Explainable Reasoning for Human-Robot Collaboration

RSMG 3: THESIS PROPOSAL

Oliver Michael Kamperis
School of Computer Science
University of Birmingham
United Kingdom
oxk312@student.bham.ac.uk
Primary Supervisor: Dr. Mohan Sridharan (90%)
Secondary Supervisor: Prof. David Parker (10%)

December 20, 2019

ABSTRACT

The objective of this thesis is to provide robots with the ability to communicate with humans during a collaborative task. To achieve this objective, this thesis will develop a novel decision making and reasoning model for robots which will enable them to provide explanatory descriptions of their beliefs, knowledge, experiences and reasoning processes, on-demand and at an appropriate abstraction level.

The development and adoption of new generations of robots such as self-driving cars and service robots has the potential to significantly improve human quality of life. However, the extent of their capabilities has lead to a distinct reluctance for humans to trust them. Robots may often do things that a human normally would not do, or behave in unexpected ways in sensitive environments that could put human safety at risk. It is thus natural to expect a robot to justify its behaviour by explaining the reasoning with which its decisions are made, and the beliefs that informed that reasoning. This ability will allow humans to better understand the behaviour of robots to provide [Sheh 2017]:

1. *Trust* - Humans tend to trust systems that they believe they understand.
2. *Accountability* - When things go wrong people will demand to know if robots are to blame.
3. *Ease of Debugging* - It is easier to fix faults when decision making models are transparent.

Existing approaches to decision making in Artificial Intelligence (AI) and robotics are predominately based on opaque black-box models whose content cannot be easily interpreted or directly explained. This problem has lead to the emergence of *eXplainable Artificial Intelligence* (XAI) [DARPA 2016]. Many approaches in this field involve the use of post-hoc approximations of Machine Learning (ML) models to provide only partial transparency of the models' input-output mapping through statistical analysis. This is insufficient for the needs of human-robot collaboration because such approaches do not provide accurate explanatory descriptions of how a robot's knowledge and experiences inform its beliefs, how these beliefs guide its reasoning and how this reasoning leads to its decisions.

This thesis takes an alternative and more promising approach towards explanation generation for human-robot collaboration which falls within the recently established field of *explainable agency* [Langley et al. 2017]. The principal behind the approach of this thesis is to build the ability to explain a robot's beliefs, knowledge and experiences and how these guide its reasoning, directly into the implementation of its reasoning model at all levels, and as a central part of its design specification.

Contents

1 Introduction **3**

2 Explainability **4**

3 Problem Statement **5**

4 Related Work **7**
4.1 Guiding Principles . . . 7
4.2 Explanation Generation Systems . . . 8
4.3 Axes of Explanation . . . 9
4.4 Mixed Logical and Probabilistic Reasoning . . . 10

5 Proposed Architecture **11**

6 Plan of Future Work **14**

7 Proposed Proof-Of-Concept Domain: Robotics in Hospitality **16**
7.1 Advantages of the Hospitality Domain . . . 16
7.2 Previous Work regarding Robotics in Hospitality . . . 16

Appendices **17**

A Action Languages **17**
A.1 Transition Diagrams . . . 17
A.2 Review of Action Languages . . . 17

B Hierarchical Task Network Planning **18**

C Answer Set Programming **19**
C.1 ASP Systems . . . 20
C.2 Grounders . . . 20
C.3 Solvers . . . 20
C.4 Combined Systems . . . 20

D Probabilistic Planning **21**
D.1 Probabilistic Policies . . . 21
D.2 Markov Decision Processes . . . 21

E Nature of Explanations **22**
E.1 Models of Explanation . . . 22
E.2 The Causal Model of Explanation . . . 22
E.3 Constrastive Explanation . . . 23
E.4 The Abnormal Conditions Model . . . 23
E.5 Goal Based Explanation . . . 23

1 Introduction

Mobile robots are becoming increasingly prevalent in human society and their development is now receiving tremendous amounts of attention from a widespread international research community. *Artificial Intelligence* (AI) has also become a vast and ubiquitous field, rich with mature technologies applicable to robotics applications [Russell and Norvig 2016]. The rapid growth of these fields seems motivated by the desire to automate complex, mundane and dangerous tasks, through which we may gain significantly in terms of freedom, liberty and quality of life. However, as robots gain in both intelligence and autonomy to become capable of performing ever more complex and important tasks, it becomes increasingly critical that they be able to interact, collaborate and converse effectively with humans [Langley et al. 2017].

Emerging generations of mobile robots, such as *service robots*, may soon be deployed to collaborate with us in our homes and workplaces. These robots will allow the automation of almost all laborious tasks whose complexity usually reserves them only for humans [Thrun 2004]. The extent of these robots' capabilities has however lead to a reluctance for humans to trust them to preform many of the tasks which make them so desirable [Andras et al. 2018]. This is because the level autonomy achieved by these robots requires them to make a range of decisions[1] which humans lack complete control over. It is therefore natural for people to expect such a robot be *explainable*. An *explainable robot* is one whose decision making processes can be understood by humans. This capability, often referred to as *explainability*, may allow humans to better understand the behaviour of robots and provide the trust required to deploy them into society. This is especially important in high-stakes or safety critical applications such as disaster rescue and law enforcement.

The need for explainability will arise often in human-robot collaboration. This is because humans usually question decisions made by others which they themselves do not understand or consider to be abnormal [Hilton and Slugoski 1986]. This will occur frequently as a robot may often be forced to adapt its plans as events diverge from its expectations or do things that a human normally would not do [Fox et al. 2017]. Furthermore, only a single decision made by an autonomous system, which cannot be understood by a human, can lead to mistrust if left unexplained [Andras et al. 2018]. This is because humans scrutinise autonomous systems significantly more than other humans [Dietvorst et al. 2015]. Providing explainability is thus of key importance in enabling human-robot collaboration. The following examples illustrate scenarios in which explainability is crucial and show how explanatory content must vary according to context:

Example 1.1 *A self-driving car is transporting a passenger along a familiar route. The car receives information that the road ahead is blocked and takes an unusual turn. In response, its passenger may inquire as to why it did so, and will likely be satisfied with a brief response stating that the road was blocked, which forced it to take an alternative route.*

Example 1.2 *A rescue robot operating in a volatile disaster zone locates a human survivor but determines that the chance of successfully rescuing them is sufficiently low enough to warrant their abandonment to search for another, easier to rescue, survivor. In this case, the robot's commander may demand more detailed justification for its decision.*

There is also substantial legal pressure for explainability. This has arisen from concerns in EU and US governments that AI may make unfair or discriminatory decisions, replicate or develop biases, and behave in unexpected ways in sensitive environments which can put human safety at risk [Wachter et al. 2017a]. The EU's General Data Protection Regulation (GDPR) proposed to mandate a "right to an explanation" about all decisions made by AI systems [Wachter et al. 2017b]. The Civil Law Rules on Robotics also call for robotic systems to include transparency tools to explain the rationale and logic behind their decision making processes [EU 2017a]. Finally, the EU parliament's committee on Civil Liberties, Justice, and Home Affairs state that all future civil laws addressing robotics must comply with the GDPR [EU 2017b].

These issues have lead to recent interest in *eXplainable Artificial Intelligence* (XAI) [XAI 2017], which is now receiving considerable attention from researchers [Adadi and Berrada 2018, Došilović et al. 2018] and substantial investments from official bodies, including the US Defense Advanced Research Projects Agency (DARPA) who launched the XAI program [DARPA 2016]. DARPA states that for an agent to be considered within the scope of XAI it should; justify its actions, explain when it will succeed or fail, and allow the user to understand when it can trust the agent. There has been much work towards this objective [Montavon et al. 2017, Guidotti et al. 2018], including particularly successful work on explaining the predictions of any classifier [Ribeiro et al. 2016]. However, this work has focused largely on producing post-hoc predictions of the input-output mapping of Machine Learning (ML) models. This is insufficient for the needs of human-robot collaboration because they cannot explain, in human-understandable terms; the underlying reasoning with which decisions are made, the beliefs that inform this reasoning, and the provenance of those beliefs.

Doran [Doran et al. 2017] proposes four distinctions to categorise XAI; (1) *opaque systems* whose model is invisible, (2) *interpretable systems* grant partial transparency through statistical analysis, (3) *comprehensible systems* provide partial transparency through symbolic outputs, and (4) *explainable systems* provide total transparency of content and reasoning processes through automated production of human-understandable explanations. Importantly, most current XAI systems fall within the interpretable or comprehensible distinctions, and few come close to an explainable system.

[1] The term *decision* is used to refer to any choice that may be made by a robot, such as *action selection* or *object classification*.

2 Explainability

The capacity for explainability in human-robot collaboration seems clear and has been described as "not an academic exercise" [Langley et al. 2017]. Yet despite that there now exists many advanced technologies for robotics applications, such as for recursive state estimation and sequential decision making, only recently has any significant work been done towards the development of goal-driven explainable robots[2], as supported by a recent survey [Anjomshoae et al. 2019].

There exist numerous approaches to achieving explainability in robotics, characterised under a range of terms such as; explicability, interpretability and comprehensibility [Doran et al. 2017, Chakraborti et al. 2019]. Many of which are not clearly defined, have significant overlap, or are not strictly relevant to robotics. This thesis reduces the range of past approaches to just two classes most fundamental for robotics known as *Explainable Agency* and *Explicable Planning*:

1. *Explainable Agency* - This involves a robot interactively explaining itself by generating human-understandable explanatory descriptions, usually via natural language, of its knowledge, beliefs and experiences, and how these guide its reasoning about; planning, diagnostics and inference [Meadows et al. 2016, Langley et al. 2017]. Achieving this requires building the ability for a robot to explain its reasoning directly into its reasoning model.
2. *Explicable Planning* - This involves a robot generating plans such that they conform to the expectations of humans [Sreedharan et al. 2017], usually formulated as a *model reconciliation problem* [Sreedharan et al. 2019]. Achieving this requires injecting biases to a planner such that generated plans are understandable to humans.

The latter of these attempts to make the behaviour of robots understandable by forcing them to only ever make decisions that conform to human expectations. This prevents the robot from acting in a manner that might be considered abnormal, and thus humans will question the robot's behaviour less often. This is however difficult to achieve because; the robot must sacrifice plan optimality in favour of explicability, it is challenging to determine the humans' expectations, and there may be multiple humans with conflicting expectations. There also exists many scenarios in which conforming to human expectations is impossible or where sacrificing optimality is extremely undesirable, such as those in examples 1.1 and 1.2. Furthermore, explaining a robot's behaviour requires much more than simply stating its plans. Plans are bound to fail for a variety of reasons. When this occurs, the robot must be able to dynamically re-plan and explain; the circumstances that lead to plan failure and its resulting decisions. Explicable planning is insufficient in such situations.

To allow humans to truly understand the behaviour of robots requires that we enable robots with the ability to answer explanatory questions about their actions and behaviour by providing human understandable explanatory descriptions of the underlying reasoning behind which their decisions are made, at all possible levels. In this way, humans may be able to understand the decisions made by robots by attributing a decision to the robot's reasons for selecting it, regardless of the criteria used in its selection. This is the central concept behind approaches within the class of *explainable agency*. To understand the explanatory content that must be provided by an *explainable agent* we need to define an explanation:

Definition 2.1 (An Explanation) *An explanation is an description of how ones knowledge and experiences informs their beliefs, how their beliefs guide their reasoning, and how their reasoning leads to their decisions.*

As humans we often provide such explanations, and robots should too. When asked to explain ourselves we state the reasoning that lead to our decisions based on our beliefs, and how we formed those beliefs based on our current knowledge and past experiences. The process of selecting, constructing and providing such explanations in robotics and AI referred to as *explanation generation*. Based on this definition of an explanation and prior work [Langley et al. 2017], the following three functional abilities are identified which a robot must be enabled with to achieve explainable agency:

1. Explain decisions made during both plan generation and execution.
2. Describe how its knowledge, beliefs and experiences guided its reasoning and lead to its decisions.
3. Communicate all levels of its decision making and reasoning processes in human understandable terms.

It is the opinion of the author that explainable agency is the most promising approach to explainability for human-robot collaboration. Despite this, there does not exist any system in the literature which simultaneously satisfies all of the above capabilities. Most previous work towards explainability in human-robot collaboration has focused on explicable planning. Explainable agency has in contrast received far less attention and remains in its infancy as a research field.

Enabling a robot with the ability of explainable agency for human-robot collaboration thus forms the primary goal of this thesis. This thesis will focus on enabling robots to provide explanations that go further than simply stating decisions made. It will seek to allow robots to expose the true underlying reasoning with which their decisions are made, the beliefs that guided this reasoning, and the provenance of those beliefs. This problem has very frequently been identified as a flaw in previous explainable systems and yet still remains unsolved [Doran et al. 2017, Abdul et al. 2018, Miller 2019]. This will produce more human-like explanations which will allow humans to better understand the behaviour of robots.

[2]The terms *robot* and *agent* are henceforth used interchangeably in this report, as are *explainable robot* and *explainable agent*.

3 Problem Statement

To understand the contributions of this thesis we need to define its scope within the context of human-robot collaboration. There has been significant research on human-robot interaction, but these emphasise joint activity in pursuit of a common goal [Langley et al. 2017]. This thesis instead focuses on the more challenging scenario in which the robot receives initial instructions, carries them out with little to no human supervision, and then must explain itself in retrospect of completing its task. This scenario is referred to as human-robot collaboration and will occur frequently in many domains of interest such as hospitality and disaster rescue. This poses significant design challenges regarding the way in which the robot stores and extracts its knowledge, beliefs, and memory of past experiences [Meadows et al. 2016, Fox et al. 2017].

Many approaches to acquiring expertise in AI, such as deep learning, involve data-driven ML algorithms which produce opaque black-box reactive decision making models whose content cannot be explained directly. Most attempts to explain these black-box models employ the use of a secondary post-hoc model-of-self which sits above the primary decision making model and exists entirely for the purposes of explanation generation. The model-of-self is however only an approximation of the original and as such, explanatory outputs are often inaccurate, unreliable or incomplete. Ultimately, these outputs aren't faithful to the decision making model's actual internal content. As such, practitioners have cautioned against their use [Lipton 2016, Rudin 2019]. Whilst they provide partial transparency into the input-output mapping of black-box models, they lack majorly in their ability to state the underlying reasoning behind which decisions are made.

This thesis distinctly disagrees with such work and finds them to be a poor approach towards explainable agency. It is not possible to achieve many of the desired capabilities of an explainable agent with reactive decision making models. These include; stating actions it plans to execute towards its goals, its corresponding beliefs at each stage of its plan, and its episodic memory of past events. Approximate explanations of ML models also require significant time to produce and thus cannot be provided interactively. It is clearly insufficient to simply design a decision making model and attempt to explain it after the fact because the ability to explain is tightly linked to the knowledge representation and reasoning methodology and its implementation. It is only possible to extract and explain content from a model which represents it such as to support such explanation. To properly realise explainable agency, we must accept that black-box models are insufficient for our needs, depart from their use, and develop novel goal-driven reasoning models designed to support explanation as a central part of their specification. This principal is stated in the thesis problem statement:

Statement 3.1 (Problem Statement) *How is it best to develop a goal-driven decision making and reasoning model for robots which intrinsically supports the ability of explainable agency in the context of human-robot collaboration?*

This is the primary open problem which motivates this thesis. It is the opinion of the author that explainability should be a primary design consideration of any human-robot collaborative system. The thesis contributions are now stated:

Statement 3.2 (Thesis Contribution) *Develop a decision making and reasoning model for a robot collaborating with humans in which the ability to generate explanatory descriptions of its beliefs, knowledge and experiences, and how these guide all aspects of its reasoning about planning, diagnostics and inference, is fully coupled to the implementation.*

Realising this objective will require defining a theory of explanations in the context of human robot collaboration, including specifying the range of explanations to be supported and characterising explanations over multiple abstractions across a "space of explanations". Further, this theory must be instantiated by developing a reasoning model in which every component necessary to support generation of such explanations is bound to the model. This demands a system for automatically extracting the correct information from the reasoning model given a position in this space. Finally, this must be coupled to a natural language interface, from which questions may be received and explanations provided.

