Thesis Proposal

Improving Transparency and Understandability of Multi-Objective Probabilistic Planning

Roykrong Sukkerd

Institute for Software Research
Carnegie Mellon University
rsukkerd@cs.cmu.edu

December 2017

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

Thesis Committee

Prof. David Garlan (co-chair)
Institute for Software Research
Carnegie Mellon University

Prof. Reid Simmons (co-chair)
Robotics Institute
Carnegie Mellon University

Prof. Manuela Veloso
Computer Science Department
Carnegie Mellon University

Prof. Bogdan Vasilescu
Institute for Software Research
Carnegie Mellon University

Prof. Bashar Nuseibeh
Department of Computing / Centre for Research in Computing
The Open University

Contents

Abstract

Understanding the reasoning behind automated planning is essential to building confidence in autonomous systems. Multi-objective probabilistic planning has been used in many domains for synthesizing behavior for autonomous systems in uncertain environment, where the systems must optimize their behavior for multiple qualities of concern. When the autonomous systems make decisions (i.e., generate planning solutions), it is natural for the humans overseeing or interacting with those systems to wish to understand what the planning solutions are, and what qualities those solutions have. One approach for achieving this is for the autonomous systems to provide an explanation that describes and helps making sense of an automatically generated planning solution, which I refer to it as a *"what-explanation"*. Beyond that, to have calibrated confidence in the decisions made by the autonomous systems, it is also important to understand their mechanisms for reasoning about qualities of planning solutions. In particular, this means understanding why the planning process selected the solution it did, in terms of the qualities of concern of the planning problem. This requires an explanation that justifies why the selected solution is best with respect to the qualities of concern, and additionally explains how those qualities are evaluated, which I refer to as a *"why-explanation"*.

In this work, I focus on explainability of probabilistic planning problems formulated as Markov decision processes (MDPs). As for the multi-objective optimization part of the planning, I consider the use of analytic models of the qualities of a planning solution to be optimized (referred to as *quality attributes*), which must be encoded in a MDP model as reward functions, and the use of utility models to impose preferences among those quality attributes. Understanding such kind of automated planning is challenging because its reasoning mechanism comprises multiple analyses of quality-attribute evaluation, as well as complex preference and value-tradeoff analyses in the case of conflicting quality optimization objectives. Those analyses are opaque in the existing planning approaches and tools. Particularly, it is challenging for humans to understand how the planning process makes assessment of the quality attributes, and how it makes value tradeoffs among conflicting quality objectives to determine the best solution in a given circumstance. Moreover, the sequential and nondeterministic nature of policies make it difficult for humans to make sense of solutions of MDP-based planning.

In this thesis, I propose to promote transparency and understandability of such kind of automated planning, by enabling autonomous systems to explain and justify their planning solutions, on the basis of the corresponding quality attributes and their optimization objectives. The goal of my proposed approach is to automatically generate *what-explanation* and *why-explanation* for solution policies of multi-objective MDPs. The proposed approach comprises four elements: (i) Explainable planning language. This language is designed to preserve the semantics of quality attributes, their evaluation models, and their utility models in the planning problem formulation. This language makes it possible to construct explanation and justification for planning solutions from their corresponding problem definitions, in a way that reflects the underlying intents of the problems. (ii) Policy summarization. This is a method for describing a probabilistic policy concisely (i.e., *what-explanation*). (iii) Contrastive quality-attribute-based justification. This is a method for generating a justification for selecting a particular solution policy (i.e., part of *why-explanation*). Such justification is based on contrasting the selected solution policy to some alternative policies on

the basis of their quality-attribute properties, and indicating the value judgment that considers the selected policy as being superior. Lastly, (iv) Quality-attribute evaluation explanation. This is a method for providing evidence supporting the claims made about quality-attribute properties in the justification (i.e., part of *why-explanation*). The approach leverages the explicit analytic models of quality-attribute evaluation to identify and describe the determining factors of the quality-attribute properties of a policy.

1 Introduction

Systems today are becoming increasingly more autonomous. They perform more critical tasks in several domains, work more closely with humans, and become more integrated in our everyday lives. In many important domains of applications, autonomous systems must make decisions, both for their own behavior and for recommendations they provide to humans as decision support, that involve optimizing the behavior for multiple qualities of concern, in the presence of uncertainty. It is important for humans who work or interact with those autonomous systems to build trust and confidence in the systems' decisions. At the same time, they should be able to recognize when the systems make wrong decisions, in order to intervene appropriately.

One approach to solve decision making and planning problems in the domains where autonomous systems have multiple quality-optimization objectives and operate in the presence of uncertainty is to formulate the problems as Markov Decision Processes (MDPs). Existing work has demonstrated how a system and its environment can be modeled in the MDP framework, and how the metrics for evaluating qualities of a strategy of the system to achieve a given goal (i.e., a policy [1]) are defined. Such qualities of concern of a policy are referred to as *quality attributes*. In current approaches, the metrics for evaluating quality attributes are encoded as the reward structure of a MDP. Thus, solving for an optimal policy (i.e., one with the maximum expected cumulative reward) for such a MDP yields a strategy that is optimized for the given quality attributes. Since a decision making or planning problem can have multiple quality attributes, their optimization objectives can be conflicting in some situations. In order to determine a single optimal policy, preference over the quality attributes must be taken into account. To this end, utility functions are used to encode such preferences. Therefore, the reward structure of a MDP essentially assigns utility values to the characteristics of the system, its operations, and its environment, as a means for characterizing the underlying quality-attribute properties of a policy.

To build and calibrate confidence in decisions made by autonomous systems via multi-objective probabilistic planning discussed above, it is important for humans overseeing or interacting with the systems to understand the reasoning mechanism in terms of the underlying semantics of the problem domains. To this end, humans must first know what decisions the systems have made, and what quality-attribute properties those decisions entail. One approach for achieving this is for the autonomous systems to provide an explanation that describes and helps making sense of their generated MDP policies, which I refer to as a *"what-explanation"*. Then, humans must understand why the planning processes of the autonomous systems generated the policies they did, in terms of the quality attributes of the respective planning problems. This requires an explanation that justifies why a generated policy is best with respect to the corresponding quality attributes, and additionally explains how those quality attributes are evaluated, which I refer to as a *"why-explanation"*.

However, such reasoning mechanisms of autonomous systems today are opaque. They are not built for easy understanding of *what* the systems have decided to do, and *why* the systems have decided to do them. As a result, mistakes made by the autonomous systems may be left uncaught. Particularly, when a system must make value tradeoffs when its quality-optimization objectives are

[1] A policy is a function mapping from a set of states to a set of actions. It describes what action should be taken in a given world state.

conflicting, the system's decision may disagree with humans' preferences and value judgments. Furthermore, since the system may assess the quality attributes based on the predictions of some hidden or future states, if the predictions are wrong, then the system may generate a suboptimal policy as a result. Lacking an understanding of how the system assesses the quality attributes and how it makes value tradeoffs among competing optimization objectives, humans may not know when autonomous systems make good decisions (i.e., ones that are based on correct assessments of the quality attributes and agree with humans' preferences of them) or otherwise.

However, automatically generating *what-explanation* and *why-explanation* for multi-objective probabilistic planning has a number of challenges. Regarding generating *what-explanation*, policies of MDPs can be complex due to multi-step state evolution and nondeterministic transitions. Therefore, it is difficult for a human to make sense of a complex policy, and it is difficult to describe such a policy in a concise yet sufficiently complete way. Regarding generating *why-explanation*, the concerned planning mechanism involves multiple analyses of quality-attribute evaluations, and of complex preferences and value-tradeoff. As discussed earlier, to solve for an optimal policy of multi-objective probabilistic planning (formulated as a MDP) is to find a policy that has the maximum expected cumulative reward, where the reward structure encodes the utility values characterizing the underlying quality-attribute properties of a policy. There are three explainability challenges to this planning mechanism:

- There is no traceability from the expected cumulative reward of a policy (i.e., the indicator of how desirable a policy is) to the quality attribute properties of that policy. That is, given a reward value, one cannot determine its corresponding quality attribute value(s), especially because different kinds of quality attributes can be quantified in different ways. In other words, one cannot know *quantitatively* how good or bad a policy is in terms of the individual quality attributes of concern, even when one knows the reward value of the policy. Since there are no other constructs in the standard MDP framework that contain information about quality attribute values, one cannot explain a solution policy on the basis of its qualities from the planning problem definition.

- Preferences over quality attribute values, both within the same type and across different types of attributes, are implicit in a reward structure. That means, determining, from a reward structure, how important a particular quality attribute value would be to the *overall* objective of a planning problem is difficult. When there are multiple conflicting objectives in a planning problem (i.e., there are multiple quality attributes, and improving some quality attributes causes some others to degrade), it is important to understand what tradeoffs are made in determining a "best" solution policy. Since the importance of individual quality attribute values to the overall planning objective is hidden in the reward structure, one cannot justify a solution policy on the basis of the *utilities* of its qualities (i.e., how important the qualities are) from the planning problem definition.

- The evaluations of quality attributes are implicit in a reward structure. That is, there is no traceability from the quality attribute values of a policy to the characteristics of the states and actions that determine those values. To evaluate quality attributes of a policy, there must be some predetermined measurement models that map characteristics of the building blocks

of the policy (i.e., states and actions) to the corresponding quality attribute values. However, once one encodes these measurement models of quality attributes as a reward structure, the relationships between the state-action characteristics of a policy and its quality attribute values become implicit. Consequently, assuming that one knows the quality attribute values of a particular policy, one cannot explain those values on the basis of the characteristics of the policy. In other words, when a justification (i.e., *"why-explanation"*) makes a claim that a policy has particular quality attribute values, one may not be confident about the claim due to the lack of supporting evidence – in terms of the policy's characteristics associated with those quality attributes.

These challenges must be addressed to enable autonomous systems to explain and justify their solution policies in terms of the quality attributes of the respective planning problems.

To promote explainability of multi-objective probabilistic planning as discussed, my work addresses the following research questions:

RQ1. How can we generate an explanation of a probabilistic policy (i.e., making sense of the policy) in a concise and sufficiently complete way?

RQ2. How can we generate an explanation for why a particular policy is selected in a given planning problem (i.e., justifying the policy) in a concise and sufficiently complete way?

RQ3. How can we provide supporting evidence for the claims made in a justification for selecting a policy in a given planning problem?

In this thesis, I propose to promote transparency and understandability of such kind of automated planning, by enabling autonomous systems to explain and justify their planning solutions, on the basis of the corresponding quality attributes and their optimization objectives. The goal of my proposed approach is to automatically generate *what-explanation* and *why-explanation* for solution policies of multi-objective MDPs. The proposed approach comprises four elements: (i) *Explainable planning language*. This language is designed to preserve the semantics of quality attributes, their evaluation models, and their utility models in the planning problem formulation. This language makes it possible to construct explanation and justification for planning solutions from their corresponding problem definitions, in a way that reflects the underlying intents of the problems. (ii) *Policy summarization*. This is a method for describing a probabilistic policy concisely (i.e., *what-explanation*). (iii) *Contrastive quality-attribute-based justification*. This is a method for generating a justification for selecting a particular solution policy (i.e., part of *why-explanation*). Such justification is based on contrasting the selected solution policy to some alternative policies on the basis of their quality-attribute properties, and indicating the value judgment that considers the selected policy as being superior. Lastly, (iv) *Quality-attribute evaluation explanation*. This is a method for providing evidence supporting the claims made about quality-attribute properties in the justification (i.e., part of *why-explanation*). The approach leverages the explicit analytic models of quality-attribute evaluation to identify and describe the determining factors of the quality-attribute properties of a policy.

2 Thesis

2.1 Thesis Statement

My thesis is:

> *We can improve transparency of autonomous systems' reasoning, and thereby allow humans to calibrate their confidence in the decisions made by autonomous systems, by enabling the systems to explain and justify their behavior generated from multi-objective probabilistic planning.*

Next, I elaborate on the thesis statement.

