives.

For instance, a schedule that puts many patients up for service in each day will have high revenue; however, it will also have high overtime. On the other hand, if a schedule puts few patients up for service in each day (e.g., the schedule has few same-day bookings), then the advance-booking queue will be long, thus increasing the lead time experienced by the patients. Such schedule will have little to no overtime, but it will potentially have idle time.

These explanations are helpful for the users (e.g., the clinic administrators) to determine which scheduling option is best for their clinic type. For instance, publicly funded clinics may prioritize utilization of the available capacity to meet all demand, while private clinics may prioritize revenue gained from patients. Public clinics may value the patients' wait time less than

Scenario ID	Improvement in Objective(s)	Deterioration in Objective(s)	Symmetrical Conflict?
1	R	OT	Yes
	OT	R	
2	R	OT	Yes
	OT	R	
3	R	OT	Yes
	OT	R	
4	R	OT, IT	Yes
	OT, IT	R	
5	R, LT	OT, IT	Yes
	OT, IT	R, LT	
6	R	OT, IT	Yes
	OT, IT	R	
	IT	R, OT	No
7	OT	IT	No
	IT	OT, S	No
8	OT, LT	IT	Yes
	IT	OT, LT	
9	OT, LT	IT	Yes
	IT	OT, LT	

Table 3.3: Summary of the competing objectives explained by our policy explanations in each scenario of the clinic scheduling problem. The objectives are: the per-day average revenue (R), overtime cost (OT), idle time cost (IT), lead time cost (LT), and cost of changing the ABP (S).

private clinics do. Our explanation approach can enable the users to better understand the trade-off space of their scheduling problem, and decide on which scheduling option aligns best with their priorities.

The following is an example of explanations from Scenario # 1: *"I'm planning to follow this scheduling policy:* [table representing the chosen scheduling policy] . *It is expected to have 48.50 dollars in revenue; have 0.0 dollar in idle time cost; have 3.33 dollars in overtime cost; have 0.0 dollar in appointment lead time cost; and have 0.0 dollar in switching ABP cost per day on average. It has the lowest expected switching ABP cost, appointment lead time cost, and idle time cost.*

Alternatively, following this scheduling policy: [table representing the first alternative scheduling policy] *would increase the revenue to 48.99 dollars in revenue (an increase of 0.44 dollar) per day on average. However, I didn't choose that policy because it would increase the overtime cost to 3.93 dollars (an increase of 0.60 dollar) per day on average. The increase in revenue is not worth the increase in overtime cost.*

Alternatively, following this scheduling policy [table representing the second alternative scheduling policy] *would reduce the overtime cost to 3.30 dollars (a decrease of 0.03 dollar) per day on average. However, I didn't choose that policy because it would reduce the revenue to 48.47 dollars (a decrease of 0.03 dollar). The decrease in overtime cost does not make up for the decrease in revenue."*

The policy explanations identify the *context dependent* competing objectives and the tradeoffs in each problem scenario. Many of which are not obvious from the cost profiles of the objectives. Table 3.3 summarizes the tradeoff insights from the policy explanations. For each scenario, each sub-row shows the improvement in certain objective(s) results in the deterioration in certain other objective(s). For most of the objective conflicts across all scenarios, the improving objectives and the deteriorating objectives are symmetrical. That is, with respect to the solution scheduling policy, the objective conflict is symmetrical if improving the objective i results in deteriorating the objective j *and vice versa*.

Asymmetrical conflicts indicate a more complex tradeoff situation. For instance, in Scenario #7, with respect to the solution scheduling policy, reducing the overtime (OT) (by accepting fewer same-day bookings) will result in increasing the idle time (IT) (some of the patients placed on the queue will not show up for their appointments, resulting in some idle time). However, reducing the idle time (by accepting more same-day bookings) will increase both the overtime (the clinic staffs have to work longer hours in a day) and the overhead cost of changing ABP (S) (the clinic has to change its ABP more often).

An important observation from the generated policy explanations for all scenarios is that although the clinic scheduling domain has 5 optimization objectives that can theoretically have multiple combinations of competing objectives, only a small number of objectives are competing *at once*. For instance, Scenarios #1, 2, and 3 only have two competing objectives, namely, revenue and overtime cost. In Scenario #4, three objectives R, OT, IT are involved in the tradeoff, but OT and IT have correlated effects from improving R, and are not competing between themselves.

Some insights on the competing objectives provided by the generated contrastive explanations can be non-intuitive. For instance, one may reasonably expect that, minimizing the average overtime (OT) should often be in conflict with minimizing the average appointment lead time (LT), since the clinic would try to not have a long service period, and have to put more patient requests on the queue. However, this is only true in one of 9 scenarios: Scenario #5.

3.7 Related Work

There are numerous works in the field of explainable AI (XAI) [40], but the area that is closely related to our work is explainable agency. [6] provides a systematic literature review of works on explainable agency for robots and intelligent agents. [79] presents a taxonomy of explainability and a framework designed to enable comparison and evaluation of explainability in human–agent systems.

Many works in human-robot interaction focus on enabling robotic agents to communicate their goals and objectives to humans. For instance, [43] contributes an approach to enable a robot to communicate its objective function to humans by selecting scenarios to demonstrate its most informative behaviors. Similarly, [60] contributes an approach to automatically generate a robot's trajectory demonstrations with particular critical points to improve human observers' abilities to understand and generalize the robot's state preferences. Their works share a similar goal with ours, which is to communicate robots' preferences, but the focuses of their contributions are rather in specific domains. Other works such as [56, 57] propose approaches to explain multi-criteria decision making in non-sequential contexts, but a user-study validation is lacking.

Many other explainable agent approaches on explaining why agents make certain decisions have been proposed. Some recent works use argumentation-based approaches for explainable decision making. For instance, [24] provides argumentative and natural language explanations for scheduling of why a schedule is feasible, efficient or satisfying fixed user decisions. [105] presents context-based, explainable decision making via Decision Graphs with Context (DGC). More closely related to our work are [5, 44] which use reward decomposition for explaining the decisions of reinforcement learning agents. The approach decomposes rewards into sums of semantically meaningful reward types, so that actions can be compared in terms of tradeoffs among the types. These works share a similar feature of explanation with ours in terms of illustrating the agent's tradeoffs. However, their work focuses on explaining the tradeoffs of each individual action, while our work differs in that we focus on explaining the tradeoffs of an entire policy via identifying some potential alternative policies.

Contrastive explanations have been suggested for explainable AI [69], and have been widely used in the field. Some examples of recent works include [90] which provides explanations consisting of information that may be absent in the user's abstract model of the robot's task and show why the foil doesn't apply in the true situation; and [51] generates explanations for how two sets of plans differ in behavior by inferring LTL specifications that describe temporal differences between two sets of plan traces. Our work differs in the use of contrastive explanation in that we contrast alternative solutions on the basis of their objective values, in order to explain the tradeoff space of the planning objectives.

Chapter 4

Empirical Evaluation of Explainable Planning

The goal of our explainable planning approach is to help the end-users better understand the planning agent's reasoning, as well as become more confident about their assessment of the agent's decisions. In this thesis, we focus on consequence-oriented, contrastive explanation of planning. That is, we explain planning decisions in terms of their direct consequences on the objectives of interest, as well as the contrast to the counterfactual consequences of some alternative decisions. Given this focus, we evaluate the effectiveness of our approach via human subjects study, to determine how well our explanation approach can enable the users to understand planning decisions and to assess whether those decisions are appropriate.

We use a task-oriented evaluation for our explainable planning approach. That is, we evaluate our approach based on how much better the users are at performing some tasks when given consequence-oriented contrastive explanations, compared to baseline explanations. In this case, we consider a task of the user to be assessing the agent's planning decisions to determine whether they are the best decisions for a given problem context. Such task reflects a potential use case of explainable agency in human-agent collaboration settings, where the user oversees the agent's decision making and can intervene when they think the agent may be making wrong decisions. In the context of multi-objective planning, which is the focus of this thesis, best decisions depend on which preference model is used. Thus, the user's task is to determine whether the agent's decisions are optimal with respect to an appropriate preference model for a given problem context.

We conducted a human subjects experiment to evaluate the effectiveness of our explainable planning approach in two aspects. First is the impact of our contrastive explanations on the users' ability to soundly assess the planning agent's decisions. This ability reflects how well the users understand the agent's rationale. We measure it based on the correctness of the users' determination of whether the agent's decisions are best for a given problem context. Second is the impact of our contrastive explanations on the users' confidence in their assessment of the agent's decisions. This is a subjective measure that indicates how reliable the users perceive their understanding of the agent's decisions to be. We compare our explanation approach to a baseline explanation approach in which the users are only given information about the direct consequences of the agent's decisions (i.e., the objective values of the solution policy). Our central hypotheses for the experiment are that the users who receive consequence-oriented contrastive explanations from the agent are more likely to be correct in assessing the agent's decisions, and are more confident in their assessment. Our experimental results supported both hypotheses.

This chapter is structured as follows. Section 4.1 describes our user study design, which is based on indoor mobile robot navigation scenarios. Section 4.1.4 describes the experiment design. Section 4.2 presents the analysis and results of the experiment. Section 4.3 discusses the results and their implications.

4.1 User Study Design

We use the following scenario and task in our user study:

Scenario We created a user-study scenario in which the user is tasking the mobile robot to deliver a package at some destination in the building, similar to the motivating example in Section 3.1.1, where the robot has to plan a navigation that minimizes the travel time, expected collisions, and intrusiveness to the building occupants. The user has a particular preference for those concerns; however, they do not know a priori whether the robot's planning objective function aligns with their preference. Once the robot has computed its optimal navigation policy, it will present the policy, as well as the estimated travel time, expected collisions, and intrusiveness of the policy, to the user.

Task The user's task is to determine whether the robot's proposed navigation policy is the best available option according to their preference for the three concerns. In the user study, to have the ground truth answer for such task, we give a predefined hypothetical preference, in the form of a *cost profile*, to each study participant and instruct them to apply that preference when evaluating the robot's proposed navigation policy. The cost profile consists of a dollar amount per unit of each of the quality attributes (QAs) (i.e., the optimization objectives of the task) in this domain (e.g., \$1 per 1 second of travel time; \$2 per 0.1 expected collision). Each participant having a known fixed preference in the study allows us to have the ground truth of the preference alignment between the participant (i.e., the user role) and the robot.

See Appendix A for the complete instructions given to the participants in our user study.

Next, we discuss in further details how we design the questions to stimulate the scenarios and tasks for the participants.

4.1.1 Questions Design

Each participant is given a *cost profile* to use as their hypothetical preference, and a series of three *navigation-planning scenarios* plus one additional validating scenario (see Section 4.1.3). Each navigation-planning scenario consists of: (a) a building map, and (b) a proposed navigation policy from the robot. All building maps in all scenarios have the same topological structure, but each map is unique in its random placement of obstacles and random locations of public, semi-private, and private areas. In every map, the robot has the same starting location and goal location. The proposed navigation policy from the robot is an optimal policy with respect to the robot's cost function, which is hidden from the participant and may or may not be the same as the cost profile of the participant.

Presented with one navigation-planning scenario at a time, the participant is asked to:

1. Indicate whether they think the robot's proposed navigation policy is the best option according to their given cost profile, and
2. Rate their confidence in their answer on a 5-point Likert scale.

We refer to a pair of a cost profile and a navigation-planning scenario as a *question item*, which defines the problem context for the participants to answer the two questions described above. For this study, we generate 16 different cost profiles, which uniformly representing diverse user preferences for the travel time, expected collisions, and intrusiveness objectives of the navigation problem. We randomly generate 5 different building maps for the navigation-planning scenarios. That is, we have a total of 80 unique question items.

In the study, each participant is randomly assigned a cost profile to use as their preference, and is presented with three randomly selected navigation-planning scenarios (plus one additional validating scenario, which we do not include in the analysis) as described above. For half of the navigation planning scenarios each participant is given, we generate the robot's proposed policy in each scenario is optimal with respect to the participant's assigned preference (i.e., the robot's policy is aligned with the user's preference). We refer to these as *"preference-aligned" scenarios*. For the remaining scenarios, we generate the robot's proposed policy in each scenario is suboptimal with respect to the participant's assigned preference (but is still Pareto optimal). We compute such policy from a randomly sampled objective function that is misaligned with the participant's given cost profile. We refer to these as *"preference-misaligned" scenarios*. Thus, if the participants answer the question "Is the robot's proposed navigation policy is the best option, given your cost profile?" randomly, they have a 0.5 probability of being correct.

4.1.2 Participants' Qualification Test

Since we prescribe a known preference to each participant in the form of a cost profile to use in the study, we must ensure that the participant understands how to interpret and apply the cost profile. To this end, we pre-screen the study participants by giving them a qualification test that assesses their ability to perform simple arithmetic for policy cost calculation and to evaluate policies based on their costs. This test determines their eligibility to participate in the study. The specific qualification test we use is the following:

> Please answer the following questions. Your answers will determine your eligibility to participate in this study. You are allowed to use a calculator.
>
> Suppose you task a mobile robot to deliver a package at some destination. Suppose the robot then informs you that it has computed a navigation route (”route A”) that will take 6 minutes, but the route has 2 segments where the robot has 20% chance of colliding with obstacles (i.e., the expected value of collision is 0.4).
>
> Suppose that you wish to quantify how ”costly” the route is, by assigning \$1 to each minute of the route, and \$2 to each 0.1 expected collision of the route.
>
> Q1: What is the cost of the route, considering time of the route alone?

Q2: What is the cost of the route, considering expected collision of the route alone?

Q3: What is the cost of the route, considering both time and expected collision of the route?

Q4: Suppose that there is an alternative navigation route (”route B”) that will take 8 minutes, but the route has 1 segment where the robot has 20% chance of colliding with obstacles (i.e., the expected value of collision is 0.2). Which route is better (i.e., less ”costly”)?

We only accept the participants who answered all of the questions correctly into the study.

4.1.3 Validating Participants' Answers

In addition to pre-screening the study participants via a qualification test, we also use the following approach to validate that each participant understands the task correctly and pays attention to completing it. First, for each navigation-planning scenario, we ask the participant to provide the total cost of the robot's proposed navigation policy, based on their assigned cost profile. This is to check whether they understand how to apply a given cost profile to evaluate a policy. The participants are allowed to use a calculator. Second, we embed an "easy scenario" in addition to the three navigation-planning scenarios for each participant, as described in Section 4.1.1. The robot's proposed navigation policy for that scenario is either the only feasible non-dominated policy, or a severely suboptimal policy. This is to check whether the participant pays attention to identify when the robot presents a clearly best policy versus a clearly bad policy. We exclude the data from any participant who fails to provide reasonable answers for the total cost calculation and the "easy scenario".

4.1.4 Experiment Design

We recruited 106 participants on the Amazon Mechanical Turk (MTurk) platform, using the following standard criteria for selecting legitimate MTurk participants for human subjects research: the participants must be located in the United States or Canada, have completed over 50,000 HITs (Human Intelligence Tasks), and have over 97% approval rating for those HITs. Additionally, the participants must have passed our qualification test designed to verify that the participants understand how to apply a given cost profile to evaluate policies (see section 4.1.2).

We used a between-subject study design, where we randomly assigned each participant to either the control group or the treatment group. The participants in both groups first received an introduction of the robot navigation planning, and an instruction of their task as users to determine whether the robot's proposed navigation policy for each given planning scenario is the best option according to their cost profile. After the introduction and instruction, each participant is given a cost profile and a series of three navigation-planning scenarios, plus one additional validating scenario. The participant is instructed to use the assigned cost profile in all scenarios presented. The participant must answer all questions in each scenario before proceeding to the next. The participants in both groups received the same set of navigation-planning scenarios, in the same order. Each pairing of a navigation-planning scenario and a cost profile is assigned to three participants in the control group and three participants in the treatment group. In both the control and treatment groups, half of the navigation-planning scenarios are “preference-aligned” scenarios and the other half are “preference-misaligned” scenarios.

Using the validation criteria discussed in Section 4.1.3, we eliminated 7 participants who gave invalid answers. We collected valid data from the remaining 49 and 50 participants in the control group and the treatment group, respectively.

Control Group For each navigation-planning scenario, participants in the control group are given: (a) the visualization of the robot’s proposed navigation policy on the map, and (b) the estimated travel time, expected collision, and intrusiveness of the robot’s policy. See Figure 4.1 for an example scenario.

Treatment Group For each navigation-planning scenario, participants in the treatment group are given the same information as the control group, plus the consequence-oriented contrastive explanations. The number of alternatives presented in the contrastive explanations is between 1-3 depending on whether the robot’s proposed navigation policy can be improved with respect to each objective individually. See Figures 4.2, 4.3, and 4.4 for an example scenario and the generated explanations.

In each scenario, the participants in both the control and treatment groups are asked to: (i) indicate whether they think the robot’s proposed navigation policy is the best option according to their cost profile, and (ii) rate their confidence in their answer on a 5-point Liker scale. See Figure 4.5 for the question form.

We measure the following dependent variables:

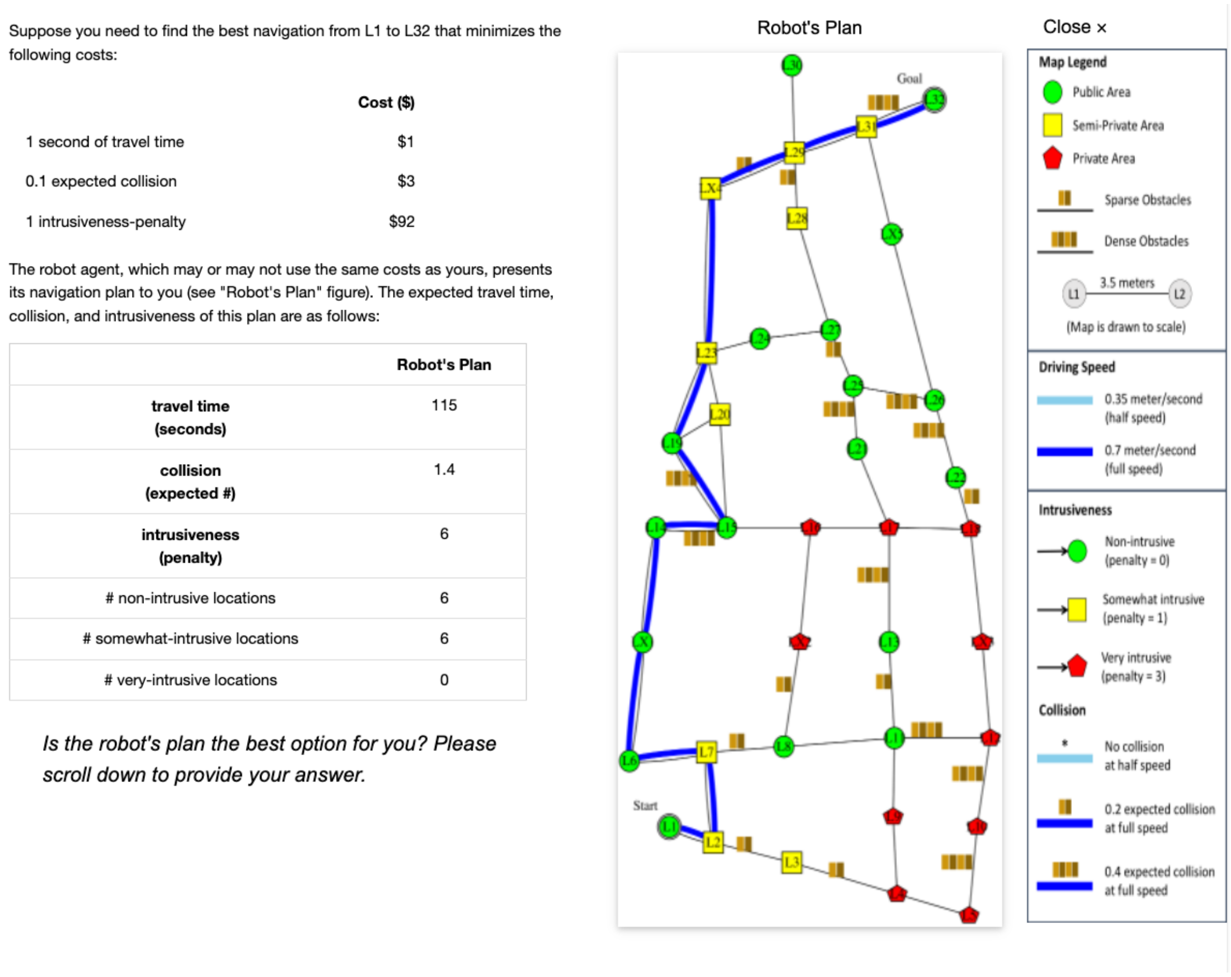
Correctness The binary correctness of each participant, for each scenario, in determining whether the robot’s proposed navigation policy is the best option with respect to their (given) preference.