It is however challenging to achieve this objective because a robot will be equipped with many different descriptions of knowledge and uncertainty. The robot will rarely have complete and correct knowledge of the domain state during planning, but usually has access to rich commonsense domain knowledge that holds in all but a few exceptional circumstances, such as "books are usually in the library, but cookbooks may be in the kitchen". The robot can also extract new unreliable information from its sensors during plan execution, and the associated uncertainty can be modelled probabilistically, such as "I am 90% certain the robotics book is on the desk". Simultaneously representing and reasoning with these different knowledge elements such that they support explanation is highly non-trivial. Further, observations acquired during plan execution may complement or contradict the robot's current beliefs. The robot is also unlikely to have access to a complete description of its domain's dynamics and may represent its current understanding both via axiomatic logic rules and probabilistic state transition functions. Unfortunately, humans rarely have the time or expertise to provide elaborate feedback or comprehensive knowledge to robots, being often only able of providing a coarse and inaccurate description of a domain [Keil 2003]. There is thus more to making and executing plans than generating a sequence of actions and blindly executing them [Roberts et al. 2018]. Plan failure will occur, when it does, the robot must diagnose and explain this failure, either revising its beliefs to re-plan, or initiate learning to discover previously unknown domain dynamics. Explaining a robot's behaviour in this scenario is challenging but necessary.

Most past work towards the development of goal-driven explainable robots provide only conceptual frameworks or lack thorough evaluation in sufficiently complex domains [Anjomshoae et al. 2019]. Many practitioners, such as those from the eXplAInable Planning (XAIP) community, provide few technical details of their implementations and only brief descriptions of their design methodologies, making it difficult to reproduce their results. There is very little work that defines any formal mechanism for extracting content from a robot's decision making model relevant to a given question. Example explanations are often provided, but no information is given as to how the explanatory content was selected from an often vast quantity of possible candidates. Furthermore, such explanations often only state actions planned by a robot, and fail to describe the rationale and reasoning behind which those actions were chosen over possible alternatives.

To understand why explanation generation for human-robot collaboration is challenging we need some definitions. A transparent system is here defined as one whose internal content can be directly seen and studied by a human. However, a robot will usually process and store vast quantities of information within a variety of different data structures whose elements may be highly interrelated. A significant portion of past work simply employs some such transparent encoding, from which it is theoretically possible to generate coherent explanations. However, simply exposing the entirety of a robot's reasoning model's content in human-readable format may be transparency, but it certainly does not by itself constitute explanation generation. To generate human-understandable explanations requires additional components.

The key concept is that an explanation is an answer to an explanatory question. Therefore, a human-understandable explanation is here defined as one that explicitly answers an explanatory question by providing only the information relevant to the specific question and context under consideration. In order to generate such explanations we must characterise explanations in the context of human-robot collaboration, and define a methodology for evaluating and selecting only the content from a reasoning model relevant to a given explanation. There exist many proposed models of the human process behind which we select content for explanation. For example, Grice [Grice et al. 1975] states four maxims that should be adhered to during conversation; (1) make your contribution at least as informative as is required and no more, (2) make your contribution one that is true and have evidence to support, (3) only provide information related to the conversation, (4) always avoid obscurity and ambiguity, be brief and orderly. Adhering to such models is clearly of importance when seeking to generate relevant and context specific explanations in human-robot collaboration.

It is also fundamental to understand that explanations typically form part of a conversational process in which questions are posed interactively and the corresponding explanations are expected to be immediate and on-demand [Hilton 1990]. The nature of a conversation thus defines the process behind which humans seek for explanations. Initial questions are often abstract, with the respective explanation establishing context, but only rarely satisfying the explainee. Further questions extend the initial such that the conversational process incrementally drills-down upon the specific thing the explainee wants explained. Following from this claim, we thus require another component, and corresponding methodology, for moving automatically between different possible descriptions of a robot's reasoning model throughout an interactive process until the robot arrives at the description the explainee desires, at an appropriate level of abstraction.

This thesis decomposes the interactive explanation generation process performed by an explainable robot into four distinct parts of a closed loop as shown in Figure 1. The question pre-processor deals with *question reception* by interpreting questions given by human collaborators and decomposing them into a constrained natural language format. This is then passed directly to the robot's reasoning model which performs *content extraction* by selecting the appropriate content to answer the given question. The explanation post-processor then performs *explanation communication* by formatting the extracted content to compose an intuitive explanation and presenting it to the user. The user may then provide feedback which is used to revise choices made in the content extraction phase, or terminate the interaction. This thesis however only concerns itself with developing the reasoning model such that it supports content extraction according to the above definitions of a human-understandable explanation. Existing tools for Natural Language Processing (NLP) and Natural Language Generation (NLG) will be used in both the post- and pre-processing phases. The assumption is made that it is always possible to obtain explicit questions and all questions have a corresponding explanation. This thesis also does not consider facets of the human conversation such as argumentation and deception. It is important to note that this interactive process is not conversational as the robot does not probe the user with questions.

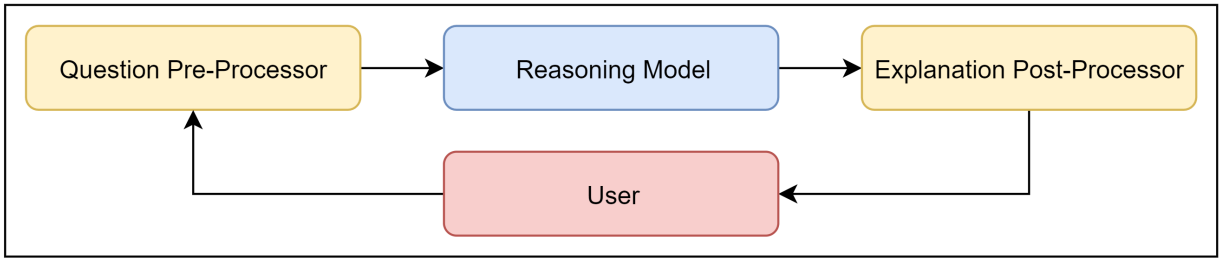

Figure 1: Components of an Interactive Explanation generation process.

4 Related Work

This section reviews past work related to this thesis. Firstly, sub-section 4.1 lists various guiding principles identified in previous work towards the development of explainable agents and explanation generation systems in robotics and AI. Sub-section 4.2 then reviews previous approaches towards generating and providing explanations within the context of human-robot collaboration. This shows that there exists a wide variety of differing techniques towards selecting, formatting and presenting explanations to humans. The advantages and limitations of each of these are identified and discussed. Sub-section 4.3 then details a formalism for characterising and constructing explanations as answers to explanatory questions and as part of a conversational process, called the axes of explanation. Finally, in sub-section 4.4 the range of existing techniques towards mixed logical and probabilistic reasoning in robotics applications are reviewed.

4.1 Guiding Principles

Various guiding principles towards the development of explainable agents, the design of explanation generation systems and the nature of explanations in the context of human-robot collaboration have already been identified in recent work. These may be used to great advantage when approaching the objectives of this thesis and as such are stated here.

Langley et al. [Langley et al. 2017] state the desired capabilities of an explainable agent as:

1. Given a *complex set of objectives* that require an agent's extended activity over time;
2. Given *background knowledge* about categories, relations, and activities that are relevant to these objectives;
3. Produce *records of decisions* made during plan generation, execution, and monitoring in pursuit of these aims;
4. Produce *summary reports*, in human accessible terms, of the agent's mental and physical activities;
5. Produce *understandable answers to questions* that are posed about specific choices and the reasons for them.

They also identify three primary "elements" for achieving explainable agency:

1. *Represent knowledge in a way that supports explanation*, including elements that describe domain properties, states and actions in terms that humans find intuitive, as well as presenting the choices that were available during planning, selections made by the agent, and criteria used to make its decisions.
2. *Have an episodic memory* that records states and actions considered during planning, justifications for action selection, a record of action execution and state changes, and events that resulted in plan revision.
3. *Be able to access and extract content from episodic memory to answer questions* about their experiences. This requires the ability to interpret at least constrained natural language, retrieve relevant structures from memory, and report answers terms that humans find comprehensible.

Sridharan and Meadows [Sridharan and Meadows 2019] state the following principles for explanation generation:

1. Explanations should *present context-specific information* relevant to the task, domain and the question under consideration, at an appropriate level of abstraction.
2. Explanations should be able to *describe knowledge, beliefs, actions, goals, decisions, rationale for decisions, and underlying strategies* or models in real-time.
3. Explanation generation systems should *have minimal task or domain-specific components*.
4. Explanation generation systems should *model and use human understanding and feedback to inform its choices* while constructing explanations.
5. Explanation generation systems should *use knowledge elements that support non-monotonic revision* based on immediate or delayed observations obtained from active exploration or reactive action execution.

Finally, Miller [Miller 2019] summarises the major findings from the social sciences regarding explanations:

1. *Explanations are contrastive* - They are sought in response to particular counterfactual cases, e.g. people do not just ask why some event X happened, but rather why X happened instead of some other event Y.
2. *Explanations are selected in a biased manner* - People rarely expect an explanation that consists of a complete cause of an event and usually select one cause from an infinite number of possible causes to be the explanation.
3. *Probabilities probably don't matter* - Referring to probabilities or statistical relationships is less effective then referring to causal relationships, and the most likely explanation is not necessarily the best.
4. *Explanations are social* - They transfer knowledge presented as part of a social conversation or interaction, note that this does not demand that explanations be given in a natural language.

4.2 Explanation Generation Systems

This section reviews the range of past approaches towards generating and providing explanations for human-robot collaboration. The advantages and limitations of each are identified and discussed with regard to the goals of this thesis.

Fox [Fox et al. 2017] proposes eXplAInable Planning (XAIP). They characterise many of the questions that need to be answered by a robot but maintain that there is no clear way to define what constitutes a good explanation. They propose a general methodology for providing explanations about; why the planner chose particular actions over others, why the planner's decision was the best at the time of planning, why it believed certain actions could not be executed, and why/if it needed to re-plan after taking observations. Their illustrative examples provide explanations which expose a large proportion of the robot's reasoning but they do not abstract explanations to a level of detail appropriate for the context. Explanations are instead presented similarly to how content is internally represented by the robot. This means that explanations are often over-informative or may not be understandable by a non-expert. Zakershahrak [Zakershahrak et al. 2019] presents an approach which seeks to allow humans to better understand explanations by making them in an online fashion throughout execution and thus spreading out the information provided. However, this is often insufficient for explaining a robot's behaviour after an extended period of operation without human supervision. In contrast, Amir and Amir [Amir and Amir 2018] propose the HIGHLIGHTS algorithm which instead generates abstract summary reports of an agent's behaviour over an extended period, rather than explaining decisions individually. This is more applicable to human-robot collaboration but reduces the ability for the robot to answer distinct questions. In past work, the difficulty of providing relevant and context specific explanations in response to explanatory questions is thus often avoided entirely by employing some other explanatory technique which can no longer answer such questions.

A number of explanation generation systems have been developed based on the concept of constrastive explanation. A constrastive explanation may state, why a robot chose an action X (the actual) over some other action Y (the hypothetical), referred to as the fact and foil respectively. The fundamental concept is that by exploiting constrastiveness, we gain an explicit focus on the most relevant part of a robot's decision making model that the user wants explained [Lipton 1990]. See appendix E.3 for a detailed discussion of constrastive explanation. Borgo [Borgo et al. 2018] has developed one such system called XAI-PLAN which is implemented on top of the ROSPlan framework [Cashmore et al. 2015] and provides an API for robotic planners employing PDDL models [Ghallab et al. 1998] in the Robot Operating System (ROS) [Quigley et al. 2009]. Their system provides explanations about initial plans and supports contrastive explanation regarding why actions were selected over alternatives suggested by the user. This highlights the differences between decisions made by the planner and what the user expected. Krarup [Krarup et al. 2019] develops a similar system for planners employing PDDL2.1 models. Their method involves generation of a hypothetical plan that encapsulates the foil in the user's question which is then compared to the robot's original plan to construct the constrastive explanation. The user can ask further questions to refine the hypothetical plan until the constrastive explanation identifies the reasons why the robot made a particular decision over that which the user expected. Korpan [Korpan and Epstein 2018] similarly developed the Why-Plan system which attempts to answer questions of the form "Why does your plan go this way?". They explain a robot's navigation by comparing the plan the robot generated with the plan the human expected it to make and constructing an explanation, using a template based approach, which identifies how the plans differ with regards to the objectives and performance criteria used by the robot. They argue that this "produces (more) meaningful human-friendly explanations". However, whilst many other practitioners support constrastive questions they do not support constrastive explanation and instead just provide two complete explanations for fact and foil [Mittelstadt et al. 2018, Miller 2019]. Many researchers argue that all why–questions ask for contrastive explanations even if the foil is implicit [Hilton and Slugoski 1986, Lipton 1990, Hilton 1990, Lombrozo 2012]. Hence it seems that supporting constrastive explanation is both advantageous and necessary when designing explanation generation systems.

Other work focuses on providing explanations for goal-reasoning robots. Mcclure [McClure 2002, McClure et al. 2003] argues that people *always prefer* explanations justifying actions to cite goals one hopes to achieve through those actions above all other reasons for that action. any explanation generation systems can state a robot's goals, but not how robot selected actions according to those goals or how those goals where originally adopted. Dannenhauer [Dannenhauer et al. 2018a] states that goal-reasoning robots may often adopt, reject or alter goals given by human participants or other robots. This is because the given goals may violate one or more of its own constraints or lead to an undesirable outcome [Dannenhauer et al. 2018b]. For a robot to explain such behaviour it should be able to state how its knowledge, beliefs and constraints affected its goal selection. Work towards this capability includes an ASP based framework for reasoning about a robot's "mental" state and deal with intentional actions [Rocio Gomez and Riley 2018].

Explanation generation and explicable planning have been combined [Chakraborti et al. 2018, Chakraborti et al. 2017]. Sreedharan [Sreedharan et al. 2017] follows the argument that explicable planning reduces the need to "explain the obvious" but does not eliminate the need for explanation generation. They propose MEGA the Multi-model Explanation Generation Algorithm which tries to bring together the advantageous properties of explicable planning *and* explanation generation. This combination may prove effective but the explanation generation capabilities are currently unsatisfactory.

There has also been a comparative study between two contrasting explanation generation systems, KRASP and UMBRA [Meadows et al. 2016]. The prior is an Answer Set Programming (ASP) based system and the latter is a cognitively inspired heuristic based system. They find that ASP based systems have many advantages over heuristic based systems. However, heuristic based systems do have some complementary capabilities including better scaling to large and complex domains. On the other hand, ASP based systems have greater domain independency but cannot explain anything beyond that which has been explicitly encoded by the designer. They are also limited due to the fact that they must compute full-models and thus cannot generate partial explanations. ASP reasoning systems are however highly transparent in that they represent the robot's knowledge, beliefs and decisions in the form of constrained natural language, and in a way that allows the reasoning that lead to such beliefs and decisions to be extracted. However, displaying an entire answer set, which may contain many tens-of-thousands of atoms, clearly is not explanation. In section 4.4 other ASP based reasoning systems are discussed in the context of mixed logical and probabilistic reasoning.

Probabilistic sequential design making algorithms such as Partially Observable Markov Decision Process (POMDP) are particularly challenging to explain because their content contains complex statistical relations. There has however been work towards explaining POMDP policies [Sukkerd et al. 2018]. These have focused primarily on labelling states and actions in order to extract explanations regarding choices made by a robot using template based approaches. These provide; minimal explanations of why the robot's chosen action is optimal [Khan et al. 2009], the robot's confidence in this choice, and the relative likelihood of possible outcomes [Wang et al. 2016]. They maintain that explaining why an action was taken under such a policy requires explaining how and why the chosen action has the highest expected utility. Such explanation can however be unsatisfactory unless accompanied by an underlying causal explanation [Miller 2019]. Explanations of POMDPs may be best used in tandem with high-level explanation generation systems such as KRASP. In section 4.4 systems which combine ASP and POMDPs are discussed to which such a technique is highly applicable.