This thesis proposes a solution to the problem of explainability of multi-objective probabilistic planning, formulated as Markov Decision Process (MDP). This work focuses on explaining (i.e., making sense of) a planning solution, i.e., a probabilistic policy, and justifying it (i.e., explaining why the planner selected this particular solution) based on the optimization objectives of the planning problem. In this thesis, I refer to measurable quantities that are to be optimized or constrained in some planning problems as *quality attributes*. The proposed approach aims at providing transparency of how an autonomous decision maker assesses problem situations in terms of the quality attributes of concern, and how it makes tradeoff among conflicting quality-attribute-based optimization objectives in finding an optimal policy.

Humans' confidence in the decisions made by autonomous systems refers to a degree of belief that the decisions, if followed accordingly, will likely yield desirable results with respect to some evaluation criteria. Having a calibrated confidence in a decision means that the humans have high confidence in the decision when in fact following the decision would likely yield desirable results (according to the available domain knowledge), and have low confidence otherwise. This thesis claims that explanations generated by the proposed approach ultimately allow humans to know when autonomous systems make good decisions and when they make bad decisions. If and when the systems make bad decisions, the explanations would allow the humans to identify what specifically the systems did wrong regarding quality-attribute evaluation and optimization. Therefore, humans can have well-calibrated and justifiable confidence in autonomous systems' decision making.

My proposed approach to explain and justify autonomous systems' behavior – generated from multi-objective probabilistic planning – consists of the following: (a) a language for expressing explainable planning problems, and a method for translating problems specified in this language to ones in standard planning languages, (b) an algorithm for generating verbal summaries of probabilistic policies, and (c) an algorithm for generating contrastive quality-attribute-based justification and its supporting evidence for why a particular policy is selected in a given planning problem.

Next, I elaborate on the different elements of the proposed approach, and state the claims of how they enable explanation and justification of autonomous systems' behavior.

> *...(a) a language for expressing explainable planning problems, and a method for translating problems specified in this language to ones in standard planning languages...*

In this work, I use the term *"explainable planning problem"* to refer to a formal representation of a planning problem that can be used as an input to automatically generate explanations – in terms of the concepts and vocabulary from the problem domain. An *explanation* in this case consists of both: (i) a description of the solution policy of a planning problem (*"what-explanation"*), and (ii) a justification of why the policy is selected as the solution (*"why-explanation"*). Existing languages that can express probabilistic planning domains with rewards (e.g., PDDL1.0+ [29], PRISM language [14, 13, 21, 12, 7]) are not suitable for expressing *explainable* multi-objective probabilistic planning problems, particularly for explaining quality attributes. In these existing planning languages, a reward construct can be used to characterize the quality attributes of a planning domain. Such reward construct essentially assigns numerical values to state-action pairs of a planning problem, indicating how preferable each state-action pair is with respect to the quality attributes. The goal of planning is then to find a policy that has a maximum expected cumulative reward value, which is a policy that is "best" with respect to those quality attributes. However, there are three explainability shortcomings of this modeling approach:

1. There is no traceability from the expected cumulative reward value of a policy (i.e., the indicator of how desirable a policy is) to the quality attribute properties of that policy. That is, given a reward value, one cannot determine its corresponding quality attribute value(s), especially because different kinds of quality attributes can be quantified in different ways (to be discussed in Section 5.1.2). In other words, one cannot know *quantitatively* how good or bad a policy is in terms of the individual quality attributes of concern, even when one knows the reward value of the policy. Since there are no other constructs in existing planning languages that contain information about quality attribute values, one cannot explain a solution policy on the basis of its qualities from the planning problem definition.

2. Preferences over quality attribute values, both within the same type and across different types of attributes, are implicit in a reward structure. That means, determining, from a reward structure, how important a particular quality attribute value would be to the *overall* objective of a planning problem is difficult. When there are multiple conflicting objectives in a planning problem (i.e., there are multiple quality attributes, and improving some quality attributes causes some others to degrade), it is important to understand what tradeoffs are made in determining a "best" solution policy. Since the importance of individual quality attribute values to the overall planning objective is hidden in the reward structure, one cannot justify a solution policy on the basis of the *utilities* of its qualities (i.e., how important the qualities are) from the planning problem definition.

3. The evaluations of quality attributes are implicit in a reward structure. That is, there is no traceability from the quality attribute values of a policy to the characteristics of the states and actions that determine those values. To evaluate quality attributes of a policy, there must be some predetermined measurement models that map characteristics of the building blocks of the policy (i.e., states and actions) to the corresponding quality attribute values. However, once one encodes these measurement models of quality attributes as a reward structure in the standard planning languages, the relationships between the state-action characteristics of a policy and its quality attribute values become implicit. Consequently, assuming that

one knows the quality attribute values of a particular policy, one cannot explain those values on the basis of the characteristics of the policy. In other words, when a justification (i.e., *"why-explanation"*) makes a claim that a policy has particular quality attribute values, one may not be confident about the claim due to the lack of supporting evidence – in terms of the policy's characteristics associated with those quality attributes.

Therefore, the first element of my proposed approach is a new *explainable* planning language that addresses these three explainability shortcomings. I claim that the proposed planning language enables automatic generation of *"what-explanation"* and *"why-explanation"* of multi-objective probabilistic planning. Additionally, such explainable planning language must be translatable to existing standard planning languages, so that existing planning engines can be used to solve *explainable* planning problems. Likewise, policies generated by existing planning engines must be translatable to policies in the proposed language.

...(b) an algorithm for generating verbal summaries of probabilistic policies...

The second element of the proposed approach is an algorithm for generating *what-explanation*, i.e., a description of a selected policy. Since a policy can contain multi-step state evolution and nondeterministic transitions, a full description of a policy can potentially be too complex and incomprehensible. Therefore, an explanation must provide a summarized description of a policy, but in a way that is consistent with its justification for the policy (i.e., the *"why-explanation"*). I claim that such automatically generated *"what-explanation"* can provide humans with sufficient insights into the planned behavior of autonomous systems, to be able to understand the accompanying *"why-explanation"* and calibrate their confidence in the systems' planned behavior.

...and (c) an algorithm for generating contrastive quality-attribute-based justification and its supporting evidence for why a particular policy is selected in a given planning problem.

I argue that, *contrastive quality-attribute-based justification* can provide sufficient insights into the reasoning behind multi-objective probabilistic planning. I use the term to refer to an argument for why a policy is selected (in some planning problem) based on a comparison between the quality attribute values of the selected policy and those of some alternative policies. In other words, it explains why the alternatives policies were not selected as the solution. I argue that this kind of justification is appropriate for explaining multi-objective probabilistic planning, because it reflects both the mathematical model of multi-objective decision-making, as well as the way people intuitively use contrastive explanations (i.e., explanations of the form *"Why P rather than Q?"*) [18, 19]. Furthermore, I argue that providing evidence supporting the claims made in a justification – that a policy has certain quality attribute values – improves transparency of the analytic models used in planning for evaluating quality attributes. Therefore, the third element of my proposed approach is an algorithm for generating *why-explanation*, i.e., a justification for why a particular policy is selected, which follows the principles described previously.

With these three elements of the proposed approach, I claim that we can enable autonomous systems to explain and justify their behavior generated from multi-objective probabilistic planning. Furthermore, I claim that the generated explanation and justification increase transparency

of the systems' reasoning mechanism, and enable the explainees to calibrate their confidence in the systems' planned behavior.

2.2 Contributions

The main contributions of this thesis will be:

- An explainable planning language for multi-objective probabilistic planning.
- An approach for generating a verbal description of a policy.
- An approach for generating a *contrastive quality-attribute-based justification* for a solution policy of multi-objective probabilistic planning.
- Improved transparency and understandability of autonomous systems' reasoning, particularly how the systems evaluate quality attributes of solution policies and how they make value tradeoffs in case of conflicting objectives.
- Concrete examples of the use of the explanation generation technique from at least three different planning domains.

3 Related Work

Explaining autonomous systems' behavior and reasoning mechanisms has been studied in different ways across different disciplines. In focusing on explanations as verbal statements, and as *what-explanation* and *why-explanation* types, I discuss the more closely related works in this section.

3.1 *Why-explanation*

First, I discuss prior works related to *why-explanation*. Such works are characterized as approaches that aim to explain why intelligent agents or systems produce particular behavior. Some existing works have investigated policy explanation for MDP-based planning systems. Elizalde et al. [6, 5] contributed an approach for explaining the commands suggested to human operators by an intelligent assistant. Their generated explanation identifies factors that are most influential to the decisions. Their approach is based on determining the variable in a given state that has the highest influence on the choice of the optimal action, by analyzing the impact of each variable on the utility value and on the optimal actions. Khan, Poupart, and Black [9] also contributed an approach for explaining an optimal action choice for a given state in a policy. Their approach explains a choice of an action by computing the frequency of reaching a goal scenario by taking the action and stating that it is the highest frequency. Khan et al. also introduced a notion of minimal sufficient explanation, which can prove that the recommended action in a given state is optimal, and uses fewest possible terms. More recently, Hayes and Shah [8] contributed algorithms and implementation for enabling robots to generate descriptions of their policies and to respond to both general and targeted queries from humans. Their supported query types include questions

about environmental conditions under which certain behavior will occur, about robot behavior that will occur under specific environmental conditions, and about why a particular behavior did not occur.

While these existing works enable policy explanation for MDP-based planning systems, they focus on explaining optimal actions in a policy solely based on either: (i) impact of action choices on the total reward or goal reachability, or (ii) conditions of the state variables under which certain actions are taken. The works by [6, 5, 9] fall into the first category. They do not address the underlying semantics of rewards, which I argue in this thesis that it is important for promoting human understanding of the decision rationale. The work by [8] falls into the second category. Their generated explanations contrast conditions under which different actions are taken, but not justify why the actions should be taken. Moreover, unlike my proposed work, they do not address the problem of multiple optimization functions and justifying optimal actions on the basis of these objectives.

Plan explanation for other planning methods has also been studied in prior work. Seegebarth et al. [25] proposed an approach for explaining hybrid planning, which combines partial-order-causal-link planning and hierarchical planning. Their explanations of a plan are proofs for the necessity of plan steps and orderings on plan steps. Their approach uses a first-order logic axiomatic system that formalizes basic arguments about a plan and the logical consequences that can be derived from them. Bidot et al. [2] presented an approach for generating verbal explanation of hybrid planning, where the explanations offer justifications for the ordering of two tasks and for the temporal position of a single task. Unlike these prior works, my approach does not aim to explain the causality of plan or policy steps. Instead, my proposed *why-explanation* focuses on explaining the relative desirability of a policy, with respect to the corresponding quality-optimization objectives.

The problem of model reconciliation in plan explanation has been posed and investigated by Chakraborti and Sreedharan et al. [3]. They argue that since humans may have domain and task models that differ from those used by AI systems, explanation can be formulated as model reconciliation problem. Their approach aims to identify minimal changes to the humans' models to bring them closer to the AI systems' models, where the models in consideration are models of classical planning problems. Although my proposed work is not primarily concerned with model reconciliation in explanation, it has some similarity to this work. Particularly, my approach uses contrastive justification for *why-explanation*, which can potentially lead to discovery of differences in the quality-attribute utility models between the human explainees' and the autonomous systems'.

Apart from plan and policy explanation, there are several other existing works on explaining intelligent systems' behavior. In the area of human-computer interaction, researchers have studied how to enable rule-based context-aware systems and machine learning applications to explain their reasoning algorithms to users. For instance, Lim and Dey et al. [4, 17, 15] investigated demand for intelligibility in context-aware systems and explanations that address different intelligibility type questions. They also contributed an implementation of automatic explanation generation for the different explanation types and decision model types, including rules, decision trees, naive Bayes, and hidden Markov models [16]. Kulesza et al. [10, 11] investigated how explanations can

affect end users' mental models of intelligent agents' personalized behavior. Their studies were conducted on explaining decision-tree bagging ensemble classifier.