Confidence The confidence level of each participant, for each scenario, in their answer, in the range of 0 (not confident at all), 1 (slightly confident), 2 (somewhat confident), 3 (fairly confident), and 4 (completely confident).

Reliable Confidence Score Combining the correctness and confidence measures, we can quantify how well the participants can assess the agent’s decisions. We assign a score ranging from -4 to $+4$ to each answer, where higher magnitude indicates higher confidence and negative value indicates incorrect answer. That is,

$$z(a,c) = \begin{cases} c & \text{if } a \text{ is correct} \\ -c & \text{if } a \text{ is incorrect} \end{cases}$$

where z is the score; a is the participant’s answer, which is either correct or incorrect; and c is the participant’s confidence level, ranging from 0 to 4 (confidence is reported on a 5-point Likert scale). The intuition is, we reward higher confidence when correct, and penalize higher confidence when incorrect.



Suppose you need to find the best navigation from L1 to L32 that minimizes the following costs:

	Cost ($)
1 second of travel time	$1
0.1 expected collision	$3
1 intrusiveness-penalty	$92

The robot agent, which may or may not use the same costs as yours, presents its navigation plan to you (see "Robot's Plan" figure). The expected travel time, collision, and intrusiveness of this plan are as follows:

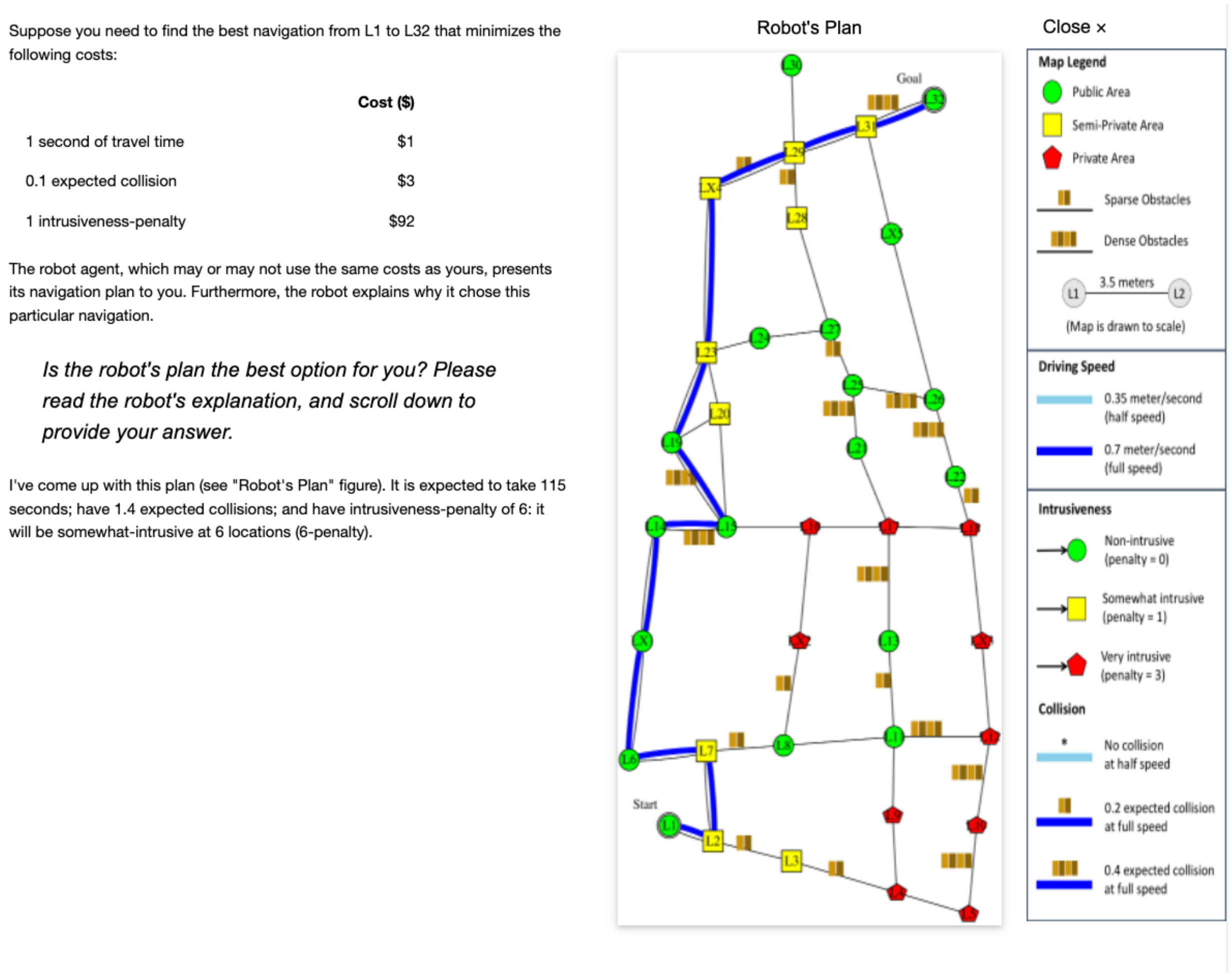
	Robot's Plan
travel time (seconds)	115
collision (expected #)	1.4
intrusiveness (penalty)	6
# non-intrusive locations	6
# somewhat-intrusive locations	6
# very-intrusive locations	0

Is the robot's plan the best option for you? Please scroll down to provide your answer.

Figure 4.1: Participants in the control group are presented with: (a) a navigation-planning scenario consisting of the map, the start and destination locations, and the cost profiles; (b) the visualization of the agent's proposed navigation policy on the map; and (c) the estimated travel time, expected collision, and intrusiveness of the agent's policy.

Hypotheses We have two central hypotheses in this study:

- **H1**: Participants in the treatment group (i.e., those who receive the consequence-oriented contrastive explanations of the policy) are more likely to correctly determine whether the robot's proposed policy is in line with their preference than participants in the control group (i.e., those who only receive the predicted objective values of the policy).
- **H2**: Participants in the treatment group have higher confidence in their determination than participants in the control group.



Suppose you need to find the best navigation from L1 to L32 that minimizes the following costs:

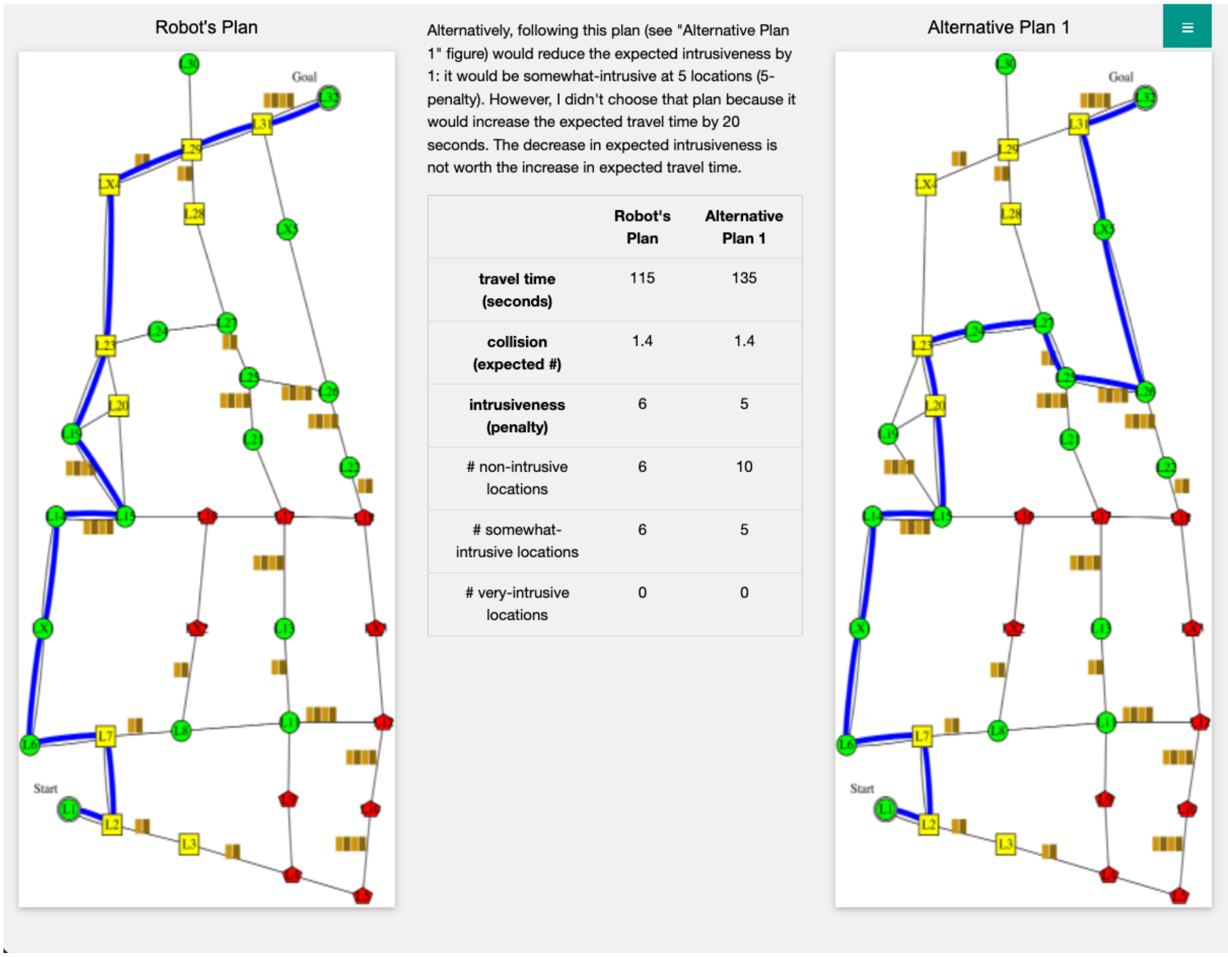
	Cost ($)
1 second of travel time	$1
0.1 expected collision	$3
1 intrusiveness-penalty	$92

The robot agent, which may or may not use the same costs as yours, presents its navigation plan to you. Furthermore, the robot explains why it chose this particular navigation.

Is the robot's plan the best option for you? Please read the robot's explanation, and scroll down to provide your answer.

I've come up with this plan (see "Robot's Plan" figure). It is expected to take 115 seconds; have 1.4 expected collisions; and have intrusiveness-penalty of 6: it will be somewhat-intrusive at 6 locations (6-penalty).

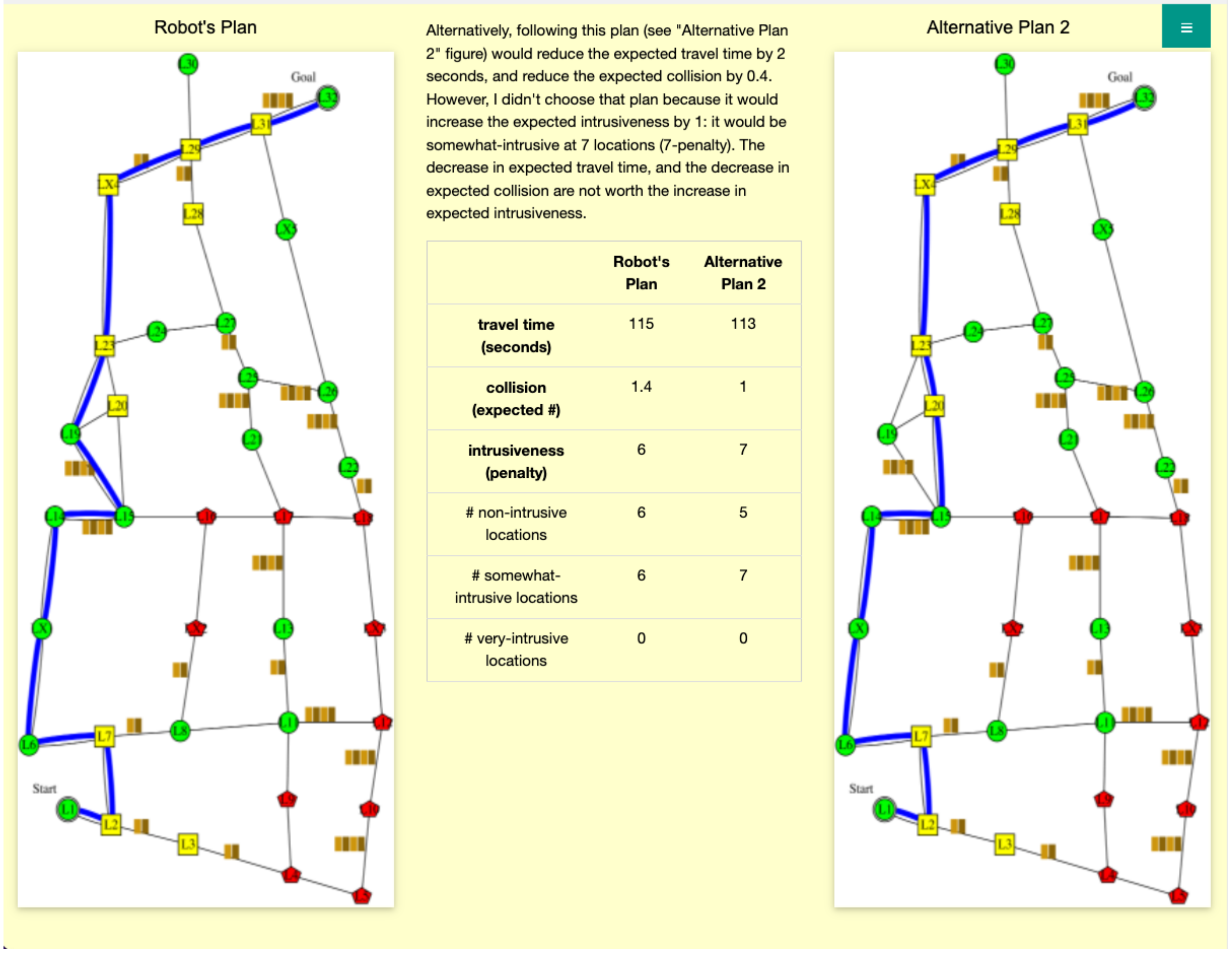
Figure 4.2: Participants in the treatment group are presented with: (a) a navigation-planning scenario consisting of the map, the start and destination locations, and the cost profiles; (b) the visualization of the agent's proposed navigation policy on the map; and (c) the first part of the agent's explanation, which describes the estimated travel time, expected collision, and intrusiveness of the agent's policy.



Alternatively, following this plan (see "Alternative Plan 1" figure) would reduce the expected intrusiveness by 1: it would be somewhat-intrusive at 5 locations (5-penalty). However, I didn't choose that plan because it would increase the expected travel time by 20 seconds. The decrease in expected intrusiveness is not worth the increase in expected travel time.

	Robot's Plan	Alternative Plan 1
travel time (seconds)	115	135
collision (expected #)	1.4	1.4
intrusiveness (penalty)	6	5
# non-intrusive locations	6	10
# somewhat-intrusive locations	6	5
# very-intrusive locations	0	0

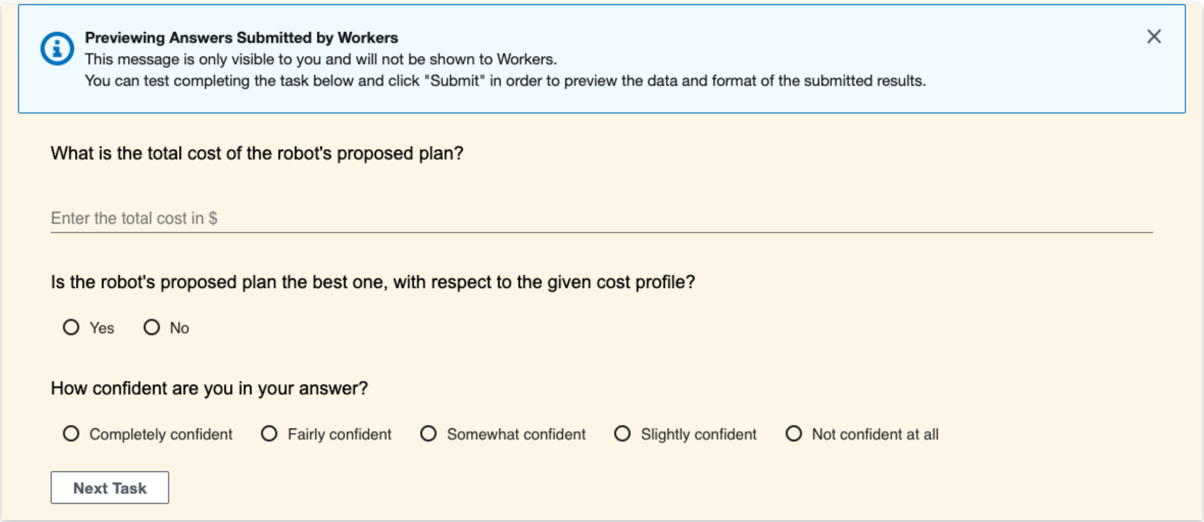
Figure 4.3: Participants in the treatment group are presented with the second part of the agent's explanation, which provides consequence-oriented contrastive explanations. This image shows the contrastive explanation of the first alternative policy.



Alternatively, following this plan (see "Alternative Plan 2" figure) would reduce the expected travel time by 2 seconds, and reduce the expected collision by 0.4. However, I didn't choose that plan because it would increase the expected intrusiveness by 1: it would be somewhat-intrusive at 7 locations (7-penalty). The decrease in expected travel time, and the decrease in expected collision are not worth the increase in expected intrusiveness.

	Robot's Plan	Alternative Plan 2
travel time (seconds)	115	113
collision (expected #)	1.4	1
intrusiveness (penalty)	6	7
# non-intrusive locations	6	5
# somewhat-intrusive locations	6	7
# very-intrusive locations	0	0

Figure 4.4: Participants in the treatment group are presented with the second part of the agent's explanation, which provides consequence-oriented contrastive explanations. This image shows the contrastive explanation of the second alternative policy.


Previewing Answers Submitted by Workers
This message is only visible to you and will not be shown to Workers.
You can test completing the task below and click "Submit" in order to preview the data and format of the submitted results.

What is the total cost of the robot's proposed plan?

Enter the total cost in $

Is the robot's proposed plan the best one, with respect to the given cost profile?

○ Yes ○ No

How confident are you in your answer?

○ Completely confident ○ Fairly confident ○ Somewhat confident ○ Slightly confident ○ Not confident at all

Figure 4.5: Participants in both the control and the treatment groups are asked to: (a) compute the total cost of the agent's proposed policy; (b) determine whether the agent's policy is the best option for their given cost profile; and (c) give a confidence rating of their answer.