4.3 Axes of Explanation

Recent work has made progress towards defining a precise model and corresponding methodology for generating relevant and context specific explanations for human-robot collaboration [Sridharan and Meadows 2019]. They develop a general theory of explanations which includes representing, reasoning with, and learning domain knowledge such that it supports explanation. This includes three fundamental axes upon which to characterise and construct explanations:

1. *Representation abstraction* - This axis models the abstraction level of the domain description used to generate an explanation. The robot may use a coarse or fine resolution description, for example it may describe it's location in the general context of what room it is currently in or what precise cell in that room it is in.
2. *Communication specificity* - This axis models what the robot focuses during communication. Explanations can be brief, describing only domain properties and beliefs most specific to a given explanation, or explanations can be elaborate, describing a broader view of the domain state and how each property effects its reasoning.
3. *Communication verbosity* - This axis models the comprehensiveness of the response provided. For example either; stating the final action a robot took before completing its goal, stating every single action it executed to achieve the goal, or stating all actions along with preconditions and expectations for the effects of each action.

The fundamental idea is that these axes model the level of detail which explanatory descriptions provide and the level of abstraction at which they are presented. An explainable robot should construct an explanation by choosing a suitable position upon these axes, revise this choice based on human feedback, and thus be able to select and move automatically between different possible descriptions of its internal model when providing answers to explanatory questions throughout a conversational process. They propose a general methodology to perform this process:

1. Choose a position along each of three axes to provide explanation.
2. Determine what needs to be described in the explanation.
3. Produce relevant, context-specific explanations that limit the use of domain-specific knowledge.
4. Use human feedback to revise the choice in Step-1.

This however tells us little of how to implement support for these axes into a robot's reasoning model. Their proposed system is ASP based, similar to KRASP, and provides the ability to select only the atoms relevant to a given explanation from an answer set. There however does not exist a precise mathematical definition of where upon these axes a given explanation falls. This is more obvious for some axes, for example; representation abstraction requires only labelling knowledge elements with their respective resolution levels. Whereas specificity is more problematic as its definition may change depending on the domain under consideration. This thesis will thus develop a richer theory of explanations, a mathematical definition of where an explanation falls upon these axes, and implement the relevant KRR mechanisms to create an concrete instance of this theory. This thesis may not adopt the same axes but the concept will be of importance.

4.4 Mixed Logical and Probabilistic Reasoning

Many current logic based approaches to computing high-level deterministic plans in robotics, such as those from the autonomous planning community, are based on first order propositional logics. These are well suited to representing and reasoning with commonsense knowledge but do not support probabilistic representations of uncertainty, and attaching probabilities to logic statements is rarely a sufficient compromise. These also require comprehensive prior knowledge of the domain dynamics, the agents' capabilities, and the preconditions and effects of their actions. Furthermore, they do not support many of the desired reasoning capabilities including; non-monotonic revision of beliefs based on integration of new unreliable sensor observations and reasoning about the occurrence of unobserved exogenous actions that may have lead to unexpected observations. The latter of which occurs particularly frequently in human-robot collaborative domains due to the unpredictable behaviour of humans. Non-monotonic logic paradigms such as Answer Set Programming (ASP) address these short comings by reducing planning and diagnostics to computing answer sets of ASP programs [Baral 2003, Lifschitz 2008, Gelfond and Kahl 2014]. On the other hand, probabilistic sequential decision making algorithms, such as Partially Observable Markov Decision Processes (POMDPs), are well suited to representing and reasoning with probabilistic representations of uncertainty to plan low-level sensing and actuation on robots. This is important because a large amount of information available to robots is represented probabilistically to capture the non-determinism in sensing and actuation that exist in real-world robotics domains. Unfortunately, these are poorly suited to reasoning with commonsense knowledge and become computationally intractable when dealing with large domains and knowledge bases, even when employing state-of-art approximate solvers [Shani et al. 2013]. Thus has arisen the need for mixed non-monotonic logical and probabilistic planning. Yet efficient coupling of logic and probabilistic reasoning remains an open problem in robotics [Zhang et al. 2014]. Many techniques have been developed attempting to solve this such as, Markov Logic Networks [Richardson and Domingos 2006], Bayesian Logic [Andersen and Hooker 1994], Independent Choice Logic [Poole 2000], Probabilistic First-Order Logic [Jaumard et al. 2006], First-Order Relational POMDPs [Sanner and Kersting 2010], and P-Log the probabilistic extension of ASP [Balai et al. 2019, Balai 2019a].

A mobile robot planning architecture has been developed which combines first-order logic and probabilistic reasoning for open world planning which can; deal with uncertain and incomplete domain knowledge, and handle and explain task failure intelligently [Hanheide et al. 2017]. The robots' knowledge was divided into layers based on three distinct types, instance, default and diagnostic, where higher level knowledge can modify lower level knowledge. Information flow is represented via a three layer system (independent of the knowledge layers), these are the; competence, belief and deliberative layers. They also provide a powerful system for assumptive planning by assigning probabilities to defaults to avoid the robot initially making unlikely assumptions. The final architecture solves four important robotic planning problems; planning under uncertainty, planning in open worlds, explaining task failure and verifying those explanations.

Recent work has developed a refinement based architecture which represents and reasons with tightly coupled transition diagrams of any given domain at multiple levels of resolution [Sridharan and Gelfond 2016, Sridharan et al. 2019]. Where the fine resolution diagram is defined as a refinement of the coarse resolution. The architecture builds upon past work combining the complementary capabilities of ASP, a non-monotonic logical declarative programming paradigm supporting commonsense reasoning, with POMDP probabilistic planning [Zhang et al. 2015]. The action language AL_d is extended, to allow non-boolean fluents (amoungst other additions), and translated into ASP. For any given goal, non-monotonic logical inference, which includes default knowledge, computes a sequence of abstract actions which deterministically guide the robot. Each such abstract action is then implemented as a sequence of concrete actions, by zooming to and reasoning probabilistically over just the relevant part of the fine resolution system description. The computational tractability of POMDP policy generation is improved significantly by heuristically generating multinomial priors for POMDP state estimation from the ASP answer sets. Observations obtained during execution of the probabilistic policy are added to the ASP knowledge base. When observations contradict the robot's beliefs the architecture can perform diagnostics to explain them, either by hypothesising the occurrence of unobserved exogenous actions or generating indirect exceptions to initial state defaults. A similar technique, removes the POMDP planner used for executing each abstract action. Sensor observations instead perform incremental Bayesian updates to a probability distribution over just the subset of the domain relevant to each particular abstract action [Colaco and Sridharan 2015].

Other work combines P-Log with POMDPs to support mixed COmmonsense Reasoning and Probabilistic Planning (CORPP) [Zhang and Stone 2015]. The resulting system uses P-Log to generate informative priors for a POMDP solver by eliminating state variables that are irrelevant to the optimal policy, showing significant improvements in efficiency and accuracy of the resulting POMDP policy. Similar work has developed a framework for HRI implemented in OpenDial, which combined P-Log with MLNs based on the concept of probabilistic rules which define mappings between logical conditions and probabilistic effects [Lison 2015]. Further work expanded CORPP producing the Interleaved COmmonsense Reasoning and Probabilistic Planning system (iCORPP) [Zhang et al. 2017, Zhang and Stone 2017]. This system decomposes a planning problem into two sub-problems; commonsense reasoning and probabilistic planning, described as focusing on; "understanding the world" and "accomplishing the task", respectively. This technique allows the robot to reason about exogenous action occurrence and reduces the state space for POMDP probabilistic planning.

5 Proposed Architecture

This section describes the proposed proof-of-concept architecture which will be developed by this thesis. Its design is based on past work on representing and reasoning with tightly coupled transition diagrams of any given domain at multiple levels of resolution [Sridharan et al. 2019], and mixed logical and probabilistic reasoning [Zhang et al. 2015].

The architecture consists of two primary components; an ASP based non-monotonic logical reasoner which computes deterministic abstract plans and a POMDP based probabilistic executor which computes non-deterministic concrete plans, as shown in Figure 2. Note that each component will be composed of multiple functional elements abstracted away in this diagram. This thesis focuses primarily on explaining the decision making and reasoning of the ASP based logical reasoner and explaining that of the POMDP based probabilistic executor is left as an extension for future work.

The logical reasoner reasons with multiple system descriptions of any given domain at different levels of resolution, including those with *more than two resolutions*, written in the action language AL_d [Gelfond and Inclezan 2013]. In AL_d a dynamic domain defined by a system description SD defines a transition diagram $\tau(SD)$. A trajectory through $\tau(SD)$ is a solution to a planning problem in SD, see Section A.1 for the definition of a transition diagram. Let SD_r denote a system description of AL_d at a resolution level of r. Then if $r = 0$, the resolution type of $\tau(SD_r)$ is *fine*, and otherwise it is *coarse*. The logical reasoner computes sequences of abstract actions over each such transition diagram to form abstract plans at all levels of resolution. Each abstract action from the plan at resolution level $r = 1$ is then passed to the POMDP probabilistic executor and implemented as a sequence of concrete actions by zooming to and reasoning probabilistically over just the relevant part of the fine resolution system description at $r = 0$. Sensor measurements taken during probabilistic execution are committed as observations to the recorded history of the logical reasoner. Records of such observations are maintained at all levels of resolution. When observations contradict the robot's beliefs, non-monotonic logical inference is used to formulate and solve a diagnostics problem. Contradictions may be solved by retracting assumptions about the initial domain state or hypothesising the occurrence of unobserved exogenous actions.

The current architecture falls close to the above definition of a transparent system as it is currently incapable of directly answering explanatory questions. In order to support this ability, the architecture will implement the axes of explanation. The architecture further expands upon this work to expose a greater proportion of the robot's reasoning processes, such as its expectations for the effect of any action, and if/why it believes a given action to be executable in any given state.

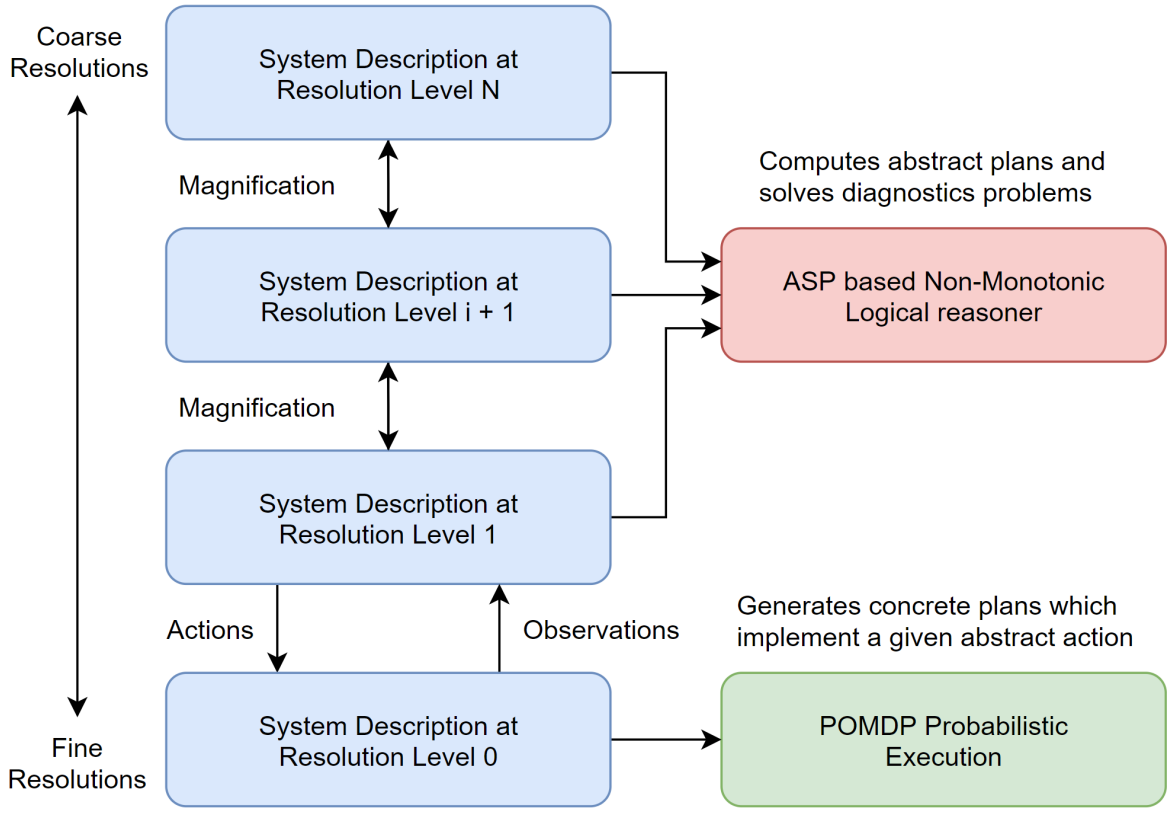

Figure 2: High-level description of the primary components of the proposed architecture.

The current progress towards the development of the architecture consists only of the ASP based logical reasoning component. The role of the logical reasoner can be thought of as analogous to that of the work performed by a human logistician. It contains the logic rules required to reason, through non-monotonic logical inference, with the robot's knowledge, beliefs and experiences in order to formulate and solve both planning and complex diagnostics problems.

The logical reasoner is split into six distinct *modules* based on their particular functional role within the component. The idea of partitioning the logical reasoner into modules takes advantage of Clingo's (the ASP solver used by the architecture) *program parts* described in section 3.1.15 of [Gebser et al. 2019b]. These modules allow the Clingo program to be partitioned into distinct individual pieces which may be selectively grounded and solved. This can be used to implement *multi-shot solving* [Gebser et al. 2019c] and integrate new observations into the program without having to re-ground the entire program [Gebser et al. 2015b]. The following six modules are used by the logical reasoner:

1. *Controller* - This is the "master" that governs the control flow between all other modules.
2. *System description* - This module includes the domain dependant action language system description.
3. *Recorded history* - This module includes a set of statements defining the observations made by the robot.
4. *Base module* - This module includes a set of general axioms used by all solver calls and system descriptions.
5. *Diagnostics module* - This module solves diagnostics problems by reasoning with default knowledge to resolve a complete and consistent initial model of the domain state, and reasons about the possible occurrence of unobserved exogenous actions to explain observations that contradict beliefs.
6. *Planning module* - This module computes minimal, complete and correct plans, such that all goals are satisfied in the minimum possible number of steps and also preforms optimisation of cost functions.

Figure 3 shows a graphical model of these modules, the control flow between which is indicated by the arrows. The *ASP Controller* is the "master" module which governs the control flow between the other five modules. The green modules, the *system description* and *recorded history*, contain the robot's axiomatic knowledge of the domain's dynamics and memory of past observations, respectively. The system description and recorded history modules contain all different descriptions at every level of resolution but they can be selectively loaded. Note that the current implementation allows updates to the recorded history, but since learning is not yet implemented, no automatic revisions are made to the system description. The three modules contained within the large grey box, the *planning module*, *base module* and *diagnostics module*, contain the ASP code used to reason with the system description and recorded history to compute plans and perform diagnostics. Each of these modules includes one or more prioritised CR-rules [Balduccini and Gelfond 2003] translated according to a novel algorithm that extends algorithm 1 of [Balai et al. 2013a] to allow preferences over CR-rules and CR-rules with cardinality constraints. These CR-rules allow defeasible reasoning such as indirect exceptions to initial defaults. The logical reasoner operates according to the following informal algorithm:

Algorithm 1: Architecture Control Flow

```
Invoke Diagnostics - Resolve a consistent model of any initial knowledge;
Invoke Planner - Compute an initial minimal and complete plan that satisfies all goals;
if ASP solver returned UNSAT then
    return Program inconsistent or goals unsatisfiable
end
while Goals not completed do
    Take observations for the effects of previous action and preconditions of next action;
    Commit records of observations to the recorded history;
    if Observations contradict beliefs then
        Invoke Diagnostics - Resolve a consistent model of the recorded history to explain unexpected observations;
        if Plan invalidated by observations then
            Invoke Planner - Compute a new minimal and complete plan that satisfies all goals;
        end
    end
    if ASP solver returned UNSAT then
        return Goals unsatisfiable
    end
    Pass next abstract action to the next level of resolution;
    Commit a record of the executed action to the recorded history;
end
return Success
```

The majority of work done so far has focused on the development of the *diagnostics module* to solve complex diagnostics problems. The need to preform diagnostics arises when observations committed to the recorded history contradict the robot's current beliefs. The system automatically requests such state observations in order to check the effects and preconditions of actions executed by the robot. Contradictions can be resolved either by; retracting assumptions by generating exceptions to initial state defaults, or by hypothesising the occurrence of unobserved exogenous actions. As standard, the architecture will attempt to solve diagnostics problems by retracting assumptions of the initial state before hypothesising any exogenous actions. It is however possible to override and swap this order on a fluent by fluent basis.