3.2 *What-explanation*

Second, I discuss prior works related to *what-explanation*. Such works are characterized as approaches that describe and help making sense of behavior of intelligent agents or systems. As mentioned in Section 3.1, the work by Hayes and Shah [8] also tackles the problem of describing policies. Their approach enables a robot to describe its behavior under different environmental conditions. This work is the most closely related to my proposed approach for generating *what-explanation*. I plan to adopt some of their techniques to synthesize policy descriptions. However, the requirements of my proposed approach for *what-explanation* differ from Hayes and Shah's work in that a policy description must allow the explainee to understand the accompanying justification (i.e., *why-explanation*) for the policy.

In the area of human-robot interaction, prior works have investigated approaches to enable autonomous robots to communicate with humans through verbal statements. St. Clair and Matarić [26] presented an approach for robots to communicate role allocations during a task collaboration with humans, where the coordination problem is formulated as an MDP. Role allocation narrations are phrases such as "I'll take care of X" and "You handle X". While their work contributes to communicating robots' policies to humans, it differs from my proposed *what-explanation* in that their approach assumes to have a set of predefined, domain-dependent policies and a corresponding set of predefined verbal phrases describing the policies. St. Clair and Matarić's approach also describes a policy by providing feedback about actions given the current state. In contrast, my proposed *what-explanation* would need to describe arbitrary synthesized policies, before their executions occur.

In the mobile robots domain, Rosenthal, Selvaraj, Veloso, and Perera [24, 22] investigated approaches for generating narrations of mobile robot experience. They contributed route verbalization algorithms and introduced verbalization spaces. In [24], their approach describes routes taken by robots, as well as concepts associated with routes such as distance, speed, and time of travel. Although such description of route properties is similar to my approach of describing quality-attribute properties of a policy, the key differences are: (i) my approach provides a general mechanism for evaluating and describing quality-attribute properties of a policy, and (ii) my approach uses quality attributes as the basis for generating contrastive justification for a solution policy. [22] contributes dynamic generation and refinement of route verbalization. This work aims to personalize robot narrations to user preferences. The key differences between verbalization in this work and my proposed approach regarding route properties are as stated previously.

There is also a research area in summarizing or generating narratives of perceived behavior. Veloso et al. [27] presented robot commentators for AIBO robot soccer games. Their approach enables the robot commentators to announced events of a game by detecting events from the game controller and from their own vision, and producing verbal utterances describing the game. Their *commentator algorithm* shares some similar goals with my proposed *what-explanation*, namely, to identify relevant elements or features from low-level information about a behavior, and to summarize the behavior over a period of time in verbal form. However, the key differences are: (i) my

work aims to summarize contigent behavior that has not already been executed and observed, and (ii) the filtering of relevant elements of a behavior is based on their relevance and impact on the quality attributes of the corresponding planning problem. Similarly, summarizing perceived behavior have also been studied in other application areas, including military exercises [20], video conferencing [28], and sports games [1].

4 Background

This section introduces *Markov decision processes (MDPs)*, the probabilistic planning model used in this work. Section 4.1 provides the formal definition of an MDP, and an overview of algorithms for solving for optimal policies of MDPs. Furthermore, Section 4.2 introduces quantitative verification and policy synthesis for MDPs, which are the key techniques I use in this work.

4.1 Markov Decision Processes

In this work, I formulate planning problems as Markov decision processes (MDPs) with rewards associated with state-action pairs.

Definition 4.1. *Markov decision process.* An MDP is a tuple $\mathcal{M} = \langle S, s_0, A, \mathcal{P}r, AP, Lab, r\rangle$, where: (i) S is a finite set of states; (ii) $s_0 \in S$ is the initial state; (iii) A is a finite set of actions; (iv) $\mathcal{P}r : S \times A \times S \to [0,1]$ is a probabilistic transition function, where $\sum_{s' \in S} \mathcal{P}r(s,a,s') \in \{0,1\}$ for all $s \in S, a \in A$; (v) AP is a set of atomic propositions; (vi) $Lab : S \to 2^{AP}$ is a labeling function, such that $p \in Lab(s)$ if and only if the atomic proposition p is true in state s; and (vii) $r : S \times A \to \mathbb{R}_{\geq 0}$ is a reward function, associating each state-action pair with a non-negative value.

The choice of action to take at each step of the execution of an MDP $\mathcal{M}$ is made by a *policy*. In this work, I consider only *memoryless* policies, which base their decisions only on the current state. Formally, a policy is a function $\pi : S \to A$ such that, for any state $s, \pi(s) \in A(s) = \{a \in A \mid \mathcal{P}r(s,a,s') > 0 \text{ for some } s' \in S\}$. In this work, I consider an MDP planning problem as the following.

Problem 1. Let $\mathcal{M}$ be an MDP and $p \in AP$ an atomic proposition. Generate a policy $\pi^{max}_{\mathcal{M}}(r, \mathtt{F}\, p)$ for $\mathcal{M}$ that maximizes the expected value of the accumulated reward to reach a state $s_n \in S$ such that $p \in Lab(s_n)$. More formally, the expected accumulated reward value of a policy π starting from an initial state s is:

$$V(\pi, s) = \begin{cases} r(s, stop) & \text{if } p \in Lab(s) \\ r(s, \pi(s)) \; + \; \sum_{s'} \mathcal{P}r(s' \mid s, \pi(s))\, V(\pi, s') & \text{otherwise,} \end{cases}$$

where *stop* is a special action designating a termination. Thus, the aim of the planning is to generate a policy π^* such that $\pi^* = \operatorname{argmax}_{\pi} V(\pi, s_0)$.

If there is a path starting from the initial state s_0 that never reaches a state satisfying the proposition p, then the maximum value of $V(\pi, s_0)$ is infinity. Therefore, in some planning problems, it is more appropriate to use a *cost function* $c : S \times A \to \mathbb{R}_{\geq 0}$ instead of a reward function. Cost functions are appropriate for planning problems whose goal conditions (i.e., represented by the proposition p) are not guaranteed to be met after taking some finite sequence of actions. An example is a problem of finding a shortest path to a goal. The aim of such planning is therefore to generate a policy $\pi^* = \underset{\pi}{\operatorname{argmin}}\, V(\pi, s_0)$ instead.

There are several algorithms to compute an optimal policy (i.e., one with maximum reward or minimum cost), including value iteration, policy iteration, and linear programming [23]. Problem 1 can be solved by first performing a graph analysis of the MDP model to identify states for which the reward (or cost) is infinite [7], then using one of the MDP algorithms.

4.2 Quantitative Verification and Policy Synthesis

My proposed approach for generating explanations leverages the quantitative verification and policy synthesis techniques, provided by the probabilistic model checking tool, PRISM [14]. More specifically, I use *numerical queries* [7] on *discrete-time Markov chains (DTMCs)* to determine: (i) probabilities of the model satisfying some properties, and (ii) some reward values of the models.

First, I provide the definition of a fragment of the property specification language from PRISM I use in this work.

Definition 4.2. *Property Specification*. A *property* is a formula ϕ derived from the grammar:

$\phi ::= \mathtt{P}_{\bowtie p}[\psi] \mid \mathtt{R}^{r}_{\bowtie x}[\rho]$

$\psi ::= \mathtt{true} \mid b \mid \psi \wedge \psi \mid \neg\, \psi \mid \mathtt{X}\, \psi \mid \psi\, \mathtt{U}^{\leq k}\, \psi \mid \psi\, \mathtt{U}\, \psi$

$\rho ::= \mathtt{C}^{\leq k} \mid \mathtt{C} \mid \mathtt{F}\, b$

where $b \in AP$ is an atomic proposition, $\bowtie \in \{\leq, <, \geq, >\}, p \in [0,1], r$ is a reward function, $x \in \mathbb{R}_{\geq 0}$, and $k \in \mathbb{N}$. ψ is a linear temporal logic (LTL) formula and ρ is a reward objective, where $\mathtt{C}^{\leq k}$ denotes a bounded cumulative reward, $\mathtt{C}$ denotes a total reward, and $\mathtt{F}\, b$ denotes a reachability reward.

A property ϕ of the form $\mathtt{P}_{\bowtie p}[\psi]$ of a DTMC $\mathcal{C}$ states that the probability that $\mathcal{C}$ satisfies the LTL property ψ meets the threshold p. Similarly, a property of the form $\mathtt{R}^{r}_{\bowtie x}[\rho]$ states that the *expected cumulative* value of the reward function r (accumulated according to the objective ρ) meets the threshold x.

Problem 2. Given Definition 4.2 and let $\mathcal{C}$ be a DTMC. A *numerical query* of the form $\mathtt{P}_{=?}[\psi]$ is a problem of determining a probability of property ψ being satisfied in $\mathcal{C}$. Likewise, a numerical query of the form $\mathtt{R}^{\mathtt{r}}_{=?}[\rho]$ is a problem of determining an expected reward of reward function r accumulated until k steps have taken, indefinitely, or until a state satisfying b is reached in $\mathcal{C}$, when objective ρ is $\mathtt{C}^{\leq k}$, $\mathtt{C}$, and $\mathtt{F}\, b$, respectively.

I use this numerical query technique to generate descriptions of quality-attribute properties of MDP policies, as discussed in Section 5.4.1.

The other key technique on which my approach build is *multi-objective policy synthesis* for MDPs. It is a problem of generating a policy that satisfies one or more (probabilistic or reward-based) properties, while maximizes some reward or probability of satisfying some LTL property. Problem 1 is a special case of this problem, where there is only one reachability property $\mathtt{F}\, p$ and a reward function r.

Problem 3. Given Definition 4.2 and let $\mathcal{M}$ be an MDP. A *multi-objective policy synthesis* of the form $\mathtt{multi}(\mathtt{P}_{\mathtt{min}=?}[\psi], \phi)$ or $\mathtt{multi}(\mathtt{P}_{\mathtt{max}=?}[\psi], \phi)$ is a problem of finding a policy that minimizes or maximizes the probability of satisfying the LTL property ψ, while satisfies the property ϕ, respectively. Likewise, $\mathtt{multi}(\mathtt{R}^{\mathtt{r}}_{\mathtt{min}=?}[\rho], \phi)$ or $\mathtt{multi}(\mathtt{R}^{\mathtt{r}}_{\mathtt{max}=?}[\rho], \phi)$ is a problem of finding a policy that minimizes or maximizes the cumulative reward r value, while satisfies the property ϕ, respectively.

I use this multi-objective policy synthesis technique to generate alternative policies for contrastive justification, as discussed in Section 5.4.2.

5 Proposed Approach

This section describes the proposed approach in greater detail. As mentioned in Section 2.1, the approach is based primarily on designing an *explainable planning language*, which enables automatic generation of explanations and justifications of solution policies from planning problem definitions, and on *contrastive quality-attribute-based justification*. Therefore, this section begins by discussing the design and formalism of the proposed language in Section 5.1. Section 5.2 describes the translation between the explainable planning representation and the regular MDP representation. Then, Section 5.3 describes the proposed algorithm for summarizing policies – to provide descriptions of solution policies (i.e., *"what-explanations"*) as well as alternative policies (discussed in *why-explanations*). Section 5.4 discusses the proposed algorithm for generating justifications (i.e., *"why-explanations"*) for solution policies. Such justifications identify alternative policies, and argue why the original solution policies are superior to the alternatives with respect to the quality attributes of concern. Finally, Section 5.5 describes the approach to explain how the quality attribute values stated in a justification are evaluated.