4.2 Analysis and Results

We collected 297 valid data points from 99 participants. Each participant provides three data points from working with three navigation-planning scenarios (we exclude the validating scenarios from the analyses). From the total of 80 unique questions items we generated, as described in Section 4.1.1, 48 unique question items were randomly selected to be used in the study.

Our goal is to analyze the effects of the explanation types (i.e., consequence-oriented contrastive explanations vs. baseline explanations) and the scenario types (i.e, "preference-aligned" vs. "preference-misaligned" scenarios) on the correctness of the participants' answers and their confidence levels. These are the *fixed effects*, as both factors are controlled in our experiment. However, since each participant provided multiple data points (i.e., each participant worked with three scenarios), and different question items (i.e., 48 question items) were used in the study, there may be *random effects* from the individual participants (e.g., some participants may be better at the task than others) and from the questions (e.g., some questions may be more difficult to answer than others). Therefore, in our statistical analyses, we accounted for those random effects using mixed-effects models [67]. The results presented are from the random intercept-only model, as the other possible mixed-effects models produce similar results (i.e., not statistically significantly different).

Table 4.1 shows the regression analysis results. First column shows the mixed-effects logistic regression result for the *Correctness* binary outcome variable. Second and third columns show the linear mixed-effects regression results for the *Confidence* and the *Reliable Confidence Score* (or *Score* for short) outcome variables, respectively. For all regression analyses, the predictor variables are: (i) binary flag *is given contrastive explanations* indicating whether the participants are given contrastive explanations (i.e., whether they are in the treatment group, as opposed to the control group), and (ii) binary flag *is preference-misaligned scenario* indicating whether the

	Dependent variable:		
	Correctness	Confidence	Score
is given contrastive explanations	1.34*** (0.32)	0.42** (0.17)	1.72*** (0.36)
is preference-misaligned scenario	−1.01*** (0.33)	−0.01 (0.09)	−1.10*** (0.37)
(Intercept)	0.75*** (0.27)	2.83*** (0.12)	0.86*** (0.32)
Random Effects			
# of Participants	99	99	99
Participants Variance	0.25	0.54	1.06
# of Question Items	48	48	48
Question Items Variance	0.29	0.00	0.45
Residual Variance		0.42	6.22
Observations	297	297	297
Marginal R^2	0.16	0.04	0.12
Conditional R^2	0.27	0.58	0.29
Note:			*p<0.1; **p<0.05; ***p<0.01

Table 4.1: User Study Results

participants are given a “preference-misaligned” scenario, as opposed to a “preference-aligned” scenario. There is no statistically significant interaction between the two predictor variables on either the correctness or confidence. Thus, we do not include the interaction between the predictor variables in our final regression models.

All regression analyses account for two random effects: the random intercepts for individual participants and different question items. We use the random intercept-only models in our final regression models, as the other possible mixed-effects models produce similar results (i.e., are not statistically significantly different).

4.2.1 Participants’ Correctness

To test hypothesis H1, we use mixed-effects logistic regression to find the correlation between the experimental conditions (i.e., control vs. treatment group, and preference-aligned vs. preference-misaligned scenario) and the correctness of the participants’ answers, while accounting for the random intercepts for individual participants and different question items.

The R notation for this mixed-effect model is:

$$\text{correctness} \sim \text{group} + \text{scenario} + (1 \mid \text{participant}) + (1 \mid \text{questionItem})$$

where “group” is 0 and 1 for the control and treatment group, respectively, and “scenario” is 0 and 1 for the preference-aligned and preference-misaligned scenario, respectively. We use glmer model from lme4 [9] R package with family $=$ “binomial” for a mixed-effect logistic regression model.

Result Our consequence-oriented contrastive explanations have a significant positive effect on the participants’ correctness. The logistic regression coefficient of “group” is 1.34. That is, the *odds* of the participants in the treatment group being correct is on average $e^{1.34} \approx 3.8$ times higher than that of the participants in the control group, with 95% confidence interval $[2.03, 7.12]$. (Fixed-effect logistic regression coefficient estimate = 1.34; standard error = 0.32.) **H1 is supported**.

The type of scenarios has a significant effect on the participants’ correctness. Although, it is at an order of magnitude smaller than the positive effect of the contrastive explanations. The logistic regression coefficient of “scenario” is -1.01. That is, the odds of the participants being correct (regardless of which group they are in) is on average $e^{-1.01} \approx 0.36$ times lower in the “preference-misaligned” scenario than in the “preference-aligned” scenario, with 95% confidence interval $[0.19, 0.70]$. (Fixed-effect logistic regression coefficient estimate = -1.01; standard error = 0.33.)

Table 4.2 shows the probabilities of the participants correctly answering the questions, in each of the experimental conditions. The probabilities are calculated from the logistic regression:

$$log\left(\frac{p}{(1-p)}\right) = \beta_0 + \beta_1 \cdot \text{group} + \beta_2 \cdot \text{scenario}$$

where $\beta_0 = 0.75, \beta_1 = 1.34$, and $\beta_2 = -1.01$ (from the *Correctness* column in Table 4.1).

Pr(Correct)	Preference-aligned Scenario (scenario $= 0$)	Preference-misaligned Scenario (scenario $= 1$)	Overall
Control Group (group $= 0$)	0.68	0.43	0.56
Treatment Group (group $= 1$)	0.89	0.75	0.82

Table 4.2: Probabilities of the participants correctly assessing the agent's policy, in each of the experimental conditions.

4.2.2 Participants' Confidence

To test hypothesis H2, we use linear mixed-effects regression to find the correlation between the experimental conditions (i.e., control vs. treatment group, and preference-aligned vs. preference-misaligned scenario) and the confidence levels of the participants, while accounting for the random intercepts for individual participants and different question items.

The R notation for this mixed-effect model is:

$$\text{confidence} \sim \text{group} + \text{scenario} + (1 \mid \text{participant}) + (1 \mid \text{questionItem})$$

where "group" is 0 and 1 for the control and treatment group, respectively, and "scenario" is 0 and 1 for the preference-aligned and preference-misaligned scenario, respectively. We use lmer model from lme4 [9] R package for a mixed-effect linear regression model.

Result Our consequence-oriented contrastive explanations have a significant positive effect on the participants' confidence. The linear regression coefficient of "group" is 0.42. That is, the confidence level of the participants in the treatment group is on average 0.42 higher than that of the participants in the control group, with 95% confidence interval $[0.09, 0.74]$. (Fixed-effect linear regression coefficient estimate = 0.42; standard error = 0.17.) **H2 is supported**. The effect size is medium: Westfall et al.'s [100] $d = 0.43$, analogous to Cohen's d.

We do not observe a statistically significant effect of the type of scenarios on the participants' confidence ($p = 0.89$).

Table 4.3 shows the confidence levels of the participants in their assessment of the agent's policy, in the control group and the treatment group. The confidence levels are calculated from the linear regression:

$$y = \beta_0 + \beta_1 \cdot \text{group}$$

where $\beta_0 = 2.83$ and $\beta_1 = 0.42$ (from the *Confidence* column in Table 4.1).

4.2.3 Participants' Reliable Confidence Scores

Similar to Section 4.2.2, we use linear mixed-effects regression to find the correlation between the experimental conditions and the scores, while accounting for the random intercepts for individual participants and different question items.

	Confidence Level (Overall)
Control Group (group = 0)	2.8 out of 4
Treatment Group (group = 1)	3.2 out of 4

Table 4.3: Confidence levels of the participants in their assessment of the agent's policy, in the control group and the treatment group. Scenario type does not have a statistically significant effect on the confidence.

Result Our consequence-oriented contrastive explanations have a significant positive effect on the participants' scores. See the *Score* column in Table 4.1 for results. The linear regression coefficient of "group" is 1.72. That is, the score of the participants in the treatment group is on average 1.72 higher than that of the participants in the control group, with 95% confidence interval $[1.03, 2.42]$. (Fixed-effect linear regression coefficient estimate = 1.72; standard error = 0.36.) The effect size is medium: Westfall et al.'s $d = 0.62$.

The type of scenarios has a significant effect of the participants' scores. Although, it is at an order of magnitude smaller than the positive effect of the contrastive explanations. The linear regression coefficient of "scenario" is -1.10. That is, the score of the participants (regardless of which group they are in) is on average 1.10 lower in the "preference-misaligned" scenario than in the "preference-aligned" scenario, with 95% confidence interval $[-1.81, -0.38]$. (Fixed-effect linear regression coefficient estimate = -1.10; standard error = 0.37.) The effect size is medium: Westfall et al.'s $d = 0.40$.

4.3 Discussion

The results from our user study show that our consequence-oriented contrastive explanation approach significantly improves the users' ability to correctly determine whether the agent's planning solution is best with respect to their preferences, as well as their confidence in the determination. The users who are given the contrastive explanations are, on average, 3.8 times more likely to be correct than those who are given the baseline explanations, which only provides the expected objective values of the agent's policies. The contrastive explanations provide a moderate improvement in the users' confidence – with a medium effect size $d = 0.43$. As shown in Table 4.2, overall, the users who receive the baseline explanations have, on average, 0.56 probability of being correct, which is approximately the random chance. Given the consequence-oriented contrastive explanations, the users have, on average, 0.82 probability of being correct. Table 4.3 shows that contrastive explanations improve the users' confidence level on average from 2.8 (i.e., slightly-confident to somewhat-confident) to 3.2 (i.e., somewhat-confident to fairly-confident) out of 4 (i.e., completely-confident).

The results also show that, when the agent's objective function is misaligned with the user's preference, it is more difficult for the user to recognize that the agent's planning solution is not the best option. On average, the users are 0.36 times less likely to be correct in this type of scenarios. Notably, without contrastive explanations, the users have worse-than-random chance, 0.43 of correctly recognizing the misalignment. Although, it does not appear that such preference-misaligned scenarios significantly affect the users' confidence in their answers.

Recommendation for Explainable Planning and Agency

Aside from serving as a proxy for measuring the user's understanding of the agent's rationale in multi-objective planning, we argue that the user's ability to assess whether the agent's decisions are best according to their own preference is useful in a wide range of planning and sequential decision-making applications. Potential misalignment between the agent's objective function and the user's preference is inevitable, due to changing preferences over time and contexts, and other limitations of preference learning [37, 59]. Our results show that the users have a particularly low probability of correctly assessing the agent's decisions *when facing value misalignment and given baseline explanations*. As shown in Table 4.2, the probability of being correct given baseline explanations is 0.68 and 0.43 in the "preference-aligned" and "preference-misaligned" scenarios, respectively. That means, in absence of sufficient explanations, the users have the tendency to agree with the agent's decisions, and potentially overtrust the agent even when it in fact made the wrong decisions. Thus, we argue that there is a need to improve the users' assessment capability to allow them to properly calibrate their trust in the agent's decisions.

To this end, as our results demonstrate significant positive effects of our explanation approach on the users' assessment capability and confidence, we recommend that consequence-oriented contrastive explanation is a promising approach for explainable planning and explainable agency.

Chapter 5

User-Guided Explainable Planning

In Chapter 3 on explainable planning for multi-objective Markov decision processes, we contribute a contrastive explanation approach that explains the expected consequences of the agent's policy on its objectives and the tradeoff rationale to resolve any competing objectives. The key part of our approach relies on computing a set of alternative policies to be used as *foils* in a contrastive explanation, explaining why the agent had chosen its solution policy by means of arguing against other "foil" alternatives. The effectiveness of contrastive explanations depends on choosing the appropriate foils. A reasonable choice for contrastive foils is a small set of Pareto optimal alternative policies nearby the agent's policy in the value space – as they indicate the inflection point of the agent's tradeoff decisions.

However, this approach does not take into account the *explainee*, who may have expectations or a mental model of how the agent should perform its task. The shortcoming of our prior explanation approach is that the generated foils in a contrastive explanation are oblivious to any questions the explainee may have about the agent's policy. The explainee may have in mind an alternative behavior that they think the agent should follow, but such behavior may be sub optimal (e.g., due to the explainee's crude mental model of the agent's task). Thus, it cannot be selected as a contrastive foil by the approach presented in Chapter 3. Generally, if the explainee has a *why-not* alternative behavior in mind that is not addressed by the generated foils in a contrastive explanation, then the explanation becomes less effective.

In this chapter, we propose a theoretical framework and approaches for *user-guided* explainable planning, in which we generate explanations for the agent's policy in response to the explainee's specific questions. The explainee is able to ask "why-not"-style questions about a specific behavior or properties of behavior that the agent's policy does not exhibit. Our approach generates contrastive explanations with foils that directly address the behavior or properties of behavior in question.

5.1 Overview of Approach

The goal of our approach is to enable automated planning agents to answer the users' questions about specific hypothetical behavior of their interest. Such questions are in the form of *why-not* queries, in which the users ask the agent: *"Why do you not plan to do X instead?"*, where *X* is a placeholder for a description of a (partial) behavior or a pattern of behavior that differs from one that the agent had originally planned. In this work, we focus on MDP planning with multiple optimization objectives. Our approach aims to provide *consequence-oriented contrastive explanations* to the users' why-not queries. Namely, it explains what the subsequent decisions and their expected effects on the agent's planning objectives would be, if the agent were to plan to act according to what the users queried. It then argues that the expected effects on the planning objectives are less desirable, with respect to the agent's perspective, compared to those of the agent's original planned behavior.

The main idea of our approach is to formulate the user's why-not query as a constraint on the planning MDP model, and compute a solution policy for the constrained MDP as the alternative behavior of the agent that conforms to the user's query. This alternative policy of the agent will be used as a foil in a contrastive explanation for why the agent did not plan to act as the user expected. Our approach focuses on the types of why-not queries that can be described as: (i) state-action decisions in the agent's policy (Markov queries), (ii) sequential or temporal patterns in the agent's policy (non-Markov queries), or (iii) bounds on the expected objective values of the agent's policy.

The important consideration in handle non-Markov queries in the user-guided explanation framework is: the computed policies for contrastive foils that satisfy the queried temporal patterns may be non-Markovian with respect to the state abstraction in the original MDP problem. The difference between the Markovian reward assumption in the agent's original reasoning, and the non-Markovian approach used to generate user-queried contrastive foils, must be acknowledged. Nonetheless, we posit that allowing contrastive foils to challenge the assumption in the agent's formal planning framework can be informative for the users to understand and assess the optimality criteria they desire for the agent's task.

5.1.1 Motivating Example

Consider the following indoor delivery robot scenario: Suppose the robot is tasked to navigate across the building to deliver an item, and it has an objective to minimize the duration of the task. Along the way to the destination, a locked door is blocking the robot from traveling the shortest path. Upon reviewing the robot's policy, the user sees that when the robot arrives at the locked door, it turns to the adjacent hallway and takes the more roundabout path to the destination. The user has a knowledge that there are people behind the locked door and that the robot is able to call people for help to unlock the door and let it pass through. The user thus asks the question: why would the robot not call for help and wait until the door is unlocked? Such query describes only a partial behavior. However, the only rational behavior of the agent – with respect to its time minimization objective – once the door is unlocked is to continue traveling the shortest path to its target. An explanation to this query therefore must answer why the robot does not call for help, wait until the door is unlocked, and then travel through the shortest path.

In this work, we propose to respond to such a why-not query by explaining the impact of the agent behaving according to the query on the subsequent decisions it would make, and the overall impact on the planning objectives. It argues that such impact is less desirable (with respect to the agent's cost objective) compared to that of the agent's original policy. For instance, an explanation in the delivery robot scenario could be that: if the robot were to wait for the people to unlock the door, then the overall duration of the task is longer than if the robot were to take the roundabout path according to its original policy (because the wait time is long).

5.1.2 User-Guided Explainable Planning Framework

We propose a framework for user-guided explainable MDP-based planning, which builds upon our explainable MDP-based planning framework (XMDP) presented in Chapter 3. Our framework supports the user to guide which part of the agent's behavior the explanation should address, and what "foil" behavior should be addressed in the contrastive explanation. We propose a consequence-oriented contrastive explanation approach for answering three types of why-not queries:

- *Single-decision query*. This is a state-action decision that is *not* made in the agent's policy. We also refer to this type of queries as *Markov queries*, as the action decision depends only on the current state.
- *Sequential pattern query*: This is a sequential or temporal pattern of the state variables visited in the agent's policy. We also refer to this type of queries as *non-Markov queries*.
- *Value-based query*. This is a bound on a particular expected objective value of the agent's policy.

We provide a set of query templates that the users can select and instantiate to ask the agent why-not questions about its policy.

Additionally, in Section 6.2, we discuss an early investigation and a future work direction of an approach to support iterative query and explanation, enabling the user to ask a follow-up query after each time they receiving an explanation to their previous query. The goal of this approach is to allow the users to iteratively refine their queries as they gain more understanding of the agent's planning rationale, and to explore a larger space of the agent's possible behavior. We propose an early prototype of this approach by building and maintaining a tree structure of *hypothetical* planning models, where each hypothetical model is used to generate a why-not explanation in response to a user query asked on its parent hypothetical model, combined with the previously asked queries along the path from the parent model to the root. The tree of hypothetical models allows the user to ask a new query at any node in the tree, effectively exploring the space of possible behavior of the agent and the space of explanations.

Next, we discuss the details of our user-guided explainable planning approach. Section 5.2 provides the background on linear temporal logic and Markov decision process planning with linear temporal logic constraints, which are the formal frameworks we use to capture the user's why-not queries and generate explanations. Section 5.3 discusses the types of queries our approach supports and how we formalize those queries. Section 5.4 discusses our methods to generate why-not explanations for the types of queries.

5.2 Background

This section provides a background on technical approaches we use in capturing and generating explanations to why-not queries. Section 5.2.1 discusses Linear Temporal Logic (LTL), which we use for formulating behavioral queries from the users. Section 5.2.2 discusses the approach for Markov decision process planning with an LTL property as a constraint, which we use for finding an appropriate behavioral foil for a contrastive explanation to a why-not behavioral query.

5.2.1 Linear Temporal Logic

Linear Temporal Logic (LTL) provides an expressive grammar for describing temporal behavior of a system [75]. An LTL specification φ is constructed from a set of *atomic propositions* Π over the states of the system S, the standard Boolean operators, and a set of temporal operators. The truth value of φ is determined on an infinite sequence of states (i.e, a trace) $\tau = s_0, s_1, ...$, where each state s_i has truth assignments for all propositions in Π. The notation $\tau, t \models \varphi$ indicates that φ holds at time t. The trace τ satisfies φ (denoted by $\tau \models \varphi$) if and only if $\tau, 0 \models \varphi$. The minimal syntax of LTL can be described as follows:

$$\varphi ::= p \,|\, \neg\varphi_1 \,|\, \varphi_1 \vee \varphi_2 \,|\, \mathbf{X}\varphi_1 \,|\, \varphi_1 \mathbf{U} \varphi_2 \tag{5.1}$$

where $p \in \Pi$ is an atomic proposition; φ_1 and φ_2 are valid LTL formulas. The operator $\mathbf{X}$ is the 'next' operator, where $\mathbf{X}\varphi$ evaluates to true at time t if φ evaluates to true at time $t+1$. The operator $\mathbf{U}$ is the 'until' operator, where $\varphi_1\mathbf{U}\varphi_2$ evaluates to true at time t_1 if φ_2 evaluates to true at some time $t_2 \geq t_1$, and φ_1 evaluates to true for all time steps t in the range $t_1 \leq t \leq t_2$.