A novel contribution of particular interest is the logical reasoner's ability to determine whether observations committed to the recorded history contradict the robot's beliefs without having to make costly calls to the ASP solver, as shown in algorithm 1. There are also conditions upon which, despite observations contradicting beliefs, the original plan is still valid. This occurs only when no action preconditions or action effects are violated by the observations. Thus, the system is able to minimise the number of necessary calls to the planning module to only those that are absolutely required. The precise methodology behind which this process is performed is not described here as it has not yet been reviewed.

In past work, diagnostics problems could be resolved only by encoding domain dependant non-prioritised consistency restoring rules (CR-rules) [Sridharan et al. 2019]. The proposed architecture extends this and the complex preference relationships between exceptions to initial state defaults and exogenous actions are instead reduced to an optimisation problem by automatically expanding a set of domain-independent CR-rules. It is also possible for initial state defaults to define a discrete hierarchy of possible initial states, from the most likely to the least likely. Thus avoiding assigning probabilities to possible initial states of the domain. Multiple possible states with equal likelihood can also be specified at the same level in the hierarchy, thus eliminating a set of unlikely states rather than specifying the most likely. This is important because the robot will rarely have access to complete correct knowledge of the initial domain state but usually has access to rich commonsense knowledge such as "books are usually in the library, but cookbooks may be in the kitchen". The robot can use this commonsense knowledge to produce the complete belief state it requires for planning.

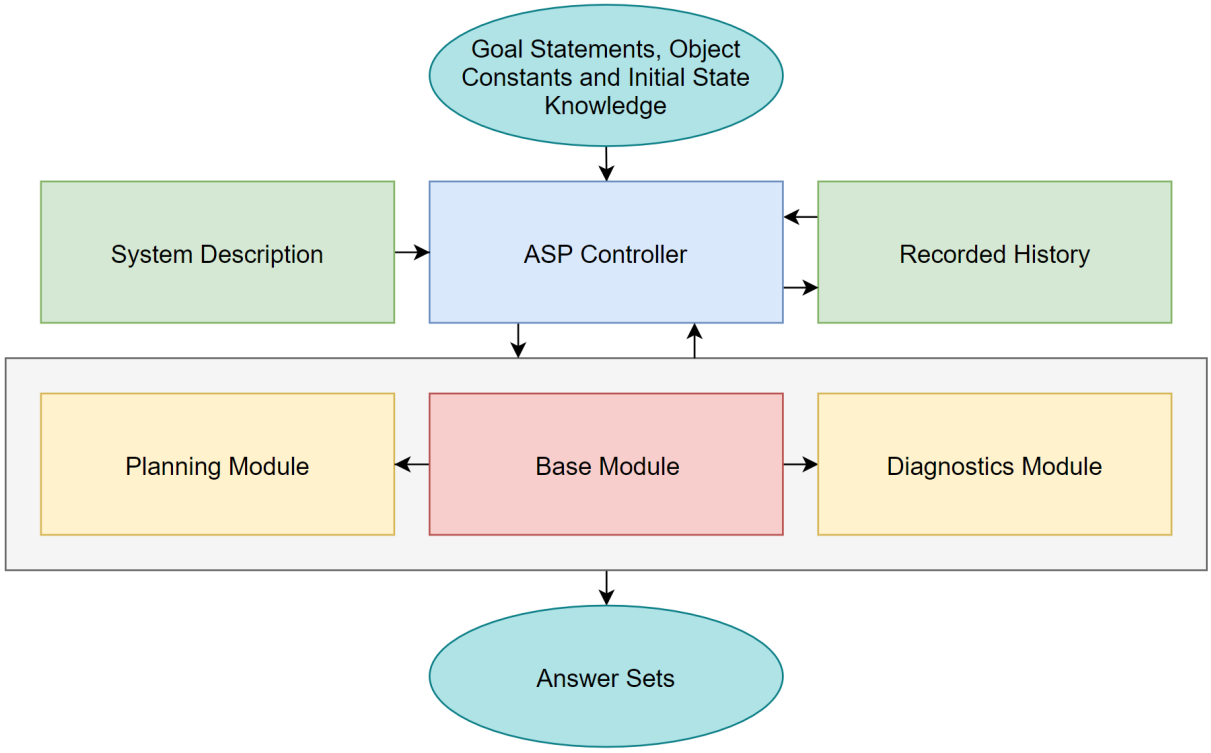

Figure 3: Diagram of the ASP based logical reasoner. It takes as input a set of goal statements, object constants, and initial state knowledge relevant to those goals. Outputs are answer sets containing abstract plans. The ASP controller manages the control flow between various components, including; loading and grounding the system description, maintaining the recorded history of observations, and invoking the relevant modules in the logical reasoner to solve planning and diagnostics problems. The logical reasoner (grey box) contains the planning, diagnostics and base modules. The base module is always invoked during solver calls along with either the planning module or diagnostics module (but never both simultaneously) to solve planning or diagnostics problems respectively. The name logical reasoner is derived from the idea that it contains all the logical axioms that a human logistician may use to solve such problems.

6 Plan of Future Work

Throughout the second year of this thesis, the theories, methodologies, and software required for generating explanations in the context of human-robot collaboration will be developed. This will require completion of the following tasks:

1. Specify and instantiate a theory of explanations for human-robot collaboration:
   - Define the range of explanations to be generated with clear justification for these choices.
   - Characterise explanations by defining where a given explanation sits upon the axes of explanation.
   - Develop a concrete software instance of an explanation generation system that implements this theory.
2. Expand the proposed knowledge representation and reasoning model to support this theory:
   - Support automated reasoning at multiple levels of resolution.
   - Support automatic extraction of the correct content from the model given a position on these axes.
3. Provide examples of execution traces and generated explanations to demonstrate the contributions of the thesis:
   - Perform experimental trails of the reasoning model on multiple simulated domains.
   - Combine the explanation generation system with pre-existing software for NLP and NLG.
   - Demonstrate that a natural language explanation can be generated given content selected from the model.
4. Perform evaluative user testing on a variety of human users:
   - Initial tests on users from research institutes with prior expert knowledge of robotics.
   - Primary tests on various users from the general public to evaluate performance with actual intended users.

Current work is focused on extending the knowledge representation and reasoning model towards supporting reasoning at multiple levels of resolution. This is primarily to facilitate multi-level planning, a process which involves passing the effects of actions planned at the higher levels as goals to the lower levels. This will allow explanations of plans and executed actions at multiple different abstractions, but requires an extension of the notion of refinement given in previous work [Sridharan et al. 2019]. Initially, this work will be tested on a basic office domain at two levels of resolution. This domain involves a mobile service robot collecting and delivering books for office workers from connected libraries. Testing will occur in a two dimensional simulated environment and there is currently no plan to extend testing to a real robot. The assumption is made within this simulation that observations are fully deterministic. The multi-level planning systems will be complete by, and presented in, the RSMG 4 report due on the 15th April 2020.

Work will begin on design and development of the explanation generation systems commencing the first quarter of 2020. In particular, development and instantiation of the axes of explanation, and what it means for an explanation to sit on a given position within the space of explanations defined by these axes, will figure centrally at this stage. This requires ensuring that explanations are possible at all abstractions available in the reasoning model, and for each type of question, there is a satisfiable explanation for any given position upon these axes. This further requires a precise definition, for each supported question type, of the nature of each axis and what it means to move along each of them.

Initially, only what and why type explanations/questions will be supported. These include explanations to questions such as “What is your plan?” and “Why did you execute the sequence of actions you did?” respectively. The latter of which is particularly interesting because the robot’s final sequence of executed actions may be very different from its initial plan or the sequence of actions the human user themselves would have taken. This can occur because assumptions the robot made about the initial state of the domain were incorrect or because the activities of humans changed the state unbeknown to the robot. This is a challenging scenario to explain as it requires the robot to maintain an episodic memory of its past experiences, decisions it made and the reasoning which lead to such decisions. From this, a robot may store and retrieve information to answer questions about decisions it made during plan execution, such as initiating diagnostics and re-planning. This is an important ability for human-robot collaboration and yet still remains unsolved. Once the support for plain what and why type questions is complete, these will be extended to further support what-if (or hypothetical whats) and why-constrastive type explanations/questions. These include explanations to questions such as “Would your plan be valid given some hypothetical observation O at time T?” and “Why did you take action A (that the robot actually executed) over some other action B (that the user themselves would have taken)?” respectively. The RSMG 4 report will include examples of at least plain what and why type explanations generated by the architecture.

Through the period commencing completion of the multi-level planning and continuing through the second quarter of the second year, the additional example domains will be developed. These will be; (1) the hospitality domain, which will serve as the primary domain for user studies discussed below, (2) a construction domain which will serve as a test bed for more complex planning problems and a possible extension to multi-robot planning, and (3) a blocks world tutorial domain for use in an online appendix to demonstrate translation of action language AL_d to the proposed architecture.

In parallel with development of these example domains, experimental trials will be performed to test the capabilities of the architecture. This will involve unit testing performed across all example domains. Such unit tests will include generating explanations for all possible question types for all combinations of the positions on the axes of explanation and checking that such explanations are generated correctly according to the theory of explanations given by this thesis.

Upon completion of these example domains, user studies will commence. Initially, in the third quarter of the second year, user studies will be restricted to review by a small number of fellow research students. This will give a relatively rough evaluation of the suitability of the proposed explanation generation techniques based on the views of those equipped with prior expert understanding of robotics and AI. This evaluation will be used to make necessary changes to the explanation generation system before any attempt is made to test the system on the general public. These evaluation results, and any changes made based on them, will be presented in the RSMG 5 report due on the 21st October 2020.

Since explainable agency is motivated by a human desire to understand the behaviour of robots it is of crucial importance that explainable agents be evaluated through the judgements of humans [Langley et al. 2017]. However, a recent study of literature regarding explainable agency found that less than one quarter of 62 reviewed papers had "clear content descriptions and presentation of their explanation methods" or "good" presentation of results, with approximately a third not providing any evaluation results at all [Anjomshoae et al. 2019]. Thus, following the RSMG 5 meeting, will commence more comprehensive user studies on a wider audience, from research institutes and the general public, to evaluate the proposed architecture against the actual intended users. Evaluation measure may include the following:

- Subjective measures:
  - Peoples' opinions of the suitability and clarity of various explanations.
  - Peoples' ratings of the degree to which they trust the robot after collaborating with it.
- Objective measures:
  - Peoples' ability to correctly summarise an explanation that a robot has given them.
  - Peoples' ability to predict a robot's behaviour based only on communication with the robot on prior tasks.

This thesis will focus primarily on objective evaluation measures. Particularly, studies will involve the user posing a specific question to the robot, allowing them to read the explanation, and then asking the user to describe to the examiner what the robot's answer was. From this, some metric may be devised, based on the user's ability to correctly summarise the given explanation, of how "understandable" the explanation was. Although the evaluation methodology has not yet been fully developed, evaluation results are expected to provide significant insight into the effectiveness of the proposed reasoning model and explanation generation systems. The results will also be necessary to defend the contributions of this thesis and show that significant progress has been made towards explainable reasoning for human-robot collaboration. Further, evaluative results may identify gaps that will highlight paths for future work.

If there is sufficient time in the third year and the thesis group finds it reasonable, the proposed architecture will be extended. Possible extensions may include; (1) addition of a POMDP component as shown in Figure 2 based on work from [Sridharan et al. 2019] for constructing priors from ASP answer sets, to demonstrate explanation of mixed logical and probabilistic reasoning, and (2) and supporting centralised and decentralised network-aware reasoning to explain communications diagnostics in multi-robot teams, based on work from [Balduccini et al. 2014].

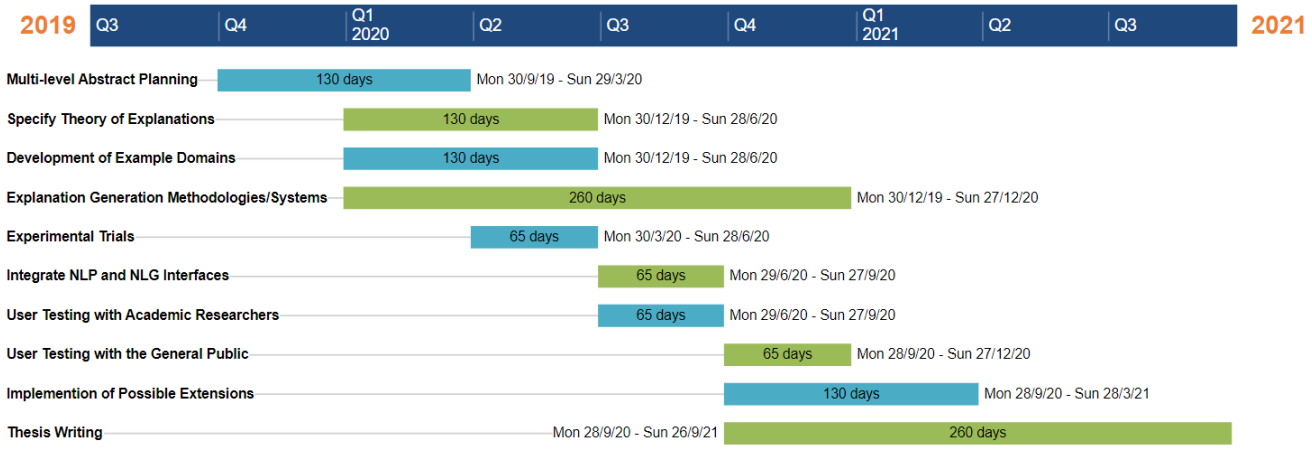

Figure 4: Gantt chart of plan of future work.

7 Proposed Proof-Of-Concept Domain: Robotics in Hospitality

The proposed proof-of-concept domain for simulated testing of the proposed architecture is a hospitality domain staffed by service robots. The hospitality domain may contain multiple different robots with heterogeneous abilities, each of which can be assigned any of a large number of different possible tasks. These may include; moving luggage into guests' rooms, bringing guests room service, cooking and cleaning. Each of these pose interesting and challenging problems in robotics, including navigation through a highly dynamic domain filled with human collaborators and complex manipulation actions of previously unseen objects. There will also exist a variety of human collaborators in the domain such as; guests, staff members, and maintenance engineers. Each of these may vary significantly both in their understanding of the functional abilities of the robot and their level of authority over the control of the robots.

7.1 Advantages of the Hospitality Domain

In traditional robotics domains, such as those in the manufacturing industry, the domain is designed specifically around the incorporation of robots to simultaneously maximize safety and productivity. Robots deployed in such domains are usually enclosed in safety cages and equipped with complete and correct domain knowledge. Furthermore, human operators will have a strong understanding of the robots' functional abilities and operational safety criteria.

