28th International Conference on
Automated Planning and Scheduling

June 24-–29, 2018, Delft, the Netherlands

XAIP 2018

Proceedings of the 1st Workshop on

Explainable Planning

Edited by:

Daniele Magazzeni, David Smith, Pat Langley, Susanne Biundo

Organization

Daniele Magazzeni
King's College London, UK

David Smith

Pat Langley
Institute for the Study of Learning and Expertise, USA

Susanne Biundo
Ulm University, Germany

Program Committee

- Susanne Biundo (Ulm University)
- John Bresina (NASA)
- Tathagata Chakraborti (Arizona State University)
- Dustin Dannenhauer (Naval Research Laboratory)
- Jeremy Frank (NASA)
- Pat Langley (University of Auckland)
- Daniele Magazzeni (King's College London)
- Matthew Molineaux (Knexus Research Corporation)
- Mark Roberts (Naval Research Laboratory)
- David Smith
- Siddharth Srivastava (ASU)
- Kartik Talamadupula (IBM)

Foreword

As AI is increasingly being adopted into application solutions, the challenge of supporting interaction with humans is becoming more apparent. Partly this is to support integrated working styles, in which humans and intelligent systems cooperate in problem-solving, but it is also a necessary step in the process of building trust as humans invest greater authority and responsibility in intelligent systems. Explainability poses challenges for many types of AI systems, including planning and scheduling (PS) systems. For example, how should a PS system justify that a plan or schedule is correct, or good, or respects supplied preferences? How can the PS system explain particular steps, ordering decisions, or resource choices? How can a PS system explain that no solution is possible, or what relaxations of the constraints would allow a solution? How can a PS system respond to questions like "what is the hard part?" or "why is this taking so long?". These are all difficult questions that can require analysis of plan or schedule structure, analysis of the goals, constraints, and preferences, and potentially hypothetical reasoning.

While the wider area of explainable AI (XAI) has been mostly focused on explaining machine/deep learning techniques, in the last two years explainable planning has received a growing attention, and a number of teams started to explore this research area. This first workshop on explainable planning is a demonstration of this. XAIP-18 has seen an excellent response in terms of submissions and participation. 13 papers have been accepted for oral presentation, and the program is completed by an invited talk. These proceedings show that there is an intense research activity around explainable planning and that explainable planning can play a key role and make important contributions to the research agenda on explainable AI.

We would like to thank the authors who submitted their papers and all workshop attendees. Moreover, we sincerely thank the program committee for their great work in reviewing the papers.

Daniele Magazzeni, David Smith, Pat Langley, Susanne Biundo
June 2018

Contents

Invited Talk: Relating XAI (Explainable AI) to XAIP (XAI Planning)

David W. Aha
Naval Research Laboratory; Navy Center for Applied Research in AI; Washington DC
david.aha@nrl.navy.mil

Abstract

The DARPA Explainable AI (XAI) program is a high-profile effort, among many, whose objective is to encourage research on AI systems whose models and decisions are more accessible and transparent to users. Yet the common focus of DARPA XAI's 11 projects is machine learning; it could have been called XML rather than XAI. Still, it is raising awareness that AI researchers need to collaborate with social scientists, and others, on the design and evaluation of XAI systems. This also applies broadly to other XAI efforts, including those of interest to the ICAPS community. In this talk, I'll summarize the objectives and status of DARPA XAI, emphasizing some topics of interest to XAIP. I'll also discuss/relate some work on XAIP that has appeared at the IJCAI-17 XAI Workshop, or will appear at the upcoming IJCAI/ECAI-18 XAI Workshop, which has a broad XAI focus (I.e., not limited to ML).

Human-Aware Planning Revisited: *A Tale of Three Models*

Tathagata Chakraborti and **Sarath Sreedharan** and **Subbarao Kambhampati**
Arizona State University, Tempe, AZ 85281 USA
{ tchakra2, ssreedh3, rao } @ asu.edu

Abstract

Human-aware planning requires an agent to be aware of the mental model of the humans, in addition to their physical or capability model. This not only allows an agent to envisage the desired roles of the human in a joint plan but also anticipate how its plan will be *perceived* by the latter. The human mental model becomes especially useful in the context of an explainable planning (XAIP) agent since an explanatory process cannot be a soliloquy, i.e. it must incorporate the human's beliefs and expectations of the planner. In this paper, we survey our recent efforts in this direction.

Cognitive AI teaming (Chakraborti *et al.* 2017a) requires a planner to perform argumentation over a set of models during the plan generation process. This is illustrated in Figure 1. Here, $\mathcal{M}^R$ is the model of the agent embodying the planner (e.g. a robot), and $\mathcal{M}^H$ is the model of the human in the loop. Further, $\mathcal{M}^R_h$ is the model the human thinks the robot has, and $\mathcal{M}^H_r$ is the model that the robot thinks the human has. Finally, $\tilde{\mathcal{M}}^R_h$ is the robot's approximation of $\mathcal{M}^R_h$; for the rest of the paper we will be using $\mathcal{M}^R_h$ to refer to both since, for all intents and purposes, this is all the robot has access to. Note that the *human mental model* $\mathcal{M}^R_h$ is in addition to the (robot's belief of the) *human model* $\mathcal{M}^H_r$ traditionally encountered in human-robot teaming (HRT) settings and is, in essence, the fundamental thesis of the recent works on *plan explanations* (Chakraborti *et al.* 2017b) and *explicable planning* (Zhang *et al.* 2017). The need for explicable planning or plan explanations occurs when the models – $\mathcal{M}^R$ and $\mathcal{M}^R_h$ – diverge so that the optimal plans in the respective models may not be the same and hence *optimal behavior of the robot in its own model is inexplicable to the human*. This is also true for discrepancies between $\mathcal{M}^H$ and $\mathcal{M}^H_r$ when the robot might reveal unrealistic expectations of the human in a joint plan.

An explainable planning (XAIP) agent (Fox *et al.* 2017; Langley *et al.* 2017; Weld and Bansal 2018) should be able to able to deal with such model differences and participate in explanatory dialog with the human such that both of them can be on the same page during a collaborative activity. This is referred to as *model reconciliation* (Chakraborti *et al.* 2017b) and forms the core of the explanatory process of an XAIP agent. In this paper, we look at the scope of problems engendered by this multi-model setting and describe the recent work in this direction. Specifically –

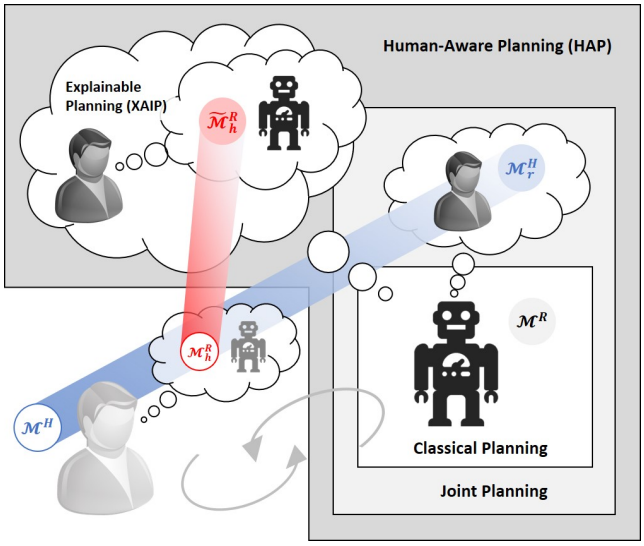


Figure 1: Argumentation over multiple models during the deliberative process of a human-aware planner (e.g. robot).

- We outline the scope of behaviors engendered by human-aware planning, including joint planning as studied in teaming using the *human model*, as well as explicable planning with the *human mental model*;
- We situate the plan explanation problem in the context of *perceived* inexplicability of the robot's plans or behaviors due to differences in these models;
  - We discuss how the plan explanation process can be seen as one of *model reconciliation* where $\mathcal{M}^R_h$ (and/or $\mathcal{M}^H_r$) is brought closer to $\mathcal{M}^R$ ($\mathcal{M}^H$);
  - We discuss how explicability and explanation costs can be traded off during plan generation;
  - We discuss how this process can be adapted to handle uncertainty or multiple humans in the loop;
  - We discuss results of a user study that testify to the usefulness of the model reconciliation process;
- We point to ongoing work in the space of abstractions and deception using the human mental model.

Background

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

This forms the classical definition of a planning problem (Russell and Norvig 2003) whose models are represented in the syntax of PDDL (McDermott *et al.* 1998). The solution to the planning problem is a sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. The cost of a plan π is given by $C(\pi, \mathcal{M}) = \sum_{a \in \pi} c_a$ if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; ∞ otherwise. The cheapest plan $\pi^* = \arg\min_{\pi} C(\pi, \mathcal{M})$ is the (cost) optimal plan with cost $C^*_{\mathcal{M}}$.

In previous work (Nguyen *et al.* 2017) we introduced an updated representation of planning problems in the form of *annotated* models to account for uncertainty or incompleteness over the definition of a planning model. In addition to the standard preconditions and effects associated with actions, it introduces the notion of *possible* preconditions and effects which may or may not be realized in practice.

An Incomplete (Annotated) Model is the tuple $\mathbb{M} = \langle \mathbb{D}, \mathbb{I}, \mathbb{G} \rangle$ with a domain $\mathbb{D} = \langle F, \mathbb{A} \rangle$ – where F is a finite set of fluents that define a state $s \subseteq F$, and $\mathbb{A}$ is a finite set of annotated actions – and annotated initial and goal states $\mathbb{I} = \langle \mathcal{I}^0, \mathcal{I}^+ \rangle$, $\mathbb{G} = \langle \mathcal{G}^0, \mathcal{G}^+ \rangle$; $\mathcal{I}^0, \mathcal{G}^0, \mathcal{I}^+, \mathcal{G}^+ \subseteq F$. Action $a \in \mathbb{A}$ is a tuple $\langle c_a, \mathit{pre}(a), \widetilde{\mathit{pre}}(a), \mathit{eff}^{\pm}(a), \widetilde{\mathit{eff}}^{\pm}(a) \rangle$ where c_a is the cost and, in addition to its *known* preconditions and add/delete effects $\mathit{pre}(a), \mathit{eff}^{\pm}(a), \subseteq F$ each action also contains *possible preconditions* $\widetilde{\mathit{pre}}(a) \subseteq F$ containing propositions that action a *might* need as preconditions, and *possible add (delete) effects* $\widetilde{\mathit{eff}}^{\pm}(a) \subseteq F$) containing propositions that the action a *might* add (delete, respectively) after execution. Similarly, $\mathcal{I}^0, \mathcal{G}^0$ (and $\mathcal{I}^+, \mathcal{G}^+$) are the known (and possible) parts of the initial and goal states.

Each possible condition $f \in \widetilde{\mathit{pre}}(a) \cup \widetilde{\mathit{eff}}^{\pm}(a)$ has a probability $p(f)$ associated with it denoting how likely it is to appear as a known condition in the ground truth model – i.e. $p(f)$ measures the confidence with which that condition has been learned. The sets of known and possible conditions in $\mathcal{M}$ are called $\mathbb{S}_k(\Gamma(\mathcal{M}))$ and $\mathbb{S}_p(\Gamma(\mathcal{M}))$. Here Γ is a mapping function that converts domain model conditions into propositions in a meta space (Chakraborti *et al.* 2017b).

An *instantiation* of an annotated model $\mathbb{M}$ is a classical planning model where a subset of the possible conditions have been realized – given by the tuple $inst(\mathbb{M}) = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$, initial and goal states $\mathcal{I} = \mathcal{I}^0 \cup \chi$; $\chi \subseteq \mathcal{I}^+$ and $\mathcal{G} = \mathcal{G}^0 \cup \chi$; $\chi \subseteq \mathcal{G}^+$ respectively, and action $A \ni a = \langle c_a, \mathit{pre}(a) \leftarrow \mathit{pre}(a) \cup \chi;\ \chi \subseteq \widetilde{\mathit{pre}}(a), \mathit{eff}^{\pm}(a) \leftarrow \mathit{eff}^{\pm}(a) \cup \chi;\ \chi \subseteq \widetilde{\mathit{eff}}^{\pm}(a) \rangle$. Given an annotated model with k possible conditions, there may be 2^k such instantiations, which forms its *completion set*.

The Likelihood $\mathcal{L}$ of an instantiation $inst(\mathbb{M})$ of the annotated model $\mathbb{M}$ is given by –

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

Such models turn out to be especially useful for the representation and learning of human (mental) models from observations, where uncertainty after the learning process can be represented in terms of model annotations as in (Nguyen *et al.* 2017; Bryce *et al.* 2016). Let $\mathbb{M}^R_H$ be the culmination of a model learning process and $\{\mathcal{M}^R_{h_i}\}_i$ be the completion set of $\mathbb{M}^R_H$. Note that one of these models – $g(\mathbb{M}^R_H)$ – is the actual ground truth (i.e. the human's real mental model).

The USAR Domain

We will illustrate the algorithms in this paper in a typical (Bartlett 2015) Urban Search And Reconnaissance (USAR) tasks where a remote robot is put into disaster response operation often controlled partly or fully by an external human commander who orchestrates the entire operation. The robot's job in such scenarios is to infiltrate areas that may be otherwise harmful to humans, and report on its surroundings as and when required / instructed by the external supervisor. The external usually has a map of the environment, but this map may no longer be accurate in the event of the disaster – e.g. new paths may have opened up, or older paths may no longer be available, due to rubble from collapsed structures like walls and doors. The robot (internal) however may not need to inform the external of all these changes so as not to cause information overload of the commander who may be otherwise engaged in orchestrating the entire operation. The robot is thus delegated high level tasks but is often left to compute the plans itself since it may have a better understanding of the environment. However, the robot's actions also contribute to the overall situational awareness of the external, who may require explanations on the robots plans when necessary. In general, differences in the models of the human and the robot can manifest in any form (e.g. the robot may have lost some capability or its goals may have changed). In the current setup, we deal with differences in the map of the environment as available to the human-robot team, i.e. these differences can then be compiled to differences only in the initial state of the planning problem (the human model has the original unaffected model of the world). This makes no difference to the proposed algorithms which treat all model changes equally.

The USAR domain is also ideal for visualizing to non-expert participants in comparison to, for example, logistics-type domains which should ideally be evaluated by experts. This became an important factor while designing the user studies. The USAR domain is thus at once close to motion planning as easily interpreted by non-experts but also incorporates typical aspects of task plans such as preconditions and effects in terms of rubble removal, collapsed halls, etc. and relevant abilities of the robot. As such, simulated USAR scenarios provide an ideal testbed for developing algorithms for effective human-robot interaction. Figure 2:B illustrates our setup (https://youtu.be/40Xol2GY7zE).

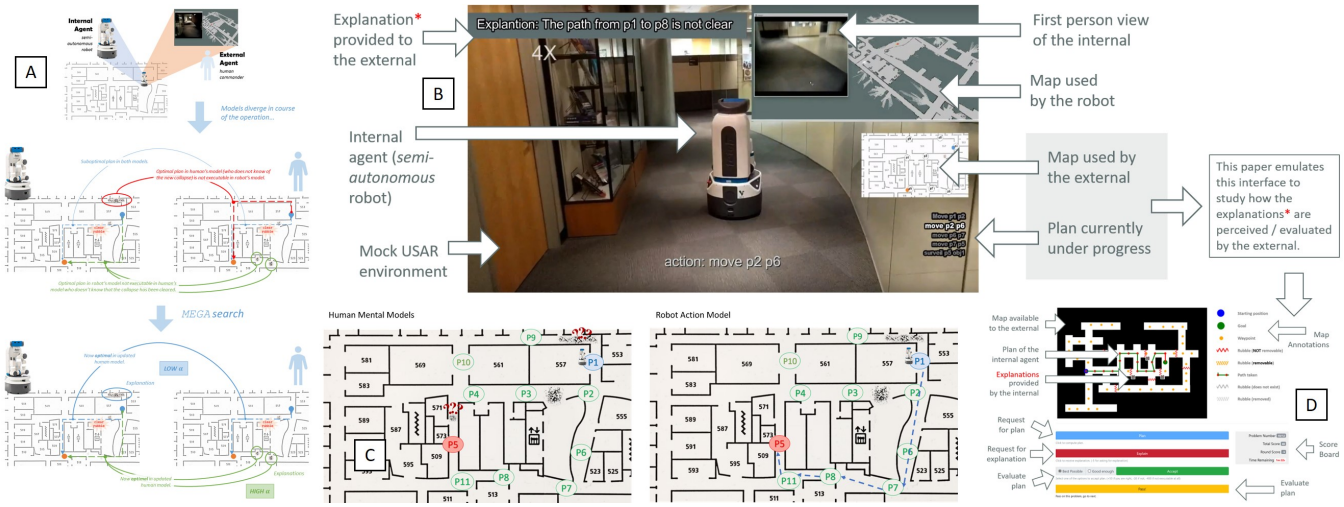


Figure 2: A mock USAR setting for studying the human-robot relationship in a typical disaster response team.

Human-Aware Planning Revisited

The human-aware planning paradigm introduces the *mental model* of the human in the loop into a planner's deliberative process, in addition to the planner's own model in the classical sense and the robot's estimate of the *human model*.

A Human-Aware Planning (HAP) Problem is given by the tuple $\Psi = \langle \mathcal{M}^R, \mathcal{M}_r^H, \mathcal{M}_h^R \rangle$ where $\mathcal{M}^R = \langle D^R, \mathcal{I}^R, \mathcal{G}^R \rangle$ is the planner's model of a planning problem, while $\mathcal{M}_h^R = \langle D_h^R, \mathcal{I}_h^R, \mathcal{G}_h^R \rangle$ is the human's understanding of the same, and $\mathcal{M}_r^H = \langle D_r^H, \mathcal{I}_r^H, \mathcal{G}_r^H \rangle$ is the planner's belief of the human's capability model.