In addition to the minimal syntax, LTL formulas can be described using the additional first order logic operators $\wedge$ ('and') and $\implies$ ('implies'), and the additional higher-order temporal operators $\mathbf{F}$ ('eventually'), $\mathbf{G}$ ('globally'), $\mathbf{W}$ ('weak until'), and $\mathbf{R}$ ('release'). $\mathbf{F}\varphi$ evaluates to true at time t if φ evaluates to true for some $t' \geq t$. $\mathbf{G}\varphi$ evaluates to true at time t if φ evaluates to true for all $t' \geq t$. $\varphi_1\mathbf{W}\varphi_2$ is equivalent to $\varphi_1\mathbf{U}(\varphi_2 \vee \mathbf{G}\varphi_1)$, which is similar to the 'until' operator semantics except that it does not require φ_2 to eventually be true. $\varphi_1\mathbf{R}\varphi_2$ evaluates to true at time step t if either: φ_2 holds true until $t_1 \geq t$, at which φ_1 also becomes true, or φ_1 never became true and φ_2 holds true for all time steps $t' \geq t$.

To determine whether a trace τ satisfies an LTL property φ, we can construct a *deterministic Rabin automaton* (DRA) from φ that *accepts* the run of τ on the Rabin automaton if and only if $\tau \models \varphi$.

Definition 5.2.1. A *deterministic Rabin automaton* is a tuple $\mathcal{R} = \langle Q, \Sigma, \delta, q_0, F \rangle$ where Q is the set of states; Σ is the input alphabet; $\delta : Q \times \Sigma \to Q$ is the transition function; q_0 is the initial state; and F is the acceptance condition: $F = \{(G_1, B_1), ..., (G_{n_F}, B_{n_F})\}$ where $G_i, B_i \subset Q$ for $i = 1, ...n_F$.

A *run* of a Rabin automaton is an infinite sequence of states $r = q_0, q_1, ...$ where $q_0 \in Q$ and for all $i > 0, q_{i+1} \in \delta(q_i, \sigma$ for some input $\sigma \in \Sigma$. Given a run r of the Rabin automaton, $\inf(r) \in Q$ is the set of states that are visited infinitely often in the sequence r. A run r is *accepting* if there exists $i \in 1, ..., n_F$ such that:

$$\inf(r) \cap G_i \neq \emptyset \quad \text{and} \quad \inf(r) \cap B_i = \emptyset \tag{5.2}$$

We can construct a DRA corresponding to any LTL property φ defined over a set of atomic propositions Π, denoted as $\mathcal{R}_\varphi$. The DRA $\mathcal{R}_\varphi$ has the input alphabet $\Sigma = 2^\Pi$, and accepts all and only *words* over the alphabet Π that satisfy φ [83].

In this work, we use LTL formulas to capture behavior or properties of behavior of the system that the users would like to query. Section 5.3.1 discusses our approach to formulate behavior-based queries from the users in LTL. The formulation of DRAs corresponding to LTL formulas enables us to generate why-not contrastive explanations in response to the behavioral queries, as discussed in Section 5.4.2.

5.2.2 Markov Decision Process Planning with Linear Temporal Logic Constraints

We can solve for a policy for a MDP such that the resulting traces of the MDP satisfy a LTL property. We use a *labeled* Markov decision process as a planning model, where the labels on the states are the truth assignments of all atomic propositions Π. In other words, the labels provide descriptions for the states using propositional logic. The LTL formulas that are applicable to the labeled MDP are built from the same atomic propositions Π.

In this section, we describe the formal technique proposed by [82] for computing a policy for a labeled MDP that satisfies a given LTL property. Their approach focuses on computing a policy that satisfies a given LTL property, but not simultaneously optimizing for other objective functions (i.e., objectives that are unrelated to satisfying the LTL property). For our use case, we need to additionally handle multi-objective optimization as well as LTL constraint satisfaction. Section 5.4.2 discusses our approach that applies the existing technique from [82] to do so.

Definition 5.2.2. A *labeled Markov decision process* is a tuple $\mathcal{M} = \langle S, \mathcal{A}, P, s_0, \Pi, L\rangle$, where S is the finite set of states of the MDP; A is the finite set of possible actions and $\mathcal{A} : S \to 2^A$ is the mapping from states to their applicable actions; $P : S \times A \times S \to [0, 1]$ is the transition probability function; $s_0 \in S$ is the initial state; Π is the set of atomic propositions, and $L : S \to 2^\Pi$ is the labeling function that labels the states with atomic propositions.

Given a labeled MDP $M = \langle S, \mathcal{A}, P, s_0, \Pi, L\rangle$ and an LTL specification φ. We want to find a *finite memory policy* for $\mathcal{M}$ such that the induced Markov chain has all of its runs satisfy φ with probability one.

Definition 5.2.3. A *finite memory policy* for $\mathcal{M}$ is a function $\pi : S^+ \to A$ such that $\pi(s_0s_1...s_n) \in \mathcal{A}(s_n)$ for all $s_0s_1...s_n \in S^+$, where S^+ is the set of all finite sequences of states in S.

A finite memory policy π for an MDP $\mathcal{M}$ induces a Markov chain with finite memory, denoted by $\mathcal{M}_\pi$. A *run* of a Markov chain is an infinite sequence of states $s_0, s_1, ...$, where s_0 is the initial state of $\mathcal{M}$, and for all i, $P(s_i, a, s_{i+1)}$ is nonzero for some action $a \in A$ of $\mathcal{M}$.

To compute a finite memory policy π^* for $\mathcal{M}$ such that the runs of $\mathcal{M}_{\pi^*}$ satisfy the LTL property φ, we can compose $\mathcal{M}$ and the deterministic Rabin automaton (DRA) $\mathcal{R}_\varphi = \langle Q, \Sigma, \delta, q_0, F\rangle$, whose acceptance condition corresponds to satisfaction of φ. The composition is a *Rabin weighted product MDP*, denoted as $\mathcal{P}$. The set of states $S_\mathcal{P}$ in $\mathcal{P}$ is a set of augmented states with components from the states in $\mathcal{M}$ and $\mathcal{R}_\varphi$. The set of actions $\mathcal{A}_\mathcal{P}$ in $\mathcal{P}$ is identical to the set of actions in $\mathcal{M}$.

Definition 5.2.4. A *Rabin weighted product MDP* or *product MDP* between a labeled MDP $\mathcal{M} = \langle S, \mathcal{A}, P, s_0, \Pi, L\rangle$ and a DRA $\mathcal{R}_\varphi = \langle Q, \Sigma, \delta, q_0, F\rangle$ is defined as a tuple $\mathcal{P} = \langle S_\mathcal{P}, \mathcal{A}_\mathcal{P}, P_\mathcal{P}, s_{\mathcal{P}0}, F_\mathcal{P}, W_\mathcal{P}\rangle$, where:

- $S_\mathcal{P} = S \times Q$ is the set of states.

- $\mathcal{A}_{\mathcal{P}}$ is the mapping from states to actions from the MDP $\mathcal{M}$: $\mathcal{A}_{\mathcal{P}}((s,q)) = \mathcal{A}(s)$.
- $P_{\mathcal{P}}$ is the transition probability function:

$$P_{\mathcal{P}}(s_{\mathcal{P}}, a, s'_{\mathcal{P}}) = \begin{cases} P(s,a,s') & \text{if } q' = \delta(q, L(s)) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

  where $s_{\mathcal{P}} = (s,q) \in S_{\mathcal{P}}$ and $s'_{\mathcal{P}} = (s',q') \in S_{\mathcal{P}}$.
- $s_{\mathcal{P}0} = (s_0, q_0) \in S_{\mathcal{P}}$ is the initial state.
- $F_{\mathcal{P}}$ is the acceptance condition given by

$$F_{\mathcal{P}} = \{(\mathcal{G}_1, \mathcal{B}_1), ..., (\mathcal{G}_{n_F}, \mathcal{B}_{n_F})\}$$

  where $\mathcal{G}_i = S \times G_i$ and $\mathcal{B}_i = S \times B_i$.
- $W_{\mathcal{P}} = \{W^i_{\mathcal{P}}\}_{i=1}^{n_F}$ is the set of reward functions corresponding to the acceptance condition. Each reward function $W^i_{\mathcal{P}} : S_{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is defined by:

$$W^i_{\mathcal{P}}(s_{\mathcal{P}}) = \begin{cases} w_G & \text{if } s_{\mathcal{P}} \in \mathcal{G}_i \\ w_B & \text{if } s_{\mathcal{P}} \in \mathcal{B}_i \\ 0 & \text{if } s_{\mathcal{P}} \in S_{\mathcal{P}} \setminus (\mathcal{G}_i \cup \mathcal{B}_i) \end{cases} \tag{5.4}$$

  where $w_G > 0$ is a positive reward and $w_B < 0$ is a negative reward. Note that our approach in Section 5.4.2 uses cost functions instead of reward functions. This is to make the model of the acceptance condition compatible with cost objective optimization in our planning applications. Nonetheless, the principles of the approach here hold for our approach as well.

To find a finite memory policy π^* for $\mathcal{M}$ such that $\mathcal{M}_{\pi^*}$ satisfies φ, it is sufficient to consider *stationary policies* for the corresponding product MDP $\mathcal{P}$. A stationary policy for $\mathcal{P}$ has a corresponding finite memory policy for $\mathcal{M}$.

Definition 5.2.5. A *stationary policy* π for a product MDP $\mathcal{P}$ is a function $\pi : S_{\mathcal{P}} \to A_{\mathcal{P}}$ that maps states to actions.

Let π be a stationary policy for $\mathcal{P}$, the Markov chain induced by applying π to $\mathcal{P}$ is denoted by $\mathcal{P}_\pi$. Let $r = s_{\mathcal{P}0}s_{\mathcal{P}1}...$ be a run of $\mathcal{P}_\pi$.

Definition 5.2.6. For any stationary policy π of a Rabin weighted MDP $\mathcal{P}$ that is a product of a MDP $\mathcal{M}$ and a DRA $\mathcal{R}_\varphi$ corresponding to a LTL formula φ, the induced Markov chain $\mathcal{M}_\pi$ satisfies φ with probability 1 if:

$$\begin{aligned} &Pr(\{r : \exists(\mathcal{G}_i, \mathcal{B}_i) \in F_{\mathcal{P}}(s) \\ &\mathit{inf}(r) \cap \mathcal{G}_i \neq \emptyset \wedge \mathit{inf}(r) \cap \mathcal{B}_i = \emptyset\}) = 1 \end{aligned}$$

where r is a run of $\mathcal{P}_\pi$.

Let i be an index of the Rabin acceptance condition $W_{\mathcal{P}}$ for the property φ. Each reward function $W^i_{\mathcal{P}}$ is designed to bias the policy towards satisfying the Rabin acceptance condition $(\mathcal{G}_i, \mathcal{B}_i)$: The states in $\mathcal{B}_i$ and in $\mathcal{G}_i$ are assigned negative reward w_B and positive reward w_G to incentivize the policy to only visit $\mathcal{B}_i$ states finitely often, and to visit $\mathcal{G}_i$ states infinitely often, respectively. The remaining states are assigned a neutral reward of 0. We rewrite the reward function $W^i_{\mathcal{P}}$ as a vector $\mathbf{W}^i \in \mathbb{R}^{|S_{\mathcal{P}}|}$ of dimension $|S_{\mathcal{P}}|$.

Definition 5.2.7. For $i \in \{1, ..., n_F\}$, the *expected discounted utility* for a policy π for $\mathcal{P}^i$ with a discount factor $0 < \gamma < 1$ is a vector $\mathbf{U}^i_\pi = [U^i_\pi(s_0)...U^i_\pi(s_{|S_{\mathcal{P}}|})]$ for $s_k \in S_{\mathcal{P}}$ such that:

$$\mathbf{U}^i_\pi = \sum_{n=0}^{\infty} \gamma^n P^n_\pi \mathbf{W}^i \tag{5.5}$$

where P_π is a $|S_{\mathcal{P}}| \times |S_{\mathcal{P}}|$ matrix of the probabilities $P_{\mathcal{P}}(s_{\mathcal{P}}, \pi(s_{\mathcal{P}}), s'_{\mathcal{P}})$.

Definition 5.2.8. For $i \in \{1, ..., n_F\}$ of the Rabin acceptance condition, a policy that maximizes the expected discounted utility for every state is an *optimal policy* $\boldsymbol{\pi}^*{}_i = [\pi^*_i(s_0)...\pi^*_i(s_{|S_{\mathcal{P}}|})]$:

$$\boldsymbol{\pi}^*{}_i = \arg\max_{\pi} \sum_{n=0}^{\infty} \gamma^n P^n_\pi \mathbf{W}^i \tag{5.6}$$

We can solve for a set policies $\{\pi^*_i\}^{n_F}_{i=1}$ where π^*_i is an optimal policy under the reward function $W^i_{\mathcal{P}}$. Then, we can use Definition 5.2.6 to determine whether any policy π^*_i satisfies φ with probability 1, by using existing efficient model checking algorithms [8].

Theorem 1. *Given a labeled MDP $\mathcal{M}$ and a LTL formula φ with a corresponding Rabin weighted product MDP $\mathcal{P}$. If there exists a (finite memory) policy $\bar{\pi}$ for $\mathcal{M}$ such that $\mathcal{M}_{\bar{\pi}}$ satisfies φ with probability 1, then there exists $i^* \in \{1, ..., n_F\}, \gamma^* \in [0, 1)$, and $w^*_B < 0$ such that any algorithm that optimizes the expected future utility of $\mathcal{P}^{i^*}$ with $\gamma \geq \gamma^*$ and $w_B \leq w^*_B$ will find a stationary policy for $\mathcal{P}$ that corresponds to $\bar{\pi}$.*

Theorem 1, proven by [81], shows that optimizing the expected discounted utility of $\mathcal{P}^i$, using sufficiently large γ and $|w_B|$, produces a policy π^*_i such that the induced Markov chain $\mathcal{M}_{\pi^*_i}$ satisfies φ with probability 1 *if* such a policy exists.

In our work, we use MDP planning with a LTL constraint to find an alternative behavior of the agent that conforms to a partial behavior or properties of a behavior that the user has queried. Section 5.4.2 discusses our approach to formulate a Rabin weighted product MDP from the agent's task planning labeled MDP and the DRA capturing a behavioral query from the user. An optimal policy computed from the product MDP is used as a foil in a why-not contrastive explanation to the query.

5.3 Why-Not User Queries

Users can pose a why-not question to ask a planning agent to explain (from the agent's perspective) the absence of a specific behavior or property of a behavior that they expected the agent to exhibit. A queried behavior or property of a behavior can be in many forms. In this work, we investigate an approach to capture why-not questions from two categories: behavior-based questions and value-based questions. Behavior-based questions target specific decisions or trajectories of decisions in an agent's policy. On the other hand, value-based questions target the evaluation of the agent's policy with respect to specific planning objectives. In this section, we discuss our approach to formulate why-not questions as behavior-based and value-based queries on Markov decision process policies.

5.3.1 Behavior-based Queries

Behavior-based queries describe how the user expects the agent to act differently from what the agent had planned. We consider two ways of expressing behavior-based queries: via Markov queries and non-Markov, or sequential-behavior, queries.

Markov Queries

Queries about alternative decisions at certain states in an agent's policy can capture basic why-not questions about the agent's behavior. For instance, in a shared autonomy setting of a human-UAVs team, a question may arise: why would the robot not transfer the control to its human operator when it gets near the boundary of its typical operating conditions, e.g., the robot is flying close to an adversarial terrain. In this case, a query is an alternative action of the robot – transferring control to the human – in a particular state – when the robot is flying close to an adversarial terrain. In this work, we refer to such query as a Markov query, where the decision to take a particular action only depends on the current state. More specifically, a Markov query is a pair of state and action: $\langle s_q, a'_q \rangle$, where s_q is a reachable state in the agent's policy being queried, and a'_q is an alternative action that is not taken in state s_q in the agent's policy.

Note that a query state s_q must be a fully described state (i.e., it must specifies the values of all state variables). This is to avoid any ambiguity in a single-decision query. However, if the users' why-not questions must be specified in terms of a subset of state variables, then sequential-behavior queries can be used, where propositions on the state variables can be described in the linear temporal logic properties.

We will discuss our approach to generate consequence-oriented contrastive explanations for Markov queries in Section 5.4.1.

Sequential-Behavior (non-Markov) Queries

Beyond Markov queries, users may have questions about particular patterns in an agent's behavior that are sequential or temporal in nature. As planning is sequential, such questions can capture users' expectation of an agent's behavior with greater expressivity than Markov queries. Sequential patterns in a policy can be expressed as a Linear Temporal Logic (LTL) property. In this work, we consider a sequential-behavior (or a non-Markov) query to capture a question: why does the agent not follow a particular sequential or temporal pattern φ, starting from a particular state s_q. Our approach assumes that user questions are already given in the form of linear temporal logic queries. The problem of translating user questions in natural language into the corresponding linear temporal logic queries is out of scope of this work.

User queries expressed as arbitrary LTL properties can be used in our approach. Table 5.1 shows an example set of interpretable LTL templates that can be used as a basis for forming sequential-behavior queries. These LTL templates are commonly used in software engineering

Template	Formula	Meaning
Global	$\mathbf{G}p$	p is true throughout the policy.
Eventuality	$\mathbf{F}p$	p eventually occurs (may later become false).
At-most-once	$\mathbf{G}(p \rightarrow (p\,\mathbf{W}\,\mathbf{F}\neg p))$	Only one contiguous interval exists where p is true.
Stability	$\mathbf{FG}p \wedge \mathbf{G}(p \rightarrow (p\,\mathbf{W}\,\mathbf{G}\neg p))$	p eventually occurs and stays true forever.
Until	$p\,\mathbf{U}\,q$	p is true until q eventually occurs.
Response	$\mathbf{G}(p \rightarrow \mathbf{XF}q)$	If p occurs, q eventually follows.
Precedence	$(q \wedge \neg p)\,\mathbf{R}\,\neg p$	If p occurs, q occurred in the past.

Table 5.1: Commonly used interpretable LTL templates [52] that can be used as a basis for forming sequential-behavior queries, where p and q are state propositions.

and verification [29], and have been used in explaining temporal differences between two sets of plan traces [52].

Formally, a why-not sequential-behavior query is a tuple $\mathcal{Q} = \langle s_q, \varphi \rangle$, where s_q is the query's starting state and φ is a LTL property describing the sequential behavior in question. Suppose that the agent had originally planned a policy π_0 for its task, and the user asks a why-not question about this policy. Here we assume that the user asks the question at a particular state, s_q, in the policy π_0. The sequential behavior in question, φ, is *not* satisfied by the policy π_0 starting from state s_q. Importantly, we assume that all state propositions used in a query φ are modeled the agent's world model.

The specific forms of φ in user queries will depend on problem domains. The following are some examples of the LTL properties for motion planning for mobile robots [34, 35, 55], and example properties derived from the basic subset of the LTL templates shown in Table 5.1: $\mathbf{G}\,p, \mathbf{F}\,p$, and $p\,\mathbf{U}\,q$.

Sequencing The user asks why the agent's policy does not follow a particular sequence of state propositions $p_1, p_2, ..., p_n$ in order. Here the states are not necessarily immediately following one another, as long as the order in the sequence is followed. For instance, in navigation-related domains, this can be a query about following a particular path (i.e., a sequence of locations) that is different from what the agent originally planned. The corresponding LTL property for the why-not query is: $p_1 \wedge \mathbf{F}\,p_2 \wedge ... \wedge \mathbf{F}\,p_n$

Avoiding Prohibited States The user asks why the agent does not avoid certain prohibited state propositions $o_1, o_2, ..., o_n$ while reaching for a goal s_g. For instance, in a mobile robot navigation scenario, the agent should avoid the locations $o_1, o_2, ..., o_n$ which have obstacles or are in hazardous zones, while navigating to a destination – instead of traveling through them as the agent originally planned. The corresponding LTL property for the why-not query is: $\neg(o_1 \vee o_2 \vee ... \vee o_n)\,\mathbf{U}\,s_g$.