The human robot-collaborative problem posed by a hospitality domain is in contrast significantly more challenging than that of traditional robotics domains. The challenge is two fold. Firstly the domain itself is substantially more complex and dynamic due mainly to the uncertainty introduced by the unpredictable nature of the behaviour of humans occupying the domain. This is problematic for planning and diagnostics as the robot will often encounter events or properties it did not expect or account for in its plans. Secondly the range of different humans collaborators, each with differing requirements from the robots, means that explanatory requests may vary significantly between different people. For example, because guests may have very little understanding of the decision making capabilities of a robot, their queries may be significantly more obscure and demand more abstract explanations than those of an experienced staff member. It is also notable that the domain is safety critical. The service robots will operate in close proximity to humans and as such they must operate with high reliability to minimise risk of potential harm to humans. Furthermore, if a robot is to cause harm to a human, it must provide detailed justifications for the actions which lead to the incident.

The robotic hotel domain therefore presents an ideal environment for testing explainable robotics systems both in simulation and in the real world. The primary reasons for the domain choice are summarised as:

- It presents a variety of interesting challenges for robotics which generalise well to other similar domains.
- It is highly demanding for explanation generation due to the large variety of possible questions that may be posed at significantly different levels of abstraction.
- The domain dynamics are well understood and easily expandable, such that they can be encoded easily in simulation both for a small and large hotels.

7.2 Previous Work regarding Robotics in Hospitality

There has been significant research regarding the use of robotics in hospitality [Ivanov et al. 2019]. This includes long term assessments of HRI between guests and service robots deployed in hotel domains and how they impact upon the comfort of guests [Pinillos et al. 2016, Rodriguez-Lizundia et al. 2015, Zalama et al. 2014]. Hotels have been identified as a complex, yet highly commercially viable application of robotics [Ivanov and Webster 2017a]. Some researchers have emphasised that hospitality companies are bound to eventually deliver the majority of their services through robots [Ivanov 2019]. Indeed it can be seen that significant commercial interest exists for robotics in hospitality, with the worlds first robotic hotel "Henn-na hotel" opening in 2015 in Japan [Osawa et al. 2017], with multiple other similar hotels planned to open in 2019. However, the Henn-na hotel focuses on the idea of creating a "unique experience" for consumers where the existence of robots serves mostly as a novelty to attract more guests. Although they also claim a significant reduction in operating costs by employing the use of robots rather than large numbers of human workers.

A recent study [Ivanov and Webster 2017b] has made suggestions to hospitality companies to make hotels more "robot inclusive" (the extent to which the design of the domain supports reliable operation of robots [Tan et al. 2016]) to make deployment of robots easier and more financially viable. They describe how less robot inclusive domains require more intelligent robots and vice versa, using hospitality as an example. However, creation of sufficiently robot inclusive environments requires a bespoke design for each such domain which will not generalise to other, even very similar, domains and will cause major issues for engineers and owners resulting from greater design complexity and build costs. This therefore seems to be a crude and unrealistic solution. Leading to the hypothesis that the more beneficial solution is to design robots to be capable of dealing with the true nature of complex real world human-robot collaborative domains.

Appendices

A Action Languages

There have been many approaches towards reasoning about action effects and state transitions in robotics domains. One of the most popular of which are *action languages* [Gelfond and Lifschitz 1998], also known as *planning languages*. Action languages are formal models of parts of natural language used for describing *transition diagrams* of dynamic systems [Gelfond and Lifschitz 1998]. Action languages are used in robotics to model, amoungst other things, the effects and preconditions of a robot's actions. Many have been developed and used successfully for automated planning. Recent action languages are often viewed as high level notations of Answer Set Programs (ASP) [Babb 2015].

A.1 Transition Diagrams

In action languages a dynamic domain defined by a *system description* SD is modelled by a *transition diagram* $\tau(SD)$ [Gelfond and Kahl 2014]. A transition diagram is a directed graph whose nodes define possible states of the domain and whose arcs are labelled by actions, the effects of which can change the state between those connected by that arc. A single state transition of a transition diagram can be described by $\langle \sigma_0, \{a_1, ..., a_k\}, \sigma_1 \rangle$, where $\{a_1, ..., a_k\}$ is a arbitrarily large set of actions which if executed simultaneously in a state σ_0 would result in a state σ_1. This guarantees that the representation is *Markovian*, whereby the effect of an action depends only on the current state and knowledge of any past states would carry no additional information about their effects. A transition diagram T is called *deterministic* if for every state σ and set of actions $\{a_1, ..., a_k\}$ there is at most one state σ' such that $\langle \sigma_0, \{a_1, ..., a_k\}, \sigma' \rangle$. A path $\langle \sigma_0, a_0, \sigma_1, ..., a_{n-1}, \sigma_n \rangle$ of the transition diagram $\tau(SD)$ represents a possible *trajectory* of the system from its initial state σ_0 to its final goal state σ_n. The transition diagram of a dynamic system contains within it all possible trajectories of that system. For a planning problem, a trajectory of $\tau(SD)$ represents a complete plan that satisfies every goal.

A.2 Review of Action Languages

The first action language, called STRIPS, [Fikes and Nilsson 1971] was developed to deal with simple robotic planning problems such as performing navigation and modelling the position of domain entities over time. STRIPS was able to reason about the effects and preconditions of a robot's available control actions in order to generate plans and update the robot's internal belief of the domain state during execution. It did this using first-order predicate calculus to deal with the large number of facts and relations required for a robotic planning problem. STRIPS was successfully used on the infamous "Shakey Robot" in 1966-1972. This was the worlds first ever intelligent mobile robot [SHAKEY 1972]. This illustrated the potential power of action languages and paved the way their continued use in robotics applications.

Action language ADL [Pednault 1989] was later developed which addressed and solved a major limitation of STRIPS operators. This was that all consequences of every possible control action in all possible states had to be listed. ADL instead allowed context specific action effects, substantially reducing the number of rules needed to define domain dynamics. A significant issue which troubled early action languages was the *frame problem*, which occurs from the need to specify what has not changed over time. It can be solved by finding a representation of the *inertial axiom*, a default stating that things usually remain the same unless they are forced to change. Importantly, both STRIPS and ADL solve the frame problem because they are able to determine which domain properties are not affected by the execution of a particular action and should therefore keep their original value in the resulting state [Downs et al. 1994].

The Planning Domain Definition Language (PDDL) [Ghallab et al. 1998] was then produced in an attempt to standardise action languages. PDDL is generally considered the standard formalisms for modelling planning problems. The original PDDL has since has many extensions, including PDDL 2.1 which allows modelling of mixed discrete-continuous dynamics for richer temporal planning [Fox and Long 2003] and has recently been employed in logic based architectures [Batusov and Soutchanski 2019] including combined Constraint Logic and Answer Set Progranning (CLASP) [Banbara et al. 2017, Balduccini et al. 2018]. The Action Notation Markup Language (ANML) [Smith et al. 2008] also builds upon PDDL to support modelling of temporal constraints, resource usage, and generative and HTN planning.

Action languages B (section 5 of [Gelfond and Lifschitz 1998]), C [Giunchiglia and Lifschitz 1998], and $C+$ (section 4 of [Giunchiglia et al. 2004]), were then introduced which further improved upon STRIPS and ADL by allowing definitions of the indirect effects of actions by describing the dependency between *fluents* (domain properties that can change with time) and thus solved the *ramification problem*. Action language B allows recursive definitions, which play an important role in reasoning about actions and describing transitive relations between fluents. Like STRIPS and ADL the frame problem is solved in B by incorporating the inertial axiom. However, it is difficult to describe fluents whose temporal behaviour is not inertial in B. Action languages C and its extension $C+$ attempt to solve this

issue by allowing inertial axioms to be freely included or not included, thus allowing other assumptions about the temporal behaviour of fluents to be postulated. On the other hand, recursive definitions cannot be so easily expressed in C and $C+$. Action language BC [Lee et al. 2013], its extended version $BC+$ [Babb 2015] and its probabilistic extension [Lee and Wang 2018] have also been developed in an attempt to combine the advantageous features of B and $C+$ by including *default rules* [Reiter 1980] which allow one to derive the conclusion of a rule from the premise if its justification can be assumed consistently. These default rules allow *causal laws* (describing action effects) which specify default values of fluents at arbitrary time steps. These languages are designed specifically to take advantage of powerful constructs provided by modern ASP solvers such as choice rules, aggregates, and abstract constraint atoms.

Action language AL_d [Gelfond and Inclezan 2009, Gelfond and Inclezan 2013] improved upon its predecessor B by making a distinction between *inertial fluents* (also called *basic fluents* in the literature) and *defined fluents*. Inertial fluents obey the laws of inertia and can be directly changed by actions, defined fluents on the other hand do not obey inertial laws and cannot be directly changed by actions, instead they are defined in terms of other fluents. AL_d also introduced the concept of a *fluent dependency graph* which formally describes the dependency between fluents in a system. The fluent dependency graph (page 170 [Gelfond and Kahl 2014]) allows powerful assertions to be made about the validity of a system description, such as *weak acyclicity* and *well-foundedness*. AL_d was designed to be inherently well defined under answer set semantics. AL_d has been extended by [Sridharan et al. 2019] supporting many additional features. Non-boolean fluents allow more compact modelling of system descriptions. Non-deterministic casual laws describe the stochastic effects of an agent's actions. Basic fluents are partitioned into *basic physical* and *basic knowledge* fluents, along with a similar partitioning of actions into *physical actions* and *knowledge producing actions*. Finally, the observability of physical fluents is represented explicitly. This describes whether their value can be directly observed by knowledge producing actions or indirectly inferred by inferring their value from other knowledge.

B Hierarchical Task Network Planning

Hierarchical Task Networks (HTNs) are a planning paradigm in which the objective is not to achieve some goals, but rather to preform some set of tasks [Erol et al. 1994, Ghallab et al. 2004]. A HTN contains primitive operators similar to those used in classical planning but also a set of *methods* which may be used to decompose tasks into a sequence of sub-tasks. The central idea is that a task is recursively decomposed into smaller and smaller sub-tasks tasks until only a sequence of primitive, directly executable, tasks remain. HTN planning is considerably more expressive than classical planning because classical planning cannot express undecidable planning problems [Ghallab et al. 2004]. However, a common criticism of HTN planning is its brittleness caused by the critical reliance on their methods being complete and correct. This means that HTNs generally require comprehensive prior knowledge of the domain and its dynamics.

- A HTN domain is defined by the tuple $D = \langle O, M \rangle$,
- a task network by the tuple $W = \langle U, C \rangle$,
- and a HTN planning problem by the tuple $P = \langle s_0, W_0, D \rangle$.

Where O is a set of operators (primitive and non-primitive tasks), M is a set of methods, U is a set of task nodes, C is a set of ordering constraints over M, s_0 is the initial state and W_0 is the initial task network. The output is a task network containing only a sequence of primitive directly executable tasks. This is obtained by recursively decomposing the tasks in W_0 using methods from M. Each method $m \in M$ accomplishes a single non-primitive task $task(m)$. m contains a set of sub-tasks $subtasks(m)$ and order constraints $constr(m)$ over these tasks. If these sub-tasks are completed in an order which satisfies these constraints then $task(m)$ is complete. Intuitively, the final task network is obtained from the initial by replacing tasks from W_0 with sub-tasks from methods in M until only primitive tasks remain. The full implementation of HTN planning allows arbitrary ordering constrains over task networks including precedence (equivalent to edges of a task network) and before constraints (task preconditions), see [Ghallab et al. 2004] for details.

HTNs have been used for many planning problems and are particularly popular in video games. Many games developers argue the HTN planning producing more natural and human-like agent behaviour [Kelly et al. 2007, Kelly et al. 2008]. Applications of HTNs in the games industry includes Minecraft [Wichlacz et al. 2019] and Guerilla Games' Horizon Zero Dawn which simulates the behaviour of teams of multiple robots with heterogeneous capabilities [Guerrilla 2017]. HTN planning is relatively simple to implement and as such there is a wide variety of efficient HTN solvers available including the free-ware SHOP [SHOP 2019], PyHop [Nau 2019] and PyHTN solver [Shafranov 2019] which is based on the HTN domain descriptions used by Guerilla Games. There also exists an extension of HTNs called Hierarchical Goal Networks (HGNs) which supports goal-reasoning capabilities, goal-based plan repair, planning with incomplete models, and temporal and cost-aware planning [Shivashankar et al. 2013, Shivashankar 2015, Alford et al. 2016]. HGNs are significantly more complex than standard HTNs but are both more expressive and less brittle. There has also been work towards autonomous construction of HTNs for human-robot collaboration [Hayes and Scassellati 2016].

C Answer Set Programming

Answer Set Programming (ASP) is a well known declarative programming paradigm that has established itself as a prominent technique for Knowledge Representation and Reasoning (KRR). ASP stands as one of the most effective axiomatic logic based approaches to artificial intelligence due to its ability to precisely capture deafeasible reasoning, making it a powerful tool for representing and reasoning with commonsense knowledge. Its elaboration tolerance makes it particularly effective for knowledge intensive combinatorial optimization problems, such as planning, scheduling and diagnostics [Kaminski et al. 2017]. There has been many applications of ASP to date [Esra Erdem 2016, Grasso et al. 2013] such as; coordinating teams of Unmanned Aerial Vehicles (UAVs) [Balduccini et al. 2014], planning and monitoring systems for housekeeping robots [Aker et al. 2012], and an ASP extension of the popular Robot Operating System (ROS) [Quigley et al. 2009, Andres et al. 2013]. Other application areas of ASP include robotics [Erdem and Patoglu 2018], healthcare [Dodaro et al. 2018, Erdem et al. 2011, Gebser et al. 2010] and industry [Falkner et al. 2018].

The successfulness of ASP is generally attributed to the versatility, expressiveness and readability of its modelling language and to the widespread availability of highly efficient ASP solvers [Gebser et al. 2018]. ASP is defined by *stable model semantics* [Gelfond and Lifschitz 1988] and has its roots in logic programming and non-monotonic logic [Gelfond and Lifschitz 1991]. It is closely related to formalisms such as Propositional Satisfiability (SAT), Satisfiability Modulo Theories (SMT), Quantified Boolean Formulas (QBF) and Constraint Programming (CP) [Gebser et al. 2017].

The central concept of ASP is to represent a problem as a *program*, whose *models*, called *answer sets* correspond to the possible solutions to that problem. Where the problem is formulated by a set of logical *axioms* which define its nature. The main component of an ASP program is the *rule*, which consists of a *head* and *body* of the form shown in rule 1.

$$a_0 \mid ... \mid a_m \leftarrow a_{m+1}\ ,\ ...\ ,\ a_n, not\ a_{n+1}\ ,\ ...\ ,\ not\ a_o. \tag{1}$$

Where: a is an *atom* of the form $p(\bar{t})$ or its classical negation $-p(\bar{t})$, p is a *predicate symbol* and $\bar{t}$ is a vector of terms defining the parameters of p composed of function symbols and variables. Atoms a_1 to a_m are called head atoms, a_{m+1} to a_n and $not\ a_{n+1}$ to $not\ a_o$ are called positive and negative body literals, respectively. The logical connective *not* stands for *negation as failure* (also known as *default negation*), as such a negative body literal of the form $not\ a$ is often referred to as a *naf-literal*. In contrast to *classical negation*, which stands for "a is believed to be false", negation as failure stands for "a is not believed to be true", i.e. a can take three possible truth values, *true*, *false* or *unknown*. The body of a rule is a logical conjunction, possibly involving negation, and the head is either an atomic formula or a logical disjunction. An ASP program is a collection of such rules. Intuitively, a rule functions as such, if the body of a rule is satisfied then so must its head. An answer set of an ASP program is a set of head atoms which simultaneously satisfy all rules of the program. If an ASP program has at least one answer set it is said to be *satisfiable*, else it is *unsatisfiable*.