The solution to the human-aware planning problem is a joint plan (Chakraborti *et al.* 2015) $\pi = \langle a_1, a_2, \ldots, a_n \rangle$; $a_i \in \{D^R \cup D_r^H\}$ such that $\delta_\Psi(\mathcal{I}^R \cup \mathcal{I}_r^H, \pi) \models \mathcal{G}^R \cup \mathcal{G}_r^H$. The robot's component in the plan is referred to as $\pi(R) = \langle a_i \mid a_i \in \pi \wedge D^R \rangle$; and similarly $\pi(H)$ for the human. Efforts to make planning more "human-aware" has largely focused on adapting $\pi(R)$ to meet the demands of $\pi(H)$ such as in (Alami *et al.* 2006; 2014; Cirillo *et al.* 2010; Koeckemann *et al.* 2014; Tomic *et al.* 2014; Cirillo 2010; Chakraborti *et al.* 2015; 2016; Talamadupula *et al.* 2014; Zhang *et al.* 2015) in the context of human-robot teams where a robot sacrifices optimality in its own model in favor of globally optimal joint plans. From the perspective of an XAIP agent, computation of the joint plan becomes more interesting when considering $\mathcal{M}_h^R$ as well, i.e. how $\pi(R)$ is *perceived* by the human. One solution is to be "explicable", i.e. make the robot conform to what the human expects of it.

Explicable Planning

An "explicable" solution to the human-aware planning problem is a plan π such that (1) it is executable (but may no longer be optimal) in the robot's model but is (2) "closer" to the expected plan in the human's model –

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$; and

(2) $C(\pi, \mathcal{M}_h^R) \approx C^*_{\mathcal{M}_h^R}$.

Such a plan is referred to as explicable because the human can explain it in their current mental model. "Closeness" or distance to the expected plan is modeled here in terms of cost optimality, but in general this can be any metric such as plan similarity (Srivastava *et al.* 2007; Nguyen *et al.* 2012). In existing literature (Zhang *et al.* 2017; 2016; Kulkarni *et al.* 2016) this has been achieved by modifying the search process so that the heuristic that guides the search is driven by the robot's knowledge of the human mental model. Such a heuristic can be either derived directly (Kulkarni *et al.* 2016) from the mental model or *learned* (Zhang *et al.* 2017) through interactions in the form of affinity functions between plans and their purported goals. The solutions generated this way satisfy the planner's goal, as required by Condition (1), but are also biased towards the human's expectations as required by Condition (2) above.

It is interesting to note that, while mental modeling allows for human-awareness in the positive sense, it can also open up pathways for deception. Indeed, recent work (Kulkarni *et al.* 2018) has looked at how the concept of explicability can be flipped to obfuscate a robot's intentions.

Plan Explanations

The other approach would be to compute optimal plans in the planner's model (which may appear as inexplicable to the human) and provide an explanation of that plan in terms of the model differences – this is referred to as the process of *model reconciliation* (Chakraborti *et al.* 2017b). Although explanation of plans has been investigated in the past (c.f. (Kambhampati 1990; Sohrabi *et al.* 2011; Seegebarth *et al.* 2012; Meadows *et al.* 2013)), much of that work has involved the planner explaining its decisions with respect to its own model (i.e. current state, actions and goals) and assuming that this "*soliloquy*" also helps the human in the loop. While such a sanguine assumption may well be requited when the human is an expert "debugger" and is intimately familiar with the agent's innards, it is completely unrealistic in most human-AI interaction scenarios, where

the humans may have a domain and task model that differs significantly from that used by the planner. We posit then that explanations should be seen as the robot's attempt to move the human's model to be in conformance with its own. The model reconciliation process thus forms the core of the explanation process for an XAIP agent and is thus the focus of the rest of the paper.

Our view of explanation as a model reconciliation process is supported by studies in the field of psychology which stipulate that explanations *"privilege a subset of beliefs, excluding possibilities inconsistent with those beliefs... can serve as a source of constraint in reasoning..."* (Lombrozo 2006). This is achieved in our case by the appropriate change in the expectation of the model that is believed to have engendered the plan in question. Further, authors in (Lombrozo 2012) also underline that explanations are *"typically contrastive... the contrast provides a constraint on what should figure in a selected explanation..."* - this is especially relevant in order for an explanation to be self-contained and unambiguous. Hence the requirement of optimality, which not only ensures that the current plan is valid in the updated model, but is also better than other alternatives or foils (Miller 2017).

The model reconciliation viewpoint can explain many phenomena in both explanation and transparency – e.g. the fact that well-performing, efficient teams require less, not more, explicit communication (Entin and Serfaty 1999) and the characteristics of effective team debriefing (Tannenbaum and Cerasoli 2013) after a mission or project.

The optimality criterion, and argumentation over the human mental model, makes the problem fundamentally different from model change algorithms in (Göbelbecker *et al.* 2010; Herzig *et al.* 2014; Eiter *et al.* 2010; Bryce *et al.* 2016; Porteous *et al.* 2015) which focus more on the feasibility of plans or correctness of domains.

The Model Reconciliation Process

The explanation process, in response to a plan π that the robot has come up with and is perceived as inexplicable by the human, begins with the following question –

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

This questions can arise due to one or both of two causes –

- $\mathcal{M}_h^R$, i.e. the human's approximation of the robot model is wrong. Here, since it knows its own ground truth model, the robot can use an approximation of the human mental model (known unknown) to perform model reconciliation so that both of them are on the same page.
- $\mathcal{M}_r^H$, i.e. the robot's approximation of the human model is wrong. The above approach would not work here since the robot does not know what it does not know (i.e. the real human model is an unknown unknown). However, if the above approach fails to provide a satisfactory response from the human, then the robot can conclude it must be because of this and seek out more information to update its understanding of the human model.

For the first case, the model reconciliation approach would be to provide an (1) explanation or model update $\mathcal{E}$ such that the (2) robot optimal plan is (3) feasible and at least as good as the foil in the updated model, i.e.

(1) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$; and

(2) $C(\pi, \mathcal{M}^R) = C^*_{\mathcal{M}^R}$;

(3) $\delta_{\widehat{\mathcal{M}}_h^R}(\widehat{\mathcal{I}}_h^R, \pi) \models \widehat{\mathcal{G}}_h^R \ \wedge \ C(\pi, \widehat{\mathcal{M}}_h^R) < C(\hat{\pi}, \widehat{\mathcal{M}}_h^R)$.

The question can also be posed in the following form –

Q: Why plan π?

This, in essence, involves an implicit quantifier over all possible foils. Condition (3) above then must ensure that plan π is now also optimal in the updated mental model –

(3) $C(\pi, \widehat{\mathcal{M}}_h^R) = C^*_{\widehat{\mathcal{M}}_h^R}$.

In (Chakraborti *et al.* 2017b) we explore different model reconciliation processes considering four characteristics –

R1. **Completeness -** Explanations of a plan should be able to be compared and contrasted against other alternatives, so that no better solution exists. We enforce this property by requiring that in the updated human model the plan being explained is optimal – i.e. Conditions (3).

R2. **Conciseness -** Explanation should be concise so that they are easily understandable to the explainee. Larger an explanation is, the harder it is for the human to incorporate that information into her deliberative process.

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

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

A Minimally Complete Explanation (MCE) is the shortest explanation that satisfies conditions (1) and (2).

A Minimally Monotonic Explanation (MME) is the shortest explanation that is both complete and monotonic.

A Plan Patch Explanation (PPE) only includes (all the) model updates pertaining to actions in the plan π.

A Model Patch Explanation (MPE) includes all the model updates $|\mathcal{M}^R \ \Delta \ \mathcal{M}_h^R|$.

The requirements outlined above are thus often at odds - an explanation that is very easy to compute may be very hard to comprehend (c.f. Table 1). A detailed account of these explanations can be found in (Chakraborti *et al.* 2017b); we will concentrate on MCEs for the rest of the paper.

Remark Note that during model reconciliation process, the robot model need not be the ground truth. However, the robot can only explain with respect to what it believes to be true. This can, of course, be wrong and be refined iteratively through interaction with the human, as demonstrated in a decision support setting in (Sengupta *et al.* 2017).

Table 1: Requirements for different types of explanations.

Explanation Type	R1	R2	R3	R4
Plan Patch Explanation	✗	✓	✗	✓
Model Patch Explanation	✓	✗	✓	✓
Minimally Complete Explanation	✓	✓	✗	?
Minimally Monotonic Explanation	✓	✓	✓	?

Remark We insisted that explanations must be compatible with the planners model. If this is relaxed, it allows the planner to generate "explanations" that it knows are false and deceive the human. In recent work (Chakraborti and Kambhampati 2018), we have shown that participants in a study were generally positive towards lying for the greater good especially when those actions would not be determined by their teammate, but is loath to suspend normative behavior, robot or not, in the event that they would be caught in that act (unless the robot is the recipient of the misinformation!).

Remark While in this line of work, we are concerned more with the generation of the *content* of explanations rather than the actual delivery of this information, there has been some recent work to this end. Depending on the type of interaction between the planner and the human, this can be achieved by means of natural language dialog (Perera *et al.* 2016), in the form of a graphical user interface (Sengupta *et al.* 2017; Chakraborti *et al.* 2018b) or even in mixed-reality interfaces (Chakraborti *et al.* 2018d; 2018c).

How to chose between Explicability/Explanations?

The two processes of explanations and explicability are intrinsically related in an agent's deliberative process (c.f. Figure 5) – it can generate a explicable plan to the best of its ability or it can provide explanations whenever required, or it can even opt for a combination of both if the expected human plan is too costly in its own model (e.g. the human might not be aware of some safety constraints) or the cost of communication overhead for explanations is too high (e.g. limited communication bandwidth). In the following discussion, we try to attain the sweet spot between plan explanations and explicability during the decision making process.

From the perspective of design of autonomy, the explicability versus explanations trade-off has two interesting implications – (1) the agent can now not only explain but also *plan* in the multi-model setting with the trade-off between compromise on its optimality and possible explanations in mind; and (2) the argumentation process is known to be a crucial function of the reasoning capabilities of humans (Mercier and Sperber 2010), and now by extension of autonomous agents as well, as a result of these algorithms that incorporate the explanation generation process into the decision making process itself. General argumentation frameworks for resolving disputes over plans have indeed been explored before (Belesiotis *et al.* 2010; Emele *et al.* 2011). Other forms of argumentation (Russell and Wefald 1991) has been aimed at meta-level reasoning of resource usage or cost of solutions. Our work can be seen as the specific case where the argumentation process is over a set of constraints that trade-off the quality of a plan and

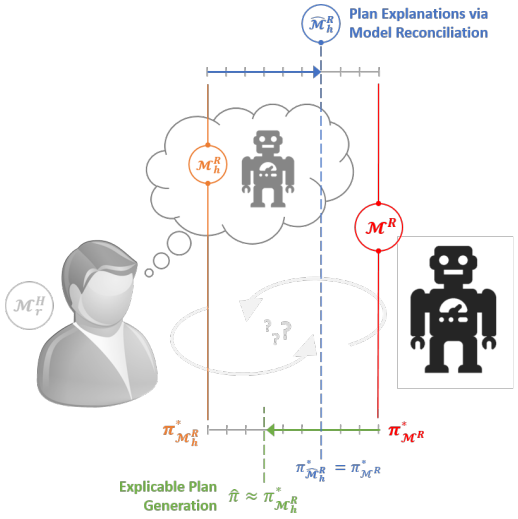


Figure 3: Balancing explicability and explanations in HAP.

the cost of explaining it. *This is, in fact, the first of its kind algorithm that can achieve this.*

The result of a trade off in the relative cost of explicability and explanations during the plan generation process is a plan π and an explanation $\mathcal{E}$ such that (1) π is executable in the robot's model, and with the explanation (2) in the form of model updates it is (3) optimal in the updated human model while (4) the cost (length) of the explanations, and the cost of deviation from optimality in its own model to be explicable to the human, is traded off according to a constant α –

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$;

(2) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$;

(3) $C(\pi, \widehat{\mathcal{M}}_h^R) = C^*_{\widehat{\mathcal{M}}_h^R}$; and

(4) $\pi = \arg\min_\pi \{ |\mathcal{E}| + \alpha \times | C(\pi, \mathcal{M}^R) - C^*_{\mathcal{M}^R} | \}$.

With higher values of α the planner generates plans that require more explanation; with lower α it will generate more explicable plans. Thus, using this hyperparameter, an autonomous agent can deliberate over the trade-off in the costs it incurs in being explicable to the human (second minimizing term in (4)) versus explaining its decisions (first minimizing term in (4)). Note that this trade-off is irrespective of the cognitive burden of those decisions on the human in the loop. For example, for a robot in a collapsed building during a search and rescue task, may have limited bandwidth for communication and hence prefer to be explicable instead.

Demonstration Figure 2:A illustrates a section of the environment where this whole scenario plays out. The orange marks indicate rubble that has blocked a passage, while the green marks indicate collapsed walls. The robot, currently located at the position marked with a blue **O**, is tasked with taking a picture at location marked with an orange **O** in the figure. The external commander's expects the robot to take the path shown in red, which is no longer possible. The robot armed with `MEGA`* has two choices – it can either follow the green path and explain the revealed passageway due to the

collapse, or compromise on its optimal path, clear the rubble and proceed along the blue path. A video demonstration can be viewed at https://youtu.be/Yzp4FU6Vn0M. The first part of the video demonstrates the plan generated by MEGA* for low α values. As expected, it chooses the blue path that requires the least amount of explanation, i.e. the most explicable plan. In fact, the robot only needs to explain a single initial state change to make its plan optimal –

```
remove-has-initial-state-clear_path p1 p8
```

This is an instance where the plan closest to the human expectation, i.e. the most explicable plan, still requires an explanation. Moreover, in order to follow this plan, the robot must perform the costly `clear_passage p2 p3` action to traverse the corridor between `p2` and `p3`, which it could have avoided in its optimal plan (shown in green). Indeed, MEGA* switches to the robot's optimal plan for higher values of α along with the following explanation –

```
add-has-initial-state-clear_path p6 p7
add-has-initial-state-clear_path p7 p5
remove-has-initial-state-clear_path p1 p8
```

What happens if the mental model is unknown?

The model reconciliation process described above is only feasible if inconsistencies of the robot model with the human mental model are known precisely. Although we made this assumption so far as a first step towards formalizing the model reconciliation process, this can be hard to achieve in practice. Instead, the agent may end up having to explain its decisions with respect to a *set of possible models* which is its best estimation of the human's knowledge state learned in the process of interactions. In this situation, the robot can try to compute MCEs for each possible configuration. However, this can result in situations where the explanations computed for individual models independently are not consistent across all possible target domains. Thus, in the case of model uncertainty, such an approach cannot guarantee that the resulting explanation will be acceptable. Instead, we want to find an explanation such that $\forall i\ \pi^*_{\widehat{\mathcal{M}}^R_{h_i}} \equiv \pi^*_{\mathcal{M}^R}$.

This is a single model update that makes the given plan optimal (and hence explained) in all the updated domains (or in all possible domains). At first glance, it appears that such an approach, even though desirable, might turn out to be prohibitively expensive especially since solving for a *single* MCE involves search in the model space where each search node is an optimal planning problem (Chakraborti *et al.* 2017b). However, it turns out that the same search strategy can be employed here as well by representing the human mental model as an *annotated* model (Sreedharan *et al.* 2018a). Condition (3) for an MCE now becomes –

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

This is hard to achieve since it is not known which is the actual mental model of the human. So we want to preserve the optimality criterion for all (or as many) instantiations of the incomplete estimation of the explainee's mental model. Keeping this in mind, we define *robustness* of an explanation for an incomplete mental models as the probability mass of models where it is a valid explanation.

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

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

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

This means a conformant explanation ensures that the given plan is explained in all the models in the completion set of the human model.

Demonstration. Consider now that the robot is located at P1 (blue) and needs to collect data from P5 (c.f. Figure 2:C). While the human commander understands the goal, she is under the false impression that the paths from P1 to P9 and P4 to P5 are unusable (red question marks). She is also unaware of the robot's inability to use its hands. On the other hand, while the robot does not have a complete picture of the human's mental model, it understands that any differences between the models are related to (1) Path from P1 to P9; (2) Path from P4 to P5; (3) Robot's ability to use its hands; and (4) Whether the Robot needs its arm to clear rubble. Thus, from the robot's perspective, the human model can be one of sixteen possible models (one of which is the actual mental model). Here, a conformant explanation for the optimal robot plan (blue) is as follows (a demonstration can be viewed at https://youtu.be/bLqrtffW6Ng) –

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

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

- Increasing the cost of model updates involving uncertain conditions relative to those involving known preconditions or effects. This ensures that the search prefers explanations that contain known conditions. By definition, such explanations will not have superfluous information.
- However, such explanations may not exist. Instead, we can convert conformant explanations into *conditional* ones by turning each model update for an annotated condition into a question and only provide an explanation if the human's response warrants it – e.g. instead of asking the human to update the precondition of `clear_passage`, the robot can first ask if the human thinks that action has a precondition `hand_usable`.

Thus, one way of removing superfluous explanations is to engage the human in conversation and reduce the size of the completion set. Consider the following exchange –

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

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

Note that in asking questions such as these, the robot is trying to exploit the human's (lack of) knowledge of the problem in order to provide more concise explanations. This can be construed as a case of lying by omission and can raise interesting ethical considerations (Chakraborti and Kambhampati 2018). Humans, during an explanation process, tend to undergo this same "selection" process (Miller 2017) as well in determining which of the many reasons that could explain an event is worth highlighting. It is worthwhile investigating similar behavior for autonomous agents.

Anytime Explanations Since dealing with model uncertainty can be computationally expensive, we relax the minimality requirement and introduce an anytime depth first explanation generation algorithm. This is explained in detail in (Sreedharan *et al.* 2018a).

What if there are multiple humans in the loop?

While generating explanations for a *set of models*, the robot is essentially trying to cater to multiple human models at the same time. We posit then that the same approaches can be adopted to situations when there are *multiple humans* in the loop instead of a single human whose model is not known with certainty. As before, computing separate explanations (Chakraborti *et al.* 2017b) for each agent can result in situations where the explanations computed for individual models independently are not consistent across all the possible target domains. In the case of multiple teammates being explained to, this may cause confusion and loss of trust, especially in teaming with humans who are known (Cooke *et al.* 2013) to rely on shared mental models. Thus *conformant explanations* can find useful applications in dealing with not only model uncertainty but also model multiplicity.

In order to do this, from the set of target human mental models we construct an annotated model so that *the preconditions and effects that appear in all target models become necessary ones, and those that appear in just a subset are possible ones*. As before, we find a single explanation that is a satisfactory explanation for the entire set of models, without having to repeat the standard MRP process over all possible models while coming up with an explanation that can satisfy all of them and thus establish common ground (Chakraborti *et al.* 2018c; Sreedharan *et al.* 2018b).