Coverage The user asks why the agent does not cover all regions $r_1, r_2, ..., r_n$ of interest, in any order. For instance, in a mobile service robot scenario, the agent should visit the rooms and areas that it misses in its original policy. The corresponding LTL property for the why-not query is: $\mathbf{F} r_1 \wedge \mathbf{F} r_2 \wedge ... \wedge \mathbf{F} r_n$.

Holding-Until The user asks why the agent does not maintain certain conditions $c_1, c_2, ..., c_n$, until reaching a certain state proposition p. For instance, in a mobile service robot scenario, the agent should not enter a target room until it receives a permission to enter – instead of proceeding without permission as the agent originally planned. (Note that we assume that all state propositions used in a query are modeled the agent's world model. That is, in this example, the agent is assumed to have a knowledge of permissions to enter rooms.) The corresponding LTL property for the why-not query is: $(c_1 \wedge c_2 \wedge ... \wedge c_n) \, \mathbf{U} \, p$.

Mutual Exclusion The user asks why the agent holds these particular conditions $c_1, c_2, ..., c_n$ at once. For instance, the agent plans to use a combination of configuration settings that the user thinks should not be used together. The corresponding LTL property for the why-not query is: $\mathbf{G} \neg (c_1 \wedge c_2 \wedge ... \wedge c_n)$.

Note that the LTL property in a sequential-behavior query does not necessarily describe a complete policy of the agent. An explanation to a query $\mathcal{Q} = \langle s_q, \varphi \rangle$ essentially predicts what would happen if the agent behaves in a way that satisfy φ starting from state s_q, and contrasts the expected consequences of that behavior to those of the agent's original policy. In Section 5.4.2, we discuss our approach to generate contrastive explanations for why-not sequential-behavior queries.

5.3.2 Value-based Queries

We use value-based queries to capture quantitative properties that the user expects the agent's behavior to have. A quantitative property of an agent's behavior is expressed as an upper bound on the expected value of the agent's policy of a particular minimization objective (or, a lower bound for a maximization objective). For instance, consider an indoor mobile robot navigation scenario similar to one in Section 3.1.1. Suppose the robot is tasked to navigate from one room to another (e.g., to deliver an item). The user may expect that the robot should be able to complete the task within 4 minutes. If the robot's policy turns out to have a longer expected travel time than that, the user may ask a value-based query, such as, "Why not complete the task within 4 minutes?".

More formally, let $\mathcal{M}$ and π_0 denote a planning problem and the agent's solution policy, respectively. A value-based query on the policy π_0 on an objective ϕ is expressed as a constraint on the ϕ-value function, denoted as J_ϕ, for the planning problem $\mathcal{M}$: $J_\phi(s_q) \leq \theta_\phi$, where s_q is a query state (which can be the initial state of $\mathcal{M}$), and θ_ϕ is an upper bound value obtained from the user, such that $\theta_\phi < J_\phi^{\pi_0}(s_q)$. That is, the query upper bound value should be less than what the agent's current policy exhibits. The value θ_ϕ can be specified directly by the user. Otherwise, if the user only indicates that they expected the agent to perform better in objective ϕ than its current policy does, our approach will formulate the query using the constraint $J_\phi(s_q) \leq J_\phi^{\pi_0}(s_q) - \delta_\phi$ instead, where δ_ϕ is an appropriate offset as discussed in Chapter 3.

To respond to a value-based query, we use a contrastive explanation to explain the tradeoff decision that the agent made involving the objective in query. For instance, in the robot navigation scenario discussed above, an explanation in response to the query could be that the robot could take another path to the target room that is shorter in distance. However, the alternative path would be more intrusive to the building occupants as it goes pass private offices areas. Note that this contrastive explanation approach is the same the technical approach as described in Section 3.4.2. The difference is that instead of explaining all tradeoff decisions that the agent made, an explanation to a value-based query targets a tradeoff involving a specific objective of interest to the user.

To find an alternative policy for a contrastive explanation, we use the mixed-integer linear programming approach described in Section 3.5 to solve $\mathcal{M}$ with the following constraint on the ϕ-value function: $J_\phi(s_q) \leq \theta_\phi$.

In the case where no policy exists that satisfies the hard constraint from the query, the explanation will first indicate that θ_ϕ is not achievable. However, as the user's query can be imprecise and can have tolerance to a small value difference, our approach attempts to find an alternative policy whose ϕ value is close to θ_ϕ. To this end, we re-solve $\mathcal{M}$ again the following constraints: $J_\phi(s_q) < J_\phi^{\pi_0}(s_q)$ and $J_\phi(s_q) \leq_{\text{soft}} \theta_\phi$, using our approach discussed in Section 3.5.2. The first constraint is a hard constraint, ensuring that any alternative policy to π_0 will have an improvement in the ϕ objective. The second constraint is a soft constraint, which allows some violation of the upper bound θ_ϕ, depending on the design of the penalty function. Let π' denote a constraint satisfying policy, the resulting explanation is in the following form:

"I cannot find a policy whose expected [Obj. ϕ] *value is within* [θ_ϕ unit(s)]. *However, I found this policy* [π'] *that has the expected* [Obj. ϕ] *value of* [ϕ-value of π'].*"*, followed by a consequence-oriented contrastive explanation between the original solution policy π_0 and the computed alternative π', as discussed in Section 3.4.2.

5.4 Why-Not Explanations of User Queries

Why-not explanations allow users to ask about the absence of specific behaviors, or properties of behaviors, of agents. These can allow the users to address their unmet expectations of the agents' behaviors, or to explore potential alternative behaviors. This section presents our approaches to generate why-not explanations for behavior-based queries.[1] Section 5.4.1 and Section 5.4.2 discuss our explanation approaches for Markov queries and non-Markov queries, respectively.

5.4.1 Markov Query Why-Not Explanations

A Markov query captures a why-not question about an alternative decision at a certain state in an agent's policy. We propose to use a consequence-oriented contrastive explanation to respond to such query. That is, we explain that if the agent were to take the queried alternative action

[1] The explanation approach for value-based queries is discussed in Section 5.3.2.

at the queried state, the impact it would have on the subsequent decisions and on the planning objectives is less desirable (with respect to the agent's cost objective) compared to that of the agent's original policy.

Consider the example scenario described in Section 5.3.1 of shared autonomy in a human-UAVs team. A question may arise: why would the robot not transfer the control to its human operator when it gets near the boundary of its typical operating conditions, e.g., the robot is flying close to an adversarial terrain. In this scenario, an example of a consequence-oriented explanation could be that transferring the control to the human operator would take time and that increases the risk of being attacked by the adversary. Thus, the robot chooses to not transfer the control.

Our approach aims to generate consequence-oriented contrastive explanations for why-not Markov queries. Formally, this type of query is a state-action pair, denoted as $\langle s_q, a_q \rangle$. An explanation to a query presents the following information:

- Subsequent decisions that would follow if the planning agent took action a_q in state s_q – i.e., what a full alternative policy would look like.
- Impact of the query decision on the quality attributes of the whole alternative policy, in contrast to those of the agent's original policy. This provides a justification for the planning agent's original decision to not take action a_q in state s_q.

To find an alternative policy that takes the query decision $\langle s_q, a_q \rangle$, we also need to ensure that the policy has the *net effect* of the query action a_q. That is, the subsequent actions in the policy after a_q is taken do not revert its effect (an action's *effect* is defined in terms of a subset of state variables it changes). When the effect of a query action is reversible, such guarantee is generally not possible in arbitrary planning [36]. This is because there may be many possible sequences of state-action pairs after $\langle s_q, a_q \rangle$ that can undo the effect of a_q – *even when the sequences do not cycle back to the state* s_q. For instance, a mobile robot moves from point A to B, changes its configuration, and then moves back from point B to A. In this case, the net effect of the first move action is reverted (i.e., the robot's location is back to A), without the robot returning to its initial state (i.e., the robot's configuration has changed).

For MDP planning with *all positive costs*, we use the following heuristic to lessen this issue: Assume that the user has an intention for their query decision to lead to an improvement in a particular planning objective, which we will referred to as a *target objective*, compared to the original policy. If we find an alternative policy that takes the query decision and improves the target objective, that policy is likely to not revert the net effect of the query action – as reverting it would likely not improve the objective and accumulate more costs. (Note that improving a target objective will necessarily deteriorate at least one other planning objective, since the original policy is Pareto optimal.)

An exception when this heuristic will not work to this is if the query decision leads to worse value for the target objective. In such case, our approach will determine that there is no alternative policy, constrained by the query decision, that improves the target objective. Our approach will then compute a best alternative policy with respect to the original planning cost function, constrained only by the query decision but not by the target objective value. In this situation, if the effect of the query action is reversible, its net effect may be reverted. This is a key limitation of explaining a single Markov query approach. In Section 6.2, we will discuss an early investigation of a potential approach to mitigate this issue by means of iterative query and explanation.

Note that however, if the effect of the query action is irreversible, then our approach will be able to find an alternative policy with the intended effect of the query decision – even if that does not lead to the intended improvement of the target objective. For instance, in the shared autonomy example discussed above, the query action of transferring control to the human operator has an irreversible effect. Suppose the target objective of this query is to reduce the risk of getting attacked by the adversary. Our approach can find an alternative policy that does transfer control to the human operator when the robot is flying close to an adversarial terrain, and show that the action in fact leads to the increase risk of being attacked (due to the introduced robot-to-operator handover delay).

Problem Statement and Approach Generating a why-not explanation for a Markov query on a policy is a problem of finding an alternative policy that conforms to the query, and contrasting it to the policy being queried. The input to the problem is a tuple $\langle \mathcal{M}, \pi_0, q \rangle$, where $\mathcal{M}$ is a Markov decision process of the original planning problem, π_0 is an optimal policy computed from $\mathcal{M}$, and q is a why-not Markov query on π_0 at a state s_q. The output is an alternative policy π'_q that diverges from the original policy π_0 at state s_q, and its contrastive explanation to π_0 based on the expected planning objective values.

We assume an explicit user intention is given along with the query decision, in a form of which planning objective they expect to have improved. With this additional input, a why-not Markov query is described as a tuple $q = \langle s_q, a_q, \phi \rangle$, where s_q denotes a query state, a_q denotes a query action, and ϕ denotes a target objective. We assume that s_q is a reachable state within the policy π_0 being queried, and ϕ is one of the component objectives of the cost function of $\mathcal{M}$.

To find an alternative policy π'_q, we use planning with constraints based on the query q. We create a new planning problem $\mathcal{M}'_q$. Informally, $\mathcal{M}'_q$ models a similar planning problem to $\mathcal{M}$. It aims for the same goal as $\mathcal{M}$, but differs in that it starts from a query state (hence the subscription q), and has a constraint on the target objective ϕ. More formally, $\mathcal{M}'_q$ is equivalent to $\mathcal{M}$ except in the following ways: First, s_q is set to be the initial state of $\mathcal{M}'_q$. Second, a_q is set to be the only applicable action in state s_q. Third, we add a constraint to the planning problem such that a solution policy to $\mathcal{M}'_q$ must yield a better expected objective value of ϕ (e.g., lower value when ϕ is a minimization objective) than that of the policy π_0 starting from state s_q. Specifically, let $J_\phi^{\pi_0}$ be the value function for the objective ϕ of the policy π_0. $J_\phi^{\pi_0}(s_q)$ is the expected ϕ-value of the policy π_0 starting from state s_q. We denote this quantity as θ_ϕ, representing the threshold value of the ϕ-value constraint on $\mathcal{M}'_q$. Formally, the ϕ-value constraint is: $J_\phi^{\pi}(s_q) \leq \theta_\phi$ for any solution policy π of $\mathcal{M}'_q$. We use the mixed-integer linear programming approach described in Section 3.5.1 to solve for an optimal policy of $\mathcal{M}'_q$ with ϕ-value constraint. We denoted such policy as π'_q.

Since π'_q is computed from an initial state s_q, when providing a contrastive explanation, we must contrast π'_q to the partial policy that is a trajectory following π_0 that starts from state s_q. We denote such partial policy as π_{0q}, which can be obtained by filtering only the state-action pairs (s, a) in π_0 such that s is reachable from s_q following the state trajectory induced by π_0. If π'_q exists, then we use the policy explanation scheme described in Section 3.4 to generate an explanation, contrasting π'_q to π_{0q}, in the following form:

"If I were to [(1) perform action a_q] *when* [(2) in state s_q], *I would subsequently* [(3) follow this alternative policy π'_q]. *Although this would* [(4) improve the target objective ϕ – and other objectives if any – by these amounts], *it would also* [(5) worsen these other objectives by these amounts]."

The statement fragments (1) and (2) reiterate the state and action query. (3) describes the subsequent alternative policy that the agent would follow after taking action a_q in state s_q. The policy is described using a graphical illustration, as discussed in Section 3.4.1. (4) describes the quantitative improvement in the target objective ϕ, and potentially any other objective improvement, achieved by this alternative policy π'_q, compared to π_{0q}. (5) describes the quantitative compromise on other objective(s). Note that since π_0 is a Pareto optimal policy for $\mathcal{M}$, there is no policy that dominates it on all objectives. Thus, for any π'_q, there must be at least one compromised objective value compared to that of π_{0q}.

However, if π'_q does not exist (i.e., there is no policy in $\mathcal{M}'_q$ that satisfies the ϕ-value constraint), that means the query decision does not lead to the user's intended effect on the target objective. In this case, we remove the ϕ-value constraint and solve for an optimal policy for $\mathcal{M}'_q$, denoted as $\tilde{\pi}'_q$. Similar to the explanation scheme above, we generate an explanation contrasting $\tilde{\pi}'_q$ to π_{0q} in the following form:

"If I were to [(1) perform action a_q] *when* [(2) in state s_q], *I would subsequently* [(3) follow this alternative policy $\tilde{\pi}'_q$]. *However, this does not* [(4) improve the target objective ϕ, as value of ϕ remains the same or worsen by this amount]. [(5) Improvement or deterioration in other objectives, if any]."

The statement fragments (1), (2), and (3) are generated the same way as the explanation scheme above. (4) indicates that the alternative policy $\tilde{\pi}'_q$ does not lead to an improvement in ϕ, or even worsen it. Optionally, if $\tilde{\pi}'_q$ yields different values of other planning objectives, (5) describes those differences.

Our why-not explanation approach demonstrates to the user what would happen if the planning agent decides to take a particular action in a particular state, in terms of the subsequent decisions to the query point, and the expected consequences of those decisions on the planning objectives. With this knowledge of the impact of changing a particular decision of the planning agent's policy, it is up to the user to determine whether the change is desirable.

5.4.2 Sequential-Behavior (non-Markov) Query Why-Not Explanations

A sequential-behavior query captures a why-not question about a partial behavior, or a pattern of behavior, in an agent's policy that is sequential or temporal in nature. In this work, we explore an approach to explain a sequential-behavior query via a contrastive explanation where the foil is an alternative solution policy that subsumes the sequential behavior in question. Often, a sequential behavior posed by the user will not describe a full behavior of the agent from the queried point on, as such can be too complex for the user to determine, or too precise or tedious for the user to specify. Thus, a contrastive explanation must complete the behavior gap by determining a foil that is a rational full behavior of the agent – with respect to its planning objectives.

As an example, consider the following indoor delivery robot scenario: Suppose the robot is tasked to navigate across the building to deliver an item, and it has an objective to minimize the duration of the task. Along the way to the destination, a locked door is blocking the robot from traveling the shortest path. Upon reviewing the robot's policy, the user sees that when the robot arrives at the locked door, it turns to the adjacent hallway and takes the more roundabout path to the destination. The user has a knowledge that there are people behind the locked door and that the robot is able to call people for help to unlock the door and let it pass through. The user thus asks the question: why would the robot not call for help and wait until the door is unlocked? Such query describes only a partial behavior. However, the only rational behavior of the agent – with respect to its time minimization objective – once the door is unlocked is to continue traveling the shortest path to its target. An explanation to this query therefore must answer why the robot does not call for help, wait until the door is unlocked, and then travel through the shortest path.

As with our explanation approach for Markov queries, we propose to respond to a sequential-behavior query by explaining the impact of the agent behaving according to the query on the subsequent decisions it would make, and the overall impact on the planning objectives. It argues that such impact is less desirable (with respect to the agent's cost objective) compared to that of the agent's original policy. For instance, an explanation in the delivery robot scenario could be that: if the robot were to wait for the people to unlock the door, then the overall duration of the task is longer than if the robot were to take the roundabout path according to its original policy (because the wait time is long).

Problem Statement and Approach Generating a why-not explanation for a sequential behavior query on a policy is a problem of finding an alternative policy that conforms to the query, and contrasting it to the policy being queried. The input to the problem is a tuple $\langle \mathcal{M}, \pi_0, \mathcal{Q} \rangle$, where $\mathcal{M}$ is a Markov decision process of the original planning problem, π_0 is an optimal policy computed from $\mathcal{M}$, and $\mathcal{Q}$ is a why-not sequential-behavior query on π_0: $\mathcal{Q} = \langle s_q, \varphi \rangle$, where s_q is the query's starting state and φ is a Linear Temporal Logic (LTL) property describing the sequential behavior in question. The output is a finite-memory alternative policy π'_q that satisfies the property φ starting at state s_q, and its contrastive explanation to π_0 based on the expected planning objective values.

To find an alternative policy π'_q, we impose the LTL specification φ from the query as a constraint on the planning MDP $\mathcal{M}$, and solve for an optimal policy where the initial state is s_q. A LTL formula capturing a sequential-behavior query is built of atomic propositions Π that are over the states of the MDP $\mathcal{M}$. A finite-memory policy π^* for $\mathcal{M}$ satisfies a LTL specification φ if all the runs of the induced Markov chain $\mathcal{M}_{\pi^*}$ satisfy φ with probability one – where a run of a Markov chain is a possible sequence of states described by its stochastic process.

We base our approach on the work from [82], which contributes an approach to find a finite-memory policy for a MDP that satisfies a LTL constraint. We adapt their problem formulation such that it can find a policy that minimizes the expected total objective cost of the MDP *in addition to* satisfying the LTL constraint. Specifically, the approach from [82] constructs a product MDP $\mathcal{P}$ from the original MDP $\mathcal{M}$ and a deterministic Rabin automaton (DRA) generated from the LTL property constraint. The cost function $W_{\mathcal{P}}$ of the product MDP $\mathcal{P}$ is defined from the acceptance condition of the Rabin automaton. Our approach adds to the formulation the additional cost function $C_{\mathcal{P}}$ of the product MDP $\mathcal{P}$ that captures the optimization objectives from the original MDP $\mathcal{M}$. This construction allows any algorithm that optimizes the expected total cost to be applied to find a constraint-satisfying optimal policy for the original MDP.

We first construct a deterministic Rabin automaton (DRA) from a given LTL property φ, denoted as $\mathcal{R}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is the set of states; Σ is the input alphabet; $\delta : Q \times \Sigma \to Q$ is the transition function; q_0 is the initial state; and F is the acceptance condition corresponding to satisfaction of φ: $F = \{(G_1, B_1), ..., (G_{n_F}, B_{n_F})\}$, where $G_i, B_i \subset Q$ for $i = 1, ..., n_F$. A run of R_φ is accepting if and only if for some pair (G_i, B_i), some of the states in G_i are visited infinitely often, and the states in B_i are visited finitely often.