Note how the elements of this approach address the research questions in Section 1:

RQ1. *How can we generate an explanation of a probabilistic policy (i.e., making sense of the policy) in a concise and sufficiently complete way?* Section 5.3 discusses a potential approach to summarize a probabilistic policy in a way that still allows the readers to understand the accompanying justification for the policy.

RQ2. *How can we generate an explanation for why a particular policy is selected in a given planning problem (i.e., justifying the policy) in a concise and sufficiently complete way?* Section 5.4 describes how we can generate contrastive justifications for solution policies on the basis of their respective quality attributes of concern.

RQ3. *How can we provide supporting evidence for the claims made in a justification for selecting a policy in a given planning problem?* Section 5.5 describes how we can support claims made about quality attribute values of a policy by explaining the corresponding evaluation models and determining factors.

5.1 Explainable Planning Language

As mentioned in Section 2.1, I propose to design a language for expressing *explainable planning problems*, which are formal representations from which explanations and justifications for the solution policies can be synthesized. Such language must overcome the three explainability shortcomings of existing standard planning languages. That is, an explainable planning language must be able to preserve the *semantics of quality attributes* of planning problems it represents, namely, the names, measurement models and units of the quality attributes. Moreover, it must also be able to preserve preference models of the quality attributes.

Therefore, I propose to extend the formal definition of probabilistic planning with rewards to *explicitly* represent quality attribute models and their corresponding preference models. Specifically, I propose that the extension must allow us to represent the following:

- *Arbitrary characteristics of states and actions* [2].
- *Analytic models for evaluating quality attributes*. These are measurement models for quantifying different qualities of a policy, based on the characteristics of states and actions in the policy.
- *Utility models of quality attributes*. These include utility models of individual quality attributes, which define how desirable different values of the same quality attribute are relative to one another, and an aggregate utility model, which defines the combined desirability of values of all quality attributes.

Note that in addition to these new elements, a probabilistic planning problem also defines probabilistic transition functions and goal predicates, which are the elements to which my proposed approach does not add any extension or make any modification.

Next, I discuss the importance of representing these elements in planning problem definitions for generating *what-explanation* and *why-explanation* of solution policies, and how they can be formalized.

5.1.1 Characteristics of States and Actions

As discussed in Section 2.1, one of the explainability shortcomings (specifically, #3) of existing planning languages is that there is no traceability from the quality attribute values of a policy to their determining factors. By making the characteristics of the building blocks of a policy explicit, especially those of the actions (which, as noted, existing planning languages do not represent), we

[2]Note that while many existing planning languages can express arbitrary characteristics of states (i.e., encoding state variables), they are not designed to explicitly represent arbitrary characteristics of actions.

can have models that relate the state-action characteristics to the corresponding quality attribute values and vice versa, allowing us to explain the causes of the policy's qualities. Moreover, these state-action characteristics may also be used for describing a policy (i.e., *"what-explanation"*), especially when the policy is probabilistic.

To represent characteristics of states, we use *state variables*. We define a state space $\mathcal{S}$ of a planning problem using finite-domain representation (FDR). $\mathcal{V}$ is a set of state variables and $\mathcal{T}$ is a set of state variable types, where each variable $v \in \mathcal{V}$ has an associated type $\tau \in \mathcal{T}$, which defines a finite domain $\mathcal{D}_\tau$. A state $s \in \mathcal{S}$ is a variable assignment over $\mathcal{V}$. In other words, s is a function on $\mathcal{V}$ such that $s(v : \tau) \in \mathcal{D}_\tau$. The values of state variables describe the characteristics of a state.

To represent characteristics of actions, we use *action properties*. Apart from the fact that they are associated with actions instead of states, action properties differ from state variables in that each action property has a unique type. Before discussing how we can represent action properties, I first discuss how we represent actions in a way that facilitates automatic generation of *what-explanations*. We define an action space $\mathcal{A}$ of a planning problem using action types ACT. An action type $\tau_{act} \in ACT$ definition consists of an action name $act \in Act$ and a parameter list $(p_1 : \tau_1, \ldots, p_k : \tau_k)$, where the types of the action parameters are the state variable types, i.e., $\tau_1, \ldots, \tau_k \in \mathcal{T}$. An action $a \in \mathcal{A}$ is a ground instance of an action type τ_{act} with a structure $act(p_1 : \tau_1, \ldots, p_k : \tau_k)$ with a substitution $\theta = \{p_1 \leftarrow u_1, \ldots, p_k \leftarrow u_k\}$, where each u_i is a value of type τ_i.

Now I discuss how we can represent action properties. Each action type τ_{act} is associated with a (possibly empty) set of *action property types* $PROP_{act}$. An action property type $\tau_{prop} \in PROP_{act}$ defines a (possibly infinite) domain $\mathcal{D}_{prop}$ of property values. These property values represent the characteristics of an action which are independent of the states to which the action is applied. That is, each ground instance of τ_{act}, i.e., an action a, has an associated property value for each action property type $\tau_{prop} \in PROP_{act}$.

In the next sections, I discuss how state variables and action properties constructs are used for generating explanations, both directly and indirectly as parts of the other language constructs for explainability.

5.1.2 Quality-Attribute Analytic Models

A *quality attribute* (QA) is a characteristic of a policy that is important in a planning problem. In this work, I consider the kinds of quality attributes that can be numerically quantified, and that the objectives of planning is to optimize (i.e., minimize or maximize) the quality attributes. A single planning problem can have multiple quality attributes, and their respective optimization objectives can potentially be conflicting – that is, two or more individual objectives cannot be achieved simultaneously without a tradeoff.

As discussed in Section 2.1, in existing planning languages, there is no traceability from reward values to the quantitative characteristics of a policy that matter to the planning problem (explainability shortcoming #1). By making the quality attributes and their evaluation models explicit, we can generate explanations that: (i) provide quantitative, problem-domain-specific measures of how good or bad a policy is with respect to the different qualities of concern, and (ii) describe how the

derived quality attribute values are calculated – these are values derived from other characteristics of a policy.

To represent a quality attribute, we can define a function that maps the characteristics of each building block of a policy (i.e., each state-action pair) to the corresponding value representing the particular quality of that building block. Then, these values of all building blocks of a policy are aggregated to form a single *quality attribute value* of the policy. The advantage of this approach of modeling quality attributes is that we can use the existing solution methods for finding optimal-reward policies to optimize the quality attributes. That is because numerical values (associated with some quality attributes) assigned to the state-action building blocks of a policy can be encoded as a reward structure.

Formally, a quality attribute i can be represented as follows. We define a *quality attribute function* $\mathcal{Q}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ that assigns non-negative real values characterizing the QA i to state-action pairs. An aggregate of all $\mathcal{Q}_i(s,a)$ values of all state-action pairs in a policy π is a *quality attribute value* of π (to be discussed later in this section). $\mathcal{Q}_i$ function calculates a value of a given state-action pair from particular subsets of the state variables and of the action properties. In other words, we can define a sub-QA function $\mathcal{Q}_i^{act}$ associated with a particular action type $\tau_{act} \in ACT$ for a QA i as $\mathcal{Q}_i^{act} : \mathcal{D}_{v_j} \times \ldots \times \mathcal{D}_{v_k} \times \mathcal{D}_{prop_m} \times \ldots \times \mathcal{D}_{prop_n} \to \mathbb{R}_{\geq 0}$, where $\mathcal{D}_{v_j}, \ldots, \mathcal{D}_{v_k}$ are the domains of the state variables $v_j, \ldots, v_k$, and $\mathcal{D}_{prop_m}, \ldots, \mathcal{D}_{prop_n}$ are the domains of the action properties, i.e., the action property types $\tau_{prop_m}, \ldots, \tau_{prop_n}$ of τ_{act}. Such definition of $\mathcal{Q}_i^{act}$ function for each QA i and each action type τ_{act} makes explicit how a particular characteristic of concern of a policy is measured.

Quality attributes can be categorized into different kinds depending on how they are quantified. Such categorization is helpful for formal modeling of QAs, as well as for explaining the QAs and how they are evaluated. Particularly, how $\mathcal{Q}_i(s,a)$ values of all state-action pairs in a policy π are aggregated to form a single QA i value of π, and how to interpret such value, depend on the kind of QA i. I propose that we can categorize QAs into 4 groups based on how they are quantified: standard measurement, non-standard measurement, number of occurrences of event, and probability of event. Next, I describe the semantics of these different kinds of QAs, how they can be formalized in *explainable* planning problem definitions, and the implications on how we explain QAs.

Standard measurement A QA of this kind can be measured in a standard, scientific way – either directly or by deriving from other measurements. Such QA is quantified by its magnitude – e.g., duration, length, rate of change, utilization, etc. – typically measured in real values. Some examples of these QAs are: response time (seconds) of a web application, network throughput (number of transactions per second), and energy consumption (watt-hour) of a system. There are two kinds of such QAs which differ in their temporal nature: *cumulative* and *instantaneous* quality attributes.

Cumulative Quality Attributes A cumulative QA of a policy is characterized by a value that accrues additively over the execution of the policy, starting at some initial state. Some examples are total execution time, total energy consumed, and total number of customer requests served. In

this case, a *cumulative QA value* of a policy π starting at an initial state s_0, denoted by $V_i(\pi, s_0)$, is an expected cumulative value of $\mathcal{Q}_i(s,a)$ for all state-action pairs in π reachable from s_0, as calculated by:

$$V_i(\pi, s) = \begin{cases} \mathcal{Q}_i(s, \mathit{stop}) & \text{if } s \text{ is an absorbing state} \\ \mathcal{Q}_i(s, \pi(s)) \; + \; \sum_{s'} \mathcal{P}r(s' \mid s, \pi(s))\, V_i(\pi, s') & \text{otherwise,} \end{cases} \tag{1}$$

where *stop* is a special action designating a termination.

Instantaneous Quality Attributes An instantaneous QA of a policy is characterized by an instantaneous (or a small time-window) measurement of a potentially varying value throughout the execution of the policy. Instantaneous QAs differ from cumulative QAs in that measurement values from various steps in a policy are not to be summed up; each of them characterizes the policy at a particular step only. Some examples are response time of a web application and network throughput. In this case, there are two ways to describe an *instantaneous QA value*: as an expected value at a particular step in a policy, and as a summary value of all steps in a policy.

An expected value of an instantaneous QA i of a policy π, starting at an initial state s_0, at its nth step, denoted by $V_i(\pi, s_0, n)$, is calculated by:

$$V_i(\pi, s_0, n) = \begin{cases} \mathcal{Q}_i(s_0, \pi(s_0)) & \text{if } n = 0 \\ \mathbb{E}[Q_n] & \text{if } n > 0, \end{cases}$$

where s_0 is an initial state, and Q_n is a random variable whose possible outcomes are $\mathcal{Q}_i(s_j, \pi(s_j))$ for all possible states s_j at the nth step in π, and the probabilities of these outcomes are the probabilities of the states s_j at the nth step in π, denoted as $P(s_j)$. Such a probability is calculated by:

$$P(s_j) = \begin{cases} 1 & \text{if } s_j \text{ is an initial state} \\ \sum_{s_p} \mathcal{P}r(s_j \mid s_p, \pi(s_p))\, P(s_p) & \text{otherwise,} \end{cases}$$

where s_p is a parent of s_j in π. Therefore, we can rewrite $V_i(\pi, s_0, n)$ as:

$$V_i(\pi, s_0, n) = \begin{cases} \mathcal{Q}_i(s_0, \pi(s_0)) & \text{if } n = 0 \\ \sum_{s_j} \mathcal{Q}_i(s_j, \pi(s_j))\, P(s_j) & \text{if } n > 0. \end{cases} \tag{2}$$

For easy interpretation of multiple instantaneous values throughout a policy, we can compute a *summary value* of the policy for each instantaneous QA. A *summary function* $\mathcal{G}_i : \Pi \times \mathcal{S} \to \mathbb{R}$ is defined for each instantaneous QA i to calculate a value describing the aggregate of all distributions of the instantaneous QA values at all steps throughout a given policy, starting from a given initial state. Some examples of summary functions are: average of the expected values of the response time at all steps, and expected number of times the response time exceed 10 seconds. However, it is important to note that each summary function $\mathcal{G}_i$ must be designed such that the following

condition holds: any pair of policy π and initial state s_0 that has a more *preferable* summary value $\mathcal{G}_i(\pi, s_0)$ must always have a higher utility value $U_i^{total}(\pi, s_0)$ (see Equation 5). This condition is to ensure that the explainee receives consistent information: if a solution policy has a better (summarized) value of an instantaneous QA than its alternative does, then it should be assigned a higher total utility with respect to that QA, and vice versa.