Demonstration We go back to our use case, now with *two* human teammates, one external and one internal. A video of the demonstration is available at https://youtu.be/hlPTmggRTQA. The robot is now positioned at `P1` and is expected to collect data from location `P5`. Before the robot can perform its `surveil` action, it needs to obtain a set of tools from the internal human agent. The human agent is initially located at `P10` and is capable of traveling to reachable

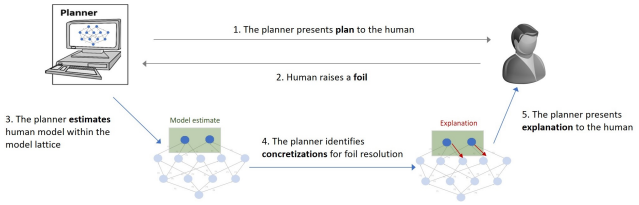


Figure 4: Hierarchical explanation generation approach.

locations to meet the robot for the handover. Here the external commander incorrectly believes that the path from `P1` to `P9` is clear and while the one from `P2` to `P3` is closed. The internal human agent, on the other hand, not only believes in the errors mentioned above but is also under the assumption that the path from `P4` to `P5` is not traversable. Due to these different initial states, each of these agents ends up generating a different optimal plan. The plan expected by the external commander requires the robot to move to location `P10` (via `P9`) to meet the human. After collecting the package from the internal agent, the commander expects it to set off to `P5` via `P4`. The internal agent, on the other hand, believes that he needs to travel to `P9` to hand over the package. As he believes that the corridor from `P4` to `P5` is blocked, he expects the robot to take the longer route to `P5` through `P6`, `P7`, and `P8` (orange). Finally, the optimal plan for the robot (blue) involves the robot meeting the human at `P4` on its way to `P5`. Using `MEGA*-Conformant`, we find the smallest explanation, which can explain this plan to both humans –

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

While the last two model changes are equally relevant for both the agents, the first change is specifically designed to help the internal. The first update helps convince the human that the robot can indeed reach the goal through `P4`, while the next two help convince both agents as to why the agents should meet at `P4` rather than other locations.

How to model human expertise?

Most of the above discussion has focused on generating explanations in cases where both the human and the robot understands the task at the same granularity. Applying model reconciliation without acknowledging the difference in the level of "expertise" can lead to confusion and information overload. Indeed, explanation generation techniques for machine learning systems have explicitly modeled this difference (Ribeiro *et al.* 2016; 2018).

In (Sreedharan *et al.* 2018c), authors have looked at ways of generating explanations when the human has access to only an abstract version of the model of the robot. Specifically, they focus on state abstractions where the abstract model was formed by projecting out a certain subset of state fluents (Srivastava *et al.* 2016), though the principles are likely to carry over to other types of abstraction as well (e.g. temporal abstractions of the types discussed in (Marthi *et al.* 2007)). Since the abstract model may be logically weaker, the human may incorrectly believe that an optimal plan sug-

gested by the planner is suboptimal. When presented with the plan π, the human can respond by either presenting a set of *foils* F. In such cases, the *explanation* takes the form of information about a set of state properties which when introduced into the human model resolves or invalidates the set of foils. Thus, the explanation can be uniquely represented by a sequence of propositions $\epsilon = \langle p_1, .., p_k \rangle$ as follows –

(1) A set of foils $F = \{\pi_1, .., \pi_m\}$ such that $\forall \pi \in F$, $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \not\models \mathcal{G}^R$ and $\delta_{\mathcal{M}_h^R}(\mathcal{I}_h^R, \pi) \not\models \mathcal{G}_h^R$

(2) An explanation $\epsilon = \langle p_1, ..., p_k \rangle$, such that $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$;

(3) $\forall \pi \in F$, $\delta_{\widehat{\mathcal{M}}_h^R}(\mathcal{I}_h^R, \pi) \not\models \mathcal{G}_h^R$

One of the main challenges with this method is the uncertainty about the human model. To address this, the authors build a lattice of possible models from the task model called *model lattice* ($\mathcal{L}$) as shown in Figure 4. The lattice consists of possible abstractions of the concrete task model and an edge exists between two models if there exists a single predicate that can be projected from one model to create the other. The foils are used to estimate the possible human model and use this estimate to compute the explanations.

How do humans reconcile models?

The design of "human-aware" algorithms is, of course, incomplete without evaluations of the same with actual humans in the loop. Thus, in the final part of this discussion, we will report on the the salient findings from a controlled user study we undertook recently in order to evaluate the usefulness of the model reconciliation approach. A detailed report of the study can be read at (Chakraborti *et al.* 2018a).

Experimental Setup For the study, we exposed the external commanders interface (c.f. Figure 2:D) to participants who, based on their map (which they are told may differ from the robots) had to identify if a given plan (which may be optimal in the robot model or explicable or even balanced) looks optimal or satisficing to them. If the player is unsure, they can ask for an explanation. The explanations provided are one of the types described before.

Study-1 In the first set of experiments, participants assumed the role of the explainer. It was found that, when left to themselves, they generated explanations of the type MPE or (if communication was restricted) MCE. Further, in subjective responses, they considered model reconciliation as necessary and sufficient for the explanation process.

Study-2 Here, participants assumed the role of the explainer, and had to identify, on the basis of explanations provided the quality of the given plan. We found that the participants were indeed able to distinguish between optimal plans (when provided with MCEs or MPEs) and (perceived) satisficing plans (when provided with PPEs) and were in general overwhelmingly in favor of model reconciliation as an effective form of explanation. We further found that explicable plans indeed reduced the call of explanations, while balanced plans preserved their outlook towards the explanations while allowing the robot to trade-off its communication cost with the optimality of its plans.

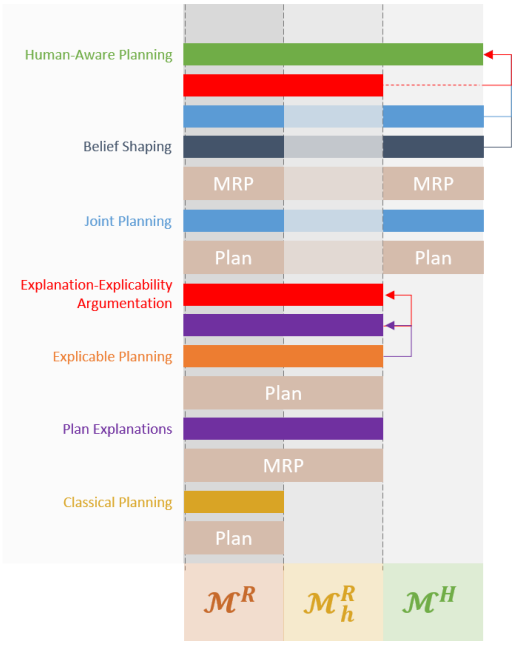


Figure 5: A subsumption architecture for HAP.

Work in Progress

So far we have have not considered differences in the cost function which, though falls under the scope of model differences to be explained, can introduce interesting challenges to the model reconciliation problem. It may be possible to learn such functions through interactions with the human as in (Zhang *et al.* 2017; Kulkarni *et al.* 2016). Further, we have not modeled the computational capability of the human which also affect the process – this can be potentially handled by modeling $\epsilon-$optimal humans or by considering top-K plans (Katz *et al.* 2018) while checking the optimality condition during model space search. Finally, we do not consider system level constraints (such as time and resources) in the explanations which remain out of scope of explanations viewed as a model reconciliation process.

Conclusion

The different behaviors engendered by multi-model argumentation can be *composed* to form more and more sophisticated forms of human-aware behavior. We thus conclude with a hierarchical composition of behaviors in the form of a subsumption architecture for human-aware planning, inspired by (Brooks 1986). This is illustrated in Figure 5. The basic reasoning engines are the Plan and MRP (Model Reconciliation) modules. The former accepts model(s) of planning problems and produces a plan, the latter accepts the same and an produces a new model. The former operates in plan space and gives rise to classical, joint and explicable planning depending on the models it is operating on, while the latter operates in model space to produce explanations and belief shaping behavior. These are then composed to form argumentation modules for trading of explanations and explicability and human-aware planning in general.

Acknowledgements This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014-16-1-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, N00014-18-1-2442 and the NASA grant NNX17AD06G. The first author is also supported by the IBM Ph.D. Fellowship from 2016-18.

References

Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Toward Human-Aware Robot Task Planning. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006.

Rachid Alami, Mamoun Gharbi, Benjamin Vadant, Raphaël Lallement, and Adolfo Suarez. On human-aware task and motion planning abilities for a teammate robot. In *Human-Robot Collaboration for Industrial Manufacturing Workshop, RSS*, 2014.

Cade Earl Bartlett. Communication between Teammates in Urban Search and Rescue. *Thesis*, 2015.

Alexandros Belesiotis, Michael Rovatsos, and Iyad Rahwan. Agreeing on plans through iterated disputes. In *AAMAS*, pages 765–772, 2010.

Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

Dan Bryce, J. Benton, and Michael W. Boldt. Maintaining Evolving Domain Models. In *IJCAI*, 2016.

T. Chakraborti and S. Kambhampati. Algorithms for the Greater Good! On Mental Modeling and Acceptable Symbiosis in Human-AI Collaboration. *ArXiv e-prints*, January 2018.

T. Chakraborti, G. Briggs, K. Talamadupula, Yu Zhang, M. Scheutz, D. Smith, and S. Kambhampati. Planning for serendipity. In *IROS*, 2015.

Tathagata Chakraborti, Yu Zhang, David Smith, and Subbarao Kambhampati. Planning with Resource Conflicts in Human-Robot Cohabitation. In *AAMAS*, 2016.

Tathagata Chakraborti, Subbarao Kambhampati, Matthias Scheutz, and Yu Zhang. AI Challenges in Human-Robot Cognitive Teaming. *arXiv preprint arXiv:1707.04775*, 2017.

Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*, 2017.

T. Chakraborti, S. Sreedharan, S. Grover, and S. Kambhampati. Plan Explanations as Model Reconciliation – An Empirical Study. *ArXiv e-prints*, February 2018.

Tathagata Chakraborti, Fadnis P. Kshitij, Kartik Talamadupula, Mishal Dholakia, Biplav Srivastava, Jeffrey O Kephart, and Rachel KE Bellamy. Visualizations for an explainable planning agent. *ICAPS UISP*, 2018.

Tathagata Chakraborti, Sailik Sengupta, and Subbarao Kambhampati. MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making. *ICAPS UISP*, 2018.

Tathagata Chakraborti, Sarath Sreedharan, Anagha Kulkarni, and Subbarao Kambhampati. Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots in a Mixed-Reality Workspace. In *HRI 2018 Workshop on Virtual, Augmented and Mixed-Reality for Human-Robot Interactions (VAM-HRI'18)*, 2018.

Marcello Cirillo, Lars Karlsson, and Alessandro Saffiotti. Human-aware Task Planning: An Application to Mobile Robots. *ACM Transactions on Intelligent Systems and Technology*, 2010.

Marcello Cirillo. *Planning in inhabited environments: human-aware task planning and activity recognition*. PhD thesis, Örebro university, 2010.

Nancy J Cooke, Jamie C Gorman, Christopher W Myers, and Jasmine L Duran. Interactive team cognition. *Cognitive science*, 37(2):255–285, 2013.

Thomas Eiter, Esra Erdem, Michael Fink, and Ján Senko. Updating action domain descriptions. *Artificial intelligence*, 2010.

Chukwuemeka D Emele, Timothy J Norman, and Simon Parsons. Argumentation strategies for plan resourcing. In *AAMAS*, 2011.

Elliot E Entin and Daniel Serfaty. Adaptive team coordination. *Human factors*, 41(2):312–325, 1999.

Maria Fox, Derek Long, and Daniele Magazzeni. Explainable Planning. In *IJCAI XAI Workshop*, 2017.

Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. 2010.

Andreas Herzig, Viviane Menezes, Leliane Nunes de Barros, and Renata Wassermann. On the revision of planning tasks. In *ECAI*, 2014.

Subbarao Kambhampati. A classification of plan modification strategies based on coverage and information requirements. In *AAAI 1990 Spring Symposium on Case Based Reasoning*, 1990.

Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A Novel Iterative Approach to Top-k Planning. In *ICAPS*, 2018.

Uwe Koeckemann, Federico Pecora, and Lars Karlsson. Grandpa Hates Robots - Interaction Constraints for Planning in Inhabited Environments. In *AAAI*, 2014.

Anagha Kulkarni, Tathagata Chakraborti, Yantian Zha, Satya Gautam Vadlamudi, Yu Zhang, and Subbarao Kambhampati. Explicable Robot Planning as Minimizing Distance from Expected Behavior. *CoRR*, abs/1611.05497, 2016.

Anagha Kulkarni, Siddharth Srivastava, and Subbarao Kambhampati. Implicit robot-human communication in adversarial and collaborative environments. *CoRR*, abs/1802.06137, 2018.

Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable Agency for Intelligent Autonomous Systems. In *AAAI/IAAI*, 2017.

Tania Lombrozo. The Structure and Function of Explanations . *Trends in Cognitive Sciences*, 10(10):464 – 470, 2006.

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

Bhaskara Marthi, Stuart J Russell, and Jason Andrew Wolfe. Angelic semantics for high-level actions. In *ICAPS*, pages 232–239, 2007.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. 1998.

Ben Leon Meadows, Pat Langley, and Miranda Jane Emery. Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In *AAAI Workshop: Plan, Activity, and Intent Recognition*, volume 13, page 13, 2013.

Hugo Mercier and Dan Sperber. Why Do Humans Reason? Arguments for an Argumentative Theory. *Behavioral and Brain Sciences*, 2010.

Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *CoRR*, abs/1706.07269, 2017.

Tuan Anh Nguyen, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 2012.

Tuan Nguyen, Sreedharan, and Subbarao Kambhampati. Robust planning with incomplete domain models. *Artificial Intelligence*, 245:134 – 161, 2017.

Vittorio Perera, Sai P. Selvaraj, Stephanie Rosenthal, and Manuela Veloso. Dynamic Generation and Refinement of Robot Verbalization. In *RO-MAN*, Columbia University, NY, August 2016.

Julie Porteous, Alan Lindsay, Jonathon Read, Mark Truran, and Marc Cavazza. Automated extension of narrative planning domains with antonymic operators. In *AAMAS*, pages 1547–1555, 2015.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD*, pages 1135–1144. ACM, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2003.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991.

Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *ICAPS*, 2012.

Sailik Sengupta, Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. Radar-a proactive decision support system for human-in-the-loop planning. In *AAAI Fall Symposium on Human-Agent Groups*, 2017.

Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11)*, pages 261–267, San Francisco, USA, August 2011.

Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Handling Model Uncertainty and Multiplicity in Explanations as Model Reconciliation. In *ICAPS*, 2018.

Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Handling model uncertainty and multiplicity in explanations via model reconciliation. 2018.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise-level modeling for user specific robot-behavior explanations. *arXiv preprint arXiv:1802.06895*, 2018.

Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. Domain independent approaches for finding diverse plans. In *IJCAI*, pages 2016–2022, 2007.

Siddharth Srivastava, Stuart J Russell, and Alessandro Pinto. Metaphysics of planning domain descriptions. In *AAAI*, 2016.

Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz, and Subbarao Kambhampati. Coordination in human-robot teams using mental modeling and plan recognition. In *IROS*, 2014.

Scott I Tannenbaum and Christopher P Cerasoli. Do team and individual debriefs enhance performance? a meta-analysis. *Human factors*, 55(1):231–245, 2013.

Stevan Tomic, Federico Pecora, and Alessandro Saffiotti. Too Cool for School??? Adding Social Constraints in Human Aware Planning. In *Workshop on Cognitive Robotics (CogRob)*, 2014.

Daniel S Weld and Gagan Bansal. Intelligible artificial intelligence. *arXiv preprint arXiv:1803.04263*, 2018.

Yu Zhang, Vignesh Narayanan, Tathagata Chakraborty, and Subbarao Kambhampati. A human factors analysis of proactive assistance in human-robot teaming. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

Yu Zhang, Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan Explicability for Robot Task Planning. In *RSS Workshop on Planning for Human-Robot Interaction: Shared Autonomy and Collaborative Robotics*, 2016.

Yu Zhang, Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan Explicability and Predictability for Robot Task Planning. In *ICRA*, 2017.

Explaining Rebel Behavior in Goal Reasoning Agents

Dustin Dannenhauer[1] and **Michael W. Floyd**[2] and **Daniele Magazzeni**[3] and **David W. Aha**[4]
[1]NRC Postdoctoral Fellow; NRL; Navy Center for Applied Research in AI; Washington, DC
[2]Knexus Research Corporation; Springfield, VA
[3]King's College London, Strand, London WC2R 2LS
[4]Naval Research Laboratory; Navy Center for Applied Research in AI; Washington DC

{dustin.dannenhauer.ctr, david.aha}@nrl.navy.mil, michael.floyd@knexusresearch.com, daniele.magazzeni@kcl.ac.uk

Abstract

Generating human-comprehensible explanations is an important requirement for autonomous systems in human-agent teaming environments. Humans and agents often have their own knowledge of the world, knowledge of objectives being pursued and tasks being performed, and their own constraints. Given these differences, an agent may be issued goals that violate its own constraints or preferences, or are undesirable for the team's task. Numerous situations may arise where rebellion by dropping or changing goals leads to a more beneficial outcome. Agents with goal reasoning capabilities may rebel by rejecting or altering the goals and plans expected of them by human teammates. Explanations help build trust and understanding between the human and agent, leading to greater overall effectiveness. In this paper we outline motivating examples for explainable rebellious behavior in goal reasoning systems and identify open research questions.

Introduction

In recent years there has been increased interest in autonomous agents capable of rebellion (Briggs and Scheutz 2016). Rebellious agents are agents that may reject, revise, or in some form protest a goal issued to them by another agent (including humans). This ability to rebel is necessary for any agent that receives goals from multiple sources (including self-provided goals) that may conflict. Consider service robots that give tours or deliver goods (e.g., a hotel room service robot); these agents should reject goals that could lead to any number of undesirable situations, such as those that damage the robot. Many positive reasons for rebellion have been described (Aha and Coman 2017; Briggs and Scheutz 2015), including:

- **Differential Information Access**: The agent may have access to information the human does not have. A simple example is when an agent is helping a human to move a box in a warehouse and, while carrying the object, the agent observes a harmful obstacle behind the human. The agent stops moving and informs the human, rebelling against the given goal of moving the box to the target destination.
- **Oversubscription**: The agent is tasked with goals from two teammates, and thus may reject goals from one of them. For example, an agent on a team is tasked by a supervisor with obtaining information (by mapping an environment and taking pictures and video) while other team members are responsible for moving obstacles to clear a path. When another agent asks the surveillance agent to help move an obstacle, the agent may reject that goal since it does not align with its current surveillance goal.
- **Ethical Conflict**: The agent has a conflict with the ethics of a given goal. For example, an agent may be asked to take a harmful action against another human, violating the agent's ethical code of not harming any humans. This may involve hard constraints, such as not taking any action that could have a harmful effect (Briggs and Scheutz 2015), or a preference for avoiding states with low ethical scores.
- **Impasse**: A provided goal may not be achievable due to resources (e.g., battery level) or obstacles.
- **Task Violation**: A provided goal may violate some global constraints or preferences of the agent, such as delivering a package to a destination outside the country or safe area. These constraints may be similar to ethics, but may be more task-specific.
- **Safety**: An autonomous vehicle may be given a goal to reach a destination in a short period of time. However, the plan to reach the destination could become dangerous. An autonomous car may need to violate traffic laws or drive off-road, or a drone may need to travel too fast to avoid obstacles (i.e., if flying in a forest or indoor environments such as in urban search and rescue settings).