The original MDP is modeled as a labeled MDP with costs: $\mathcal{M} = \langle S, \mathcal{A}, P, C, s_0, \Pi, L \rangle$, where S is the set of states of the MDP; A is the set of actions and $\mathcal{A} : S \to 2^A$ is the mapping from states to actions; $P : S \times A \times S \to [0, 1]$ is the transition probability function; $C : S \to \mathbb{R}_{\geq 0}$ is the cost function – which can have multiple sub components capturing multiple objectives; $s_0 \in S$ is the initial state – this is the query state s_q; Π is the set of atomic proposition, and $L : S \to 2^\Pi$ is the labeling function labeling states with atomic propositions. Given the DRA $\mathcal{R}_\varphi$ and the labeled MDP $\mathcal{M}$, we construct a *Rabin weighted product MDP*, denoted as $\mathcal{P}$, as follows:

$$\mathcal{P} = \langle S_\mathcal{P}, \mathcal{A}_\mathcal{P}, P_\mathcal{P}, s_{\mathcal{P}0}, F_\mathcal{P}, W_\mathcal{P} + C_\mathcal{P} \rangle \tag{5.7}$$

where:

- $S_\mathcal{P} = S \times Q$ is the set of states.
- $\mathcal{A}_\mathcal{P}$ is the mapping from states to actions from the original MDP $\mathcal{M}$: $\mathcal{A}_\mathcal{P}((s, q)) = \mathcal{A}(s)$.
- $P_\mathcal{P}$ is the transition probability function:

$$P_\mathcal{P}(s_\mathcal{P}, a, s'_\mathcal{P}) = \begin{cases} P(s, a, s') & \text{if } q' = \delta(q, L(s)) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

  where $s_\mathcal{P} = (s, q) \in S_\mathcal{P}$ and $s'_\mathcal{P} = (s', q') \in S_\mathcal{P}$.
- $s_{\mathcal{P}0} = (s_0, q_0) \in S_\mathcal{P}$ is the initial state.
- $F_\mathcal{P}$ is the acceptance condition given by

$$F_\mathcal{P} = \{(\mathcal{G}_1, \mathcal{B}_1), ..., (\mathcal{G}_{n_F}, \mathcal{B}_{n_F})\}$$

  where $\mathcal{G}_i = S \times G_i$ and $\mathcal{B}_i = S \times B_i$.
- $W_\mathcal{P} = \{W^i_\mathcal{P}\}_{i=1}^{n_F}$ is the set of cost functions corresponding to the acceptance condition. Each cost function $W^i_\mathcal{P} : S_\mathcal{P} \to \mathbb{R}_{\geq 0}$ is defined by:

$$W^i_\mathcal{P}(s_\mathcal{P}) = \begin{cases} w_G & \text{if } s_\mathcal{P} \in \mathcal{G}_i \\ w_B & \text{if } s_\mathcal{P} \in \mathcal{B}_i \\ w_\epsilon & \text{if } s_\mathcal{P} \in S_\mathcal{P} \setminus (\mathcal{G}_i \cup \mathcal{B}_i) \end{cases} \tag{5.9}$$

  where $w_G = 0, w_B > 0$ is a positive cost, and $w_\epsilon > 0$ is a relatively small positive cost $w_\epsilon \ll w_B$. Note that the original approach in [82] uses reward functions for $W_\mathcal{P}$ instead of cost functions. However, as many application domains we explore in our work are concerned with minimization objectives (e.g., task duration, intrusiveness, etc.), our approach modifies the formulation of $W_\mathcal{P}$ to use cost functions instead.

- $C_{\mathcal{P}} = \{C^i_{\mathcal{P}}\}_{i=1}^{n_F}$ is the set of cost functions, one for each index i of the acceptance condition, that captures the optimization objectives from the original MDP $\mathcal{M}$. Each cost function $C^i_{\mathcal{P}} : S_{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is defined by:

  $$C^i_{\mathcal{P}}(s_{\mathcal{P}}) = \begin{cases} 0 & \text{if } s_{\mathcal{P}} \in \mathcal{G}_i \\ C(s) & \text{otherwise, where } s_{\mathcal{P}} = (s, q) \text{ for some } q. \end{cases} \tag{5.10}$$

  The cost function $C_{\mathcal{P}}$ is designed to not interfere with the effects of $W_{\mathcal{P}}$ for the acceptance condition. For each index i of the acceptance condition, any state in $\mathcal{G}_i$ will have a $w_G = 0$ combined cost; any state in $\mathcal{B}_i$ will have a combined cost that is $\geq w_B$; and any other state will have a combined cost that is $\geq w_\epsilon$.

We use $\mathcal{P}^i$ to denote a product MDP with the cost function $W^i_{\mathcal{P}}$ corresponding to $(\mathcal{G}_i, \mathcal{B}_i)$ in the acceptance condition $F_{\mathcal{P}}$. The combined cost function of $\mathcal{P}^i$ is therefore $W^i_{\mathcal{P}} + C^i_{\mathcal{P}}$.

Cost Design The design of the cost functions in $W_{\mathcal{P}}$ incentivizes a policy towards satisfying the Rabin automaton's acceptance condition. The cost function $W^i_{\mathcal{P}}$ for each index i of the acceptance condition assigns a positive cost w_B to states $s_{\mathcal{P}} \in \mathcal{B}_i$ since they should be visited only finitely often (i.e., minimize or avoid the visits to states in $\mathcal{B}_i$). Meanwhile, states $s_{\mathcal{P}} \in \mathcal{G}_i$ are assigned zero cost $w_G = 0$ to bias a policy to visit those states infinitely often (i.e., maximize the visits to states in $\mathcal{G}_i$). The remaining neutral states $s_{\mathcal{P}} \in S_{\mathcal{P}} \backslash (\mathcal{G}_i \cup \mathcal{B}_i)$ are assigned a relatively small cost $w_\epsilon \ll w_B$. That is, it biases a policy more towards minimizing the visits to states $\mathcal{B}_i$ than to the neutral states.

The challenge in adding the cost functions $C_{\mathcal{P}}$ to the product MDP $\mathcal{P}$ formulation is that it must not interfere with the effects of the cost functions $W_{\mathcal{P}}$ for the acceptance condition, as discussed above. To this end, we must design the values of w_B and w_ϵ for $W_{\mathcal{P}}$ to the appropriate values given the cost function C of the original MDP $\mathcal{M}$. For each index i of the acceptance condition, the *combined cost* of a state $s_{\mathcal{P}} \in S_{\mathcal{P}}$ is:

$$W^i_{\mathcal{P}}(s_{\mathcal{P}}) + C^i_{\mathcal{P}}(s_{\mathcal{P}}) = \begin{cases} w_G & \text{if } s_{\mathcal{P}} \in \mathcal{G}_i \\ w_B + C(s) & \text{if } s_{\mathcal{P}} \in \mathcal{B}_i\text{, where } s_{\mathcal{P}} = (s, q) \text{ for some } q \\ w_\epsilon + C(s) & \text{otherwise, where } s_{\mathcal{P}} = (s, q) \text{ for some } q \end{cases} \tag{5.11}$$

where $w_G = 0$ and $s \in S$ is a state in the original MDP $\mathcal{M}$. For a given problem domain, w_B must be set to a sufficiently large value relative to $C(s)$, to bias a policy towards satisfying the Rabin automaton's acceptance condition and simultaneously optimizing the cost objective C.

Solving Product MDPs A *stationary policy* (see Definition 5.2.5) for the product MDP $\mathcal{P}$ is $\pi : S_{\mathcal{P}} \to A_{\mathcal{P}}$, and it has a corresponding *finite-memory policy* (see Definition 5.2.3) for the original MDP $\mathcal{M}$. Let $\mathcal{P}_\pi$ denote the induced Markov chain from applying π to $\mathcal{P}$. Let $\mathcal{M}_\pi$ denote the the induced Markov chain from applying the corresponding finite-memory policy of π to $\mathcal{M}$. From Definition 5.2.6, we know that $\mathcal{M}_\pi$ satisfies φ with probability 1 if

$$\begin{aligned} &Pr(\{r : \exists(\mathcal{G}_i, \mathcal{B}_i) \in F_{\mathcal{P}}(s) \\ &\mathit{inf}(r) \cap \mathcal{G}_i \neq \emptyset \wedge \mathit{inf}(r) \cap \mathcal{B}_i = \emptyset\}) = 1 \end{aligned}$$

where r is a run of $\mathcal{P}_{\pi}$ initialized at $s_{\mathcal{P}0}$, and $inf(r) \in S_{\mathcal{P}}$ is the set of states that are visited infinitely often in the sequence r.

From Theorem 1, we know that if there exists a (finite-memory) policy $\bar{\pi}$ for the original MDP $\mathcal{M}$ such that $\mathcal{M}_{\bar{\pi}}$ satisfies φ with probability 1, then there exists $i^* \in \{1, ..., n_F\}$ and $w_B^* > 0$ such that any algorithm that optimizes the expected total cost of $\mathcal{P}^{i^*}$ with $w_B \geq w_B^*$ will find a stationary policy for the product MDP $\mathcal{P}$ that corresponds to $\bar{\pi}$.

Based on Theorem 1, we can find an optimal policy for the original MDP $\mathcal{M}$ such that the induced Markov chain satisfies the LTL property φ via the following approach: From the product MDP $\mathcal{P}$, we compute a collection of stationary policies $\{\pi_i^*\}_{i=1}^{n_F}$ that optimize the expected total cost of each $\mathcal{P}^i$, by using *a sufficiently large positive cost* w_B, for each $W_{\mathcal{P}}^i$. Then, we check whether any of the policies π_i^* satisfies φ with probability 1, by using any of the existing efficient model checking algorithms [8] to check whether the induced Markov chain $\mathcal{M}_{\pi_i^*}$ satisfies φ with probability 1. If any φ-satisfying policies exist, we select one that has the lowest expected total cost. Finally, we map the stationary policy for the product MDP $\mathcal{P}$ to the corresponding finite-memory policy for the original MDP $\mathcal{M}$. This becomes the alternative policy π_q' for our contrastive explanation for the query $\mathcal{Q}$. If none of the policies $\{\pi_i^*\}_{i=1}^{n_F}$ induces a Markov chain that satisfies φ, then we can conclude that there is no policy for the original MDP $\mathcal{M}$ that satisfies the LTL property φ.

Once we have obtained the alternative policy π_q' for the query $\mathcal{Q}$, or have determined that such policy does not exist, we can generate a why-not explanation for $\mathcal{Q}$ using our consequence-oriented contrastive explanation scheme. That is, if exists, we use π_q' as the foil to the original policy π_0 for the original MDP $\mathcal{M}$ in a contrastive explanation. We use the explanation scheme described in Section 5.4.1 to explain the impact of behaving in compliance with φ from state s_q on the subsequent decisions that the agent would make (i.e., the full alternative policy π_q'), and the expected consequences of those decisions on the planning objectives – in contrast to those of the original policy. On the other hand, if π_q' does not exist, our explanation simply indicates that it is impossible for the agent to behave according to φ from state s_q.

5.5 Conclusion

In this chapter, we have contributed a formal framework and approaches for user-guided explainable planning, in the consequence-oriented contrastive explanation paradigm. We introduce three types of why-not queries that the users can employ to ask the planning agent to address unexpected decisions: single-decision (Markov) queries, sequential pattern (non-Markov) queries, and value-based queries. We propose approaches to formulate such why-not queries as constraints imposed on the agent's original MDP model, and to solve the constrained MDP to obtain alternative policies that conform to the users' queries. Such alternative policies are then used as contrastive foils against the agent's original policy in the consequence-oriented contrastive explanations. An important step for the future work is to study the impact of enabling the users to ask different types of why-not queries, on the effectiveness of the contrastive explanations they receive as a result

Chapter 6

Discussion and Future Work

In this chapter, we discuss the assumptions, the scope, and the limitations of our explainable planning approaches and frameworks presented in this thesis. We also discuss some of the potential future research directions that expand our consequence-oriented contrastive explanation paradigm to the broader explainable agency capabilities.

6.1 Assumptions, Scopes, and Limitations

In this thesis, we build the consequence-oriented contrastive explanation frameworks and approaches for Markov decision processes as a sequential decision framework. We make a set of assumptions on the technical characteristics of the problem domains that our approaches can support. In this section, we discuss these assumptions and the implications they have on the scope of problem domains to which our work is applicable.

Optimality Criteria We assume that the desirable behavior of an agent performing a given task can be captured by Markovian rewards or costs (i.e., rewards and costs depend only on the current state and action of the agent). The desirable behavior can therefore be obtained by computing a policy that maximizes the total rewards (or minimizes the total costs) over the horizon of the task. This assumption underpins all standard MDP planning and reinforcement learning approaches.

This Markovian reward assumption has an implication on the scope of the problem domains that our approaches can support. Namely, our explainable planning approaches are only applicable to problem domains in which the desirable agent behavior can be reduced to maximizing or minimizing long-term rewards or costs, respectively. We argue that there are many problem domains that fall into this scope. An extensive survey of applications of MDPs [101] shows a wide range of application areas, including agriculture, manufacturing, finance, and healthcare among many others.

We argue that the key to enable the scope of our work to include a wide range of application areas is to support a wide range of objective functions. Fortunately, this boils down to supporting a handful of different optimality criteria for solving MDPs. To this end, our explainable planning approaches support two classical optimality criteria for MDPs: the expected total cost and the long-run average cost criteria. As evident in the survey [101], these two types of cost criteria cover most of the objective functions used in 18 application areas.

Reward Decomposition Following the assumption of Markovian rewards or costs, we further assume that the rewards or costs that capture the desirable agent behavior can be decomposed into multiple interpretable components. This assumption is crucial to our consequence-oriented contrastive explanation paradigm. Our approaches rely on the interpretable components of the agent's rewards or costs to explain the agent's optimization and tradeoff decisions. For instance, in a mobile robot indoor navigation domain discussed in Chapter 3, we assume that the costs that capture the desirable behavior of the mobile robot in a navigation task can be decomposed into components corresponding to travel time, collisions with obstacles, and intrusiveness to the building occupants. Our explanation approaches leverage these components to explain the agent's decisions in terms of its reasoning to balance optimizing travel time, collisions, and intrusiveness while navigating to the destination.

This assumption on decomposable rewards or costs puts a scope on the types of problem domains our explanation approaches can support. We argue that problem domains that have structured environments will often have a set of well-defined objectives that capture various concerns in a given problem. These well-defined objectives can then be used to define the interpretable components of rewards or costs of MDPs. Problems in various domains of engineering, productions, management and logistics are some examples that have structured environments. We argue that our explanation approaches are applicable to such well structured domains.

On the other hand, most of the fundamental challenges pertaining to designing intelligent agents to interact with the real world involve highly unstructured and dynamic environments. In such domains, the expectations of a desirable agent behavior can be complex and may not be decomposable into well-defined or interpretable objectives. For instance, for reinforcement learning agents that must learn optimal policies based on reward signals received from humans, the rewards can often be sparse and non-decomposable [98]. Our explanation approaches are not applicable to such unstructured or sparse rewards domains.

Preference Models Following the assumption of decomposable rewards or costs, we further assume that the preference models for the multiple objectives are additive models [49]. An additive preference model assumes that the strength of preference for the values of one objective can be expressed independently of the values of others. Particularly, we assume that multi-objective MDPs are converted to single-objective MDPs with additive returns, using a linear scalarization function [78]. The key benefit of this approach is its computational efficiency in finding a Pareto optimal policy on the convex hull.

The key limitation of additive preference models such as linear scalarization is that they are not always adequate for expressing a wide range of user or stakeholder preferences. For instance, the user preference for one objective may depend on the value of another objective. In such cases, more complex preference models such as nonlinear scalarization [78] or generalized additive models [13] can be more suitable. In terms of the technical solutions, our explanation approaches current cannot support such models, as the agent's reasoning to balance multiple optimization objectives in such cases are mathematically different from the additive model case.

However, the concept of consequence-oriented contrastive explanations in our approaches can potentially be formalized and implemented for more complex preference models.

Agent's World Model Our user-guided explainable planning approaches in Chapter 5 assume that why-not queries from the users are expressed only in terms of the state variables, actions, and objective functions that are present in the agent's world model. In other words, the users can only ask why-not questions about the concepts that the agent has in its knowledge. This also entails that the users have a knowledge about the capabilities of the agent that may not have been exhibited in the agent's policy so far. Capabilities can include what the agent can sense and have the understanding of the stimuli (e.g., the agent understands the concept of receiving permission to enter a room), and what the agent can act (e.g., the agent is able to request help from humans). This assumption is important to the utility of a user-guided contrastive explanation paradigm: The users must be able to ask questions about the *foil* cases (i.e., decisions the agent did not make) that can help them better understand the agent's reasoning and reconcile any unexpected decisions the agent made.

We may not expect lay users to have a full knowledge of the agent's capabilities. However, we argue that this requirement can potentially be met via an additional user interaction mechanism that allows the agent to describe or explain its capabilities to the users. Such mechanism will allow the users to have a shared world model with the agent, and can therefore ask more informed why-not questions.

6.2 Future Research Directions

In this section, we discuss some of the future research directions that expand our consequence-oriented contrastive explanation paradigm to the broader explainable agency capabilities.

Human Subjects Evaluation of User-Guided Explainable Planning

In Chapter 5, we investigate a formal framework and approaches for user-guided explainable planning, in the consequence-oriented contrastive explanation paradigm. An important step for the future work is to study the impact of enabling the users to ask different types of why-not queries, on the effectiveness of the contrastive explanations they receive as a result. In particular, the study may investigate an interplay between the cognitive effort required from the users to determine what why-not questions to ask, and the benefit from receiving contrastive explanations that can directly address the users' surprises or confusions about the agent's decisions. As we discuss in this thesis, the effectiveness of explanations should be evaluated based on measurable task performance of the users. To this end, our human subjects study design in Chapter 4 can be applied to the user-guided explainable planning settings.

Summarizing Landscape of Alternative Policies

Our current approaches for contrastive explanations identify individual alternative policies to be used as contrastive foils. However, as the space of potential solution policies for a given planning problem is large, it can be more informative for the users or stakeholders to understand the landscape of alternative policies and their characteristics. An opportunity for a future research direction is to investigate approaches to summarize a large number of alternative policies, in order to provide contrastive explanations in a more appropriate level of abstractions. For instance, clustering potential alternative policies based on their structural patterns or quantitative properties, and inferring high-level properties of those clusters to summarize and explain the policies may be a promising direction to explore.

Explanations for Other Sequential Decision Frameworks

The idea of consequence-oriented contrastive explanations at a high level is broadly applicable to other types of sequential decision problems beyond Markov decision processes. However, the technical approaches for identifying alternative policies as contrastive foils must be designed for the specific mathematical frameworks of the sequential decision problems. In particular, our current explanation approaches are designed for MDPs with known transition models and known reward models. Related frameworks such as reinforcement learning (RL) and Markov games allow more flexible views on sequential decision problems, where the dynamics of the environment are unknown and there are multiple adaptive agents with interacting or competing goals [63], respectively.

We suggest a future research direction in exploring how consequence-oriented contrastive explanation paradigm can be implemented in other sequential decision frameworks such as RL and Markov games. By and large, we believe that our current technical approaches for computing alternative policies as contrastive foils, such as constrained planning and planning with linear temporal logic specifications, are transferable from MDP settings to RL or Markov games settings. We anticipate that the key challenges for implementing such techniques in the new frameworks are computational efficiency and sample efficiency in the case of RL.

Iterative Queries and Explanations

Building upon our investigation into the user-guided explainable planning approaches in Chapter 5, we discuss our idea for supporting iterative queries and explanations. As the user asks a why-not query to the agent and receives an explanation, they may wish to ask a follow-up query or refine the previous query. This can be due to multiple reasons, such as:

- The user has multiple non-overlapping queries to ask the agent.
- Via the explanation, the user has seen how their query translates to a possible full behavior of the agent, and wishes to refine their query to ask about a different behavior.
- The user has in mind a different full behavior of the agent than the one presented as a foil in the contrastive explanation, but their query under-specified the behavior.