Non-standard, non-binary measurement Some properties may 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. That is, a *non-standard, non-binary measurement* maps the characteristics of states and actions to numerical values representing the corresponding levels or degrees of some property of concern.

Consider the following example, suppose we are concerned about intrusiveness of a mobile robot, that is, we would like the robot to avoid entering private spaces and instead to use routes that go through public spaces. There is no standard measurement for intrusiveness; however, we can define a measurement that assigns different levels of intrusiveness (hence, a non-binary measurement) to the robot being in different areas on a map. Concretely, we may first define an ordinal scale, ranging from 0 to 2, for intrusiveness, where 0, 1, and 2 mean *"not intrusive"*, *"somewhat intrusive"*, and *"very intrusive"*, respectively. Then, we may define a measurement that assigns the intrusiveness level between 0 and 1 to situations in which the robot is in public or shared spaces, and between 1 to 2 when the robot is in private spaces. In this example, the intrusiveness scale and measurement are defined for a particular problem instance. Other problem instances of the same domain (i.e., mobile robots with consideration of intrusiveness) may use different scales and measurements for intrusiveness.

Since the absolute numerical values used to represent abstract properties such as intrusiveness in one problem instance do not have general meanings, unlike for standard measurements, we do not wish to communicate those values to the readers. Instead, I conjecture that it is sufficient to describe values of QAs with non-standard measurements *qualitatively* to the readers. For instance, continuing with the mobile robot example, we can use the terms *"not intrusive"*, *"somewhat intrusive"*, and *"very intrusive"* to describe the intrusiveness levels in an explanation (instead of using the numerical values 0 to 2). It is straightforward to select qualitative terms to describe non-standard QAs when we describe the properties of individual states or individual state-action pairs. However, the challenge lies in describing such properties of an entire policy, since there are different ways to aggregate non-standard measurement values of the state-action constituents to form a single QA value of that policy. Specifically, it depends on how we want to characterize the QA when considering temporality. There are at least two possibilities. The first is, the QA value of an entire policy depends on how often each of the state-action characteristics appears throughout the policy. Continuing with the mobile robot example, we may consider the robot being more intrusive the more often it drives into the private spaces. In contrast, the second possibility is that the QA value of an entire policy depends solely on whether each of the state-action characteristics appears in the policy, regardless of how often it does. For instance, we may consider the robot being equally intrusive whether it enters the private spaces once or 10 times. Given such difference

in temporal consideration, and the fact that there are no standard measurements for some abstract QAs, it is important to be explicit about how non-standard QAs are characterized *for an entire policy* when giving an explanation.

For the two kinds of temporal consideration for non-standard QAs described previously, I propose to use two kinds of *summary functions* for describing the aggregate of non-standard QA measurement values of all state-action pairs in a policy. These formal representations enable us to explain – from planning problem definitions – on how temporality plays role in characterizing non-standard QAs of a policy.

Number of occurrences of each value range If the number of times a policy exhibits each characteristic of a particular QA (e.g., *"not intrusive"*, *"somewhat intrusive"*, and *"very intrusive"* values of intrusiveness) is relevant in the planning problem, then for each QA value range, we use a summary function that calculates the expected number of occurrences of that value range throughout a policy.

In this case, when describing the QA of some policy in an explanation, we essentially describe the distribution of the different qualitative characteristics of that QA throughout the policy.

Probability of each value range If the planning problem is concerned only about whether a policy exhibits each characteristic of a particular QA at least once, regardless of the actual number of occurrences, then for each QA value range, we use a summary function that calculates the probability of that value range occurring in a policy.

In this case, when describing the QA of some policy in an explanation, we describe the probability that the policy exhibits each of the qualitative characteristics of that QA at least once.

There are also *non-standard, binary measurements* of QAs. Such measurements are categorized into two groups based on temporal consideration in the same way non-standard, non-binary measurements are. These are: number of occurrences of event, and probability of event.

Number of occurrences of event These quality attributes are quantified as expected numbers of occurrences of events of interest, where the objectives are to minimize and maximize occurrences of "bad" and "good" events, respectively. Some examples of these quality attributes are: safety of a mobile robot as measured by the number of collisions, availability of a web server as measured by the number of dropped requests, and surveillance success of a drone as measured by the number of detected targets.

An event e is defined as a predicate of states, actions, or a combination of them (more specifically, a predicate of state variables, action instances, action properties, or their combination). Currently, events that have durations (i.e., events that span across multiple transitions) are excluded from the definition. A quality attribute function $\mathcal{Q}_i$ of the number of occurrences of event e is:

$$\mathcal{Q}_i(s,a) = \begin{cases} 1 & \text{if the predicate representing } e \text{ is true given } s \text{ and } a \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the expected number of occurrences of event e of a policy π starting at an initial state s_0 is calculated by $V_i(\pi, s_0)$ as in Equation 1.

Probability of event Instead of the number of occurrences of an event of interest, we may be concerned about the probability of the event occurring at least once in a policy. This also applies to terminating events, i.e., success and failure of a mission, which occurs only once. Some examples of these quality attributes are: probability of a robot's collision, probability of a drone getting shot down (failure of a mission), and probability of rescuing victims in search-and-rescue (success of a mission).

To model probability of an event *e* as a QA, we must ensure that:

- *e* has a corresponding boolean state variable that serves as a "memory", recording whether the event has occurred so far. Once the event first occurs, such "memory" state variable must remain the same value, and
- every reachable state can reach an absorbing state, and for every absorbing state s_A, a transition into s_A is deterministic and has the form (s, stop, s_A), where s is an arbitrary state and *stop* is a special action designating a termination.

A quality attribute function $\mathcal{Q}_i$ of the probability of an event *e* is:

$$\mathcal{Q}_i(s,a) = \begin{cases} 1 & \text{if } a \text{ is } \mathit{stop} \text{ and the "memory" state variable corresp. to } e \text{ is true in } s \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the probability of event *e* of a policy π starting at an initial state s_0 is calculated by $V_i(\pi, s_0)$ as in Equation 1.

So far I have discussed how we can formalize and explicitly represent quality attributes and their evaluation methods in a planning problem definition, which allow us to describe *objective* properties of a policy in an explanation. In the next section, I will discuss how we can represent utility models of quality attributes, which express a *subjective* value judgment of the quality attribute properties.

5.1.3 Utility Models

When different quality-optimization objectives of the same planning problem are conflicting, preferences and value tradeoffs among the different quality attributes must be used to reconcile the tension. To this end, utility models can be used to represent *a priori* preference information, and single-objective optimization techniques can be applied to solve multi-objective planning. Many of the existing approaches for formulating multi-objective planning problems employ utility models. However, these approaches encode utility models implicitly as rewards in planning problem definitions (explainability shortcoming #3). That is, there is no clear relation between values describing how desirable the characteristics of policies are (i.e., utility values) and values describing the nature of the policies themselves (i.e., quality attribute values). The novelty of my proposed explainable planning language is that utility models of quality attributes can be represented explicitly, providing traceability from utility values of policies to the determining quality attribute properties. Utility models of an explainable planing problem consist of: (i) a single-attribute utility function for each of the quality attributes, and (ii) an aggregate (multi-attribute) utility function for all of

the quality attributes. These models allow us to explain preferences of quality attributes, and value tradeoffs made among conflicting quality objectives in determining a single optimal solution policy for a given planning problem.

Formally, to represent the utility models of a planning problem, we first define a single-attribute utility function $\mathcal{U}_i : \mathit{Range}(\mathcal{Q}_i) \rightarrow [0,1]$ for each quality attribute i. This function defines how desirable the different values of the QA i are relative to one another. Then, we define an *aggregate utility function* of state-action pairs $\mathcal{U} : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$, which is a linear combination of all composites of single-attribute utility functions $\mathcal{U}_i$ and their corresponding QA functions $\mathcal{Q}_i$:

$$\mathcal{U}(s,a) = \sum_i w_i \, (\mathcal{U}_i \circ \mathcal{Q}_i)(s,a), \tag{3}$$

where $0 < w_i < 1$ and $\sum_i w_i = 1$. This aggregate utility function defines how desirable the different value combinations of all QAs are. The objective of planning is to find a policy that maximizes the expected cumulative aggregate utility:

$$U^{total}(\pi, s) = \begin{cases} \mathcal{U}(s, stop) & \text{if } s \text{ is an absorbing state} \\ \mathcal{U}(s, \pi(s)) \; + \; \sum_{s'} \mathcal{P}r(s' \mid s, \pi(s)) \, U^{total}(\pi, s') & \text{otherwise,} \end{cases} \tag{4}$$

of an initial state s_0.

Note that since solving for optimal policies in MDPs only optimizes for the expected cumulative values, we must ensure that, for each quality attribute i – especially an instantaneous quality attribute, the sum of values of its corresponding utility function $\mathcal{U}_i$ of all steps throughout a policy can be used to rank policies based on the decision maker's preference with respect to the quality i.

The rationale for decoupling utility functions $\mathcal{U}_i$ from quality attribute functions $\mathcal{Q}_i$ is that we can separate the measurements of the *objective* characteristics of a policy from the measurements of the *subjective* preference over those characteristics. This is useful for clearly separating factual presentation and value-judgment presentation in a justification. It allows the readers to determine whether they agree with the value tradeoffs employed in determining an optimal policy.

In this thesis, I propose to create a new planning language that has the constructs for representing states and actions as discussed in Section 5.1.1, QA evaluation models as discussed in Section 5.1.2, and utility models as discussed in Section 5.1.3. Planning problems specified in this proposed language can be translated to ones in the regular Markov decision process (MDP) representation. I claim that, given a planning problem specified in the proposed language, we can automatically generate a *what-explanation* of a solution policy, as well as a *why-explanation* for selecting the policy on the basis of the QAs of concern.

5.2 Translation To Standard Planning Representation

Planning problems specified in the proposed explainable planning language discussed in Section 5.1 can be translated to ones specified in the regular MDP representation, and by extension, other languages that have MDP semantics (e.g., PDDL version 1.0 or higher, and PRISM language). This means we can use the existing algorithms and tools that can solve for optimal policies of

MDPs to solve *explainable* planning problems. This allows for the explainabiliy property to be added to autonomous systems with loose coupling with the planners or decision engines used by those systems. Likewise, policies generated by the planners in the regular representation can be translated into the explainable representation.

5.2.1 Explainable Representation to Standard Representation

To translate a planning problem specified in the explainable representation into the regular MDP representation is straightforward. The states, actions, and probabilistic transitions of the explainable representation correspond directly to those in the regular MDP representation. However, since actions in the regular MDP representation are atomic, while actions in the explainable representation have types, parameters, and properties, the structure of explainable actions must be recoverable from regular actions – to allow policies in the regular representation to be translated to explainable policies (Section 5.2.2). To this end, I use a naming scheme to preserve the structure of explainable actions. That is, action instances in the explainable representation are converted to action strings following the naming scheme. Furthermore, the quality-attribute functions, the single-attribute utility functions, and the weight constants of the aggregate utility function are composed to form the reward function in the regular MDP representation, following Equation 3. Such a translated regular MDP can be solved by existing planners.