While there are many motivations for positive rebellion, it makes a rebel agent less predictable to other interacting agents. Thus, any agent that can rebel should also be able to explain its behavior. Indeed, legal measures are being adopted to provide individuals affected by automated decision-making with a "right to explanation", as referred to in the recent EU General Data Protection Regulation (GDPR), in place from May 2018 (European Parliament and Council 2016).

The interpretability of AI systems has been a popular topic of workshops and related events since 2016, and in 2017 DARPA launched the Explainable AI (XAI) Program. Most of these efforts have focused on providing transparency to the decision making of machine learning (ML) systems in general, and deep networks more specifically[1].

[1]Exceptions, for example, include the broader intent of the

While XAI research on data-driven ML is well-motivated, AI Planning is well placed to address the challenges of transparency and explainability in a broad range of interactive AI systems. For example, research on Explainable Planning has focused on helping humans to understand a plan produced by the planner (e.g., (Sohrabi, Baier, and McIlraith 2011; Bidot et al. 2010)), on reconciling the models of agents and humans (e.g., (Chakraborti et al. 2017)), and on explaining why a particular action was chosen by a planner rather than a different one (e.g., (Smith 2012; Langley et al. 2017; Fox, Long, and Magazzeni 2017).

Most prior work on rebel agents considers how robots can avoid hard ethical constraints, such as actions that may harm self or others (Briggs and Scheutz 2015), or that violate a constraint in a task specification when working alongside a human (Gregg-Smith and Mayol-Cuevas 2015). These agents have rebelled against actions that have harmful effects. Instead, we are concerned with agents equipped with automated planners that may execute complex sequences of actions to achieve goals.

Since rebellious goal reasoning agents are concerned with decisions regarding which goals to pursue in addition to planning decisions, explanations will also need to consider goal reasoning decisions. Explaining such planning and goal related decisions is the focus of this paper. More specifically, we motivate why the intersection of rebel agents and explainable planning is an important research area, outline future problems for explainable rebellious planning and goal reasoning agents, and discuss several research questions that require attention.

Related Work

Briggs et al. (2015) present an approach to determine the speech directives a robot should utter when rebelling. They describe five felicity conditions that must be met before a robotic agent adopts a goal issued by a human. These conditions may not be sufficient for automated planning agents to accept or reject goals. For example, metrics are not considered, so there is no ability for an agent to reject a goal because it would be much more expensive/risky/unsafe than what the human may have assumed when issuing the goal.

To our knowledge, no prior work describes automated planning agents that can reject goals and explain their decision for rebellion. Research on goal reasoning (Vattam et al. 2013) considers agents that can perform operations on their goals, such as changing or dropping a goal via various operations (Cox, Dannenhauer, and Kondrakunta 2017; Cox 2016) and, in the formalism of the Goal Lifecycle, by applying goal refinement strategies (Roberts et al. 2016). However, in prior work these goal changes occur due to encountering unexpected external events or opportunistic situations, rather than rebelling against a provided goal. Cox and Veloso (1998) describe goal transformations that occur during plan generation in PRODIGY, where one of the transformations is the dropping of a goal. Although rebellious goal changes could be encoded in such an architecture (e.g., rebellion by dropping a goal could be considered a *retraction* operation as described in Table 1 in (Cox and Veloso 1998)), rebellious goal changes have not been considered or implemented.

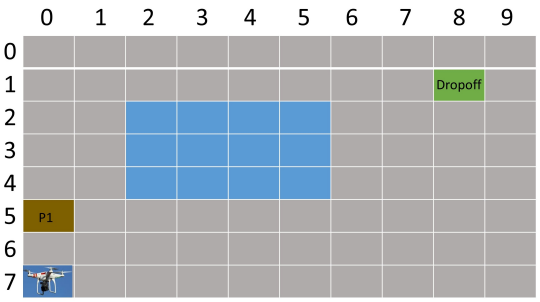


Figure 1: Top-down environment view for an autonomous delivery drone. The environment contains the drone (Row 7, Column 0), a package (Row 5, Column 0), a dropoff location (Row 1, Column 8), and a body of water (rectangular region between Row 2, Column 2 and Row 4, Column 5).

Real-world planning systems operate on large amounts of data and can generate plans that overwhelm a human operator, especially when they are subject to severe time constraints for decision making (e.g., whether to abort a mission). Thus, methods for explaining decisions (e.g., by comparing multiple plans) must generate explanations that can be quickly digested.

Motivating Example

We now provide a motivating example of a rebel agent operating in a dynamic and uncertain environment. Figure 1 displays a top-down view of a state in an environment in which an autonomous delivery drone operates (Row 7, Column 0). In this example, the agent controls the drone and we use the two terms interchangeably. This state contains a package (Row 5, Column 0), a dropoff location (Row 1, Column 8), and a large body of water (a rectangular region between Row 2, Column 2 and Row 4, Column 5). The drone's human operator provides it with goals and can communicate with it, but the operator is not physically located in the area depicted in Figure 1. For this example, we assume that the initial goal utterance provided by the operator is *"deliver the package to the dropoff location and return to your initial position"*.

In the simplest scenario, the drone would take the provided goal and generate the following plan (Figure 2): fly to the package, pick up the package, fly directly to the dropoff location, drop off the package, and fly directly to the initial location. In this version, there would be little need for explanation since the drone would achieve the specified goal using an expected plan (i.e., flying directly between all locations). Additionally, since no unexpected events or environment changes occur, the drone would have no new information to provide its operator.

Consider a variant of this scenario where the drone considers flying over water to be dangerous but the operator is

Workshops on XAI at IJCAI-17 and IJCAI-18

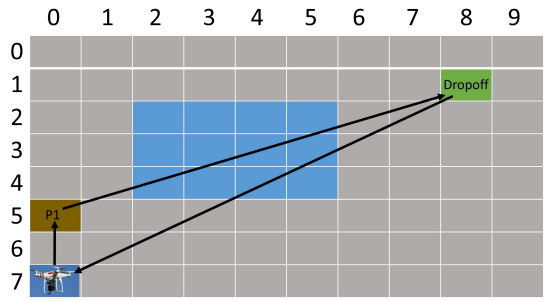


Figure 2: Visualization of drone's plan to fly directly to the package, pick up the package, fly directly to the dropoff, drop off the package, and fly directly to its initial position.

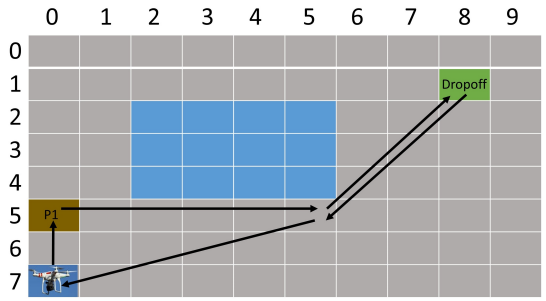


Figure 3: Variant of the drone's delivery plan that avoids flying over the water.

not aware of this water-avoidance preference. The plan generated by the drone will still deliver the package, but will take a slightly longer route that avoids the water (Figure 3). If no pre-execution communication occurred (e.g., making the operator aware of the preference or getting feedback on the generated plan), the drone would execute the plan without knowing that it was rebelling against the operator's desire for a plan that uses only direct flight routes.

Either during plan execution, if communication is possible, or during a post-mission debriefing, the operator may ask questions to improve their understanding of why the drone's plan differed from their expectations:

- **Operator**: *"Were you pursuing another goal?"*
  Agent: *"No, only the goal you provided me"*
- **Operator**: *"Was that the most efficient plan to achieve the goal?"*
  Agent: *"Yes, it was the most direct route I could take to deliver the package given my preferences"*
- **Operator**: *"Why didn't you just fly over the water'*
  Agent: *"Because flying over the water is too dangerous"*

Based on these questions and the provided explanations, the drone can convey that its act of rebellion was due to a divergent set of planning preferences, thereby leading to differences in the drone's generated plan and the plan that the operator expected to see executed.

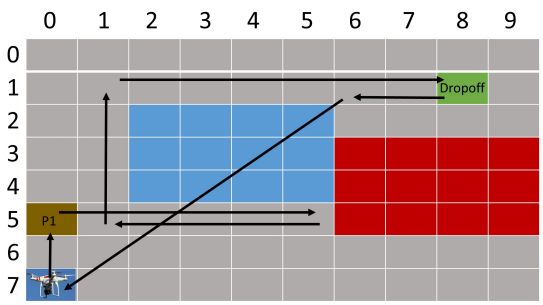


Figure 4: Variant of the drone's delivery plan that requires replanning when the drone detects the fire (rectangular region between Row 3, Column 6 and Row 5, Column 9) and when the drone detects it is low on fuel.

Consider a third variant where the operator is aware that the drone prefers to avoid flying over water. In this variant, the operator would expect the drone to fly around the water but, because they are not physically located in the environment, would be unaware of any environment changes that occur. As shown in Figure 4, when the drone reaches the water's edge it notices a large fire blocking its path to the dropoff location. This would cause the drone to replan to fly clockwise around the water. However, after delivering the package to the dropoff location, the drone realizes its fuel is lower than expected and its planner finds that the only feasible plan, due to the limited fuel, is to fly over the water.

To the operator, who is unaware of the fire, it may appear that the drone rebelled (against the expected plan or provided goal) or opportunistically modified its goals. For example, to an outside observer it would be reasonable to assume that the drone retrieved another package at the location where it detected the fire and delivered that package to a location near the top of the water body. As in the previous scenario, and most situations involving a rebel agent capable of goal reasoning, the operator may wish to query the agent about its goals and plans. For example, these questions, and possible associated explanations from the agent, include:

- **Operator**: *"Why did you stop flying towards the dropoff location and fly clockwise around the water?"*
  Agent: *"Because there was a large fire blocking my path"*
- **Operator**: *"Why did you return by flying over the water?"*
  Agent: *"Because I was low on fuel and my desire to return home outweighed my preference to avoid flying over water"*
- **Operator**: *"Why didn't you take a shorter path over the water by flying directly north after detecting the fire?*
  Agent: *"Because my original plan was to avoid the water, but I used more fuel than expected and needed to replan"*

As a final scenario, consider when the drone executes its initial plan but an unexpected opportunity presents itself. Figure 5 shows that the drone, while attempting to deliver the package, observes a serious car accident. The drone may

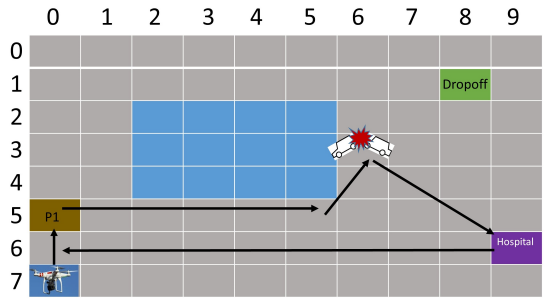


Figure 5: Variant of the drone's plan when it encounters a car crash (Row 3, Column 6) and generates a goal to assist the victims by airlifting them to the hospital (Row 6, Column 9).

have a strong internal motivation to help preserve life whenever possible, so it would rebel against its delivery goal, abandoning or suspending it, and instead formulate a new goal to assist the crash victims. The drone would generate a plan to achieve its new goal, and perform actions to airlift the car's driver to a nearby hospital[2]. Its new plan would involve airlifting the car's driver to a nearby hospital. After completing that plan and achieving the goal, the extra fuel drain from carrying a human would make the delivery goal impossible, so the drone would return to its initial position. The operator may have the following questions for the drone:

- **Operator**: *"Were you pursuing another goal?"*
  Agent: *"Yes, I formulated a new goal to assist a car crash victim"*
- **Operator**: *"Why did you help the victim instead of completing the delivery?"*
  Agent: *"I am programmed to prevent the loss of human life and the injuries looked serious."*
- **Operator**: *"Why didn't you complete the delivery afterwards?'*
  Agent: *"Because the weight of the victim drained my fuel faster than normal; I was unable to generate a plan to complete the delivery given my fuel level"*

As this motivating example demonstrates, when an agent operates in a sufficiently complex environment where it interacts with a teammate, many opportunities/needs for explanation arise. These increase when the agent and its teammate may have different (or unknown) planning preferences (e.g., water-avoidance), methods to evaluate plan quality (e.g., plan duration vs. plan safety), or observable information (e.g., an agent located in the environment vs. an operator externally located). Additionally, the knowledge that an agent can rebel or modify its own goals may make an operator more likely to assume goal changes or rebellion are the cause of unexpected behavior. Thus, the agent needs to answer such questions by explaining its plans and goals, as well as any additional information related to the reason it behaved unexpectedly.

We simplified the presentation and discussion of our examples by including assumptions, namely a lack of online communication or time-sensitive tasks preventing rapid communication, that restricted questions and explanations to occur post-run. However, the need for explanation also exists during the course of operation. For example, upon seeing a real-time report of the drone's GPS location, the operator may ask questions about why the drone appears to be deviating from expectations. Similarly, the drone could realize that its changing behavior is unexpected and provide proactive explanations that it believes will provide necessary context to the operator. Although such real-time questions and explanations could lead to unnecessary supervision of the agent (e.g., questioning every minor variance from expectations, even if the differences have no impact on overall success), they also provide the ability to incrementally correct misunderstandings and prevent the operator from becoming overwhelmed by how much the agent's behavior deviates from expectations. This is important in situations where disuse is possible, since a human may just label a robot as untrustworthy and stop using it rather than ask the tens or hundreds of post-run questions necessary to understand its behavior.

Explaining Goal Reasoning Decisions

The field of goal reasoning has seen at least three general frameworks emerge: Goal-Driven Autonomy (GDA) (Molineaux, Klenk, and Aha 2010), the Goal Lifecycle (Roberts et al. 2016), and Goal Operations and Transformations (Cox, Dannenhauer, and Kondrakunta 2017; Cox and Veloso 1998). Explainable goal reasoning is an open problem which we hope to motivate others in the community to pursue, and has been identified as an important factor for goal reasoning agents that are members of human-robot teams (Molineaux et al. 2018). Although there have been recent examples of goal reasoning agents being deployed as members of human-agent teams in complex domains (Floyd et al. 2017; Gillespie et al. 2015), these agents do not explicitly explain their behavior to human teammates. We highlight the challenges and issues that may arise to create explainable goal reasoning agents in each of these frameworks.

Explaining Goal-Driven Autonomy

- Explainable GDA agents will likely require not only a general trace, but also explanations of components that affect the pursuit of goals besides planning, including: motivator functions (Coddington et al. 2005; Muñoz-Avila, Wilson, and Aha 2015), discrepancy detection (Dannenhauer, Munoz-Avila, and Cox 2016; Karneeb et al. 2018), internal explanation (Molineaux, Kuter, and Klenk 2012), and goal selection. A goal may be abandoned because discrepancy detection (using informed expectations or goal regression expectations) notices an anomaly. A logical follow-up question is *"what was the anomaly?"* and *"why is the anomaly important?"*. When it comes to explaining discrepancy detection and why an action failed,

[2]This example assumes that a drone exists that could perform such an airlift maneuver. This may be unreasonable for current-generation drones, but is used for illustrative purposes.

and why that action is important (especially if using goal regression expectations) we may want to use a dependency graph (Ayan et al. 2007) to demonstrate that a certain part of our goal cannot be reached if this action is not performed.

- Since a human may ask why an agent formulated a new goal g', the agent may need to backtrack to the internal explanation component (e.g., that the explanation is that there is an obstacle), and then backtrack to the anomaly detection (e.g., that action 7 of the current plan failed), and then backtrack to the planning system to say *"I (the agent) chose this plan instead of others because it was more efficient"*.

Explaining the Goal Lifecycle

- The Goal Lifecycle is built on Goal-Task Network (GTN) Planning and an open question remains about how to generate explanations from GTNs. Another open question is whether the questions asked about GTNs differ from those considers by existing work of explainable planning (Fox, Long, and Magazzeni 2017).
- As goals are refined toward completion or backtracking for later processing, a number of metrics (either domain-dependent or domain-independent, like inertia) are maintained. An open question is how to use these metrics in explanations? Another open question is whether the information maintained by the Goal Lifecycle during goal refinement is enough to handle all the questions a user would ask the system. If not, what other information would be needed to answer these questions and how would the formalism change to support this?

Explaining Goal Operations and Transformations

- Performing goal operations and transformations to resolve an unachievable goal to an achievable goal is an open problem. Since goal operations are formulated similarly to planning operations on world states, the trace of goal operations could be stored. Only some goal operations have been formalized completely into operators, and they make use of an ontology. Explanations of why the agent chose to pursue goal g' when the current goal g is unachievable or undesirable may rely on explaining relationships between goal predicates that are in the ontology. The example from (Cox, Dannenhauer, and Kondrakunta 2017), where the goal of *on(B,A)* is generated when *stable-on(B,A)* is unachievable, has to do with the fact that the *on* predicate is closely related to the *stable-on* predicate in the goal predicate ontology.

Common Themes for Explainable GR Agents

- A trace of behavior will be needed to rewind decision making in all three of these frameworks. The formalism of the Goal Lifecycle contains some trace information for explanations and the MIDCA cognitive architecture from (Cox, Dannenhauer, and Kondrakunta 2017) maintains a trace of the cognitive level decision making which includes some goal reasoning decision making: goal selection, goal-related discrepancy detection, and internal explanation. However, some goal operations, such as goal transformation (one method of goal formulation) could occur at the metacognitive level, thus warranting explanations of the meta-level processes. Therefore, a trace of the metalevel could identify the meta-level reasoning processes performed by the agent and provide metalevel explanations.