There are four most fundamental types of rule. A rule with exactly one atom in the head and at least one literal in the body is called a *normal* rule. If the body if a normal rule is satisfied by the answer set of the program, then the atom in the head must be in the answer set. A rule with exactly one atom in the head and no body is called a *fact*, all atoms occurring in facts must be in all answer sets of the program. A rule with no head and at least one literal in the body is called an *integrity constraint*, no answer set of the program may satisfy its body, because the head can never be satisfied. A rule with multiple atoms in the head and any number of literals in the body is called a *disjunctive* rule, disjunctive rules are similar to normal rules, the head is satisfied if exactly one of the atoms in the head occurs in the answer set of the program. A rule with at least one atom in the head, enclosed in braces, is called a *choice* rule, these are again similar to normal rules, except they are satisfied if any, none or all of the head atoms are in the answer set of the program. Choice rules can be preceded by the aggregate statements such as $\#count$ or $\#sum$ to express *cardinality constraints* and *weight constraints* respectively [Dell'Armi et al. 2003, Calimeri et al. 2005]. For greater details of the input language of ASP the reader should refer to the ASP-core-2 language standard [Francesco Calimeri 2012].

Two special types of rule are *weak constraints* and *consistency restoring* (CR-rules) rules [Balduccini and Gelfond 2003]. CR-rules are encoded via weak constraints and a type of choice rule known as a generator rule. The process of doing this is described in algorithm 1 of [Balai et al. 2013a] and involves reducing the rule to a minimization problem. A CR-rule states that if the body of the rule is satisfied, then the head may be satisfied only under the condition that not doing so would result in program inconsistency. Thus, the resulting answer set of a CR-Prolog program should contain a minimal number of occurrences of head atoms entailed only from CR-rules. Of particular interest is the ability for CR-rules to generate indirect exceptions to default knowledge. This ability is notable for robotics applications as it allows non-monotonic revision of assumptions made about the initial state of a domain in the presence of incomplete explicit knowledge. CR-rules also provide the ability to preform abductive reasoning to explain the cause of unexpected observations that contradict a robot's beliefs by hypothesising the occurrence of unobserved exogenous actions.

C.1 ASP Systems

ASP systems generally consist of two distinct parts, the *grounder* and the *solver* [Gebser et al. 2018]. The prior of these takes an ASP program containing *non-ground rules* and returns the equivalent program containing only *ground rules*, where an expression is said to be ground, if it contains no variables [Kaminski et al. 2017]. The latter takes a ground program and *solves* it to produce one or more *answer sets* which represent the possible solutions to that program.

C.2 Grounders

Grounding is the process of replacing all variables with all possible combinations of the corresponding constants of the same sort. The process typically produces a ground program of polynomial to exponential size with respect to the original non-ground input program. Modern grounders employ various techniques for producing ground programs of minimal size, such as *simplification*, whilst still ensuring the semantics of the original program are preserved. One of the major problems that grounders must deal with is *unsafe rules*. Unsafe rules are problematic as they can result in existing program rules behaving erroneously after addition of other seemingly unrelated rules, see [Balai et al. 2013a].

One of the first grounders was *Lparse* [Syrjänen 1998]. Programs passed to *Lparse* must conform to it's ω*-restrictedness* condition [Syrjänen 2001]. This demands that each variable in a rule must occur in a positive body literal of that rule which is not mutually recursive with its head and is not unstratified and not dependent upon an unstratified predicate. Another early grounder was *Gringo* series 1 [Gebser et al. 2007]. ASP programs passed to *Gringo* must conform to it's λ*-restrictedness* condition. This condition extends ω*-restrictedness* to deal with recursion, guaranteeing a finite sized ground program. Later the *DLV* [Eiter et al. 1998] grounder and *Gringo* series 3 (and above) [Gebser et al. 2011b, Gebser et al. 2015a] were introduced. These grounders impose the simpler condition of *variable safety* which requires only that every variable in a rule appears in at least one positive body literal of that rule and yet still ensures rule safety.

C.3 Solvers

Solving ASP programs to compute answer sets is founded upon the concepts of *Propositional Satisfiability* (SAT) and *Boolean Constraint* solving. SAT solving is based upon the classical Davis-Putnam-Logemann-Loveland (DPLL) [Davis et al. 1962] procedure but current modern solvers now employ conflict-driven solving procedures [Gebser et al. 2009] based on Conflict-Driven Clause Learning (CDCL) [Marques-Silva and Sakallah 1999].

The first ASP solvers were the *DLV* [Eiter and Faber 2000, Leone et al. 2006] and *Smodels* [Niemelä et al. 2000, Simons et al. 2002] solvers and were based on the DPLL procedure adjusted for ASP solving. Recent conflict-driven ASP solvers include *Clasp* [Gebser et al. 2015c] and *WASP* [Alviano et al. 2013]. *Clasp* series 1 [Gebser et al. 2012a] was the first such solver, featuring conflict-driven learning and back-jumping. *Clasp* series 2 [Gebser et al. 2012b] extended this to support parallel search via shared memory multi-threading. *Clasp* series 3 [Gebser et al. 2015c] further expanded upon its predecessors by supporting parallel processing of disjunctive logic programs, prioritised parallel optimisation, integration of a wide range of aggregate, heuristic and meta statements, and embedded Python scripts.

There are also solvers for cluster based solving [Gebser et al. 2011a], GPU based solving [Dovier et al. 2015] and solvers for ASP variants P-Log [Balai et al. 2019], CR-Prolog [Balduccini 2007] and CASP [Balduccini et al. 2018].

C.4 Combined Systems

The earliest combined system was *Lparse* coupled with *Smodels*, they provided a basic C++ library, but required you to "manually" pipe the output of the grounder to the solver. Unfortunately, both *Lparse* and *Smodels* are now deprecated as they no longer conform to the ASP-Core-2 language standard and do not support most new ASP features.

One of the most successful modern ASP systems [Gebser et al. 2017] is *Clingo* series 5 [Gebser et al. 2014]. The Clingo system integrates the *Gringo* series 5 grounder and the *Clasp* series 3 solver. Clingo provides an API with bindings in C, C++, Lua and Python, giving control over the grounding and solving processes [Gebser et al. 2019a, Kaminski et al. 2017]. The API provides powerful features such as incremental [Gebser et al. 2008] and multi-shot solving [Gebser et al. 2015d, Gebser et al. 2019c]. These are extremely powerful techniques for automated planning as they allow parts of the program to be iterative re-grounded with respect to a successively increasing planning horizon. Clingo removes the need to completely re-ground programs when new statements are added by reusing ground rules from previous solver calls [Gebser et al. 2015b]. This is particularly powerful for mobile robot planning applications because such a robot will usually commit many state observations to the program throughout plan execution.

The *SPARC* system for sorted ASP solving has also been developed [Balai et al. 2013b, Balai et al. 2013a, Balai 2018, Balai 2019b]. The SPARC system provides a high level syntax for ASP with the ability to explicitly declare sort definitions on top of the normal language. It is able to translate its syntax into Clingo's standard language for solving.

D Probabilistic Planning

Probabilistic reasoning paradigms, such as *probabilistic sequential decision making algorithms* and *probabilistic graphical models of uncertainty*, have proved to be powerful techniques for maximising long term rewards during planning of sensing and actuation in partially observable and stochastic domains by formally modelling the non-determinism of state transitions and sensor measurements [Kaelbling et al. 1998]. These include many powerful algorithms for recursive state estimation, localisation, mapping, planning and control [Thrun et al. 2005, Koller and Friedman 2009].

D.1 Probabilistic Policies

In classical planning, problems are usually formulated as a graph search problem in which an *optimal plan* consists of a deterministic sequence of actions, that if executed in order will achieve the goal from the given starting state. However, real-world domains are stochastic in nature. This is such that, actions do not have deterministic effects, instead there is a probability distribution over the range of all possible effects that may occur from the execution of a given action. The conditional probability distribution $p(s'|s,a)$ is the probability that an action a transitions from state s to state $s^{'}$. It is therefore no longer sufficient to restrict ourselves to computing a single sequence of actions because each action has multiple possible outcomes. Thus arises the need to know, for every possible future state that may occur, what the "best" action would be. Probabilistic planners can explicitly represent and reason with uncertainty in sensing and actuation using the calculus of probability theory to compute an *optimal policy* $\pi^* : s \rightarrow a$. A policy is a naive mapping between state-action pairs which defines; for each possible state, the optimal action to take in that state. An optimal policy is one that maximizes the expected utility (expected sum of future rewards) if followed, and defines a *rational reflex agent*.

D.2 Markov Decision Processes

Markov Decision Process (MDPs) are a well understood approach to probabilistic policy generation for stochastic robotics domains. All MDPs assume that the state space is Markovian. The "standard" MDP makes the assumption of *full observability* in which the robot is able to determine the complete current domain state with absolute certainty. This is obviously an unrealistic assumption in almost all robotics domains of interest. The more general approach is the *Partially Observable Markov Decision Process* (POMDP) which relaxes this assumption [Kaelbling et al. 1998, Bai et al. 2014]. POMDPs are able to model the uncertainty in both sensing and actuation that exist in real-world robotics domains. POMDPs have been used in many applications such as; robotics, structural inspection and marketing [Cassandra 1998].

The POMDP approach is more complex than the standard MDP. However, many phenomena contribute to the problem of partial observability which motivate its requirement. The most common of which occurs in almost all domains of interest and is caused by the possibility of getting the same observation from multiple different locations. This means that the robot rarely obtains *uniquely identifying observations* and is thus unable to determine the state with certainty. The standard MDP uses a state space representation. POMDPs however, assume there exists some underlying MDP in the state space, which it cannot observe directly, and thus operates in the *belief space*. The belief space is the space of all possible belief distributions over the state space. The belief space is always continuous, even for a discrete state space. POMDPs compute the value function recursively using the Bellman equation over the belief space given in equation 2.

$$V^T(b) = \gamma \max_u \left[r(b,u) + \int V_{t-1}(b')\ p(b'|u,b)\ db' \right] \tag{2}$$

With $V_1(b) = \gamma \max_u E_x[r(x,u)]$. The control policy is obtained through the following:

$$\pi^T(b) = \gamma\ \underset{u}{argmax} \left[r(b,u) + \int V_{t-1}(b')\ p(b'|u,b)\ db' \right] \tag{3}$$

The optimal value function obtained from a policy given by equation 3 is always piece-wise linear and convex. This is such that, when the robot has a high certainty of the current state, actuation is favoured, whereas when the robot is uncertain, sensing is favoured. A POMDP is defined by the 7-tuple $(S, A, T, R, \Omega, O, \gamma)$, where; S is the set of states, A the set of available control actions, T the set of state transition probabilities $P(x|u,x')$, R the reward function $R : S \times A \rightarrow r$, Ω the set of possible observations, O the set of observation functions $P(z|x)$ and $\gamma : [0,1]$ the discount factor. There are two important factors to consider; the *reward function* is a state-action mapping defined by the designer, that tells the robot, for each possible state-action pair, the immediate reward it gains from taking that action in that state. This is important because the reward function defines the nature of the optimal policy. The *discount factor* is important as it allows the designer to tell the robot, how far into the future it should look when computing the value of a state-action pair. A discount factor of 0 defines a greedy agent which will consider only immediate rewards, and a discount factor of 1 defines an agent which considers all future rewards, into the infinite horizon, as equally valuable.

E Nature of Explanations

A recent study [Miller et al. 2017] has indicated that most work on XAI has only used the researchers' own intuition of what constitutes an effective explanation and that researchers in AI and robotics should exploit the large bodies of research from the social sciences regarding how to generate explanations that properly satisfy the needs of the actual intended target users. This section reviews these bodies of research and identifies the key features human explanation.

E.1 Models of Explanation

Grice's maxims [Grice et al. 1975] are a formal model of the human cooperative conversational process. He states a general principle that should be adhered to during a conversation; "Make your conversational contribution as much as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged" (page 45 [Grice et al. 1975]). He defines four maxims that should be obeyed to achieve this principle:

1. *Quantity*: Make your contribution at least as informative as is required and no more.
2. *Quality*: Make your contribution one that is true, do not say what you lack adequate evidence to support.
3. *Relation*: Only provide information that is related to the conversation.
4. *Manner*: Be perspicuous; Avoid obscurity and ambiguity, be brief and orderly.

The category of quantity requires explanations be complete, and both prevents wasting of time and reduces the chance of confusion through "overinformativeness". The category of quality requires explanations be sound, accurate and do not mislead the user. The category of relation is simply a strategy for achieving the previous maxims. The final category of manner is related to how an explanation is vocalised (or presented) forming part of the post processing of an explanation and is therefore not strictly relevant to this thesis. Miller [Miller 2019] notes there are certain conditions under which these maxims can be violated such as irony or metaphor. Hilton [Hilton 1990] argues that because explanations are made as part of conversations then explanations human-robot collaboration should follow these maxims.

Aristotle's Four Causes model can be employed when providing answers to why–questions [Hankinson 2001]:

1. *Material*: The substance or material of which something is made.
2. *Formal*: The form or properties of something that make it what it is.
3. *Efficient*: The proximal mechanisms of the cause something to change.
4. *Final*: The end or goal of something.

This model may be useful for analysing the type of question a robot may receive. In an explainable robot the material descries the underlying computations made my the robot, the formal describes the decision making model itself, the efficient describes the underlying reasoning behind decision making and the final level describes its goals and intentions.

E.2 The Causal Model of Explanation

The idea that explanations describing our behaviour should state causes behind which decisions are made (i.e. we usually state our reasons for taking particular actions) is well documented in the social sciences [Josephson and Josephson 1996, Hilton 1996, Hankinson 2001, Lombrozo 2006, Lombrozo 2010]. Lipton [Lipton 1990] states that "According to a causal model of explanation, we explain phenomena by giving their causes or, where the phenomena are themselves causal regularities, we explain them by giving a mechanism linking cause and effect.". For explainable agency this means that for an explanation to be useful, a human must be able to attribute the robot's decision making to its underlying reasoning directly from the information given in an explanation [Doran et al. 2017, Abdul et al. 2018]. This process should be intuitive, unambiguous and possible even for a lay-person with little to no understanding of a robot's decision making processes or the domain dynamics. Miller [Miller 2019] argues that the use of statistical relationships to explain events is unsatisfying. For robotics this means that producing explanations that cite only probabilistic representations of uncertainty are undesirable unless accompanied by a related causal explanation [Josephson and Josephson 1996].

Kulesza et al. found that the best explanations are both sound and complete, such that all underlying causes of an event or property are identified [Kulesza et al. 2013]. They identify a number of important issues for explanation generation (1) Complete but unsound explanations reduce trust, (2) Sound but incomplete explanations result in more requests for clarification, (3) Completeness is more important than soundedness. They compiled their results in three principles for explainability; (1) Be sound, (2) Be complete, (3) Don't overwhelm [Kulesza et al. 2015]. Although their research focused on debugging machine learning models, their findings have significant meaning to explainable agency for

human-robot collaboration. Namely, since principles 1 and 2 are at odds with 3, careful consideration must be made in the design of explanation generation systems such that they be able to present complete and correct information that fully answers a query without being overly verbose. The important insight here is that whilst it is critical to explain all underlying causes, this does not mean one has to explain all *possible causes*, only the "best" and most plausible causes are needed. Lombrozo found that people usually consider the best explanation to be the simplest even if it is less likely than a complex one unless significant probabilistic evidence is given to support the complex one [Lombrozo 2007].

Miller [Miller 2019] notes that "Causal attribution is not causal explanation [...] extracting a causal chain and displaying it to a person is causal attribution [...] causal attribution does not constitute giving an explanation [...] (because it is) not reasonable to expect a lay-person to be able to interpret a causal chain.". He maintains that most work on XAI simply extracts a full causal chain and displays it, which does not present a sufficient explanation for the majority of humans. It is thus important to focus the content of explanations to include only the most relevant information to a given query.

E.3 Constrastive Explanation

A large proportion of researchers from the social sciences state that all why–questions ask for contrastive explanations [Hilton and Slugoski 1986, Lipton 1990, Hilton 1990, Lombrozo 2012]. Constrastive explanations are such that the cause of an event is explained in contrast to some other event that did not occur. Such an explanation usually takes the form; "Why X and not Y?" [Miller 2019]. Where X is an event that actually occured and Y is some counterfactual hypothetical event that did not occur [Hilton 1990]. Lipton [Lipton 1990] defines X as the fact and Y as the foil in such an explanation. For robotics, this would suggest that humans are most likely to ask a robot why it took an action X rather than an action Y (that the human expected it to make) and the robot must justify its choice for X with respect to Y.