5.2.2 Standard Policy Representation to Explainable Policy

To translate a policy generated by an existing planner to an explainable policy is also straightforward. The states of the policy already correspond directly to the states in the explainable representation. The action strings of the policy can have their explainable action structures recovered via pattern matching, following the naming scheme (Section 5.2.1). Such a translated explainable policy is used to generate *what-explanation*. However, note that generating *why-explanation* does not need any translation, since all necessary constructs are already in the explainable problem representation.

In this work, I provide a method for automatically translating planning problems specified in the proposed explainable planning language into the regular MDP representation, and for translating policies in the regular representation to explainable policies.

5.3 Policy Summarization

In this section, I discuss an approach to generate a natural-language description of a policy (i.e., *what-explanation*). As a simple approach, we may describe a policy π as follows. Starting at an initial state s_0, we describe a sequence of actions in π up to either an absorbing state, or a point where there is uncertainty in the transition from a state s via an action a, inclusively. For the latter case, we create a conditional branching in the description, where each branch describes the values of the state variables that differ across all possible resulting states, i.e., the complement of the largest subset of state variables whose values are the same across all possible resulting states. For each resulting state s', we recursively describe the policy π starting at s'.

While the simple method described above provides the complete description of a policy, it can potentially be too complex, and therefore incomprehensible. This is due to multi-step state evolution and nondeterministic transitions in a probabilistic policy, especially for policies that are synthesized. Summarization of a policy is needed to generate a comprehensible *what-explanation*. To this end, a verbal summary of a policy must be concise as well as sufficiently complete for allowing the explainee to understand the justification provided for why the policy is selected as a best solution (Section 5.4). More specifically, since a justification discusses the quality-attribute properties of a solution policy and alternative policies, a description of the solution policy and each alternative should include the characteristics of the policies that are: (i) determining factors of the quality-attribute properties, and (ii) different across the solution policy and its alternatives. I plan to further investigate this problem of policy summarization in the remaining work of the thesis.

5.4 Justification of Solution Policy

In this section, I discuss the proposed approach to generate justifications for solution policies of multi-objective probabilistic planning. As mentioned in Section 2.1, I propose to use *contrastive quality-attribute-based justification* to provide *why-explanation*. The structure of such justification for a solution policy is the following. First, the quality-attribute properties of the solution policy are described. Next, an alternative policy and its quality-attribute properties are described, with an emphasis on the differences from the originally selected policy. Then, an argument against the alternative policy is given, indicating the particular differences in quality-attribute properties that make the alternative inferior to the selected policy. Essentially, the argument indicates the preferences and value tradeoffs employed in determining an optimal policy. The last two steps repeat for all relevant alternative policies.

The rationale for adopting the kind of justification described previously is that it reflects both (part of) the mathematical model of the planning as well as how people intuitively provide contrastive explanations. More specifically, the utility models of a planning problem can be reflected in an *explicit* comparison and indication of preference between the quality-attribute values of a selected solution policy and those of an alternative policy. Regarding how people provide and expect explanations, several work in cognitive and social psychology, as well as philosophy, argue that *contrastive explanation* is what people seek for when asking why-questions [18, 19]. A contrastive explanation explains the cause of an event relative to some other event that did not occur. In the context of justifying a planning solution, an "event" is a selection of a particular solution policy, and "some other event that did not occur" is a selection of some alternative policy. The literature argues that it is important that the explainee understands the *counterfactual case* [18, 19], which refers to the hypothetical outcome that did not occur. Given these insights from psychology and philosophy, I therefore propose to use contrastive explanation to provide *why-explanation* of planning, on the basis of quality attributes.

In this section, I discuss the proposed approach to generate a contrastive, quality-attribute-based justification. Section 5.4.1 describes how to generate a description of quality-attribute properties of a solution policy. Section 5.4.2 discusses how we can determine relevant alternative policies with which to contrast the solution policy. Lastly, Section 5.4.3 describes how to generate a statement of rejection of an alternative policy.

5.4.1 Describing quality-attribute properties of solution policy

As discussed previously, the first part of a justification is a description of the quality-attribute properties of a solution policy. This is essential to the explanation as it provides *objective and quantitative* evaluation of the solution policy along the various quality dimensions of concern. Each quality-attribute property is described by the name of the quality attribute, the measurement value, and the unit of measurement. The specific meanings of measurement values and units depend on the kinds of quality attributes. Such description can be generated using the QA evaluation model constructs of the proposed explainable planning language (Section 5.1.2).

Standard measurement Recall that there are two kinds of QAs that have standard measurements. I propose to describe them in the following ways:

Cumulative Quality Attributes To describe a cumulative QA property i of a policy π and an initial state s_0, we first calculate the expected cumulative value of the QA, $V_i(\pi, s_0)$, using the formula in Equation 1 and the QA function $\mathcal{Q}_i$. Then, we generate a verbal description in the following form:

> *"The expected total* ⟨`QA name`⟩ *is* ⟨`QA value`⟩⟨`QA measurement unit`⟩",

where the QA name and measurement unit are provided in the planning problem vocabulary, and the QA value is $V_i(\pi, s_0)$.

Instantaneous Quality Attributes The challenge in describing properties of an instantaneous QA is that there are multiple values from all steps throughout a policy that quantify the particular concern of the policy. Describing all individual values would be too verbose; instead, we would like to describe a summary of those values. One approach is to use a *summary function* as discussed in Section 5.1.2, which must satisfy the condition that, for any two distinct pairs of policy and initial state: $\langle \pi, s_0 \rangle$ and $\langle \pi', s_0' \rangle$, a summary value $\mathcal{G}_i(\pi, s_0)$ is more *preferable* than another summary value $\mathcal{G}_i(\pi', s_0')$ if and only if the total single-attribute utility value $U_i^{total}(\pi, s_0)$ is greater than $U_i^{total}(\pi', s_0')$. Given a summary function, we can generate a description of an instantaneous QA property in a similar way we do for a cumulative QA property. A verbal description is in the following form:

> *"The* ⟨`summary function name`⟩ *of* ⟨`QA name`⟩ *is* ⟨`summary value`⟩ ⟨`summary value unit`⟩",

where the summary function name and measurement unit are provided in the planning problem vocabulary, and the summary value is $\mathcal{G}_i(\pi, s_0)$.

Additionally, we can describe an instantaneous QA property at a single point in time – without using a summary function. That is, we describe an expected value of an instantaneous QA at some step in a policy. This may be useful when we need to generate an elaborated explanation. We can generate a verbal description in the following form:

"The expected ⟨`QA name`⟩ *at step* ⟨n⟩ *in the policy is* ⟨`QA value`⟩ ⟨`QA measurement unit`⟩",

where n is a step in the policy, and the QA value is $V_i(\pi, s_0, n)$ in Equation 2.

Non-standard, non-binary measurement As discussed in Section 5.1.2, an arbitrary measurement can be used to characterize states, actions, and events with respect to a particular property of concern that does not have an associated standard measurement. I also argue that values of such non-standard, non-binary QA properties can be described qualitatively. Recall that there are two kinds of temporal consideration for non-standard QAs. I propose to describe them in the following ways:

Number of occurrences of each value range For a non-standard QA with this kind of temporal consideration, we can report the expected number of times a policy exhibits each characteristic of the QA. For instance, recalling the mobile robot example from Section 5.1.2, we can describe the intrusiveness property of the robot's policy as follows.

> *"The robot is expected to be not intrusive for 5 steps, somewhat intrusive for 3 steps, and very intrusive for 2 steps in the policy."*

To generate such a description, we need to calculate the expected number of times a state-action pair in a policy exhibits each characteristic of the QA. We can achieve this by defining a predicate over states and actions to represent each characteristic of the QA, and defining a value function that assigns 1 to state-action pairs in which the predicate is evaluated to true, and assigns 0 otherwise. The expected cumulative value of such function over a policy is the expected number of times the policy exhibits the corresponding QA characteristic.

Probability of each value range For a non-standard QA with this kind of temporal consideration, we can report the probability of a policy exhibits each characteristic of the QA at least once. For instance, we can describe the intrusiveness property of the robot's policy as follows.

> *"The robot will eventually be not intrusive, has 0.8 probability to eventually be somewhat intrusive, and has 0.1 probability to eventually be very intrusive."*

To generate such a description, we need to calculate the probability that a policy will eventually exhibit each characteristic of the QA. We can achieve this by defining a predicate over states and actions to represent each characteristic of the QA, and using probabilistic model checking for reachability property. Specifically, we first use the policy to resolve nondeterminism in the Markov decision process to obtain a Markov chain. Then, for each characteristic of the QA, we use *numerical query* of the form $P_{=?}[F\, b]$ to query the Markov chain, where $F\, b$ is a path formula stating that b is eventually true. We define b to be true if and only if the Markov chain exhibits the particular QA characteristic. Since $F\, b$ is a path formula, b must be a proposition labeling a state; however, a QA characteristic is defined for a state-action pair, not only for a state. Therefore, to use a state-labeling proposition b to capture the QA characteristic, we need to augment each state

in the Markov chain to contain information about the action to which the state is mapped in the policy. Once we obtain the augmented Markov chain, we can perform a numerical query $P_{=?}[F\, b]$ on it to compute the probability of the QA characteristic represented by b occurring eventually.

Number of occurrences of event To describe this kind of QA of a policy, we simply calculate the expected number of occurrences of the event of interest using the approach discussed in Section 5.1.2. Then, we can generate a description in the following form:

> *"The expected number of times* ⟨`event name`⟩ *occurs is* ⟨n⟩",

where the event name is provided in the planning problem vocabulary, and n is the expected counts of the event.

Probability of event To describe this kind of QA of a policy, we simply calculate the probability of the event of interest *eventually* occurring, using the approach discussed in Section 5.1.2. Then, we can generate a description in the following form:

> *"The probability of* ⟨`event name`⟩ *occurs is* ⟨p⟩",

where the event name is provided in the planning problem vocabulary, and p is the probability of the event.

5.4.2 Determining relevant alternative polices

The second part of a justification is an argument of why the selected solution policy is the best solution, with respect to the quality attributes of concern. Since the planning problem has multiple optimization objectives, describing the quality-attribute properties of the selected policy alone is insufficient as a justification because:

- Often, there is no single solution policy that dominates in all quality attributes. In such case, an argument for why any solution policy is selected as a "best" solution must explain the preferences and value tradeoffs among the quality attributes made in determining a single optimal policy.
- In a case where there is a solution policy which dominates in all quality attributes, an explanation should indicate that the policy *objectively* has the most desirable properties for all quality attributes.

One approach to explain preferences and value tradeoffs among quality attributes is to contrast the quality-attribute properties of the selected solution policy with those of some Pareto-optimal alternative policies. The rationale for using this approach is the following. Let a policy π_1 be a Pareto-optimal solution policy, whose quality-attribute values are $q_1^1, q_2^1, \ldots, q_n^1$. By definition, there is no other policy whose all quality-attribute values are better (according to the optimization objectives of the planning problem) than those of π_1. Let π_2 be another Pareto-optimal solution policy, whose quality-attribute values are $q_1^2, q_2^2, \ldots, q_n^2$. Suppose that the policy π_1 is better than,

worse than, and equivalent to π_2 with respect to the quality attributes $1 \ldots j$, $j+1 \ldots k$, and $k+1 \ldots n$, respectively. By comparing the quality-attribute values of π_1 and π_2, we can show that there are tradeoffs between the quality attributes $1 \ldots j$ and $j+1 \ldots k$, and what the corresponding amounts are. Furthermore, suppose we indicate that π_1 is preferred to pi_2. We can then show that we accept certain tradeoffs, but not the others. Particularly, we accept the losses $\mid q^1_{j+1} - q^2_{j+1} \mid, \ldots, \mid q^1_k - q^2_k \mid$ for the gains $\mid q^1_1 - q^2_1 \mid, \ldots, \mid q^1_j - q^2_j \mid$. Such information about the preferred loss-gain can help explain the preferences among the quality attributes.