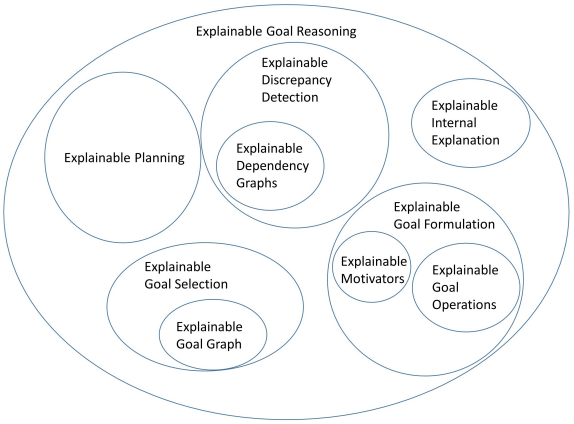


Figure 6: Multi-level overview of goal reasoning processes.

- Complex situations will call for human operators to desire multi-level explanation mechanisms that at the top level answer questions by first identifying which goal reasoning mechanism(s) were involved and then exploring the details of those specific mechanisms. The space of explanations for goal reasoning systems is likely much larger than planning systems without goal reasoning components. A preliminary multi-level overview of goal reasoning components is presented in Figure 6.

  The larger bubbles in Figure 6 represent more general explanation requests, and smaller bubbles represent more narrow requests. In attempting to learn why an agent abandoned a goal, the operator may first ask a high-level question of: *"What caused you to abandon your goal?"*. The agent may give the high-level answer: *"Because I detected an unresolvable discrepancy"*. To answer this question, the agent may first start within the largest bubble, *Explainable Goal Reasoning*, and answer with a response tied to a bubble within that bubble: *Explainable Discrepancy Detection*. When the operator asks a follow up question *"Why was the discrepancy unresolvable?"* the agent may first identify this question falls within the *Explainable Discrepancy Detection* bubble and generate an answer related to the inner bubble *Explainable Dependency Graphs*: *"I could not achieve my goal because I could not execute a specific action that achieves a specific goal condition"*. This example illustrates that explanations may be relevant to different goal reasoning processes. Before answering a question effectively, an agent may need to identify which goal reasoning components are relevant to the question.

Discussion

In the case where the agent rejects a goal, the agent needs to show (to the extent that is possible) that it could not find any way to achieve that goal without violating one of the conditions. And this is where explainable planning, and in particular the use of planning, comes into play. Given that, in general, it is not feasible to formally prove that there are no plans, the agent and the user can use the explainable planning framework (XAIP) (Fox, Long, and Magazzeni 2017) to provide a justification/understanding of why the agent rebelled.

The XAIP framework in this context can be used in two ways:

1. The rebel agent uses the XAIP framework itself, by questioning the current plans (that is not valid) and exploring alternative plans.
2. The agent and the human (or the other agent) use XAIP in cooperation, where the human questions the initial plan (trying to find feasible ways to achieve the goal).

In both scenarios, we envision XAIP being used both during and after scenarios (i.e., online vs. post-run debriefing).

Open Questions for Explainable Rebel Agents

1. Explainable rebellion: How can rebel agents explain why the rejected actions violated their ethical models (e.g., for example rejecting actions that cause harm)? Also, how can rebel agent explain why they chose one goal over another when using a metric-based evaluation (e.g., this may involve reporting that states needed to reach the goal or goal states themselves have low ethical scores)?
2. While there has been some work on implementing goal operations in various frameworks, an agent with all possible goal operations and goal strategies has not yet been realized. Two open problems are: *How can each goal operation / strategy be explained?* and in the case when multiple goal reasoning operations / strategies have been applied *How can multiple explanations be composed in a digestible way?*
3. Will explainable goal reasoning agents in complex domains with complex reasoning components run into potential space problems? Might some questions warrant explanations that are just too expensive to compute? For example, suppose an agent has executed hundreds or thousands of actions since the last time the operator interacted with the agent to give it a goal. During that time, the agent's goal may have changed multiple times due to anomalies (e.g, the goal was suspended, another goal chosen, then later the original goal was re-adopted but the old plan was invalid so agent did re-planning). Whenever the agent changes goals or plans, how many alternative plans should it store to provide evidence of its decisions? Or should the agent save computational and storage time by only computing information needed for explanations when necessary (e.g., when a question is asked)? How does the tradeoff between maintaining complete explanation knowledge and lazy on-demand explanation generation impact the amount of time necessary to generate explanations (i.e., precomputed explanations vs. dynamically generated explanations)? Does the particular domain impact these choices (e.g., the agent's available onboard hardware resources, time-sensitive nature of explanations, expected explanation needs of a particular user)?

Conclusion

An agent's behavior may be the result of changes to both the agent's goals and plans. Thus, an agent that can modify its plans or goals, and also reject plans or goals from teammates, requires increasingly sophisticated methods of explanation. In this paper, we have discussed why explanation is an important factor for rebellious goal reasoning agents and why explainable planning is a key consideration for such agents. In addition to a motivating example illustrating the potential explanations that may be required of such an agent, we have also examined the explanation needs of existing goal reasoning frameworks and many of the open questions that remain. As goal reasoning agents are deploying in increasing complex domains as part of human-agent teams, we hope this discussion will motivate the development of explanation capabilities in goal reasoning agents.

References

Aha, D. W., and Coman, A. 2017. The AI Rebellion: Changing the Narrative. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 4826–4830.

Ayan, N. F.; Kuter, U.; Yaman, F.; and Goldman, R. P. 2007. Hotride: Hierarchical ordered task replanning in dynamic environments. In *Planning and Plan Execution for Real-World Systems–Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop*.

Bidot, J.; Biundo, S.; Heinroth, T.; Minker, W.; Nothdurft, F.; and Schattenberg, B. 2010. Verbal plan explanations for hybrid planning. In *Proceedings of Multikonferenz Wirtschaftsinformatik*, 2309–2320.

Briggs, G., and Scheutz, M. 2015. "Sorry, I Can't Do That": Developing Mechanisms to Appropriately Reject Directives in Human-Robot Interactions. In *Proceedings of the AAAI Fall Symposium on AI for Human-Robot Interaction*, 32–36. AAAI Press.

Briggs, G., and Scheutz, M. 2016. The Case for Robot Disobedience. *Scientific American* 316(1):44–47.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 156–163.

Coddington, A.; Fox, M.; Gough, J.; Long, D.; and Serina, I. 2005. MADbot: A motivated and goal directed robot. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 1680–1681. AAAI Press.

Cox, M., and Veloso, M. 1998. Goal transformations in continuous planning. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*, 23–30. AAAI Press.

Cox, M. T.; Dannenhauer, D.; and Kondrakunta, S. 2017. Goal operations for cognitive systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 4385–4391. AAAI Press.

Cox, M. T. 2016. A model of planning, action, and interpretation with goal reasoning. In *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems*, 48–63. Cognitive Systems Foundation.

Dannenhauer, D.; Munoz-Avila, H.; and Cox, M. T. 2016. Informed expectations to guide GDA agents in partially observable environments. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2493–2499.

European Parliament and Council. 2016. Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union* L119:1–88.

Floyd, M. W.; Karneeb, J.; Moore, P.; and Aha, D. W. 2017. A goal reasoning agent for controlling UAVs in beyond-visual-range air combat. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4714–4721. AAAI Press.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. In *Explainable Artificial Intelligence: Papers from the IJCAI Workshop*.

Gillespie, K.; Molineaux, M.; Floyd, M. W.; Vattam, S. S.; and Aha, D. W. 2015. Goal reasoning for an autonomous squad member. In *Goal Reasoning: Papers from the ACS Workshop*, 52–67.

Gregg-Smith, A., and Mayol-Cuevas, W. W. 2015. The design and evaluation of a cooperative handheld robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1968–1975. IEEE.

Karneeb, J.; Floyd, M. W.; Moore, P.; and Aha, D. W. 2018. Distributed discrepancy detection for beyond-visual-range air combat. *AI Communications* 31(2):181–195.

Langley, P.; Meadows, B.; Sridharan, M.; and Choi, D. 2017. Explainable agency for intelligent autonomous systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 4762–4764. AAAI Press.

Molineaux, M.; Floyd, M. W.; Dannenhauer, D.; and Aha, D. W. 2018. Human-agent teaming as a common problem for goal reasoning. In *Proceedings of the AAAI Spring Symposium on Integrating Representation, Reasoning, Learning, and Execution for Goal Directed Autonomy*. AAAI Press.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010. Goal-driven autonomy in a Navy strategy simulation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press.

Molineaux, M.; Kuter, U.; and Klenk, M. 2012. DiscoverHistory: Understanding the past in planning and execution. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems*, 989–996. IFAAMAS.

Muñoz-Avila, H.; Wilson, M. A.; and Aha, D. W. 2015. Guiding the ass with goal motivation weights. In *Goal Reasoning: Papers from the ACS Workshop*, 133–145.

Roberts, M.; Shivashankar, V.; Alford, R.; Leece, M.; Gupta, S.; and Aha, D. 2016. Goal reasoning, planning, and acting with ActorSim, the actor simulator. In *Poster Proceedings of the 4th Annual Conference on Advances in Cognitive Systems*.

Smith, D. E. 2012. Planning as an iterative process. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2180–2185. AAAI Press.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. AAAI Press.

Vattam, S.; Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Breadth of approaches to goal reasoning: A research survey. In *Goal Reasoning: Papers from the ACS Workshop*.

Action Selection for Transparent Planning

Aleck M. MacNally, Nir Lipovetzky, Miquel Ramirez and Adrian R. Pearce
The University of Melbourne
Melbourne, Australia
aleck.macnally, nir.lipovetzky, miguel.ramirez, adrianrp@unimelb.edu.au

Abstract

We introduce a novel framework to formalize and solve *transparent planning tasks* by executing actions selected in a suitable and timely fashion. A *transparent planning task* is defined as a task where the objective of the agent is to communicate its true goal to observers, thereby making its intentions and its action selection *transparent*. We formally define and model these tasks as Goal POMDP's where the state space is the Cartesian product of the states of the world and a given set of hypothetical goals. Action effects are deterministic in the world states of the problem but probabilistic in the observer's beliefs. Transition probabilities are obtained from making a call to a model–based plan recognition algorithm, which we refer to as an *observer stereotype*. We propose an action selection strategy via on–line planning that seeks actions to quickly convey the goal being pursued to an observer assumed to fit a given stereotype. In order to keep run–times feasible, we propose a novel model–based plan recognition algorithm that approximates well–known probabilistic plan recognition methods. The resulting on–line planner, after being evaluated over a diverse set of domains and three different observer stereotypes, is found to convey goal information faster than purely goal–directed planners.

Introduction

Understanding the intentions and plans of agents, humans or otherwise, has been identified as crucial in key AI research areas such as intelligent user interfaces, dialogue in natural language, cooperation in multi–agent systems, and assisted cognition (Carberry 2001; Cohen, Perrault, and Allen 1981; Pentney et al. 2006; Yang 2009) and substantial literature exists proposing several different formal and computational frameworks (Avrahami-Zilberbrand and Kaminka 2005; Geib and Goldman 2009; Ramirez and Geffner 2010).

On the other hand, the complementary problem of *assisting* an observer to determine the goal being pursued, has received less attention (Chakraborti et al. 2017). Informally, and from the perspective of model–based approaches to planning and plan recognition, this is a problem of planning – seeking actions – under specific constraints or in response to *feedback* obtained from the observer. In turn, the ability of autonomous systems to generate behaviour that is, according to some measure of efficiency, *easy* to interpret is identified as a critical and complementary component

This paper has been published at AAMAS 2018

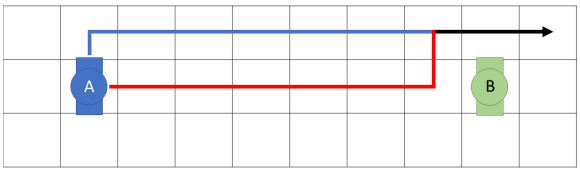


Figure 1: Shows a person, **A,** walking in a corridor who must avoid a collision with a second person, **B**. Two possible paths are shown for person **A**.

to robust intention recognition in human–in–the–loop systems and joint human–robotic teams operating in dynamic environments. Existing approaches reformulate the planning model used by the acting agent (*actor*) in a way that is beneficial to the agent that observes (*observer*). Chakraborti et al (2017) propose a framework for *Model Reconciliation* that aims to elicit changes on-line in planning models so that the cost optimal plans pursued by the actor match those considered by the observer. Moving the focus away from the dialogical dimension of the problem, Keren et al (2014) *Goal Recognition Design* aims at redesigning the environment where plans are executed to guarantee that the degree of ambiguity in optimal plans for a given set of goals is bounded. Interestingly, recent work (Masters and Sardina 2017) aims at *exploiting* inherent ambiguity in the structure of the environment to *delay* the identification of goals. In this paper we present a formal and computational framework for transparent planning, when *on–line co–ordinated* or *off–line* model reformulation is not suitable or possible. We formalize the problem and model it using the Goal POMDP framework (Kaelbling, Littman, and Cassandra 1999), and the Functional STRIPS language to describe states and actions (Geffner 2000; Bonet and Geffner 2000). The goal pursued is then a constraint on the posterior probabilities that the observer assigns to a set of *hypothetical* goals, so that the *actual* goal is the best *predictor* for the actions executed up to a given point in time. We propose an on–line, polynomial and approximate algorithm to select actions, that embeds a model–based plan recognition algorithm to procedurally determine the Goal POMDP transition probabilities required to track the progression of beliefs. We evalu-

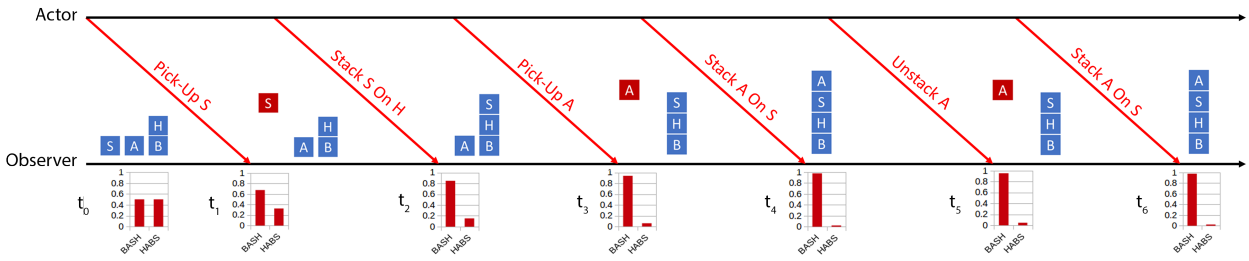


Figure 2: Shows the change in the beliefs of an observer in response to the actions it observes the actor executing. Time flows left to right. The beliefs are shown in the graphs at the base of the figure. Next to each action is a diagram illustrating the state following that action.

ate the resulting planner over several domains, which in this work are restricted to actions whose effects over state variables modelling the environment are deterministic. The results obtained indicate that the proposed scheme, accelerates the convergence of the observer's beliefs towards a probability distribution with the desired properties, and is robust with respect to the assumptions made on the posterior used by the observer to interpret observed actions.

Motivation

It is important in robot-human co-operative teams that each member knows the intentions and roles of their collaborators (Stubbs, Wettergreen, and Hinds 2007). In this paper we focus specifically on how an agent may communicate its goals to its fellow teammates. An obvious method of achieving this is to have the agent explicitly transmit all relevant information to observing agents. This can be achieved through some form of communication protocol, such as, when the observer is human, a computer screen or through natural language. Unfortunately this solution is not as straight-forward as it seems. When the observing agent is human, the internal state of an acting agent must be distilled in such a way that the information relevant to humans is accessible. However, what is accessible may depend on the human. Issues with training, language and culture will affect whether they can correctly comprehend any communication.

An explicit communication protocol requires that both communicating parties understand the protocol in its entirety. *Transparent Planning* represents a method for communicating information implicitly by acting in a way which is recognizable to observers. In comparison to explicit communication methods this form of implicit communication only requires that there is agreement regarding what constitutes natural behaviour, rather than a complete protocol.

We can see *transparent planning* used in human-human collaboration especially in cases where there is no shared communication protocol such as between an adult and a toddler (Warneken and Tomasello 2006). In the work by Warneken et al. (2006) an adult executes actions, which are chosen because they indicate what the adult is attempting to achieve, to communicate to an observing toddler. In this case the toddler does not yet understand languages but can interpret the intentions of the adult by reasoning about its actions, and then it may choose to assist.

In cases where there is an established communication protocol, there is also the problem of communicating over a channel. The form of this channel greatly changes the difficulty of the communication. For instance, if we take the example of a human supervising a team of robots, it is clear that if each of the robots was audibly broadcasting its intentions, the human would be unable to process the barrage of information. Communicating by acting transparently mostly uses a visual communication channel which, whilst subject to its own set of problems, will be useful in cases where other channels are intractable, as in the above example.

Illustration Figure 1 shows a situation in which two people in a corridor must navigate around each other in order to pass. For a successful solution each human must know which side the other will pass on. In this case **A** has decided that they will pass **B** on the their right-hand side. It is imperative that **B** understand **A**'s intention to avoid a collision. Two possible paths are illustrated in Fig. 1 from which **A** may choose. The blue path clearly communicates **A**'s intentions in advance of a possible collision, while the red path does not. In situations like this, humans do not generally broadcast their intentions audibly to other humans but instead act transparently, as in this example by moving to the decided side early on in their plan. We seek to produce this behaviour in human-robot collaborations.

In this work we do not require that the agent who is acting *transparently* commits to achieving its goal *and* communicating it *simultaneously*. The actor's objective is to communicate its goal as efficiently as possible regardless of how far this might remove the actor from the goal. We wish to have the planner use actions as the vocabulary for describing its intentions rather than for their true effects. For example in Figure 2 we can see the interaction of a human and a robot over time. The robot is planning in the BLOCKS WORDS domain (Ramirez and Geffner 2009), in which the robot must assemble alphabet blocks into a goal configuration by picking them up and putting them down. The set of possible goals $\mathcal{G}$ consists of two words: BASH and HABS. The robot's goal is BASH. The initial state can be seen on the far left of Figure 2. The optimal action for both goals is

to pick up H and then put it on the table, but if the actor executes these actions the observer will only see a partial plan that is equally good for both goals. So instead it picks up S which only occurs optimally in plans to achieve the word BASH at t_0. It then *composes* the word ASH on top of the B. This clearly conveys the goal of the actor very rapidly as can be seen by the graphs at the bottom of the figure indicating the beliefs of the observer. Once the actor has written ASH it can do no more with its limited vocabulary and therefore un-stacks A as that is its only available action. At this point we may assume that an observer will have gauged the goal of the actor already and the actor may, therefore, act towards achieving the goal. If the actor assumes that the observer is not convinced it may instead again stack A on top of S, in an attempt to convey its goal with a limited vocabulary. This behaviour is reminiscent of that required by the adults in the research by Warneken et al. (2006), in which an adult repeatedly knocks into a closed cupboard to indicate to a toddler that it intended to place an object into the cupboard.