This is challenging because the foil in a question is often left implicit. The trivial solution in this case is to assume the questions is "Why X and not not X?", or in other words "Why X and not an alternative to X". Unfortunately, this is equivalent to providing all possible causes for X, i.e. the entire class of foils [Lipton 1990]. This is unhelpful and the robot must determine the true foil, possibly by asking the human to clarify or infer it using techniques such as eye gaze.

Lipton [Lipton 1990] states that it is significantly easier to generate an explanation in response to a constrastive question than a plain-fact question because they point towards the specific thing which the explainee wants explained. Exploiting the constrastive nature of explanations is thus valuable for explainable agency because they allow explanations to be focused on just the most relevant part of a robot's decision making model that the explainee does not understand. Hence a complete explanation can be presented without describing all possible causes. Miller [Miller 2019] argues that despite this, most existing work in XAI considers only contrastive questions but not contrastive explanations. This is such that they simply provide two complete explanations for the fact and foil [Mittelstadt et al. 2018]. Miller states that good explanations should not contain causes that explain both the fact and the foil. Lipton formalises this into the general *difference condition* which states that to construct relevant constrastive explanations it is best to state only the causal differences between X and not Y. Hesslow [Hesslow 1988] also argues that the best explanations identify the greatest number of differences between fact and foil, and should not include any causes that explain both the fact and foil.

E.4 The Abnormal Conditions Model

The abnormal conditions model proposed by [Hilton and Slugoski 1986, Hilton 1990, Hilton 1996] states that an event or property is abnormal if it is considered unusual, unexpected or cannot be understood from the perspective of a human observer. Van Bouwel and Weber [Van Bouwel and Weber 2002] argue that the abnormal conditions model is directly related to constrastive explanation. They state that the majority of contrastive questions are posed when humans observe abnormal events, whereas plain-fact questions are usually asked out of curiosity. This is backed up by Hesslow [Hesslow 1988] who states that the best explanations of an abnormal event or property are those that highlight the most causal differences between that abnormal event and the event that would normally have occured. This is fundamental because it shows that questions are most likely to be posed when a person expects an event Y but then observes an event X. For robotics, this means that a robot will most commonly have to explain itself when its behaviour may be considered abnormal. This may occur when a robot has to unexpectedly change its plans according to some unanticipated observation which contradicted its expectations or when it makes a decision a human usually would not.

E.5 Goal Based Explanation

Mcclure [McClure et al. 2003] argues that people always prefer explanations justifying actions to cite goals one hopes to achieve through those actions over other causes for that action. This is such that explanations should state why an action was taken with regard to how its effects facilitate the completion of ones goals. He argues that such explanations are more informative because they allow inference of how ones intentions lead to their actions [McClure 2002]. This is despite the fact that actions are governed by other causes such as *enabling conditions* [McClure and Hilton 1998].

References

[Abdul et al. 2018] Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., and Kankanhalli, M. (2018). Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, page 582. ACM.

[Adadi and Berrada 2018] Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160.

[Aker et al. 2012] Aker, E., Patoglu, V., and Erdem, E. (2012). Answer set programming for reasoning with semantic knowledge in collaborative housekeeping robotics. *IFAC Proceedings Volumes*, 45(22):77–83.

[Alford et al. 2016] Alford, R., Shivashankar, V., Roberts, M., Frank, J., and Aha, D. W. (2016). Hierarchical planning: Relating task and goal decomposition with task sharing. In *IJCAI*, pages 3022–3029.

[Alviano et al. 2013] Alviano, M., Dodaro, C., Faber, W., Leone, N., and Ricca, F. (2013). Wasp: a native asp solver based on constraint learning.

[Amir and Amir 2018] Amir, D. and Amir, O. (2018). Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1168–1176. International Foundation for Autonomous Agents and Multiagent Systems.

[Andersen and Hooker 1994] Andersen, K. A. and Hooker, J. N. (1994). Bayesian logic. *Decis. Support Syst.*, 11(2):191–210.

[Andras et al. 2018] Andras, P., Esterle, L., Guckert, M., Han, T. A., Lewis, P. R., Milanovic, K., Payne, T., Perret, C., Pitt, J., Powers, S. T., Urquhart, N., and Wells, S. (2018). Trusting intelligent machines: Deepening trust within socio-technical systems. *IEEE Technology and Society Magazine*, 37(4):76–83.

[Andres et al. 2013] Andres, B., Obermeier, P., Sabuncu, O., Schaub, T., and Rajaratnam, D. (2013). Rosoclingo: A ros package for asp-based robot control. *CoRR*, abs/1307.7398.

[Anjomshoae et al. 2019] Anjomshoae, S., Najjar, A., Calvaresi, D., and Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems.

[Babb 2015] Babb, J. (2015). Action language bc +.

[Bai et al. 2014] Bai, H., Hsu, D., and Lee, W. S. (2014). Integrated perception and planning in the continuous space: A pomdp approach. *Int. J. Rob. Res.*, 33(9).

[Balai 2018] Balai, E. (2018). *SPARC Manual*. Texas Tech University, 2500 Broadway Lubbock TX 79409 USA.

[Balai 2019a] Balai, E. (2019a). P-log 2.0. `https://github.com/iensen/plog2.0`.

[Balai 2019b] Balai, E. (2019b). Sparc. `https://github.com/iensen/sparc`.

[Balai et al. 2013a] Balai, E., Gelfond, M., and Zhang, Y. (2013a). Sparc - sorted asp with consistency restoring rules. *CoRR*, abs/1301.1386.

[Balai et al. 2013b] Balai, E., Gelfond, M., and Zhang, Y. (2013b). Towards answer set programming with sorts. In *LPNMR*.

[Balai et al. 2019] Balai, E., Gelfond, M., and Zhang, Y. (2019). P-log: refinement and a new coherency condition. *Annals of Mathematics and Artificial Intelligence*, pages 1–44.

[Balduccini 2007] Balduccini, M. (2007). cr-models: An inference engine for cr-prolog. In *LPNMR*, pages 18–30.

[Balduccini and Gelfond 2003] Balduccini, M. and Gelfond, M. (2003). Logic programs with consistency-restoring rules.

[Balduccini et al. 2018] Balduccini, M., Magazzeni, D., Maratea, M., and Leblanc, E. (2018). CASP solutions for planning in hybrid domains. *CoRR*, abs/1704.03574.

[Balduccini et al. 2014] Balduccini, M., Regli, W. C., and Nguyen, D. N. (2014). An asp-based architecture for autonomous uavs in dynamic environments: Progress report. *CoRR*, abs/1405.1124.

[Banbara et al. 2017] Banbara, M., Kaufmann, B., Ostrowski, M., and Schaub, T. (2017). Clingcon: The next generation. *TPLP*, 17:408–461.

[Baral 2003] Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets*. Cambridge University Press.

[Batusov and Soutchanski 2019] Batusov, V. and Soutchanski, M. (2019). A logical semantics for pddl+.

[Borgo et al. 2018] Borgo, R., Cashmore, M., and Magazzeni, D. (2018). Towards providing explanations for ai planner decisions. *CoRR*, abs/1810.06338.

[Calimeri et al. 2005] Calimeri, F., Fabery, W., Leone, N., and Perri, S. (2005). Declarative and computational properties of logic programs with aggregates. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 406–411, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Cashmore et al. 2015] Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtós, N., and Carreras, M. (2015). Rosplan: Planning in the robot operating system. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 2015:333–341.

[Cassandra 1998] Cassandra, A. R. (1998). A survey of pomdp applications. In *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, volume 1724.

[Chakraborti et al. 2019] Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith, D. E., and Kambhampati, S. (2019). Explicability? legibility? predictability? transparency? privacy?security? the emerging landscape of interpretable agent behavior. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 86–96.

[Chakraborti et al. 2017] Chakraborti, T., Sreedharan, S., and Kambhampati, S. (2017). Balancing explicability and explanation in human-aware planning. *CoRR*, abs/1708.00543.

[Chakraborti et al. 2018] Chakraborti, T., Sreedharan, S., and Kambhampati, S. (2018). Explicability versus explanations in human-aware planning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2180–2182. International Foundation for Autonomous Agents and Multiagent Systems.

[Colaco and Sridharan 2015] Colaco, Z. and Sridharan, M. (2015). What happened and why? a mixed architecture for planning and explanation generation in robotics. In *In Australasian Conference on Robotics and Automation (ACRA)*.

[Dannenhauer et al. 2018a] Dannenhauer, D., Floyd, M. W., Magazzeni, D., and Aha, D. W. (2018a). Explaining rebel behaviour in goal reasoning agents.

[Dannenhauer et al. 2018b] Dannenhauer, D., Floyd, M. W., Molineaux, M., and Aha, D. W. (2018b). Learning from exploration: Towards an explainable goal reasoning agent.

[DARPA 2016] DARPA (2016). Broad agency announcement: Explainable artificial intelligence (xai). `https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf`.

[Davis et al. 1962] Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397.

[Dell'Armi et al. 2003] Dell'Armi, T., Faber, W., Ielpa, G., Leone, N., and Pfeifer, G. (2003). Aggregate functions in dlv. In *Answer Set Programming*.

[Dietvorst et al. 2015] Dietvorst, B. J., Simmons, J. P., and Massey, C. (2015). Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144(1):114.

[Dodaro et al. 2018] Dodaro, C., Galatà, G., Maratea, M., and Porro, I. (2018). An overview of asp applications in the health-care domain. In *RiCeRcA@AI*IA*.

[Doran et al. 2017] Doran, D., Schulz, S., and Besold, T. R. (2017). What does explainable AI really mean? A new conceptualization of perspectives. *CoRR*, abs/1710.00794.

[Dovier et al. 2015] Dovier, A., Formisano, A., Pontelli, E., and Vella, F. (2015). Parallel execution of the asp computation - an investigation on gpus. *CEUR Workshop Proceedings*, 1433.

[Downs et al. 1994] Downs, J., Reichgelt, H., and Shadbolt, N. (1994). Automatic derivation of world update schemes. pages 237–242.

[Došilović et al. 2018] Došilović, F., Brcic, M., and Hlupic, N. (2018). Explainable artificial intelligence: A survey.

[Eiter and Faber 2000] Eiter, T. and Faber, W. R. (2000). Declarative problem-solving using the dlv system.

[Eiter et al. 1998] Eiter, T., Leone, N., Mateis, C., Pfeifer, G., and Scarcello, F. (1998). The kr system dlv: progress report, comparisons and benchmarks. In *KR 1998*.

[Erdem et al. 2011] Erdem, E., Erdogan, H., and Öztok, U. (2011). Bioquery-asp: Querying biomedical ontologies using answer set programming. In *RuleML America*.

[Erdem and Patoglu 2018] Erdem, E. and Patoglu, V. (2018). Applications of asp in robotics. *KI - Künstliche Intelligenz*, 32:143–149.

[Erol et al. 1994] Erol, K., Hendler, J., and Nau, D. S. (1994). Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128.

[Esra Erdem 2016] Esra Erdem, Michael Gelfond, N. L. (2016). Applications of answer set programming.

[EU 2017a] EU (2017a). Civil law rules on robotics - european parliament resolution of 16 february 2017 with recommendations to the commission on civil law rules on robotics (2015/2103(inl)). `http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+TA+P8-TA-2017-0051+0+DOC+PDF+V0//EN`.

[EU 2017b] EU (2017b). European parliament committee on legal affairs, "report with recommendations to the commission on civil law rules on robotics" (2015/2103(inl). `http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+REPORT+A8-2017-0005+0+DOC+PDF+V0//EN`.

[Falkner et al. 2018] Falkner, A., Friedrich, G., Schekotihin, K., Taupe, R., and Teppan, E. C. (2018). Industrial applications of answer set programming. *KI - Kunstliche Intelligenz*, 32(2):165–176.

[Fikes and Nilsson 1971] Fikes, R. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

[Fox and Long 2003] Fox, M. and Long, D. (2003). Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124.

[Fox et al. 2017] Fox, M., Long, D., and Magazzeni, D. (2017). Explainable planning.

[Francesco Calimeri 2012] Francesco Calimeri, Wolfgang Faber, M. G. G. I. R. K. T. K. N. L. F. R. T. S. (2012). Asp-core-2: Input language format.

[Gebser et al. 2010] Gebser, M., Guziolowski, C., Ivanchev, M., Schaub, T., Siegel, A., Thiele, S., and Veber, P. (2010). Repair and prediction (under inconsistency) in large biological networks with answer set programming.

[Gebser et al. 2015a] Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., and Schaub, T. (2015a). Abstract gringo. *Theory and Practice of Logic Programming*, 15(4-5):449–463.

[Gebser et al. 2015b] Gebser, M., Janhunen, T., Jost, H., Kaminski, R., and Schaub, T. (2015b). Asp solving for expanding universes. In *LPNMR*.

[Gebser et al. 2019a] Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S., and Wanko, P. (2019a). Clingo python module version 5.3.0. `https://potassco.org/clingo/python-api/current/clingo.html`.

[Gebser et al. 2019b] Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S., and Wanko, P. (2019b). *Potassco User Guide*. University of Potsdam, Am Neuen Palais 10, 14469 Potsdam, Germany, 2nd edition.

[Gebser et al. 2008] Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Thiele, S. (2008). Engineering an incremental asp solver. pages 190–205.

[Gebser et al. 2015c] Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., and Schaub, T. (2015c). Progress in clasp series 3. In *LPNMR*.

[Gebser et al. 2014] Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2014). Clingo = asp + control: Extended report. *CoRR*, abs/1405.3694.

[Gebser et al. 2019c] Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2019c). Multi-shot asp solving with clingo. *TPLP*, 19:27–82.

[Gebser et al. 2011a] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., and Schnor, B. (2011a). Cluster-based asp solving with claspar. volume 6645, pages 364–369.

[Gebser et al. 2011b] Gebser, M., Kaminski, R., König, A., and Schaub, T. (2011b). Advances in gringo series 3. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning*, LPNMR'11, pages 345–351, Berlin, Heidelberg. Springer-Verlag.

[Gebser et al. 2015d] Gebser, M., Kaminski, R., Obermeier, P., and Schaub, T. (2015d). Ricochet robots reloaded: A case-study in multi-shot asp solving. 9060.

[Gebser et al. 2009] Gebser, M., Kaufmann, B., and Schaub, T. (2009). The conflict-driven answer set solver clasp: Progress report. volume 5753.

[Gebser et al. 2012a] Gebser, M., Kaufmann, B., and Schaub, T. (2012a). Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89.

[Gebser et al. 2012b] Gebser, M., Kaufmann, B., and Schaub, T. (2012b). Multi-threaded ASP solving with clasp. *CoRR*, abs/1210.3265.

[Gebser et al. 2018] Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., and Schaub, T. (2018). Evaluation techniques and systems for answer set programming: a survey. In *IJCAI*.

[Gebser et al. 2017] Gebser, M., Maratea, M., and Ricca, F. (2017). The sixth answer set programming competition. *J. Artif. Int. Res.*, 60(1):41–95.

[Gebser et al. 2007] Gebser, M., Schaub, T., and Thiele, S. (2007). Gringo: A new grounder for answer set programming. volume 4483, pages 266–271.

[Gelfond and Inclezan 2009] Gelfond, M. and Inclezan, D. (2009). Yet another modular action language.

[Gelfond and Inclezan 2013] Gelfond, M. and Inclezan, D. (2013). Some properties of system descriptions of ald. *Journal of Applied Non-Classical Logics*, 23:105–120.

[Gelfond and Kahl 2014] Gelfond, M. and Kahl, Y. (2014). *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, New York, NY, USA.

[Gelfond and Lifschitz 1988] Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In Kowalski, R., Bowen, and Kenneth, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press.

[Gelfond and Lifschitz 1991] Gelfond, M. and Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385.