User queries are exploratory, and one-shot explanations may not be sufficient for the user's understanding of the planning landscape and the agent's rationale. The foil in a contrastive explanation to a why-not behavioral query may not address the latent expected behavior in the user's mind, due to the inherent under-specification of a query. Moreover, even when the contrastive explanation does address the expected behavior in the user's mind, it may prompt the user to question about potential other alternative behavior.

As an early investigation and a future work direction, we explore an approach to enable iterative queries and explanations, to allow the user to more easily explore the planning space and the consequences of different behavior of the agent. The goal of this approach is to allow the user to compose and refine multiple simpler queries to form a specific complex behavior of the agent for which they would like a why-not explanation. To do so, we propose maintaining a tree structure of multiple planning model variants of the same planning domain. The root node of the tree corresponds to the original planning model, and each descendent node corresponds to a *hypothetical model* that encompasses a series of constraints imposed on the original model by the user's iterative queries. Each hypothetical model will be used to generate a constrastive explanation in which the foil addresses the complex behavioral pattern of the agent that the user is cumulatively querying so far.

Problem Statement and Approach Let $\mathcal{M}$ denote an original planning MDP, and π_0 denote an optimal policy for $\mathcal{M}$. Our goal is to allow the user to construct a complex why-not behavioral query $\mathcal{Q}$ that can be composed of multiple simpler Markov or non-Markov queries on π_0 and $\mathcal{M}$, and to generate a contrastive explanation in response.

We construct a tree structure where the root node corresponds to the original MDP $\mathcal{M}$, and the rest of the nodes correspond to different *hypothetical models* that are variants of $\mathcal{M}$. Each edge connecting a parent node to a child node corresponds to a why-not behavioral query asked on the (original or hypothetical) model in the parent node, that results in the hypothetical model in the child node. Starting from the root node, suppose the user asks a query $\mathcal{Q}_0$ on the policy π_0 for the original MDP $\mathcal{M}$. We expand the tree by creating a child node that corresponds to the hypothetical model $\mathcal{H}_1$ resulting from imposing the constraint from $\mathcal{Q}_0$ on the model $\mathcal{M}$. For instance, suppose the user query is a non-Markov query $\mathcal{Q}_0 = \langle s_{q_0}, \varphi_0 \rangle$, where φ_0 is a LTL property starting at a query state s_{q_0}. The hypothetical model is therefore a product MDP of the original MDP $\mathcal{M}$ and the DRA constructed from the LTL property φ_0 and the query state s_{q_0}: $\mathcal{H}_1 = \mathcal{M} \otimes \mathcal{R}_{\varphi_0}$. From the hypothetical model $\mathcal{H}_1$, we compute its optimal policy $\pi_{\mathcal{H}_1}$, and use it as the foil in a contrastive explanation in response to the query $\mathcal{Q}_0$.

We can recursively expand the tree from any node on which the user asks a new query. For notation purposes, we use $\mathcal{H}_0$ to denote the original MDP $\mathcal{M}$, and $\mathcal{H}_i$ for $i \geq 1$ to denote hypothetical models. Let $\mathcal{H}_i$ denote the model corresponding to the node N_i, and $\mathcal{Q}_i$ denote the new query. By construction of the tree, we know that $\mathcal{H}_i$ is a result of imposing the constraint from the query $\mathcal{Q}_{i-1}$ on the parent model $\mathcal{H}_{i-1}$ That is, $\mathcal{H}_i = \mathcal{H}_{i-1} \otimes \mathcal{R}_{\varphi_{i-1}}$, where $\mathcal{R}_{\varphi_{i-1}}$ is a DRA for the query $\mathcal{Q}_{i-1}$. In other words, $\mathcal{H}_i$ is already constrained by $\mathcal{Q}_{i-1}$. Inductively, we know that:

$$\mathcal{H}_i = \mathcal{H}_0 \otimes \mathcal{R}_{\varphi_0} \otimes \mathcal{R}_{\varphi_1} ... \otimes \mathcal{R}_{\varphi_{i-1}} \tag{6.1}$$

where $\mathcal{H}_0$ is the original MDP and $\mathcal{R}_{\varphi_j}$ for $0 \leq j \leq i-1$ are the DRAs constructed from the queries corresponding to the edges from the root node N_0 down to N_i. Therefore, when the user asks a new query $\mathcal{Q}_i$ on a model $\mathcal{H}_i$, they effectively compose the new query with the previously asked queries back to the original MDP $\mathcal{H}_0$.

Implementation-wise, however, to create a new hypothetical model $\mathcal{H}_i$ from a parent model

$\mathcal{H}_{i-1}$ and a query $\mathcal{Q}_{i-1}$, when $i > 1$, we may not want to compute a product MDP between $\mathcal{H}_{i-1}$ and the DRA $\mathcal{R}_{\varphi_{i-1}}$ as described in Equation 5.7. Doing so would result in an exponential grow in the size of the state space of the resulting product MDP at every step i when the user asks a new query, due to the cross product of the states of $\mathcal{H}_{i-1}$ and the states of $\mathcal{R}_{\varphi_{i-1}}$. Instead, we implement the following approach in order to perform cross product of the states only once: We keep track of the LTL formula φ_0 in the first query $\mathcal{Q}_0$ that the user asks on the original MDP $\mathcal{H}_0$. We construct the hypothetical model $\mathcal{H}_1$ as a product MDP between $\mathcal{H}_0$ and $\mathcal{R}_{\varphi_0}$ according to Equation 5.7. Given a new query $\mathcal{Q}_1$ on the model $\mathcal{H}_1$, we compute the conjunction of the LTL formulas $\varphi_0 \wedge \varphi_1$, where φ_1 is the LTL formula in the query $\mathcal{Q}_1$. We then compute the corresponding DRA $\mathcal{R}_{\varphi_0 \wedge \varphi_1}$. Finally, we compute the new hypothetical model $\mathcal{H}_2$ as a product MDP between the original MDP $\mathcal{H}_0$ and the DRA $\mathcal{R}_{\varphi_0 \wedge \varphi_1}$. With this approach, we can recursively compute a new hypothetical model $\mathcal{H}_i$, given a new user query $\mathcal{Q}_{i-1}$ on the existing model $\mathcal{H}_{i-1}$, for any $i > 1$ as follows:

$$\mathcal{H}_i = \mathcal{H}_0 \otimes \mathcal{R}_{\varphi_0 \wedge \varphi_1 \ldots \wedge \varphi_{i-1}} \tag{6.2}$$

where each φ_j is the LTL formula in a query $\mathcal{Q}_j$ asked on the model $\mathcal{H}_j$.

Once a new hypothetical model $\mathcal{H}_i$ is created, our goal is to generate an explanation to a why-not query captured by the LTL formula $\varphi_0 \wedge \varphi_1 ... \wedge \varphi_{i-1}$ imposed on the original MDP $\mathcal{H}_0$. To this end, we compute an optimal policy $\pi_{\mathcal{H}_i}$ for the product MDP $\mathcal{H}_i$, which , if exists, satisfies the LTL formula $\varphi_0 \wedge \varphi_1 ... \wedge \varphi_{i-1}$. We use $\pi_{\mathcal{H}_i}$ as a foil in a contrastive explanation for the original policy π_0 for $\mathcal{H}_0$.

Depth and Breadth Expansion Once the user received an explanation of $\pi_{\mathcal{H}_i}$ from the latest hypothetical model $\mathcal{H}_i$ (which corresponds to a leaf node in the tree), if the user wishes to ask further questions about the agent behavior, they may:

- Ask a new query $\mathcal{Q}_i$ on the model $\mathcal{H}_i$. This step will create a new hypothetical model $\mathcal{H}_{i+1}$, which will be added as the first child node of the node for $\mathcal{H}_i$ in the tree. We refer to this as a *depth expansion* of the tree. Such expansion allows the user to guide the explanation process to address a more specific agent behavior of their interest, since such query increases the specificity of the constraint on $\mathcal{H}_0$ from $\varphi_0 \wedge \varphi_1 ... \wedge \varphi_{i-1}$ to $\varphi_0 \wedge \varphi_1 ... \wedge \varphi_{i-1} \wedge \varphi_i$.
- Ask a new query $\mathcal{Q}'_{i-1}$ on the parent model $\mathcal{H}_{i-1}$ that is different from the previously asked query $\mathcal{Q}_{i-1}$. This step will create a new hypothetical model $\mathcal{H}'_i$, which will be added as the next sibling node of the node for $\mathcal{H}_i$ in the tree. We refer to this as a *breadth expansion* of the tree. Such expansion allows the user to explore diverse behavior of the agent, since such query changes the constraint on $\mathcal{H}_0$ from $\varphi_0 \wedge \varphi_1 ... \wedge \varphi_{i-1}$ to $\varphi_0 \wedge \varphi_1 ... \wedge \varphi'_{i-1}$.

The benefit of incrementally and iteratively asking a query and getting an explanation is not only that the user can form a complex query by building up on simpler queries as indicated in Equation 6.2, but also that an explanation given at each step i can inform the user's incremental query $\mathcal{Q}_i$ or $\mathcal{Q}'_{i-1}$. The alternating between queries and explanations allow the user queries to guide explanation generation, as well as the explanations to guide what the user will query next.

Chapter 7

Conclusion

In this thesis, we develop approaches and a framework for enabling a Markov decision process (MDP) planning agent to communicate its goals and explain the rationale for its sequential decisions to the end-users. This thesis is motivated to address problem domains in which multiple optimization objectives are involved in the planning tasks. The type of explainability problem focused on in this thesis is that of algorithm-agnostic, model-based inference reconciliation of MDP planning. In particular, our work aims at explaining to the end-users why the planning agent's solution policy is optimal with respect to the task objectives. To this end, we investigate contrastive explanations as a mean for explaining why the agent's decisions are optimal in context of other possible decisions, as optimality is inherently a comparative property.

The key challenge addressed in this thesis is in how to instantiate the concept of contrastive explanations and implement the mechanisms for an explainable MDP planning framework. In much of explainable AI literature, contrastive explanations are often viewed through the lens of causality of events, or of the properties and relations inherent to a system in different circumstances. However, these common views are not necessarily suitable or amenable to implementing a scalable approach for explaining optimality of planning solutions. Defining an appropriate formulation of contrastive explanations for this purpose, as well as designing algorithmic approaches to generate such contrastive explanations are the main goals in this thesis.

We propose a consequence-oriented contrastive explanation framework, in which an argument for a policy is in terms of its expected consequences on the task objectives, put in context of the selected viable alternatives to demonstrate the optimization and tradeoff reasoning of the agent. In Chapter 3, we develop: (a) a modeling framework that supports reward decomposition and augments MDP representation to ground the components of the reward or cost function in the domain-level concepts and semantics, to facilitate explanation generation; and (b) a method for computing policy-level contrastive foils that describe the inflection points in the agent's decision making in terms of optimization and tradeoff reasoning of the decomposed task objectives. Our method computes the contrastive foils by replanning with hard and soft constraints on the value functions of the decomposed task objectives. In Section 3.6, we demonstrate the applicability of our proposed explainable planning framework by applying our approach to three planning problem domains: waypoint-based indoor robot navigation, Unmanned Aerial Vehicles (UAV) mission planning, and outpatient clinic scheduling.

In Chapter 5, we investigate the feasibility of a user-guided approach to our consequence-oriented contrastive explanation paradigm. We explore proof-of-concept approaches to formulate "Why-Not" queries as state-action constraints and linear temporal logic (LTL) constraints on the planning problem, and to solve for satisfying policies in order to explain the full impact that the queried behavior has on the subsequent decisions and on the task objectives. Finally, in Chapter 4, we design and conduct a human subjects experiment to evaluate the effectiveness of explanations based on measurable task performance.

Appendix A

User Study Instructions

Below are the instructions given to the participants in our user study in Chapter 4.

For your convenience in performing this HIT, please set your browser to full-screen mode throughout the task.

Introduction

Our research aims to understand how people interpret and understand the objectives of an autonomous agent (e.g., a robot) by observing its behavior. In this study, consider a mobile robot navigating inside a building. Its task is to move from its current location to a given destination (see Figures 1 and 2 for illustration). The robot must navigate to the destination while optimizing for 3 objectives, namely, minimizing its (1) driving time, (2) expected collisions with obstacles, and (3) intrusiveness to human occupants in the building (see Figures 3-7 for details).

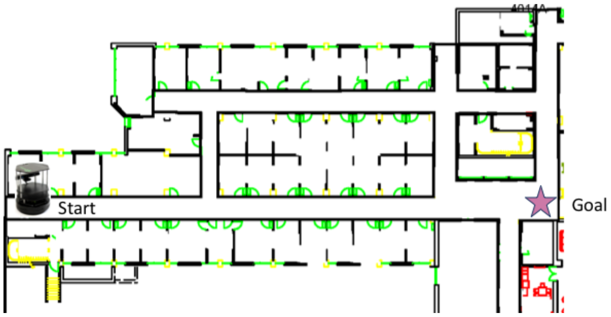


Figure 1: The robot has to navigate from its current location to a given goal.

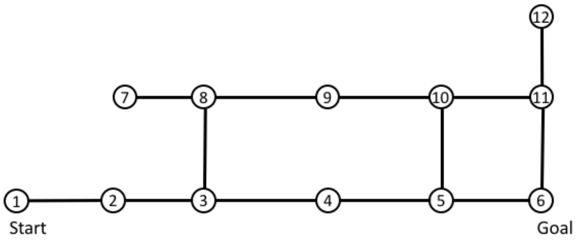


Figure 2: The map of the building in Figure 1 is represented abstractly by a set of *waypoints* and *connections* among them. Each waypoint represents a location in the building, and a connection between 2 waypoints means that Robot can move between the 2 locations.

The distances between waypoints are to scale, and waypoints 1 and 2 are 10 meters apart.

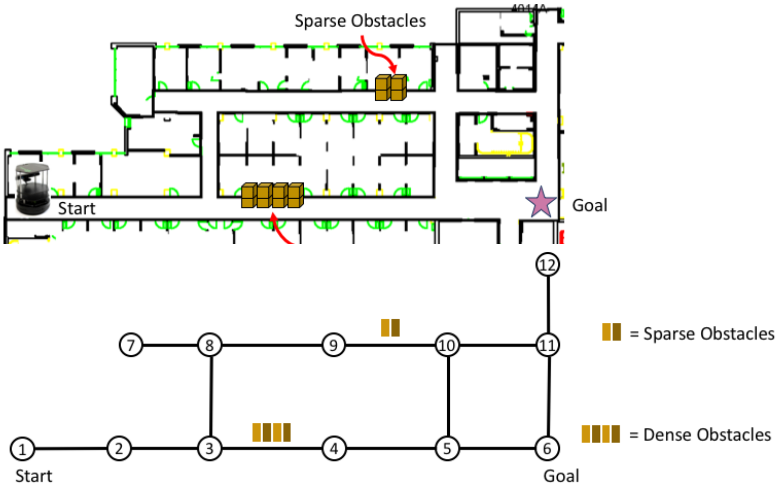
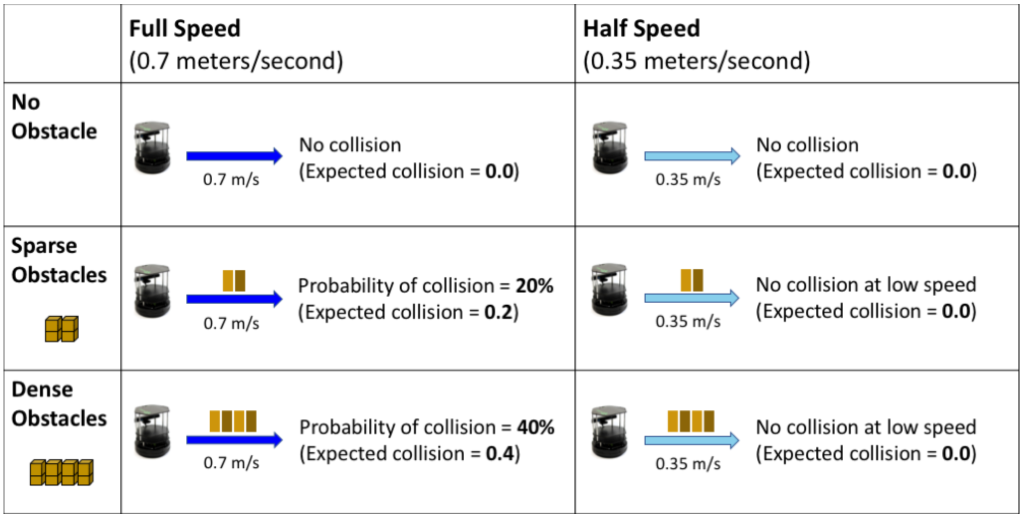


Figure 4: In an abstract map, obstacles are represented on a connection between 2 waypoints. The *expected collision* when the robot moves from one waypoint to the next depends on the speed of the robot and the obstacle density, as shown in Figure 5.



	Full Speed (0.7 meters/second)	**Half Speed** (0.35 meters/second)
No Obstacle	0.7 m/s No collision (Expected collision = **0.0**)	0.35 m/s No collision (Expected collision = **0.0**)
Sparse Obstacles	0.7 m/s Probability of collision = **20%** (Expected collision = **0.2**)	0.35 m/s No collision at low speed (Expected collision = **0.0**)
Dense Obstacles	0.7 m/s Probability of collision = **40%** (Expected collision = **0.4**)	0.35 m/s No collision at low speed (Expected collision = **0.0**)

Figure 5: The robot may drive at its full speed: 0.7 meters/second (indicated by dark-blue line), or at half speed: 0.35 meters/second (indicated by light-blue line). The robot can avoid collision with obstacles when it moves at half speed. That is, the expected collision of moving at half speed is always 0.0.

Every time the robot moves at full speed through *sparse* obstacles, and through *dense* obstacles, it has 20% and 40% probability of colliding, respectively. In other words, the expected collision of moving at full speed through each waypoint-connection with sparse obstacles and dense obstacles is 0.2 and 0.4, respectively.

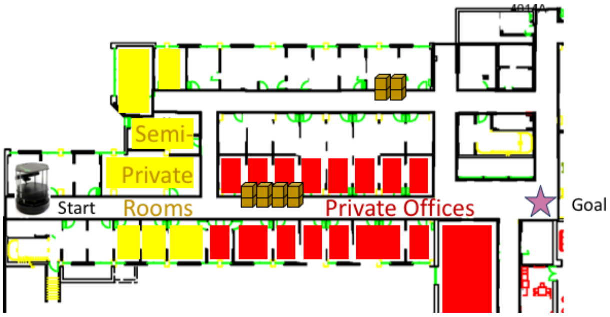


Figure 6: The building has private offices (in red) and semi-private conference rooms

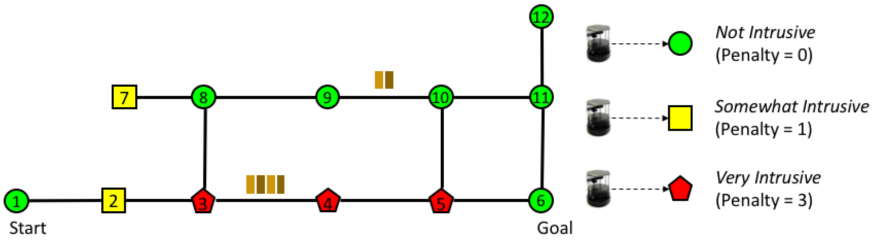


Figure 7: In an abstract map, each waypoint is color-coded to represent the *intrusiveness* of the robot if it drives to that location. When the robot drives to a PUBLIC waypoint, it does not receive any *intrusiveness penalty*. However, each time the robot drives to a SEMI-PRIVATE or a PRIVATE waypoint, it receives a penalty of 1 or 3, respectively.