Related Work

The literature combining plan recognition and automated planning generally focuses on recognizing the intentions of exogenous agents and then acting in response to those intentions (Freedman and Zilberstein 2017), this work in contrast focuses on what an observer would gauge from observing the agent under our control. Much of the literature considering the effects actions have on the mental model of observers has been focused in the area of human-aware planning (Alami et al. 2006), in particular in the area of plan explanation. Chakraborti et al. (2017) considered plan explanation by taking into account differences between the actor's action model and the actor's model assumed by the human. They formulated a method for *reconciling* the human and robot action models by suggesting changes to the human model. In contrast, we do not intend to change the human action model but to generate the most unambiguous behaviour given an *observation* model. Zhang et al. (2017) also considered the difference between the actor's action model and the human's. They use this difference to produce plans expected by a human who has full knowledge of the goal of the actor, bypassing the need to explain actions. We produce plans without assuming the observer *knows* the actor's goal, we instead select actions to communicate this goal to the observer.

We take direct inspiration from Keren et al. (2014) work on *Goal Recognition Design*, where a point is made that even if actors plan optimally, it may still be difficult to identify the goal being pursued in a timely fashion. From this observation, we note that seeking a plan which is not necessarily optimal may allow an agent to convey its goal more efficiently.

An orthogonal work to the focus on explaining actions and, more generally, directing observers towards the goals that motivate plans, recent work by Masters and Sardina (2017) explores the active manipulation of observers' beliefs, proposing algorithms to produce *deceptive* plans in the path–planning domain. We manipulate beliefs too, but with the exact *opposite* intent and over *general* planning models.

Finally, this work is inextricably linked to the notion of model–based, probabilistic plan recognition (Ramirez and Geffner 2010). *Transparent Planning* internalizes this idea and uses it as an *element* of the planning model, so the evolution of observer beliefs over time is accounted for *explicitly* while seeking actions and plans.

Planning over Goal POMDPs

Goal POMDPs (Partially Observable Goal MDP) are a well–known framework (Kaelbling, Littman, and Cassandra 1999) for agents that have incomplete state information and receive indirect evidence of the actual state of the world via sensors or by their own reasoning. In this paper we mostly follow Bonet & Geffner's (2013) presentation of Goal POMDPs and define them as a tuple $M = \langle S, b_0, b_G, A, tr, c, \Omega, q\rangle$ whose elements are as follows. S is a non-empty, discrete and finite *state space*. b_0 is the *initial belief* state, a probability distribution $P(s)$ over *every* $s \in S$, representing the probability that the agent is in state s. We seek actions that transform b_0 into a new belief state in which the set of *constraints* on the state probabilities, b_G, is satisfied. The set of *actions* that an agent may execute in a state, s, is given by the function $A(s)$ and $c(a, s)$ is the cost of performing action a in s, which is a positive real number. tr is the state transition function and $tr(s'|s, a)$ gives the probability that the agent will transition into state s' after performing a in s.

Feedback from sensors consists of a finite set of *observation tokens* Ω. The probability of a token, $\omega \in \Omega$, being obtained having performed action a in state s is given by the *sensor model* $q(\omega|s, a)$.

A common way to solve POMDPs is to formulate them as completely observable MDPs over the *belief states* of the agent (Astrom 1965; Sondik 1978). While the effects of an action will have on a state may not be exactly predicted, the effects of actions on *belief states* can. Formally, the belief b_a that results from doing action a in belief b, and belief b_a^ω that result from observing ω after doing a in b, are:

$$b_a(s) = \sum_{s' \in S} tr(s'|s, a)b(s')\,, \tag{1}$$

$$b_a(\omega) = \sum_{s \in S} q(\omega|s, a)b_a(s)\,, \tag{2}$$

$$b_a^\omega(s) = q(\omega|s, a)b_a(s)/b_a(\omega) \quad \text{if } b_a(\omega) \neq 0. \tag{3}$$

As a result, the *partially observable* problem of going from an initial state to a goal state is transformed into the *completely observable* problem of going from one *initial belief state* into a *goal belief state*. We also note that MDPs where $tr(s'|s, a) > 0$ for exactly one state s' map directly onto deterministic state models (Geffner and Bonet 2013).

We use the Functional STRIPS (Geffner 2000) *language* to describe declaratively states and transitions, as Ramirez and Geffner (Ramirez and Geffner 2011) did, for the domains discussed in the Evaluation Section. Out of the full expressiveness of Functional STRIPS we only use a subset, we note the features required by our benchmarks next. State variables X have their domains to be either Boolean, and

hence modeling the truth value of some arbitrary atomic formula, or a subset of $\mathbb{N}$ with small cardinality. S then corresponds to the set of possible valuations of X, and the value of x on state s is noted as $[x]^s$. Actions are described by a *precondition* list, $Pre(a)$, which is a list of literals of X, and an *effect* list, $Eff(a)$, which is a list of updates of the form $x := \top$ or $x := \bot$, where x is a variable with Boolean domain.

We refer the reader to (Francès and Geffner 2015) and (Bonet and Geffner 2001) for more detailed accounts of Functional STRIPS, and its extensions to deal with probabilistic effects and partially observable states, respectively.

Probabilistic Plan Recognition

A *probabilistic plan recognition problem* (Ramirez and Geffner 2010) is a tuple $T = \langle X, A, I, \mathcal{G}, O, Prob \rangle$ where:

- X and A are a set of Functional STRIPS variables and actions describing a *planning domain*,
- $I \in 2^X$ is an initial state,
- $\mathcal{G}$ is a set of candidate goal formulae from which a goal will be chosen,
- O is an *observation sequence*,
- $Prob$ is a prior probability distribution over $\mathcal{G}$, which represents previous knowledge of the likelihoods of each goal.

We note that T can be also defined over a *plan library* (Avrahami-Zilberbrand and Kaminka 2005; Geib and Goldman 2009), which is a set of sequences of actions $\pi = (a_0, \ldots, a_n)$, elicited from selecting plans that map initial state I into states that satisfy a goal $G \in \mathcal{G}$.

We assume that an observer's beliefs follow from the probability distribution, $P(G|O)$, where $G \in \mathcal{G}$, which is the solution to a plan recognition problem, T. We also assume that these *posterior* probabilities are computed via the application of Bayes' Rule as:

$$P(G|O) = \alpha\, P(O|G)\, P(G) \tag{4}$$

where α is a normalizing constant, and $P(G)$ is $Prob(G)$. Several definitions of likelihoods $P(O|G)$ have been put forward in the literature on model–based plan recognition, this work examines the following three. The first one, which we refer to as RG09, adopts a hard postulate on the rationality of agents pursuing hypothetical goals (Ramirez and Geffner 2009).

$$P(O|G) = \begin{cases} 1 & |c(G,O) - c(G)| = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $c(G, O)$ is the cost of a plan for G *constrained* to be consistent with observation sequence O, and $c(G)$ is the cost of a plan for G *without* constraints. $c(*)$ maybe computed with an optimal planner or may be approximated by a satisficing planner or heuristic. The second definition, also by Ramirez & Geffner (2010) and noted as RG10, adopts a softer postulate, preferring hypothetical goals G as an explanation for O when they minimize the difference between $c(G, O)$ and $c(G, \bar{O})$, the latter being the cost of a plan for G which is constrained to be *inconsistent* with O, that is a plan which does not feature the observations in sequence. Assuming a Boltzmann distribution over plans according to their cost, and writing $exp\{x\}$ for e^x, likelihoods $P(O|G)$ are defined as:

$$P(O|G) = \frac{1}{1 + \exp\{-\beta\, \Delta(G)\}} \tag{6}$$

where $\Delta(G) = c(G, O) - c(G, \bar{O})$ and β is a positive constant. It is often noted that $\Delta(G)$ can be sometimes difficult to calculate when the observation sequence O contains actions which are *goal landmarks* (Hoffmann, Porteous, and Sebastia 2004). In that case $c(G, \bar{O})$ can increase substantially – or be ∞ in extreme cases – leading run–times to increase significantly. Also, while Ramirez & Geffner (2010) allows O to deviate from the plans selected by the observer as the *best plan* for G, inherent in the approach is the commitment to *one* specific best plan, amongst potentially many plans with the same associated cost. Various approaches have been proposed to avoid these issues while allowing for a softer stance on rationality (Martin, Moreno, and Smith 2015; Sohrabi, Riabov, and Udrea 2016; Pereira, Oren, and Meneguzzi 2017). In this work we will concentrate our efforts to study the most recent one as it results in run–times similar to library–based approaches, after an off-line and potentially expensive elicitation of landmarks, and has been shown empirically to provide very accurate approximations to RG10. Following from the discussion in Pereira et al (2017) of possible *heuristics* to select hypothetical goals as explanations for O, we define the likelihood of O given a hypothetical goal G in terms of landmarks:

$$P(O|G) \propto \frac{|O \cap \mathcal{L}(G)|}{|\{G' | \mathcal{L}(G') \cap \mathcal{L}(G) \neq \emptyset, G' \in \mathcal{G}\}|} \tag{7}$$

in words, directly proportional to the degree O *covers* the set of landmarks for goal G, $\mathcal{L}(G)$ and inversely proportional to the degree of overlap between sets of landmarks of hypothetical goals $\mathcal{G}$. We will refer to this last approach as POM17.

Planning Transparently

While Plan Recognition is usually described as a multi–agent setting, only recently (Zhang et al. 2017) models and frameworks have been proposed that address explicitly the possible interactions between the agent being observed, or *actor*, and the agent observing the actor, which we will refer to as the *observer*. In particular we drop the key assumption in so–called *keyhole* Plan Recognition (Schmidt, Sridharan, and Goodson 1978; Kautz and Allen 1986) where the actor does not take an interest on what the observer may do or believe about its actions. We will next propose *one possible* planning model for an actor that both *knows* about the observer and *wants* to influence it in some way. In order to do so, the actor may eventually have to follow a course of action that deviates from the most efficient plan for its *actual* goal $G^* \in \mathcal{G}$, or temporarily abandon its pursuit, to pursue instead changing the observer's initial beliefs to consider G^* *more likely*, while driving those *away* from other hypothetical goals $G' \in \mathcal{G}$.

We define a transparent planning task with the tuple $\Pi = \langle X, A, s_0, \Omega, q(\cdot|\cdot), \mathcal{G}, G^*, P(\cdot|\cdot), Prob \rangle$, where:

- X and A are a set of Functional STRIPS variables and actions describing a *planning domain*,
- s_0 is a given valuation of variables X,
- $\mathcal{G}$ is a set of possible goal formulae G over literals of X, with one distinguished member G^* which models the actor's goal,
- $\Omega \subseteq X$ and $q(\phi|\psi, a)$ allow us to specify a *sensor model* when the actor has limited access to the values of X, where $a \in A$, ϕ is a conjunction of literals of every variable in Ω, and ψ is a conjunctive formulae over X,
- $P(\cdot|\cdot)$ is the *definition* posterior goal distribution $P(G|O)$ which is *assumed* by the actor to be used by the observer to make sense of its behaviour,
- $Prob$ is a *prior* probability distribution over $\mathcal{G}$.

We will refer to pairings of distributions $P(\cdot|\cdot)$ and $Prob$ as the *observer stereotype* or *stereotype* for short. Last, Π shares elements both relevant to planning and goal recognition models discussed elsewhere in the paper, this makes apparent that the actor both reasons about his own goal G^*, and influences the observer's beliefs as it generates observation sequences O made of actions $a \in A$ chosen in a suitably defined manner.

Amongst the several possible models discussed by the planning community over the years, we chose that of Kaelbling's (1999) Goal POMDPs. The transparent planning task Π can be compiled into a Goal POMDP $M(\Pi)$ as follows. States S are valuations over variables $X' = X \cup \{y\}$, y is the non–observable state variable used to keep track of the goal the observer *knows* the actor to pursue. The domain of y is given by the set $D(y) = \{KG | G \in \mathcal{G}\}$, where KG are constant symbols that stand for "the observer *knows* $G = G^*$". Such a statement is only true when indeed, the observer has identified *precisely* the goal G^*, as it is false for every goal $G' \neq G^*$, $G' \in \mathcal{G}$. Furthermore, we note that states s such that $[G]^s = \top$ and $y \neq KG$ are valid. The reason for allowing the mental state of the observer to be independent of the actual truth value of the goals $\mathcal{G}$ in a given state, is to avoid the observer beliefs diverging from the distribution $P(\cdot|\cdot)$. Such a thing can occur if we forced $y = KG$ whenever $[G]^s$ is true, such as when there exists a goal $G^{'} \in \mathcal{G}$ which has G as a landmark. Introducing such a functional dependency between X and y can indeed be justified, but it is not of immediate interest to us.

The initial belief b_0 is defined as $b_0(s) = Prob(G)$ for every state $s = s_0 \wedge (y = KG)$ with $G \in \mathcal{G}$ and $b_0(s') = 0$ otherwise. Actions a do not change the value of y so the transition probability distribution follows from

$$tr(s|s', a) = \begin{cases} P(\{a\}|[y]^{s'})P(\sigma|a) & s = f(s', \sigma) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $f(s', \sigma)$ is a function which updates s' with the set of assignments σ, and $P(\sigma|a)$ is the probability of σ given the action a (Bonet and Geffner 2001).

A belief b is a goal belief if b satisfies constraints b_G:

$$\sum_{s \in S,\, [y]^s = G^*} b(s) \geq \frac{1}{|\mathcal{G}|} + \max_{G' \in \mathcal{G} \setminus G^*} \sum_{s \in S'\, [y]^s = G'} b(s) \quad (9)$$

We want to note that alternative definitions of b_G are possible. For instance, fairly intuitive ones such as that of G^* being part of the set

$$\operatorname*{argmax}_{G' \in \mathcal{G}} \sum_{s \in S,\, [y]^s = G'} b(s) \quad (10)$$

could be useful when $Prob$ is *not* a i.i.d. over $\mathcal{G}$, otherwise b_0 would trivially satisfy the constraint. We use equation 9 as it requires that the true goal be salient beyond a normalization factor.

The tracking of beliefs in Equations 1–3 can be simplified in the following way for the domains discussed in the Evaluation section. First, in these domains the agent has access to all values in X and may not directly or indirectly know the value of y, so our sensor model becomes $\Omega = \{\perp\}$, $Q_a(\omega|s) = 1$ for every s and a. As a result of this, Equation 1 and Equation 3 have one and the same expression, i.e. $b_a(s) = b_a^{\perp}$. Second, actions in A have *deterministic* effects so the belief update in Equation 1 becomes

$$b_a(s) = \sum_{s' \in S} tr(s|s', a) b(s') = \sum_{s' \in S} P(\{a\}|[y]^{s'}) b(s') \quad (11)$$

The two simplifications above apply to fully observable, deterministic domain theories as in the case of the problems we have used in this paper. For this reason, instead of compiling Π to a Goal POMDP we can instead compile it to a *factored state model* (Frances et al. 2017) and use scalable width–based algorithms such as BFWS (Lipovetzky and Geffner 2017a) to efficiently search for sequences of actions to achieve a goal belief b_G.

The reasons for having chosen Kaelbling's classic framework over recently proposed ones such as Bonet's & Geffner (Bonet and Geffner 2016) that allows belief tracking factorization, are exposed next. First, we note that Goal POMDPs can be compiled into Bonet & Geffner framework without loss of generality. Second, and from a purely empirical standpoint, the models $M(\Pi)$ that result from the domains discussed in the Evaluation Section always have a *causal width* of 1 and hence, no computational advantage of using Bonet's & Geffner framework was to be reported, since time complexity of belief tracking is *linear* over $|\mathcal{G}|$. A recent framework we would have *wanted* to use but could not because it does not support probabilistic beliefs at the time of writing this, is the full–fledged multi–agent on–line planner recently proposed by Kominis & Geffner (2017). We look forward to reconcile this work with that of Kominis in the near future.

Action Selection

The action selection problem over the transparent planning task Π can be addressed as a *net-benefit planning problem* (Keyder and Geffner 2009) where the reward function is defined over beliefs b. The reward $R(b) = -d(b, b_T)$ considered in this work corresponds to the negative of the Euclidean distance between belief b and *target* belief state b_T, that satisfies the constraint b_G in Equation 9. We set b_T so that $P(G^*|O) = 1$, and $P(G|O) = 0$ for $G \in \mathcal{G} \setminus G^*$, which

stands for the observer being *absolutely* certain of G^* being the goal the actor pursues. In order to compute the Euclidean distance d, the probabilities assigned by belief b to goals $G \in \mathcal{G}$ are treated as real–valued vectors of dimension $|\mathcal{G}|$.

A valid plan is a sequence of actions $\pi = \langle a_0 \ldots, a_n \rangle$ that induces the sequence of observations $O_\pi = \langle a_0, \ldots, a_n \rangle$ and the sequence of belief states $\langle b_0, \ldots, b_n \rangle$, such that a_0 is applicable in b_0, $b_i = b_{a_i}$ as per Equation 11, and $b_n \models b_G$. The utility of a plan π is defined as

$$u(\pi) = \frac{\sum_i^{|\pi|} R(b_i)}{|\pi|} \tag{12}$$

in words, the average accumulated reward, and a plan π is optimal if there is no other plan π' with higher utility. Note that plan length $|\pi|$ is only used to normalise $u(\pi)$, as we set the task to seek sequences π that maximise the *average* similarity between beliefs b_i and target belief b_T.

Features over Beliefs for Width–Based Search

Width-based algorithms have been shown to scale up for classical planning problems, factored state models, and net-benefit problems (Lipovetzky, Ramirez, and Geffner 2015; Lipovetzky and Geffner 2017a; Frances et al. 2017). In this work we use BFWS, a best-first search that balances exploration and exploitation. BFWS ranks nodes in the open list by novelty (exploration term), and breaks ties by a heuristic function (exploitation term). The novelty of a newly generated state $w(s)$ is the size of the smallest subset (conjunction) Q of atoms $[x]^s$ true in s and false in all the states s' generated before s, i.e. $w(s) = min_{Q \subseteq s, Q \not\subseteq s'} |Q|$ (Lipovetzky and Geffner 2012).