Therefore, to obtain alternative policies to be discussed and contrasted with the selected solution policy in a justification, I propose a method to sample alternative policies that lie, on the Pareto surface on the n-dimensional space of n quality attributes, nearby the selected solution policy (which is also on the Pareto surface). The key idea of the proposed method is the following. Starting from the quality-attribute values of the selected solution policy π^*: $q^{\pi^*}_1, \ldots, q^{\pi^*}_n$. For each quality attribute i, we determine a new value q'_i which is more preferable than the value $q^{\pi^*}_i$. Then, we construct a new planning problem with 1 less optimization objective (excluding that of the quality attribute i), and with the constraint that the value of the quality attribute i must be q'_i or better (according to the optimization objective). Next, we find an optimal, constraint-satisfying solution policy for the new planning problem. This policy is an alternative to the selected solution policy π^* for the original planning problem. Using this method, we can obtain up to n alternative policies; each one improves at least 1 quality attribute but compromises at least 1 other quality attribute.

Algorithm 1 outlines the alternative policies generation method discussed above. For the purpose of this algorithm, we defined the expected cumulative utility function for each quality attribute i of a policy π, starting from an initial state s, as:

$$U_i^{total}(\pi, s) = \begin{cases} (\mathcal{U}_i \circ \mathcal{Q}_i)(s, stop) & \text{if } s \text{ is an absorbing state} \\ (\mathcal{U}_i \circ \mathcal{Q}_i)(s, \pi(s)) \; + \; \sum_{s'} \mathcal{P}r(s' \mid s, \pi(s)) \, U_i^{total}(\pi, s') & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathcal{U}_i$ is a single-attribute utility function and $\mathcal{Q}_i$ is a quality-attribute function.

Algorithm 1: Generate alternative policies

input : π^*: selected policy
s_0: initial state
$\mathcal{U}_1, \ldots, \mathcal{U}_n$: single-attribute utility functions of all n QAs
$w_1, \ldots, w_n$: weight constants for all single-attribute utility functions
$\Delta u_1, \ldots, \Delta u_n$: increment sizes of utilities
$\theta_1, \ldots, \theta_n$: maximum utilities
output: A set of alternative policies Π'

$\Pi' \leftarrow \varnothing$;
$D \leftarrow \{1, \ldots, n\}$ // Quality dimensions to be explored;
while $D \neq \varnothing$ **do**
  $i \leftarrow$ remove an element from D;
  $u'_i \leftarrow U_i^{total}(\pi^*, s_0)$;
  $w'_j \leftarrow$ new weight constant for all $j \neq i$, where the ratio among the new constants w'_j is equal to the ratio among the original constants w_j;
  $U'^{total}(\pi, s_0) \leftarrow$ expected cumulative aggregate utility function of all QAs $j \neq i$, with the new weight constants w'_j;
  while $u'_i \leq \theta_i$ **do**
    $u'_i \leftarrow u'_i + \Delta u_i$;
    $\pi' \leftarrow \underset{\pi}{\operatorname{argmax}}\, U'^{total}(\pi, s_0)$, subject to $U_i^{total}(\pi, s_0) \geq u'_i$;
    if π' *exists* **then**
      $\Pi' \leftarrow \Pi' \cup \{\pi'\}$;
      for $j \neq i$ **do**
        if $U_j^{total}(\pi', s_0) \geq U_j^{total}(\pi^*, s_0) + \Delta u_j$ **then**
          // Disregard quality dimensions that were sufficiently improved in this iteration in future exploration
          $D \leftarrow D - \{j\}$
        end
      end
      break;
    end
  end
end

The method of generating alternative policies presented in Algorithm 1 is only a base approach. It produces alternative policies which can be used to explain preferences and value tradeoffs among quality attributes. However, there are two challenges that this base approach does not address.

- First, the explainee may not find all alternative policies produced by the base approach relevant. For instance, the explainee may find that a presented alternative policy is too similar to the selected solution policy in terms of their quality-attribute properties. Therefore, the

explainee may not perceive the discussion of this particular alternative useful, as they would be indifferent between the two policies.

- Second, the base approach may not produce alternative policies which the explainee would find relevant. For instance, some combination of loss-gain of quality attributes between the selected solution policy and some alternative policy may best convince the explainee that the selected policy is superior to the alternative (or vice versa). Nonetheless, that particular combination of loss-gain of quality attributes is not explored by the base approach.

I plan to extend the base approach presented in Algorithm 1 to address these challenges of determining relevant alternative policies in the remaining work of this thesis.

5.4.3 Rejecting alternative policies

Given a set of Pareto-optimal alternative policies generated by Algorithm 1 or its variant, we can generate natural language statements, for each of the alternative policies, contrasting the selected solution policy with the alternative policy on the basis of the quality attributes, and rejecting the alternative. To this end, we can generate the statements of the following form:

> *"I could* [(1) vp. improve these quality dimensions to these values]*, but at the expense of* [(2) np. worsening these other quality dimensions to these values]*, by* [(3) vp. carrying out this alternative policy] *instead. However, I decided not to do that because* [(4) np. the improvement in these quality dimensions] *is not worth* [(5) np. the deterioration in these other quality dimensions].

The statement fragments (1) and (2) contrast the quality-attribute values of the selected solution policy with those of the alternative policy, by describing the gains and the losses. The statement fragment (3) describes the alternative policy (to be discussed in Section 5.3). Finally, the statement fragments (4) and (5) restate the gains and the losses but in a qualitative way – as part of justifying the rejection of the alternative policy.

Recall the mobile robot example. Below is an example explanation of the robot's behavior, with the justification for the robot's selected solution policy highlighted in bold:

> *"I'm planning to go through the path* l1-l4-l3-l5 *to get to location* l5*. This should take 38 seconds and it should have 0.2 probability of collision.* ***I could decrease time to 33 seconds, but at the expense of increasing probability of collision to 0.7, by going through the path*** l1-l2-l3-l5 ***instead. However, I decided not to do that because the decrease in time is not worth the increase in probability of collision."***

Since we have the constructs for representing quality-attribute analytic models in the proposed explainable planning language (Section 5.1.2), generating the statement fragments (1), (2), (4), and (5) is essentially similar to generating a description of the quality-attribute properties of a solution policy (Section 5.4.1). The key difference is that the former method describes only the quality-attribute properties of the alternative policy which differ *significantly* from those of the selected solution policy. To this end, one approach for determining which quality-attribute properties are

significantly different between the selected and the alternative policies is to employ the single-attribute utility functions of the quality attributes (see Equation 5). If the expected cumulative utility values of QA *i* of the two policies differ more than some threshold, then QA *i* is a significant difference. However, the key challenge is determining the appropriate utility-difference threshold for each quality attribute. I plan to address this challenge in the remaining work of this thesis.

5.5 Explaining Quality-Attribute Evaluation

In Section 5.4, I have discussed how to generate a contrastive, quality-attribute-based justification for a solution policy, which involves describing the quality-attribute properties of the solution policy and its alternatives. Given such justification, the explainee may understand what properties the solution policy has and what value tradeoffs it entails. However, as I have shown so far, the claims made about the quality-attribute properties of a policy in the justification are unsupported – they are presented merely as facts. The explainee may not take the claims for granted, but they may wish to understand the factors that determine the quality-attribute properties of a policy. Therefore, a justification should be accompanied by some supporting evidence for the quality-attribute claims it makes – unless the claims are self-evident. Such supporting evidence must reflect the quality-attribute evaluation models employed in the planning.

In this section, I discuss my plan to investigate an approach for, and address the challenges of, explaining why a policy has certain quality-attribute properties. Particularly, the goal is to explain each quality-attribute property by describing the *input* to the corresponding QA evaluation model, and relating it to the *output* of the model (i.e., the calculated QA value). The preliminary main idea for such approach is that it can leverage the language constructs representing QA evaluation models in the proposed explainable planning language (Section 5.1.2). Using these representations of QA evaluation models and probabilistic model checking, we have the means to query information about the QA-determining factors of a given policy. Such information can then be presented in addition to any description of the quality-attribute properties of a policy (e.g., either that of a selected solution policy or an alternative policy). For instance, consider the mobile robot example, whose one of the quality attributes is total execution time. The evaluation model of execution time (of a single step in a policy) takes traveling distance and driving speed as input. An explanation for the execution-time property of the robot's policy is shown in the parentheses:

> *"...the expected total execution time is 5 minutes (the expected total distance is 15 meters, and the average speed is 3 m/s)..."*

Since a QA evaluation model has clearly defined input parameters, we can trivially identify the factors that determine the quality-attribute property of a policy. However, recall that these factors are represented by state variables, action types, and action properties, which means individual building blocks of a policy (i.e., state-action pairs) can have different values of these factors. Therefore, the key challenge of describing the factors determining the quality-attribute properties of a policy lies in summarizing the values of these factors throughout the policy. Revisiting the mobile robot example, at each step in its policy, the robot may be traveling for a different distance and at a different speed. The example explanation for the execution time of the robot's policy

given above *aggregates* the traveling distances and the driving speed settings from all steps in the policy into an *expected total* distance and an *average* speed, respectively. A general approach for summarizing the state-action characteristics which determine the quality-attribute properties of a policy is needed. I plan to investigate such approach in the remaining work of this thesis.

6 Preliminary Work

This section describes my research towards the completion of this thesis.

6.1 Formal Representations for Explainable Planning

I have designed the formal representations of the proposed explainable planning language discussed in Section 5.1. I have implemented a framework for defining planning problems using the explainable planning representations in Java. I have also implemented a translation from an explainable planning problem definition (in Java) into the regular MDP representation. In this work, I use the probabilistic model checking tool, PRISM, for solving for optimal policies of MDPs. Therefore, the previously mentioned translation is implemented for the PRISM modeling language. Although, the explainable planning representations can be translated into any regular MDP modeling language of choice. Furthermore, output policies of the PRISM tool (specified in the PRISM output format) are translated into the proposed explainable representation of policies.

This preliminary work on the formal representations of the proposed explainable planning language enables the investigation of explanation generation. Specified in the proposed explainable representations, planning problems and their corresponding solution policies (solved by some planners) are the key input to the explanation generation algorithm. Nonetheless, the explainable planning representations written in Java are only meant to be the intermediate representations. To make specifying planning problems more convenient, I plan to develop a new syntax for the proposed explainable planning language, which can be translated into the intermediate explainable representations. These intermediate representations can then be translated into a regular MDP representation – to be used in the explanation generation algorithm. Defining a full syntax for the proposed explainable planning language and implementing its corresponding translation are part of the remaining work of this thesis.

6.2 Numerical Query of Quality-Attribute Properties with Probabilistic Model Checking

I have implemented the proposed approach for evaluating and describing the quality-attribute properties of a policy. I use a probabilistic model checking method to perform numerical queries to determine values of the quality attributes of a policy. More specifically, given an explainable planning problem definition and its solution policy, my approach performs the following. First, it translates the explainable planning problem into the regular MDP representation, in the PRISM language (see Section 5.2). Then, it encodes each quality-attribute evaluation model of the planning problem as a reward structure in the translated MDP. Next, the approach uses the given solution policy

to resolve nondeterminism in action choices of the MDP, to obtain a discrete-time Markov chain (DTMC) with rewards. For each of the quality attributes, the approach performs numerical query for its value on the DTMC. This numerical query is supported by PRISM. Essentially, it is done by using probabilistic model checking of reward-based properties corresponding to the encoded quality-attribute evaluation models. In the case of instantaneous quality attributes, whose values must be summarized (see Section 5.1.2), the approach evaluates the corresponding summary functions in addition to performing numerical queries of instantaneous rewards at all steps in the policy.