[Gelfond and Lifschitz 1998] Gelfond, M. and Lifschitz, V. (1998). Action languages. *Electronic Transactions on Artificial Intelligence*, 3:195–210.

[Ghallab et al. 1998] Ghallab, M., Knoblock, C., Wilkins, D., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D., Sun, Y., and Weld, D. (1998). Pddl - the planning domain definition language.

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

[Giunchiglia et al. 2004] Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., and Turner, H. (2004). Nonmonotonic causal theories. *Artif. Intell.*, 153(1-2):49–104.

[Giunchiglia and Lifschitz 1998] Giunchiglia, E. and Lifschitz, V. (1998). An action language based on causal explanation: Preliminary report. In *AAAI/IAAI*.

[Grasso et al. 2013] Grasso, G., Leone, N., and Ricca, F. (2013). Answer set programming: Language, applications and development tools. In Faber, W. and Lembo, D., editors, *Web Reasoning and Rule Systems*, pages 19–34, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Grice et al. 1975] Grice, H. P., Cole, P., Morgan, J., et al. (1975). Logic and conversation. *1975*, pages 41–58.

[Guerrilla 2017] Guerrilla (2017). `https://www.guerrilla-games.com/read/the-ai-of-horizon-zero-dawn`.

[Guidotti et al. 2018] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

[Hanheide et al. 2017] Hanheide, M., Göbelbecker, M., Horn, G. S., Pronobis, A., Sjöö, K., Aydemir, A., Jensfelt, P., Gretton, C., Dearden, R., Janícek, M., Zender, H., Kruijff, G.-J. M., Hawes, N., and Wyatt, J. L. (2017). Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 247:119–150.

[Hankinson 2001] Hankinson, R. J. (2001). *Cause and explanation in ancient Greek thought*. Oxford University Press.

[Hayes and Scassellati 2016] Hayes, B. and Scassellati, B. (2016). Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5476. IEEE.

[Hesslow 1988] Hesslow, G. (1988). The problem of causal selection. *Contemporary science and natural explanation: Commonsense conceptions of causality*, pages 11–32.

[Hilton 1990] Hilton, D. (1990). Conversational processes and causal explanation. *Psychological Bulletin*, 107:65–81.

[Hilton 1996] Hilton, D. J. (1996). Mental models and causal explanation: Judgements of probable cause and explanatory relevance. *Thinking & Reasoning*, 2(4):273–308.

[Hilton and Slugoski 1986] Hilton, D. J. and Slugoski, B. R. (1986). Knowledge-based causal attribution: The abnormal conditions focus model. *Psychological review*, 93(1):75.

[Ivanov 2019] Ivanov, S. (2019). Ultimate transformation: How will automation technologies disrupt the travel, tourism and hospitality industries?

[Ivanov et al. 2019] Ivanov, S., Gretzel, U., Berezina, K., Sigala, M., and Webster, C. (2019). Progress on robotics in hospitality and tourism: a review of the literature.

[Ivanov and Webster 2017a] Ivanov, S. and Webster, C. (2017a). Adoption of robots, artificial intelligence and service automation by travel, tourism and hospitality companies – a cost-benefit analysis.

[Ivanov and Webster 2017b] Ivanov, S. and Webster, C. (2017b). Designing robot-friendly hospitality facilities. In *Proceedings of the Scientific Conference, Tourism. Innovations. Strategies*, pages 74–81, Bourgas, Bulgaria.

[Jaumard et al. 2006] Jaumard, B., Fortin, A., Shahriar, I., and Sultana, R. (2006). First order probabilistic logic. pages 341 – 346.

[Josephson and Josephson 1996] Josephson, J. R. and Josephson, S. G. (1996). *Abductive inference: Computation, philosophy, technology*. Cambridge University Press.

[Kaelbling et al. 1998] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

[Kaminski et al. 2017] Kaminski, R., Schaub, T., and Wanko, P. (2017). *A Tutorial on Hybrid Answer Set Solving with clingo*, pages 167–203.

[Keil 2003] Keil, F. C. (2003). Folkscience: Coarse interpretations of a complex reality. *Trends in Cognitive Sciences*, 7(8):368–373.

[Kelly et al. 2007] Kelly, J.-P., Botea, A., Koenig, S., et al. (2007). Planning with hierarchical task networks in video games. In *Proceedings of the ICAPS-07 Workshop on Planning in Games*.

[Kelly et al. 2008] Kelly, J. P., Botea, A., Koenig, S., et al. (2008). Offline planning with hierarchical task networks in video games. In *AIIDE*.

[Khan et al. 2009] Khan, O. Z., Poupart, P., and Black, J. P. (2009). Minimal sufficient explanations for factored markov decision processes. In *Nineteenth International Conference on Automated Planning and Scheduling*.

[Koller and Friedman 2009] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

[Korpan and Epstein 2018] Korpan, R. and Epstein, S. L. (2018). Toward natural explanations for a robot's navigation plans.

[Krarup et al. 2019] Krarup, B., Cashmore, M., Magazzeni, D., and Miller, T. (2019). Model-based contrastive explanations for explainable planning.

[Kulesza et al. 2015] Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 126–137.

[Kulesza et al. 2013] Kulesza, T., Stumpf, S., Burnett, M. M., Yang, S., Kwan, I., and Wong, W.-K. (2013). Too much, too little, or just right? ways explanations impact end users' mental models. *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 3–10.

[Langley et al. 2017] Langley, P., Meadows, B., Sridharan, M., and Choi, D. (2017). Explainable agency for intelligent autonomous systems. In *AAAI*.

[Lee et al. 2013] Lee, J., Lifschitz, V., and Yang, F. (2013). Action language bc: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

[Lee and Wang 2018] Lee, J. and Wang, J. (2018). A probabilistic extension of action language bc+. *CoRR*, abs/1805.00634.

[Leone et al. 2006] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2006). The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7:499–562.

[Lifschitz 2008] Lifschitz, V. (2008). What is answer set programming? In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1594–1597.

[Lipton 1990] Lipton, P. (1990). Contrastive explanation. *Royal Institute of Philosophy Supplements*, 27:247–266.

[Lipton 2016] Lipton, Z. C. (2016). The mythos of model interpretability.

[Lison 2015] Lison, P. (2015). A hybrid approach to dialogue management based on probabilistic rules. *Comput. Speech Lang.*, 34(1):232–255.

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

[Lombrozo 2007] Lombrozo, T. (2007). Simplicity and probability in causal explanation. *Cognitive psychology*, 55(3):232–257.

[Lombrozo 2010] Lombrozo, T. (2010). Causal–explanatory pluralism: How intentions, functions, and mechanisms influence causal ascriptions. *Cognitive Psychology*, 61(4):303–332.

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

[Marques-Silva and Sakallah 1999] Marques-Silva, J. P. and Sakallah, K. A. (1999). Grasp: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521.

[McClure 2002] McClure, J. (2002). Goal-based explanations of actions and outcomes. *European review of social psychology*, 12(1):201–235.

[McClure and Hilton 1998] McClure, J. and Hilton, D. J. (1998). Are goals or preconditions better explanations? it depends on the question. *European Journal of Social Psychology*, 28(6):897–911.

[McClure et al. 2003] McClure, J. L., Sutton, R. M., and Hilton, D. J. (2003). The role of goal-based explanations. *Social judgments: Implicit and explicit processes*, 5.

[Meadows et al. 2016] Meadows, B., Sridharan, M., and Colaco, Z. (2016). Towards an explanation generation system for robots: Analysis and recommendations.

[Miller 2019] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38.

[Miller et al. 2017] Miller, T., Howe, P., and Sonenberg, L. (2017). Explainable ai: Beware of inmates running the asylum. *CoRR*, abs/1712.00547.

[Mittelstadt et al. 2018] Mittelstadt, B. D., Russell, C., and Wachter, S. (2018). Explaining explanations in AI. *CoRR*.

[Montavon et al. 2017] Montavon, G., Samek, W., and Müller, K. (2017). Methods for interpreting and understanding deep neural networks. *CoRR*.

[Nau 2019] Nau, D. S. (2019). Pyhop version 1.2.2. `https://bitbucket.org/dananau/pyhop/src/default/`.

[Niemelä et al. 2000] Niemelä, I., Simons, P., and Syrjänen, T. (2000). Smodels: A system for answer set programming. *CoRR*, cs.AI/0003033.

[Osawa et al. 2017] Osawa, H., Ema, A., Hattori, H., Akiya, N., Kanzaki, N., Kubo, A., Koyama, T., and Ichise, R. (2017). Analysis of robot hotel: Reconstruction of works with robots. In *26th IEEE International Symposium on Robot and Human Interactive Communication*, pages 219–223.

[Pednault 1989] Pednault, E. P. D. (1989). Adl: exploring the middle ground between strips and the situation calculus. In *KR 1989*.

[Pinillos et al. 2016] Pinillos, R., Marcos, S., Feliz, R., Zalama, E., and Gómez-García-Bermejo, J. (2016). Long-term assessment of a service robot in a hotel environment. *Robotics and Autonomous Systems*, 79:40–57.

[Poole 2000] Poole, D. (2000). Abducing through negation as failure: stable models within the independent choice logic. *J. Log. Program.*, 44:5–35.

[Quigley et al. 2009] Quigley, M., Gerkey, B. P., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system.

[Reiter 1980] Reiter, R. (1980). A logic for default reasoning. *Artif. Intell.*, 13:81–132.

[Ribeiro et al. 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *HLT-NAACL Demos*.

[Richardson and Domingos 2006] Richardson, M. and Domingos, P. (2006). Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

[Roberts et al. 2018] Roberts, M., Monteath, I., Sheh, R., Aha, D. W., Jampathom, P., Akins, K., Sydow, E., Shivashankar, V., and Sammut, C. (2018). What was i planning to do?

[Rocio Gomez and Riley 2018] Rocio Gomez, M. S. and Riley, H. (2018). Representing and reasoning with intentional actions on a robot. In *Workshop on Planning and Robotics at International Conference on Automated Planning and Scheduling*, Delft, The Netherlands.

[Rodriguez-Lizundia et al. 2015] Rodriguez-Lizundia, E., Marcos, S., Zalama, E., Gómez-García-Bermejo, J., and Gordaliza, A. (2015). A bellboy robot: Study of the effects of robot behaviour on user engagement and comfort. *Human-Computer Studies*, 82:83–95.

[Rudin 2019] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215.

[Russell and Norvig 2016] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[Sanner and Kersting 2010] Sanner, S. and Kersting, K. (2010). Symbolic dynamic programming for first-order pomdps. volume 2.

[Shafranov 2019] Shafranov, A. (2019). Pyhtn: Python htn planner. `https://github.com/mrceresa/PyHTN`.

[SHAKEY 1972] SHAKEY (1966-1972). Milestone-proposal: Shakey: The world's first mobile, intelligent robot, 1972. `http://ieeemilestones.ethw.org/Milestone-Proposal:Shakey:_The_World%E2%80%99s_First_Mobile,_Intelligent_Robot,_1972`.

[Shani et al. 2013] Shani, G., Pineau, J., and Kaplow, R. (2013). A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51.

[Sheh 2017] Sheh, R. (2017). "why did you do that?" explainable intelligent robots.

[Shivashankar 2015] Shivashankar, V. (2015). Hierarchical goal networks: Formalisms and algorithms for planning and acting.

[Shivashankar et al. 2013] Shivashankar, V., UMD EDU, R. A., Kuter, U., and Nau, D. (2013). Hierarchical goal networks and goal-driven autonomy: Going where ai planning meets goal reasoning. In *Goal Reasoning: Papers from the ACS Workshop*, page 95.

[SHOP 2019] SHOP (2019). Simple hierarchical ordered planner. `http://www.cs.umd.edu/projects/shop/index.html`.

[Simons et al. 2002] Simons, P., Niemelá, I., and Soininen, T. (2002). Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2).

[Smith et al. 2008] Smith, D. E., Frank, J., and Cushing, W. (2008). The anml language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

[Sreedharan et al. 2017] Sreedharan, S., Chakraborti, T., and Kambhampati, S. (2017). Balancing explicability and explanation in human-aware planning. *CoRR*, abs/1708.00543.

[Sreedharan et al. 2019] Sreedharan, S., Olmo, A., Mishra, A. P., and Kambhampati, S. (2019). Model-free model reconciliation. *CoRR*.

[Sridharan and Gelfond 2016] Sridharan, M. and Gelfond, M. (2016). Using knowledge representation and reasoning tools in the design of robots. In *KnowProS@IJCAI*.

[Sridharan et al. 2019] Sridharan, M., Gelfond, M., Zhang, S., and Wyatt, J. (2019). Reba: A refinement-based architecture for knowledge representation and reasoning in robotics. *Journal of Artificial Intelligence Research*, 65:87–180.

[Sridharan and Meadows 2019] Sridharan, M. and Meadows, B. (2019). A theory of explanations for human-robot collaboration.

[Sukkerd et al. 2018] Sukkerd, R., Simmons, R., and Garlan, D. (2018). Towards explainable multi-objective probabilistic planning. In *Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 19–25. ACM.

[Syrjänen 1998] Syrjänen, T. (1998). Implementation of local grounding for logic programs with stable model semantics.

[Syrjänen 2001] Syrjänen, T. (2001). Omega-restricted logic programs. pages 267–279.

[Tan et al. 2016] Tan, N., Mohan, R. E., and Watanabe, A. (2016). Toward a framework for robot-inclusive environments. *Automation in Construction*, 69:68–78.

[Thrun 2004] Thrun, S. (2004). Toward a framework for human-robot interaction. *Human-Computer Interaction*, 19(1):9–24.

[Thrun et al. 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

[Van Bouwel and Weber 2002] Van Bouwel, J. and Weber, E. (2002). Remote causes, bad explanations? *Journal for the Theory of Social Behaviour*, 32(4):437–449.

[Wachter et al. 2017a] Wachter, S., Mittelstadt, B., and Floridi, L. (2017a). Transparent, explainable, and accountable ai for robotics. *Science Robotics*, 2(6).

[Wachter et al. 2017b] Wachter, S., Mittelstadt, B., and Floridi, L. (2017b). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law, 2017*.

[Wang et al. 2016] Wang, N., Pynadath, D. V., and Hill, S. G. (2016). The impact of pomdp-generated explanations on trust and performance in human-robot teams. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pages 997–1005. International Foundation for Autonomous Agents and Multiagent Systems.

[Wichlacz et al. 2019] Wichlacz, J., Torralba, Á., and Hoffmann, J. (2019). Construction-planning models in minecraft.

[XAI 2017] XAI (2017). *IJCAI-17 Workshop on Explainable AI (XAI)*, volume 1, Melbourne, Australia.

[Zakershahrak et al. 2019] Zakershahrak, M., Gong, Z., and Zhang, Y. (2019). Online explanation generation for human-robot teaming. *CoRR*.

[Zalama et al. 2014] Zalama, E., Gómez-García-Bermejo, J., Marcos, S., Dominguez, S., Feliz, R., Pinillos, R., and López, J. (2014). Sacarino, a service robot in a hotel environment. *Advances in Intelligent Systems and Computing*, 253:3–14.

[Zhang et al. 2017] Zhang, S., Khandelwal, P., and Stone, P. (2017). Dynamically constructed (po)mdps for adaptive robot planning. In *AAAI*.

[Zhang et al. 2014] Zhang, S., Sridharan, M., Gelfond, M., and Wyatt, J. (2014). Towards an architecture for knowledge representation and reasoning in robotics. In *International conference on social robotics*, pages 400–410. Springer.

[Zhang et al. 2015] Zhang, S., Sridharan, M., and Wyatt, J. L. (2015). Mixed logical inference and probabilistic planning for robots in unreliable worlds. *IEEE Transactions on Robotics*, 31:699–713.

[Zhang and Stone 2015] Zhang, S. and Stone, P. (2015). Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot. In *AAAI Spring Symposia*.

[Zhang and Stone 2017] Zhang, S. and Stone, P. (2017). Integrated commonsense reasoning and probabilistic planning.