For the transparent planning task we extend the computation of novelty to a belief state by taking into account not only the state variables valuation $[x]^s$, but also the quantities $[Y_G]^s = \sum_{s \in S, [y]^s = G} b(s)$ corresponding with the posterior probability $P(G|O)$ assigned by the observer to each goal $G \in \mathcal{G}$. The observation sequence O corresponds to the partial plan leading to state s. Features Y_G take a precision of two decimal digits[1]. The features used to compute novelty are then $F(s) = \{ [x]^s \mid x \in X \} \cup \{ [Y_G]^s \}$, both the valuation of state variables $x \in X$ and the posteriors Y_G for each goal $G \in \mathcal{G}$. The novelty of a state becomes $w(s) = min_{Q \subseteq F(s), Q \not\subseteq F(s')} |Q|$. Extending the definition of novelty with additional features have already shown benefits in classical planning (Frances et al. 2017). Last, the exploitation term $h(s) = u(\pi)$ is set to be the utility of the partial plan π leading to s, making BFWS to prefer first novel states, breaking ties by utility.

In order to cope with the size of the state-space, and taking advantage that net-benefit problems can be solved by *online* algorithms, we adapt BFWS to solve the online action selection problem, where given a *budget* expressed in terms of time, number of generated states or simply as a search horizon on the maximum plan length, the algorithm has to return the first applicable action leading to the state with highest utility. For this setting we take advantage of a polynomial variant of BFWS where nodes whose novelty $w(s) > 1$ are pruned (Lipovetzky and Geffner 2017b). The benefit of the polynomial version known as 1-BFWS is that it does not require a budget, it simply stops when no more nodes can be expanded. Note that 1-BFWS can generate at most $\mathcal{D}_X \times \mathcal{D}_{Y_G}$ states, where $\mathcal{D}_X = \sum_{x \in X} |D(x)|$ is the sum of domains D cardinality for each state variable, and $\mathcal{D}_{Y_G} = \sum_{G \in \mathcal{G}} |D(Y_G)|$ is the sum of the cardinalities of state features' domains. As soon as 1-BFWS reaches a state s that satisfies the goal belief state b_G, then the search is stopped and the first action leading to s is returned. If no such state exists, then the search stops when no more nodes can be expanded and the action leading to the state with highest utility is returned.

We finish observing that the instance of BFWS above does not *necessarily* select actions (paths) which lead to beliefs b where G^* is true in every state s s.t. $b(s) > 0$, and therefore, guarantee that the true goal has been achieved. In practice, once G^* has been confirmed by the observer in some way, e.g. via some observation token which rules out any state where $y \neq KG^*$, one could switch to a purely goal–directed planner.

Approximating $\mathbf{P(O|G)}$ for Belief Tracking

The computational bottleneck to solve the transparent planning problem is the derivation of Δ values. We extend Ramirez & Geffner (2009) approximation for $C(G, O)$, based on Hoffmann's h_{FF} heuristic (2001), to approximate $C(G, \bar{O})$. While the value of h_{FF} on the tasks that result from compiling away O may be reasonably informative, as noted by Ramirez & Geffner (2010), it generally is much less informative for the planning task constrained to avoid plans consistent with O, as the constraints are satisfied already in the initial state.

The cost of achieving goal G without satisfying the observation sequence O, is the cost of a plan which either removes one or more observations, or changes their sequence order. Hence, in order to break the sequence of operators it is sufficient to *force* an observation to appear with a different index. The following transformation extends Ramirez & Geffner (2009) in order to achieve this.

Given an observation sequence O and observation $a_i \in O$, we construct the FSTRIPS planning problem $\Pi = \langle X', s, A', G \rangle$ required to evaluate $P(\cdot|\cdot)$ in Equation 11 as follows. The set of variables X' results from augmenting X with Boolean state variable $next$. A' is made up of copies a' of each $a \in A$, where $Pre(a') = Pre(a) \wedge next$ when $a = a_i$, $Pre(a') = Pre(a)$ otherwise. Effects of copies a' for actions $a \neq a_i$ are $Eff(a') = Eff(a) \cup \{next := \top\}$. Finally the initial state $s = f(s_0, O')$ is set to the state resulting from the application of the observation sequence $O' = \langle a_0, \ldots, a_{i-1} \rangle$ preceding observation a_i. The transformed planning problem ensures that the observation a_i is out of sequence as it is not applicable on the ith step of a plan, and importantly not even in the delete-relaxed plan that induces the value of h_{FF}. This is achieved by making sure that a different observation action that adds the variable $next$

[1] Features need to be bounded, otherwise their domain becomes non-finite.

is used first. This transformation is applied to every observation $a_i \in O$. The cost of an optimal plan in Π_i is approximated by h_{FF}. The cost $C(G, \bar{O})$ is then set to minimum cost for breaking the observation sequence O:

$$C(G, \bar{O}) = \min_{i=0,\ldots,|O|} c(\pi_{i-1}) + h_{FF}(\Pi_i, I_i) \qquad (13)$$

where $c(\pi_{i-1})$ is the cost of the observation sequence leading to a_i. Note that for each search node we do not require computing h_{FF} $|O|$ times, $C(G, \bar{O})$ can be computed from the best $C(G, \bar{O})$ value leading to the parent node.

Evaluation

Domain	RG10	RG09	POM17	IGC T	LAMA T
BLOCKS WORD	0.565	0.813	1.028	0.344	0.193
GRID NAVIGATION	0.523	0.523	1.000	0.431	0.21
GRID NAVIGATION + OBS	0.649	0.649	1.000	0.433	0.21
TICKET TO RIDE	0.571	0.634	0.959	0.654	0.386
CAMPUS	0.854	0.828	1.344	0.437	0.353
INTRUSION-DETECTION	0.774	0.774	1.919	0.535	0.244
LOGISTICS	0.442	0.369	1.011	0.21	0.556
KITCHEN	0.778	0.444	1.756	0.237	0.19
ROVER	0.750	0.597	1.775	0.34	0.23

Table 1: Evaluation of IGC and LAMA. IGC T and LAMA T is average time in seconds to select an action. Columns RG10, RG09, POM17 are average ratios $Q_\upsilon(IGC)/Q_\upsilon(LAMA)$ for the three stereotypes considered.

In order to test the usefulness of the proposed framework, algorithms and approximations discussed in the previous Sections, we have implemented an on–line planner for transparent, IGC (implicit goal communication), using the LAPKT planning framework (Ramirez, Lipovetzky, and Muise 2015). This planner, sets $P(\cdot|\cdot)$ to the approximation discussed in the previous Section, fixing $\beta = 1$ in Equation 6. A general planner for Π trivially follows from attaching procedures to the planner which provide the denotation of FSTRIPS terms corresponding to the posterior goal probabilities (Frances et al. 2017).

It may be that a regular goal-directed planner communicates its goal efficiently in the course of its plan execution, in this case there would be very little utility in the proposed methods. To determine whether this is true we evaluate IGC against the satisficing classical planner LAMA (Richter and Westphal 2010) with which we project away the variable y. Action selection in LAMA is driven mainly by the goal–directed reachability and landmark–based heuristics used to determine *helpful actions*/*preferred operators*. Run–times, quantity T in the tables, are put forward for reference only, as action selection in LAMA requires the computation of a *full* plan for goal G^* and then to commit to its first action.

We have tested IGC and LAMA over 9 domains: GRID NAVIGATION with and without obstacles, TICKET TO RIDE, BLOCKS WORD, CAMPUS, INTRUSION-DETECTION, LOGISTICS, KITCHEN and ROVER. CAMPUS, INTRUSION-DETECTION and KITCHEN are domains which have been compiled from plan libraries (Ramirez 2012), whereas the BLOCKS WORD, LOGISTICS and ROVER are domains from the International Planning Competition discussed in the literature on plan recognition as planning (Ramirez and Geffner 2010; Pereira, Oren, and Meneguzzi 2017). The GRID NAVIGATION domain specifies a simple navigation task on a 2–dimensional, 4–connected grid. The TICKET TO RIDE domain is a simplified version of the popular board game of the same name[2]. The task requires the connection of nodes representing in–game locations. Connecting two nodes is only possible if enough game pieces of a given colour have been drawn. Actions in every domain have deterministic effects and unitary costs. Experiments were conducted on a i7-6700HQ CPU running at 2.60GHz and the physical memory available to the planners was limited to 8Gb of RAM.

Experiments *simulate* the effects of the actions selected by IGC and LAMA on the beliefs of the observer. For each instance, and domain, three *observer stereotypes* are tested, one for each of the three methods, RG09, RG10 and POM17, discussed in the Section "Probabilistic Plan Recognition" to define $P(O|G)$, along with uniform priors $Prob$. The difference between the landmark graph used by LAMA and POM17 observer is that POM17 uses *causal* landmarks and no disjunctive landmarks (Keyder, Richter, and Helmert 2010; Richter 2011). It is an open question to measure the sensitivity of the observed results to the choice of algorithm to approximate action landmark sets. The simulator progresses $M(\Pi)$ as actions are requested from the planner and executed, this process continues up to the point that the current belief b satisfies b_G in Equation 9, producing a sequence of actions π. We measure performance comparing the length of sequences π generated, we denote this value as $Q_\upsilon(X)$, where υ is the *observer stereotype* (RG09, RG10, POM17) and X is a planner (IGC, LAMA).

Table 1 compares the average lengths of generated action sequences π between IGC and $LAMA$, and the results clearly show a stark difference in the performance of IGC when switching the cost–based stereotypes (RG09, RG10) for the landmark–based observer stereotype. For the former, IGC, accelerates the convergence of the observer beliefs towards those satisfying b_G. For these observer stereotypes IGC acts far more transparently and manages to convey the goal faster than the standard goal-directed $LAMA$. In the case of the POM17 stereotype in most domains the lengths are very similar, and in the specific case of ROVERS and the plan library compilations (CAMPUS, KITCHEN and INTRUSION DETECTION) are worse. The reason for this difference of performance can be traced directly back to the "loopy" behaviour demonstrated in Figure 2 which *delays* the selection of landmarks. The number of plans in the plan library compilations are quite small[3] and the set of (action) landmarks contain most if not every action used relevant to valid plans. In contrast, Richter and Westphal's use of the landmark heuristic (2010) naturally directs LAMA towards including action landmarks in plans. While LAMA may seem

[2] https://boardgamegeek.com/boardgame/9209/ticket-ride

[3] We refer the reader to the figures in pages 73–78 of M. Ramirez's thesis (2012), where a graphical representation of the plan libraries is discussed.

to have been a poor choice for a baseline, in fact it serves us to illustrate the potential of an implementation of IGC using likelihoods derived from landmarks.

Table 2 produces a ranking between IGC and $LAMA$ on the basis of how often either planner induce beliefs consistent with b_G faster than the other. Again, for the cost–based stereotypes, the proposed IGC planner clearly outperforms $LAMA$, sometimes by a wide margin (see entry for Logistics with RG10 stereotype). Interestingly, this Table also shows how the performance of IGC degrades as we move from RG10 to stereotypes which *strongly* penalize plans which do not match their definition of rationality – $P(G|O)$. This is made manifest by the results on the ROVERS domain, where we see RG09 and POM17 react in *opposite* ways to IGC. RG09 assigns null posteriors $P(G|O)$ as soon as O deviates from *the* (suboptimal) best plan, POM17 in contrast penalises IGC as it avoids executing landmark actions common to many of the hypothetical goals $\mathcal{G}$, yet instrumental to achieve G^*. Included in Table 2 is an analysis which uses a χ^2 test where the null hypothesis (H_0) is that the number of IGC wins is equal to the number of times IGC doesn't win. A result with an *, **, *** or **** indicates a rejection of H_0 with a significance level of $0.05, 0.01, 0.001$ or 0.0001 respectively. Results which are significant in the opposite direction of the hypothesis are denoted **ns**, these are considered not significant because they only suggest that IGC did not perform better than LAMA.

While the results clearly show that our proposed instantiation of IGC is overfitting RG09 and RG10, and more generally, cost–based goal recognisers, we note that the hypothesis that maximising the number of landmarks achieved by plans as a good strategy for selecting actions for cost–based recognisers, does not seem to follow from the results in Tables 1 and 2. LAMA does not convey the goal faster in most domains.

Finally, we note that assumptions on rationality from the actor and observer perspective are best tested over domains that present opportunities for divergences in terms of which action conveys more information. Such domains are essentially partially–observable domains like those discussed by Ramirez & Geffner (2011), yet to try our ideas on these, additional research is necessary to integrate algorithms for action selection over partially–observable planning models, such as $LW(1)$ (Bonet and Geffner 2014), with the width–based search methods discussed in this paper.

			RG10			RG09			POM17		
Domain	I	P	F	Δ_m	Δ_a	F	Δ_m	Δ_a	F	Δ_m	Δ_a
BLOCKS WORD	23	IGC	22^{****}	7	2.7	14	4	2.4	3^{ns}	2	1.3
		LAMA	–	–	–	3	3	1.7	4	2	1.2
CAMPUS	16	IGC	4	2	1.5	4	3	2.2	$-^{ns}$	–	–
		LAMA	–	–	–	–	–	–	5	2	1.8
GRID NAVIGATION	21	IGC	16^{*}	16	5.1	16^{*}	16	5.1	$-^{ns}$	–	–
		LAMA	–	–	–	–	–	–	–	–	–
GRID NAVIGATION (W. OBST)	24	IGC	15	7	3.0	15	7	3.0	$-^{ns}$	–	–
		LAMA	–	–	–	–	–	–	–	–	–
INTRUSION DETECTION	23	IGC	19^{***}	4	1.4	18^{**}	4	1.3	3^{ns}	3	3.0
		LAMA	–	–	–	–	–	–	17	8	3.8
KITCHEN	18	IGC	12	2	1.5	18^{****}	2	1.3	6	2	2.0
		LAMA	–	–	–	–	–	–	12	10	3.0
LOGISTICS	20	IGC	20^{****}	17	4.5	18^{***}	20	11.0	13	17	7.3
		LAMA	–	–	–	2	11	10.0	5	10	7.0
ROVER	18	IGC	13	1	1.0	14^{*}	9	5.2	2^{ns}	1	1.0
		LAMA	1	1	1.0	3	1	1.0	13	4	1.1
TICKET TO RIDE	21	IGC	20^{****}	11	3.0	19^{***}	11	2.9	3^{ns}	4	3.3
		LAMA	–	–	–	–	–	–	–	–	–

Table 2: Comparison between IGC and LAMA over stereotypes RG09, RG10 and POM17. Column F is the number of times $Q_v(IGC) < Q_v(LAMA)$ (and vice versa). Columns Δ_m and Δ_a report maximal and average difference $Q_v(X_2) - Q_v(X_1)$, where X_2 and X_1 are the slowest and fastest planner. – entries correspond to cases where IGC (or LAMA) were never quicker than the other. Number of ties follows from subtracting the sum of F values from I (# instances) for every domain and stereotype. *, **, *** and **** indicate the statistical significance of the result representing *p-values* less than 0.05, 0.01, 0.001 and 0.0001 respectively. **ns** denotes results which are significant in the opposite direction of the hypothesis, which are those where IGC did not perform better than LAMA.

Future Work

In this paper we have formalised the notion of planning for communicating goals to an observer, making some strong assumptions on the nature of the reasoning process used to make sense out of the actions in the plan. One obvious and appealing line of future work is to conduct investigations with human observers and see to what extent human cognitive processes fit the behaviours predicted by our computational model. This study would need to investigate whether this fit would vary based upon if the human knows that the agent is communicating with them or not (whether the human believes that it is in a keyhole recognition scenario or an intended recognition scenario). Such a study would verify the suitability of our framework to inform the design of systems where humans and robots need to co–operate directly or indirectly in the pursuit of a common goal.

Communicating goals is a limited form of communication between agents which does not require a previous agreement on a communication protocol to be used by all the parties involved. Another future line of research is to explore the possibilities opened up by this work to enable *implicit co–ordination* in multi–agent planning domains.

The experimental setting we use in this paper to evaluate our approach implicitly assumes that every action can be perceived by the observer. This may well be not the case in realistic, yet simple settings. Examples of these are domains where the environment in some form *prevents* the observer from sensing the actors' actions.

Last, planning to *obfuscate* the goal being pursued seems intuitively a matter of setting b_T in a suitable manner, yet finding settings with an entirely *passive* observer where doing so is meaningful seems challenging to us.

References

Alami, R.; Clodic, A.; Montreuil, V.; Sisbot, E. A.; and Chatila, R. 2006. Toward human-aware robot task planning. In *AAAI spring symposium*, 39–46.

Astrom, K. 1965. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.* 10:174–205.

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proc. IJCAI*.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Bonet, B., and Geffner, H. 2001. Gpt: A tool for planning with uncertainty and partial information. In *Proc. IJCAI Workshop on Planning with Uncertainty and Partial Information*.

Bonet, B., and Geffner, H. 2014. Belief tracking for planning with sensing: Width, complexity and approximations. *JAIR* 50:923–970.

Bonet, B., and Geffner, H. 2016. Factored probabilistic belief tracking. In *Proc. IJCAI*.

Carberry, S. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11(1-2):31–48.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proc. IJCAI*.

Cohen, P. R.; Perrault, C. R.; and Allen, J. F. 1981. Beyond question answering. In Lehnert, W., and Ringle, M., eds., *Strategies for Natural Language Processing*. LEA.

Francès, G., and Geffner, H. 2015. Modeling and computation in planning: Better heuristics from more expressive languages. In *Proc. ICAPS*, 70–78.

Frances, G.; Ramirez, M.; Lipovetzky, N.; and Geffner, H. 2017. Purely declarative action representations are overrated: Classical planning with simulators. In *Proc. IJCAI*.

Freedman, R. G., and Zilberstein, S. 2017. Integration of planning with recognition for responsive interaction using classical planners. In *AAAI*, 4581–4588.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool.

Geffner, H. 2000. Functional strips. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Kluwer. 187–205.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173(11):1101–1132.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR*.

Kaelbling, L. P.; Littman, M.; and Cassandra, A. R. 1999. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. AAAI*, 32–38.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal recognition design. In *Proc. ICAPS*.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *JAIR* 36:547–556.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *Proc. of European Conference in Artificial Intelligence (ECAI)*, 335–340.

Kominis, F., and Geffner, H. 2017. Multiagent online planning with nested beliefs and dialogue. In *Proc. ICAPS*.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. ECAI*.