The method for generating a verbal description of the quality-attribute properties of a policy has been implemented. It uses a simple template-based approach as discussed in Section 5.4.1, using the vocabulary provided in a planning problem definition. Such a verbal description of quality-attribute properties is provided as part of a justification for a solution policy.

6.3 Algorithm for Contrastive Quality-Attribute-Based Justification

As a preliminary work on generating contrastive quality-attribute-based justification, I have implemented the base approach for generating alternative policies, discussed in Section 5.4.2. The implementation uses multi-objective strategy synthesis for MDPs supported by the PRISM tool. In particular, the tool provides the capability to find a solution policy of an MDP that maximizes the expected cumulative value of some reward function, as well as satisfies the given bounds on the values of other reward functions. The implementation of the base approach uses this capability to generate Pareto-optimal alternative policies, as outlined in Algorithm 1. Particularly, the single-attribute utility functions of n individual quality attributes and the aggregate utility functions are encoded as multiple reward structures of an MDP. Each alternative policy is generated by maximizing the expected cumulative reward corresponding to an aggregate utility of $n - 1$ quality attributes, while satisfying a constraint on the reward corresponding to the single-attribute utility of the remaining quality attribute.

I have also implemented the proposed method for generating a justification discussing and contrasting the alternative policies with the selected solution policy on the basis of quality attributes, as described in Section 5.4.3. However, as discussed in Section 5.4.2, there are challenges regarding relevance of the generated alternative policies, which the base approach does not address. Thus, in this thesis, I plan to investigate new approaches for determining *relevant* alternative policies. Furthermore, my preliminary work so far has used only a simple approach for generating *what-explanation* of a policy; that is, describing the full policy (see Section 5.3). The remaining work of my thesis will investigate an approach for policy summarization.

7 Validation

In this section, I discuss my validation plan for the thesis. Recall that my thesis statement claims that we can improve transparency of autonomous systems' reasoning, and allow humans to calibrate their confidence in the systems' decisions. I plan to validate this claim in two parts. First, I will demonstrate that explanations generated by my proposed approach improve the explainees' understanding of how autonomous systems make decisions. Second, I will demonstrate that the

generated explanations enable the explainees to recognize whether the systems have made good decisions.

7.1 Improve transparency of autonomous systems' reasoning

To validate the claim that my proposed approach can improve humans' understanding of how an autonomous system reason (i.e., improve transparency), I plan to demonstrate whether, given a generated explanation, the explainees can correctly answer specific questions about decisions made by the system. These questions will assess the explainees' understanding of: (i) the implications of the system's decisions on the quality attributes of concern, (ii) the rationale for the system's decisions to take a particular action or sequence of actions, and (iii) how the system assesses the quality attributes in its decision making.

I plan to conduct the following human-subject experimental study for this validation. I will recruit participants for the study, and randomly assign them to the control group and the experimental group. Each participant will be given descriptions of several missions of an autonomous system, in the form of high-level (informal) goals and quality optimization objectives. For each mission, the participants will observe an execution of the autonomous system to complete the mission. Then, the participants in both groups are given a set of questions about the decisions made by the autonomous system to achieve its mission, as discussed previously. Both groups must answer the questions, as well as explain how they derived their answers. The control group will only be allowed to investigate the planning problem specification and the log trace of the system's execution of each mission – as if they were a developer of the system. A log trace provides primitive information about the system's execution trace, including state variables, action names, etc. A planning problem specification, which will be given to the participants will be an abstract, more human-readable version of the actual planning specification. Meanwhile, the experimental group will receive an explanation generated by the proposed approach. The correctness of all participants' answers to the questions, how they derived their answers, and the amount of time the participants take to determine the correct answers will be considered in the analyses of this study.

The hypotheses of this experiment are: (i) the participants in the experimental group provide a larger number of correct answers than those in the control group, where an answer can be considered correct only if the participant did not take an unsound approach to derive the answer or make a random guess, and (ii) the participants in the experimental group take less time to determine correct answers than those in the control groups. The results of this experiment will demonstrate whether explanations generated by my proposed approach can help humans gain more understanding of how autonomous systems make decisions, in a more efficient way compared to a baseline approach of examining (real or simulated) execution logs and investigating planning specifications and models.

7.2 Calibrate humans' confidence in autonomous systems' decisions

To validate the claim that my proposed approach can allow humans to calibrate their confidence in the decisions made by autonomous systems, I plan to demonstrate whether, given a generated explanation, the explainees can recognize when the autonomous systems make good decisions

and when they make bad decisions. Unlike the first validation method, which only concerns with whether the explainees understand the reasoning mechanism of the systems, this validation method concerns with whether the gained understanding is useful for the explainees to determine whether following the systems' decisions will likely lead to desirable outcomes. In particular, the systems may make decisions that disagree with the humans' preferences on the quality attributes. Or the systems may make incorrect assessment of some quality-attribute properties, potentially leading to a suboptimal solution policy being chosen. In those cases, the humans should have *low confidence* in the systems' decisions, and they should be able to justify why they think the systems' decisions are bad.

I plan to conduct the following human-subject experimental study for this validation. I will recruit participants for the study, and randomly assign them to the control group and the experimental group. Similar to the first experimental study, all participants will be given a high-level description of a mission of an autonomous system. However, there are two key differences. First, instead of observing an execution, the participants will be given a description of a policy of the system. Second, the participants will be given: (i) a *human-understandable* preference model of the quality attributes of the system's mission, and (ii) information about the factors that determine the quality-attribute properties, and the exact values of those factors in the mission context. I refer to these information as the "ground truth". That is, the preference model and the values of the quality-attribute determining factors given to the participants are considered the true values, which may differ from the ones used by the system.

The experimental setting is as follows. For each mission scenario, each participant is randomly given (a description of) a policy of the system. A policy is either generated from the planning problem whose models are all consistent with the "ground truth", or from one whose models differ from it. In other words, some policies given to the participants are generated from planning problems whose: (a) utility models do not represent the *true* preference of the quality attributes, or (b) whose predictions of the hidden or future values of the quality-attribute determining factors are incorrect. The task for the participants is to determine whether their given policies would likely yield desirable outcomes, given what they know about the mission, the true values of the quality-attribute determining factors, and the true preference model. All participants must justify their answers and explain how they derive their answers. The control group will only have access to the planning problem specifications *corresponding to their given policies* – as if they were a developer of the system. Similar to the first study, the provided planning problem specifications are abstracted, more human-readable versions of the actual planning specifications. Meanwhile, the experimental group will receive an explanation generated from the proposed approach. Similar to the first experimental study, the correctness of all participants' answers, how they derived their answers, and the amount of time the participants take to determine the correct answers will be considered in the analyses of this study.

The hypotheses of this experiment are the same as those of the first experiment. In this experiment, there is only one question which the participants must answer (i.e., of whether or not a given policy is likely to yield a desirable outcome of the mission). The results of this experiment will demonstrate whether explanations generated by my proposed approach can enable humans to

calibrate their confidence in the decisions made by autonomous system, in a more efficient way compared to a baseline approach of investigating planning specifications and models.

8 Research Plan

This section describes the work plan for this thesis. Table 1 lists the tasks that make up the work plan for this thesis, including their current status, and the estimated duration in months. The total estimated time for completion is 19 months.

Task	Status	Est. Time (months)
Explainable planning language design	Mostly done	1
Algorithm for describing policy (basic)	Partially done	1
Algorithm for summarizing policy (advance)	To be done	2
Algorithm for justifying policy (basic)	Done	
Algorithm for generating relevant alternative policies (advance)	To be done	2
Algorithm for explaining QA evaluations	To be done	2
Implementation of explanation generation	Partially done	3
Case studies	To be done	2
Validation	To be done	2.5
Write dissertation	To be done	3.5

Table 1: Thesis Tasks

The design of the proposed explainable planning language was presented in this thesis proposal (Section 5.1). However, refinement of the language may be required as I continue to investigate this work. The simple algorithm for describing a policy (Section 5.3) and the algorithm for generating justification for a policy, with the base approach for generating alternative policies (Section 5.4), are implemented. These work comprises the initial end-to-end implementation of explanation generation.

The key remaining tasks in this thesis are to improve upon the preliminary algorithms for describing and justifying solution policies. Investigating algorithms for generating relevant alternative policies (for contrastive justification, Section 5.4.2) and for explaining quality-attribute evaluations (Section 5.5) are the high priority tasks and to be completed first. Investigating an algorithm for summarizing policies (Section 5.3) is expected to be the most difficult task, but not a critical part of the thesis. Furthermore, to demonstrate the applicability of the proposed approach, I plan to apply the explanation generation technique to at least three case studies from different application domains.

References

[1] Nicholas D Allen, John R Templon, Patrick Summerhays McNally, Larry Birnbaum, and Kristian J Hammond. Statsmonkey: A data-driven sports narrative writer. In *AAAI Fall*

Symposium: Computational Models of Narrative, 2010.

[2] Julien Bidot, Susanne Biundo, Tobias Heinroth, Wolfgang Minker, Florian Nothdurft, and Bernd Schattenberg. Verbal plan explanations for hybrid planning. In *MKWI*, pages 2309–2320, 2010.

[3] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of IJCAI*, 2017.

[4] Anind K Dey. Explanations in context-aware systems. In *ExaCt*, pages 84–93, 2009.

[5] Francisco Elizalde, L Enrique Sucar, Manuel Luque, J Diez, and Alberto Reyes. Policy explanation in factored Markov decision processes. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM 2008)*, pages 97–104, 2008.

[6] Francisco Elizalde, Luis Enrique Sucar, Alberto Reyes, and Pablo deBuen. An MDP approach for explanation generation. In *ExaCt*, pages 28–33, 2007.

[7] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In M. Bernardo and V. Issarny, editors, *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.

[8] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 303–312. ACM, 2017.

[9] Omar Zia Khan, Pascal Poupart, and James P Black. Minimal sufficient explanations for factored Markov Decision Processes. In *ICAPS*, 2009.

[10] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more?: the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1–10. ACM, 2012.

[11] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, pages 3–10. IEEE, 2013.

[12] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[13] M. Kwiatkowska, G. Norman, and D. Parker. Advances and challenges of probabilistic model checking. In *Proc. 48th Annual Allerton Conference on Communication, Control and Computing*, pages 1691–1698. IEEE Press, 2010.

[14] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[15] Brian Y Lim and Anind K Dey. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 195–204. ACM, 2009.

[16] Brian Y Lim and Anind K Dey. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 13–22. ACM, 2010.

[17] 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. ACM, 2009.

[18] Tania Lombrozo. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470, 2006.

[19] Tania Lombrozo. Explanation and abductive inference. *Oxford handbook of thinking and reasoning*, pages 260–276, 2012.

[20] Linus J Luotsinen, Hans Fernlund, and Ladislau Bölöni. Automatic annotation of team actions in observations of embodied agents. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 9. ACM, 2007.

[21] Gethin Norman and David Parker. Quantitative verification: Formal guarantees for timeliness, reliability and performance. Technical report, The London Mathematical Society and the Smith Institute, 2014.

[22] Vittorio Perera, Sai P Selveraj, Stephanie Rosenthal, and Manuela Veloso. Dynamic generation and refinement of robot verbalization. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 212–218. IEEE, 2016.

[23] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

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

[25] 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.

[26] Aaron St Clair and Maja Mataric. How robot verbal feedback can improve team performance in human-robot task collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 213–220. ACM, 2015.

[27] Manuela Veloso, Nicholas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford, Colin McMillen, Maayan Roth, Douglas Vail, and Stefan Zickler. A team of humanoid game commentators. *International Journal of Humanoid Robotics*, 5(03):457–480, 2008.

[28] Ameni Yengui and NEJI Mahmoud. Semantic annotation formalism of video-conferencing documents. *Cognition and Exploratory Learning in Digital Age*, 2, 2009.

[29] Håkan LS Younes and Michael L Littman. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. 2004.