It is not always possible to find a way to the destination that takes the shortest amount of time, has the lowest expected collisions, and is the least intrusive to human occupants. The robot may need to make a tradeoff among these objectives. For instance, the robot may need to take a detour to the destination to minimize its intrusiveness. Or it may take the shortest route to the destination, but at the expense of being more intrusive. Which option is better depends on how one weighs the importance of each of the objectives.

Suppose a user of the robot wants the robot to deliver a package at some destination. The user has in mind how they would weigh the importance of each objective. For instance, the user may mainly care about the robot minimizing its expected collisions because the package is fragile, and not care much about the package being delivered very quickly. The user's concerns can be represented by a *cost profile*, for instance:

	Cost ($)
1 second of travel time	$1
0.1 expected collision	$9
1 intrusiveness-penalty	$5

A *total cost* of a navigation plan can be computed from the cost profile. For example, a navigation plan that takes 120 seconds, has 0.2 expected collision, and has intrusiveness penalty of 7 would have a total cost of $173:

```
Total cost = ($1 per 1 second * 120 seconds)
           + ($9 per 0.1 exp. collision * 0.2 exp. collision / 0.1)
           + ($5 per 1 intrusive penalty * 7 intrusive penalty)
           = $173.
```

A best navigation, with respect to a particular cost profile, is one that has the lowest total cost.

Unfortunately, the user cannot control how the robot weighs its objectives. The robot will make its own decision on how to navigate to the destination, and its decision may or may not be in agreement with how the user would weigh the objectives. However, the robot will present its navigation plan to the user before it is taking off.

Task

Imagine that you are the user of the robot. **You will be given a cost profile**, similar to the one shown above. You will be presented with a series of 4 navigation problems (each with a different map) and the corresponding navigation plans that the robot came up with. **You task is to determine whether the robot's plan is the best available option for you.**

Example

Suppose that you are given the cost profile above. Suppose that the robot presents the following plan and informs you that the navigation will take 80 seconds:

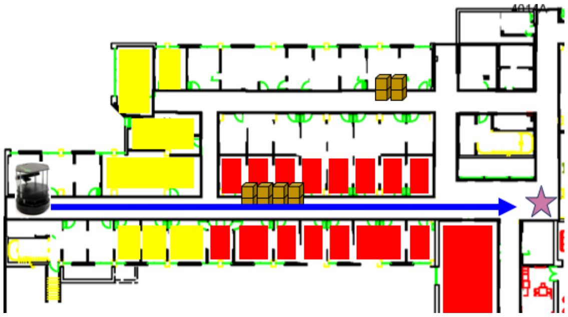


Figure 8: The robot's navigation plan. This navigation will take 80 seconds

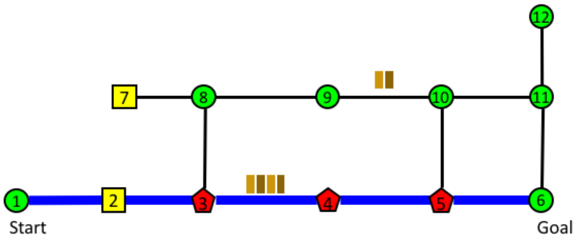


Figure 9: This is the abstract navigation plan of Figure 8. Here, the robot plans to drive at full speed through waypoints 1 to 6.

This plan takes the least amount of time. But it incurs intrusiveness-penalty of 10, since it drives through a SEMI-PRIVATE waypoint 1 time and a PRIVATE waypoint 3 times. This plan also has the expected collision of 0.4, since it moves through dense obstacles at full speed.

You can calculate the total cost of the robot's presented navigation plan in Figure 9: ($1 * 80 seconds) + ($9 * 0.4 exp. collision / 0.1) + ($5 * 10 intrusive penalty) = $166. However, this plan is NOT the best available option according to the cost profile, because the following alternative navigation would be a better option:

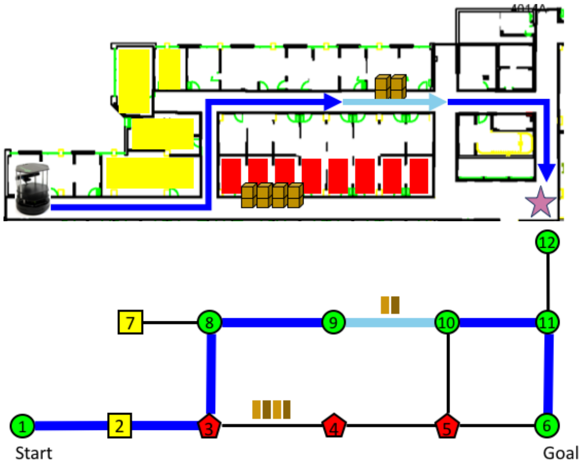


Figure 11: This is the abstract navigation plan of Figure 10. Here, the robot plans to drive through waypoints 1-2-3-8-9-10-11-6 at full speed except from waypoints 9-10, where there are sparse obstacles.

This plan incurs intrusiveness-penalty of 4 (since it drives through a SEMI-PRIVATE waypoint 1 time and a PRIVATE waypoint 1 time), which is the lowest among all possible plans. It also has the lowest expected collision (no collision). However, this plan takes more time to reach the destination than the plan in Figure 9.

You can calculate the total cost of the alternative navigation plan in Figure 11: ($1 * 128 seconds) + ($9 * 0.0 exp. collision / 0.1) + ($5 * 4 intrusive penalty) = $148. This plan costs less than the one that the robot presented.

Once you have read and understand the instruction, click the button below to continue to the task. The information presented above regarding how to interpret symbols in a map and a navigation plan (e.g., obstacles, intrusiveness, robot's speed, probability of collision, etc.) will be provided in a legend box next to each navigation problem.

Continue to the Task

Bibliography

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018. 2

[2] Tobias Ahlbrecht and Michael Winikoff. Explaining aggregate behaviour in cognitive agent simulations using explanation. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, pages 129–146. Springer, 2019. 2

[3] Jonathan Aldrich, David Garlan, Christian Kästner, Claire Le Goues, Anahita Mohseni-Kabir, Ivan Ruchkin, Selva Samuel, Bradley Schmerl, Christopher Steven Timperley, Manuela Veloso, et al. Model-Based Adaptation for Robotics Software. *IEEE Software*, 36(2), 2019. 3.1.1

[4] Eitan Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999. 3.5.1

[5] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. Explaining Reinforcement Learning to Mere Mortals: An Empirical Study. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. 3.7

[6] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019. 2, 3.7

[7] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020. 2

[8] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008. 5.2.2, 5.4.2

[9] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *arXiv preprint arXiv:1406.5823*, 2014. 4.2.1, 4.2.2

[10] Diego Bello and German Riano. Linear programming solvers for markov decision processes. In *2006 IEEE systems and information engineering design symposium*, pages 90–95. IEEE, 2006. 3.5.1

[11] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain—how planning helps to assemble your home theater. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 386–394, 2014. 2

[12] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2):49–107, 2000. 3.6.1

[13] Darius Braziunas and Craig Boutilier. Preference elicitation and generalized additive utility. In *AAAI*, volume 21. Boston, MA, 2006. 6.1

[14] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019. 2

[15] Michael Cashmore, Anna Collins, Benjamin Krarup, Senka Krivic, Daniele Magazzeni, and David Smith. Towards explainable ai planning as a service. *arXiv preprint arXiv:1908.05059*, 2019. 2

[16] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317*, 2017. 2

[17] Tathagata Chakraborti, Kshitij P Fadnis, Kartik Talamadupula, Mishal Dholakia, Biplav Srivastava, Jeffrey O Kephart, and Rachel KE Bellamy. Planning and visualization for a smart meeting room assistant. *AI Communications*, 32(1):91–99, 2019. 2

[18] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. The emerging landscape of explainable automated planning & decision making. In *IJCAI*, pages 4803–4811, 2020. 1, 1.1, 2

[19] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M Rao, et al. Interpretability of deep learning models: A survey of results. In *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*, pages 1–6. IEEE, 2017. 2

[20] Anjan Chakravartty. Perspectivism, inconsistent models, and contrastive explanation. *Studies in History and Philosophy of Science Part A*, 41(4):405–412, 2010. 2

[21] Seth Chin-Parker and Alexandra Bradner. A contrastive account of explanation generation. *Psychonomic Bulletin & Review*, 24(5):1387–1397, 2017. 2

[22] Seth Chin-Parker and Julie Cantelon. Contrastive constraints guide explanation-based category learning. *Cognitive science*, 41(6):1645–1655, 2017. 2

[23] Jason Cong, Bin Liu, and Zhiru Zhang. Scheduling With Soft Constraints. In *In Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2009. 1b, 3.5.2

[24] Kristijonas Cyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni. Argumentation for Explainable Scheduling. In *In Proceedings of the AAAI Conference on Artificial Intel-*

ligence (AAAI), 2019. 3.7

[25] Claudia D'Ambrosio, Andrea Lodi, and Silvano Martello. Piecewise Linear Approximation of Functions of Two Variables in MILP Models. *Operations Research Letters*, 38(1), 2010. 1b, 3.5.2, 3.5.2

[26] Stanislas Dehaene. The neural basis of the weber–fechner law: a logarithmic mental number line. *Trends in cognitive sciences*, 7(4):145–147, 2003. 3.5.2

[27] Thomas Dodson, Nicholas Mattei, Joshua T Guerin, and Judy Goldsmith. An english-language argumentation interface for explanation generation with markov decision processes in the domain of academic advising. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(3):1–30, 2013. 2

[28] Dmitri A. Dolgov and Edmund H. Durfee. Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005. 1b, 3.5.1, 3.5.1, 3.5.1, 3.5.1

[29] Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, pages 411–420, 1999. 5.3.1

[30] Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni, and Marcel Steinmetz. A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9818–9826, 2020. 2

[31] Thomas Eiter, Zeynep G Saribatur, and Peter Schüller. Abstraction for zooming-in to unsolvability reasons of grid-cell problems. *arXiv preprint arXiv:1909.04998*, 2019. 2

[32] Salomé Eriksson, Gabriele Röger, and Malte Helmert. Unsolvability certificates for classical planning. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017. 2

[33] Salomé Eriksson, Gabriele Röger, and Malte Helmert. A proof system for unsolvable planning tasks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 2018. 2

[34] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2020–2025. IEEE, 2005. 5.3.1

[35] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009. 5.3.1

[36] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable Planning. *arXiv preprint arXiv:1709.10256*, 2017. 1, 2, 3, 5.4.1

[37] Johannes Fürnkranz, Eyke Hüllermeier, Cynthia Rudin, Roman Slowinski, and Scott Sanner. Preference Learning. *Dagstuhl Reports*, 4(3), 2014. ISSN 2192-5283. 1, 4.3

[38] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Twentieth*

International Conference on Automated Planning and Scheduling, 2010. 2

[39] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International conference on machine learning*, pages 1792–1801. PMLR, 2018. 2

[40] David Gunning. Explainable Artificial Intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)*, 2, 2017. 2, 3, 3.7

[41] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE, 2017. 2

[42] Scott A Hissam, Sagar Chaki, and Gabriel A Moreno. High assurance for distributed cyber physical systems. In *Proceedings of the 2015 european conference on software architecture workshops*, pages 1–4, 2015. 3.6.2

[43] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. Enabling Robots to Communicate Their Objectives. *Autonomous Robots*, 43(2), 2019. 3.7

[44] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable Reinforcement Learning via Reward Decomposition. In *In Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence*, 2019. 2, 3.7

[45] Subbarao Kambhampati. Mapping and retrieval during plan reuse: A validation structure based approach. In *AAAI*, pages 170–175, 1990. 2

[46] Subbarao Kambhampati and James A Hendler. Flexible reuse of plans via annotation and verification. In *Proceedings The Fifth Conference on Artificial Intelligence Applications*, pages 37–38. IEEE Computer Society, 1989. 2

[47] Daniel Kasenberg and Matthias Scheutz. Interpretable apprenticeship learning with temporal logic specifications. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4914–4921. IEEE, 2017. 2

[48] Rick Kazman, Gregory Abowd, Len Bass, and Paul Clements. Scenario-Based Analysis of Software Architecture. *IEEE Software*, 13(6), 1996. 3.1

[49] Ralph L Keeney, Howard Raiffa, and Richard F Meyer. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993. 6.1

[50] Omar Khan, Pascal Poupart, and James Black. Minimal sufficient explanations for factored markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, pages 194–200, 2009. 2

[51] Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian Inference of Linear Temporal Logic Specifications for Contrastive Explanations. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. 3.7

[52] Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In *IJCAI*, pages 5591–5598, 2019. (document), 2, 5.1, 5.3.1

[53] Anurag Koul, Sam Greydanus, and Alan Fern. Learning finite state representations of recurrent policy networks. *arXiv preprint arXiv:1811.12530*, 2018. 2

[54] Benjamin Krarup, Michael Cashmore, Daniele Magazzeni, and Tim Miller. Model-based contrastive explanations for explainable planning. In *ICAPS 2019 Workshop on Explainable AI Planning (XAIP)*. AAAI Press, 2019. 2

[55] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009. 5.3.1

[56] Christophe Labreuche. A general framework for explaining the results of a multi-attribute preference model. *Artificial Intelligence*, 175(7-8), 2011. 3.7

[57] Christophe Labreuche and Simon Fossier. Explaining Multi-Criteria Decision Aiding Models with an Extended Shapley Value. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 3.7

[58] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In *Twenty-Ninth IAAI Conference*, 2017. 2

[59] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable Agent Alignment via Reward Modeling: A Research Direction. *arXiv preprint arXiv:1811.07871*, 2018. 1, 1.1, 4.3

[60] Shen Li, Rosario Scalise, Henny Admoni, Siddhartha S. Srinivasa, and Stephanie Rosenthal. Evaluating Critical Points in Trajectories. In *In Proceedings of the IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2017. 3.7

[61] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2119–2128, 2009. 1

[62] Peter Lipton. Contrastive explanation. *Royal Institute of Philosophy Supplements*, 27: 247–266, 1990. 2

[63] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994. 6.2

[64] Matthew R Longo and Stella F Lourenco. Spatial attention and the mental number line: Evidence for characteristic biases and compression. *Neuropsychologia*, 45(7):1400–1407, 2007. 3.5.2

[65] Maurício Cecílio Magnaguagno, RAMON FRAGA PEREIRA, Martin Duarte Móre, and Felipe Rech Meneguzzi. Web planner: A tool to develop classical planning domains and visualize heuristic state-space search. In *2017 Workshop on User Interfaces and Scheduling and Planning (UISP@ ICAPS), 2017, Estados Unidos.*, 2017. 2

[66] R Timothy Marler and Jasbir S Arora. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26(6), 2004. 3

[67] Charles E McCulloch and John M Neuhaus. Generalized Linear Mixed Models. *Encyclopedia of Biostatistics*, 4, 2005. 4.2

[68] Ben Leon Meadows, Pat Langley, and Miranda Jane Emery. Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*. Citeseer, 2013. 2

[69] Tim Miller. Explanation in Artificial Intelligence: Insights From the Social Sciences. *Artificial Intelligence*, 267, 2019. 1, 2, 3.7

[70] Tim Miller. Contrastive explanation: A structural-model approach. *The Knowledge Engineering Review*, 36, 2021. 1, 2

[71] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45, 2021. 2

[72] Gabriel Moreno, Cody Kinneer, Ashutosh Pandey, and David Garlan. Dartsim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 181–187. IEEE, 2019. 3.6.2, 3.6, 3.6.2, 3.6.2

[73] Matthew L Olson, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual states for atari agents via generative deep learning. *arXiv preprint arXiv:1909.12969*, 2019. 2

[74] Jonathan Patrick. A markov decision model for determining optimal outpatient scheduling. *Health care management science*, 15(2):91–102, 2012. 3.6.3, 3.6.3, 3.6.3

[75] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977. 5.2.1

[76] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994. 3.2.1, 3.4.1, 3.5.1, 3.5.1

[77] Peter Reichl, Sebastian Egger, Raimund Schatz, and Alessandro D'Alconzo. The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010. 3.5.2

[78] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48, 2013. 1, 3, 6.1

[79] Avi Rosenfeld and Ariella Richardson. Explainability in Human-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 33(6), 2019. 2, 3.7

[80] Stephanie Rosenthal, Sai P Selvaraj, and Manuela M Veloso. Verbalization: Narration of autonomous robot experience. In *IJCAI*, volume 16, pages 862–868, 2016. 2

[81] Dorsa Sadigh, Eric Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. Technical Report UCB/EECS-2014-166, EECS Department, University of California, Berkeley, Sep 2014. URL `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-166.html`. 5.2.2

[82] Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia. A learning based approach to control synthesis of markov decision processes for linear tem-

poral logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 1091–1096. IEEE, 2014. 4, 5.2.2, 5.4.2, 5.4.2

[83] Shmuel Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. Found. of Comp. Sci*, pages 319–327, 1988. 5.2.1

[84] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Twenty-second international conference on automated planning and scheduling*, 2012. 2

[85] Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[86] Raymond Sheh and Isaac Monteath. Introspectively assessing failures through explainable artificial intelligence. In *IROS Workshop on Introspective Methods for Reliable Autonomy*, pages 40–47. iliad-project. eu Vancouver, Canada, 2017. 2

[87] Raymond K Sheh. Different xai for different hri. In *2017 AAAI Fall Symposium Series*, 2017. 2

[88] Shirin Sohrabi, Jorge Baier, and Sheila McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 261–267, 2011. 2

[89] Sarath Sreedharan, Subbarao Kambhampati, et al. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, pages 518–526, 2018. 2

[90] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical Expertise Level Modeling for User Specific Contrastive Explanations. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 2, 3.7

[91] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that hal? explaining unsolvability of planning tasks. In *International Joint Conference on Artificial Intelligence*, 2019. 2

[92] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Tldr: Policy summarization for factored ssp problems using temporal abstractions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 272–280, 2020. 2

[93] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001, 2021. 2

[94] Roykrong Sukkerd, Reid Simmons, and David Garlan. Tradeoff-focused contrastive explanation for mdp planning. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1041–1048. IEEE, 2020. 1, 2

[95] Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*,

volume 33, pages 2514–2521, 2019. 2

[96] Felipe Trevizan, Sylvie Thiébaux, Pedro Santana, and Brian Williams. Heuristic search in dual space for constrained stochastic shortest path problems. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016. 3.5.1, 3.5.1

[97] Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark Neerincx. Contrastive explanations for reinforcement learning in terms of expected consequences. *arXiv preprint arXiv:1807.08706*, 2018. 2

[98] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 6.1

[99] Manuela M Veloso. Learning by analogical reasoning in general problem solving. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1992. 2

[100] Jacob Westfall, David A Kenny, and Charles M Judd. Statistical Power and Optimal Design in Experiments in Which Samples of Participants Respond to Samples of Stimuli. *Journal of Experimental Psychology: General*, 143(5), 2014. 4.2.2

[101] Douglas J White. A Survey of Applications of Markov Decision Processes. *Journal of the Operational Research Society*, 44(11), 1993. 3, 6.1

[102] Michael Winikoff. Debugging agent programs with why? questions. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 251–259, 2017. 2

[103] Michael Winikoff, Virginia Dignum, and Frank Dignum. Why bad coffee? explaining agent plans with valuings. In *International Conference on Computer Safety, Reliability, and Security*, pages 521–534. Springer, 2018. 2

[104] Zahra Zahedi, Alberto Olmo, Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. Towards understanding user preferences for explanation types in model reconciliation. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 648–649. IEEE, 2019. 2

[105] Zhiwei Zeng, Xiuyi Fan, Chunyan Miao, Cyril Leung, Jing Jih Chin, and Yew-Soon Ong. Context-based and Explainable Decision Making with Argumentation. In *In Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018. 3.7

[106] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021. 2