Lipovetzky, N., and Geffner, H. 2017a. Best-first width search: Exploration and exploitation in classical planning. In *Proc. AAAI*.

Lipovetzky, N., and Geffner, H. 2017b. A polynomial planning algorithm that beats lama and ff. In *Proc. ICAPS*.

Lipovetzky, N.; Ramirez, M.; and Geffner, H. 2015. Classical planning with simulators: Results on the atari video games. In *Proc. IJCAI*.

Martin, Y.; Moreno, M. D.; and Smith, D. E. 2015. A fast goal recognition technique based on interaction estimates. In *Proc. IJCAI*.

Masters, P., and Sardina, S. 2017. Deceptive path-planning. In *Proc. IJCAI*.

Pentney, W.; Popescu, A.; Wang, S.; Kautz, H.; and Philipose, M. 2006. Sensor-based understanding of daily life via large-scale use of common sense. In *Proc. AAAI-06*.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-based heuristics for goal recognition. In *Proc. AAAI*.

Ramirez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proc. IJCAI*, 1778–1783. AAAI Press.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. AAAI*.

Ramirez, M., and Geffner, H. 2011. Goal recognition over POMDPs. In *Proc. IJCAI*.

Ramirez, M.; Lipovetzky, N.; and Muise, C. 2015. Lightweight Automated Planning ToolKiT. `http://lapkt.org/`. Accessed: 2016-12-11.

Ramirez, M. 2012. *Plan recognition as planning*. Universitat Pompeu Fabra.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Richter, S. 2011. *Landmark-based heuristics and search control for automated planning*. Griffith University.

Schmidt, C.; Sridharan, N.; and Goodson, J. 1978. The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligence* 11(1:2):45–83.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *Proc. IJCAI*, 3258–3264.

Sondik, E. 1978. The optimal control of partially observable markov decision processes over the infinite horizon: discounted costs. *Operations Research* 26(2).

Stubbs, K.; Wettergreen, D.; and Hinds, P. H. 2007. Autonomy and common ground in human-robot interaction: A field study. *Intelligent Systems, IEEE* 22(2):42–50.

Warneken, F., and Tomasello, M. 2006. Altruistic helping in human infants and young chimpanzees. *Science* 311(5765):1301–1303.

Yang, Q. 2009. Activity Recognition: Linking low-level sensors to high-level intelligence. In *Proc. IJCAI-09*, 20–26.

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan explicability and predictability for robot task planning. In *In Proc. ICRA*, 1313–1320. IEEE.

Moral Permissibility of Action Plans

Felix Lindner and **Robert Mattmüller** and **Bernhard Nebel**
University of Freiburg, Germany
{lindner, mattmuel, nebel}@informatik.uni-freiburg.de

Abstract

Research in classical planning so far was mainly concerned with generating a satisficing or an optimal plan. However, if such systems are used to make decisions that are relevant to humans, one should also consider the ethical consequences that generated plans can have. We address this challenge by analyzing in how far it is possible to generalize existing approaches of machine ethics to automatic planning systems. Traditionally, ethical principles are formulated in an action-based manner, allowing to judge the execution of one action. We show how such a judgment can be generalized to plans. Further, we study the complexity of making ethical judgment about plans.

Introduction

With the advent of autonomous machines that drive on the streets or act as household robots, it has been argued that we need to add an ethical dimension to such machines leading to the development of the research area *machine ethics* (Anderson, Anderson, and Armen 2005; Anderson and Anderson 2011). One important question is how we can align the behavior of autonomous machines with the moral judgment of humans. In this context, most often the question is whether a particular action is morally obligatory, permissible or impermissible, given a particular ethical principle (Driver 2006). Judging one action is, of course, important. However, automated planning systems (Ghallab, Nau, and Traverso 2016) are faced with the problem of making a huge number of decisions about including actions into a plan. And it does not necessarily make sense to analyze the ethical contents of each such decision in isolation, but to take an ethical perspective on an entire plan (and perhaps alternative plans reaching the same goal). As an example, consider utilitarian reasoning: if every action in a plan were judged in isolation, one would not be allowed to perform an action that temporarily decreases the utility, even if this action is a necessary prerequisite for later earning a lot of utility in a globally optimal final reachable state. Judging a plan as a whole allows considering this early investment for the sake of a later benefit as permissible from a utilitarian perspective.

In this paper, we will address these problems by analyzing three problems. First, we will look at what kind of additional information we need in order to be able to make moral judgments in the context of different ethical theories. Secondly, we will propose methods to judge the ethical acceptability of a plan. We will test the proposed notions using examples from the literature on moral dilemmas.

We will not limit ourselves to one particular ethical principle, but will consider a number of different principles that have the potential to be treated computationally, similar to the HERA (Lindner, Bentzen, and Nebel 2017) approach.

Based on the work we present in this paper, a planner will not only be able to come up with plans, but also to ethically judge those plans and compare them to alternative plans that it produces or that may be suggested by a human user or another automated planner. This will enable an ethically competent planning system to discuss and *explain* why a certain—morally superior—plan should be preferred over another—morally inferior—one. Such an explanation can explicitly refer to *which* ethical principles are violated and *how* they are violated.

The remainder of the paper is structured as follows. In the next section, we introduce different ethical principles that have been discussed in the literature. Then, the planning formalism we will use throughout the paper will be specified. This is basically a propositional planning formalism extended by variables with non-binary domains, exogenous events, and moral valuations of actions and consequences. We then formalize the notions of causation and means to an end in the framework of our planning formalism. Based on that, we can then formalize different ethical principles, which we will use to analyze the computational problem of ethically validating a given plan. Finally, we sketch related work and conclude.

Ethical Principles

In moral philosophy, various ethical principles are considered. Ethical principles are descriptions of abstract rules that can be used to determine the moral permissibility of concrete courses of actions. In this section, we introduce ethical principles which embrace different views on how to assess moral permissibility of actions based on the actions' consequences: *Utilitarianism*, *deontology*, *do-no-harm principle*, and the *principle of double effect*.

The *utilitarian principle* focuses on consequences of actions. It says that an agent ought to perform the action amongst the available alternatives with the overall maximal utility. We adopt an act-utilitarian interpretation which does

not distinguish between doing and allowing, i.e. the causal structure of the situation is not taken into account. Thus the action which the agent ought to perform is the one which leads to the best possible situation, i.e. the highest utility, regardless of what the agent causes and intends.

The utilitarian principle is often contrasted with *deontology*. According to deontology, an action does not get its moral value from the consequences brought about by the action. Instead, deontology takes only the intrinsic utility of an act into account. An action is permissible according to the deontology if and only if the act itself is morally good or indifferent.

The *do-no-harm principle* is a consequentialist principle, like utilitarianism, but more restrictive in that it states that an agent may not perform an action which has any negative consequences. The do-no-harm principle is satisfied in case the agent remains inactive as there will then be no negative consequences and since we regard the act token of remaining inactive itself as neutral. The distinction between doing and allowing is relevant to this principle, as it is the causal consequences of an action which are considered. The intentions of the agents are not considered ethically relevant for our interpretation of this principle. A version of the do-no-harm principle can for instance be found in Asimov's first law of robotics forbidding robots to bring about harm by the action. A less restrictive version of this principle is the *do-no-instrumental-harm principle*. This principle allows for harm as a side effect but not as a means to ones goals.

Finally, we will consider the *principle of double effect*. Under this principle, an action is permissible if five conditions hold:

1. The act itself must be morally good or neutral.
2. A positive consequence must be intended.
3. No negative consequence may be intended.
4. No negative consequence may be a means to the goal.
5. There must be proportionally grave reasons to prefer.

A closer look reveals that the first condition of the principle of double effect implements deontology. Thus, actions are assumed to have an inherent moral value, which does not (necessarily) stem from the effect of an action. The second and third conditions take the intentions, or goals, of the agent into consideration: An agent may not have a bad consequence as a goal, but it should intend something good. The fourth condition is an implementation of the do-no-instrumental-harm principle introduced above: Morally bad consequences are permissible as side effects only. And finally, the fifth condition is a weaker version of utilitarianism: In our interpretation, the condition requires that all in all the effects of the action must yield positive utility. Thus, if the bad side effects are too severe, the principle of double effect will render the action morally impermissible.

Planning Formalism

We use a planning formalism based on SAS$^+$ (Bäckström and Nebel 1995), extended with conditional effects (Rintanen 2003) and exogenous events (Fox, Howey, and Long 2005; Cresswell and Coddington 2003).

Language. A *planning task* is a tuple $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$ consisting of the following components: $\mathcal{V}$ is a finite set of *state variables* v, each with an associated finite domain $\mathcal{D}_v$. The set of all values is denoted by $\mathcal{D} = \bigcup_{v \in \mathcal{V}} \mathcal{D}_v$. A *fact* is a pair $\langle v, d \rangle$, where $v \in \mathcal{V}$ and $d \in \mathcal{D}_v$, also written as $v{=}d$ in conditions and $v{:=}d$ in effects. We call a conjunction of facts $v_1{=}d_1 \wedge \cdots \wedge v_k{=}d_k$ *consistent* if it does not contain any two facts $v_i{=}d_i$ and $v_j{=}d_j$ such that $v_i = v_j$, but $d_i \neq d_j$. We call it a *complete conjunction*, or simply *complete* if it contains a conjunct $v{=}d$ for every variable $v \in \mathcal{V}$. Up to reordering and unnecessary repetitions of conjuncts, there is a unique complete conjunction of facts for every possible assignment of domain values to variables. Therefore, we will often identify those representations. A complete conjunction of facts s is also called a *state*, and S denotes the set of states of Π.

The set A is a set of *actions*, where an action is a pair $a = \langle pre, \mathit{eff} \rangle$. The *precondition* pre is a conjunction of facts, and the *effect* eff is a *conditional effect* in effect normal form (ENF) (Rintanen 2003), i. e., a conjunction $\mathit{eff} = \mathit{eff}_1 \wedge \cdots \wedge \mathit{eff}_k$ of sub-effects eff_i of the form $\varphi_i \rhd v_i{:=}d_i$, where φ_i is a conjunction of facts, the *effect condition*, and where $v_i{:=}d_i$ is an atomic effect (a fact). Every atomic effect may occur at most once in eff. We furthermore assume that, whenever eff includes two conjuncts $\varphi_i \rhd v_i{:=}d_i$ and $\varphi_j \rhd v_j{:=}d_j$, and $v_i = v_j$, but $d_i \neq d_j$, then $\varphi_i \wedge \varphi_j$ is inconsistent, to rule out contradictory effects. If some φ_i is the trivial condition $\top$, then the corresponding sub-effect is unconditional, and we write $v{:=}d$ instead of $\top \rhd v{:=}d$. The set of actions A is partitioned into a set A_{endo} of *endogenous actions* and a set A_{exo} of *exogenous actions*. We assume that the set of endogenous actions always contains the *empty action* ϵ, which has an empty precondition and effect, and we assume that each exogenous action is associated with a set of discrete time points $t(a)$ at which it will be automatically applied, provided that its preconditions is satisfied. This is similar in spirit to *timed facts* (Cresswell and Coddington 2003) that are made true exactly at their associated time point. The state $s_0 \in S$ is called the *initial state*, and the partial state $s_\star$ specifies the *goal* condition.

Semantics. An endogenous action $a = \langle pre, \mathit{eff} \rangle$ is applicable in state s iff $s \models pre$. For an exogenous action a to be applicable, we additionally require that s is the t-th state in the state sequence induced by the action sequence under consideration for some $t \in t(a)$. Let $\mathit{eff} = \bigwedge_{i=1}^{k} (\varphi_i \rhd v_i{:=}d_i)$ be an effect in ENF. Then the *change set* (Rintanen 2003) of eff in s, symbolically $[\mathit{eff}]_s$, is the set of facts $\bigcup_{i=1}^{k} [\varphi_i \rhd v_i{:=}d_i]_s$, where $[\varphi \rhd v{:=}d]_s = \{v{=}d\}$ if $s \models \varphi$, and $\emptyset$, otherwise. A change set will never contain two contradicting effects. Now, applying an applicable action a to s yields the state s' that has a conjunct $v{=}d$ for each $v{=}d \in [\mathit{eff}]_s$, and the conjuncts from s for all variables v that are not mentioned in the change set $[\mathit{eff}]_s$. We write $s[a]$ for s'.

For exogenous actions, we assume an *urgent* semantics. More specifically, whenever an exogenous action a_{exo} is applicable and its application in the current state leads to a

different successor state, its application is enforced. We furthermore assume that if two or more exogenous actions are applicable in the same state, they do not interfere, i. e., neither of them disables another one, nor do they have conflicting effects. Let s be a state. Then by $\Delta_{\text{exo}}(s)$ we refer to the unique state that is obtained from s by applying all applicable exogenous actions. Since exogenous actions that are applicable in the same time step do not interfere, $\Delta_{\text{exo}}(s)$ is well-defined and is obtained by the application of finitely many exogenous action occurrences. We give the following semantics to a sequence consisting of endogenous actions $\pi = \langle a_0, \ldots, a_{n-1} \rangle$: First we extend the plan by empty actions if $n - 1 < \max \bigcup_{a \in A_{\text{exo}}} t(a)$ until the highest time step of the exogenous actions equals $n - 1$. Then, we assume that the initial state s_0 is already closed under exogenous action application, i. e., that $\Delta_{\text{exo}}(s_0) = s_0$. Finally, for $i = 0, \ldots, n-1$, the next state s_{i+1} is obtained by first applying action a_i to state s_i (assuming that it is applicable), followed by closing under exogenous actions. More formally, $s_{i+1} = \Delta(s_i, a_i) := \Delta_{\text{exo}}(s_i[a_i])$. If a_i is inapplicable in s_i for some $i = 0, \ldots, n-1$, then π is inapplicable in s_0.

A state s is a goal state if $s \models s_\star$. We denote the set of goal states by $S_\star$. We call π a *plan* for Π if it is applicable in s_0 and if $s_n \in S_\star$.

Modified semantics for counterfactual reasoning. Below, we will try to answer questions of the form: "What would have happened if we had followed plan π, but without action a being part of π?", or: "What would have happened if $v{=}d$ would not have been an effect of action a?" For that, we want to be able to trace plan π while leaving out a or $v{=}d$. Unfortunately, with the semantics above, this would often simply mean that the pruned plan is no longer applicable. To avoid this, we consider an alternative semantics here. Let $\pi' = \langle a_0, \ldots, a_{n-1} \rangle$ be a pruned plan, possibly with some actions dropped or replaced by the empty action ϵ, or with some effects removed from actions. Let s_0 be the initial state. Then we define, for all $i = 0, ..., n-1$, that $s_{i+1} = \Delta(s_i, a_i)$, if a_i is applicable in s_i, and $s_{i+1} = s_i$, otherwise. In other words, if a_i is applicable in s_i, then we apply it, otherwise, we skip it. Notice that even if a_i remains applicable in s_i in π', the actual effects of a_i may differ from what happens when tracing the unpruned plan π, since some effect conditions of a_i may be satisfied for π, but not for π', or the other way around.

Moral valuations of actions and consequences. Above, we defined the planning formalism we use. To define the possible *dynamics* of the system under consideration, this is sufficient. However, in order to formally capture and reason about the *ethical principles* outlined in above, we also need to classify actions and facts with respect to their moral permissibility as either morally bad, indifferent, or good.

To that end, in the following, we assume that each planning task Π comes with a utility function u that maps endogenous actions and facts to utility values: $u\colon A_{\text{endo}} \cup (\mathcal{V} \times \mathcal{D}) \to \mathbb{R}$.

Note that we let u map to $\mathbb{R}$ instead of just $\{-1, 0, 1\}$ to allow for different degrees of how morally good or bad an action or fact may be. We need this in order to reasonably capture the utilitarian principle. We call an action a or fact f *morally bad* if $u(a) < 0$ or $u(f) < 0$, respectively. Similarly, we call an action or fact *morally indifferent* or *morally good* if its utility value is zero or higher than zero, respectively. Notice that we explicitly do *not* require that permissibility of actions and facts must be consistent in any particular sense. For instance, we do not require that an action must be classified as morally bad if one (or all) of its effects are morally bad. The rationale behind this choice is that, in terms of deontology, actions are good or bad *per se*, without regard to their actual effects. We leave enforcing such consistency to the modeler where this is desired, and emphasize that occasionally, such consistency may be explicitly *not* desired.

When using a consequentialism view, we will judge the moral value of a plan by the utility value of its final state, which is defined to be the sum over the utility values of all facts in the final state: $u(s) = \sum_{\{v=d \mid s \models v=d\}} u(v{=}d)$. If we want to consider also the utility value of intermediate states of a plan, one would need to propagate the relevant facts to the final state. This again would be something the modeler is responsible for.

Means to an End

Existing Proposal

To derive the means of a plan, Govindarajulu and Bringsjord (2017) propose the following definition of a relation between two effects in the plan:

> Given a plan ρ, we say an effect e_1 is used as means for another effect e_2, if $e_1 \in pre(a_1)$, a_1 is an action in the plan and $e_2 \in additions(a_2)$, and a_1 comes before a_2.

The purpose of this definition is to check whether or not a given plan violates the fourth condition of the double-effect principle requiring that no morally bad effect is used as a means to bring about a morally good effect (see section on ethical principles). However, the definition does not capture the intuition about what it means that an effect is used as a means for another effect. One can easily think of a plan with two actions a_1, a_2, such that e_1 is a precondition of a_1, which makes one part of a goal true, and e_2 be an effect of a_2 which makes another part of the goal true. Then, e_2 does not depend on e_1 in any way.

That said, the Govindarajulu and Bringsjord's (2017) definition does not take into account that different actions in an action plan can contribute to different parts of the goal. For a more demonstrative example consider an action plan for a tea-serving robot to accomplish $s_\star \equiv bobHT{=}\top \wedge aliceHT{=}\top$. An action plan could be $\pi = \langle announceTea, serveBob, serveAlice \rangle$, that is, first the robot creates the desire for tea by announcing tea time, then the robot brings Bob and Alice some tea. We assume the planning task $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$:

- $\mathcal{V} = \{bobHT, aliceHT, bobWT, aliceWT\}$
- $A = \{announceTea, serveBob, serveAlice\}$

- $announceTea = \langle \top, bobWT{:=}\top \wedge aliceWT{:=}\top \rangle$
- $serveBob = \langle bobWT{=}\top, bobHT{:=}\top \rangle$
- $serveAlice = \langle aliceWT{=}\top, aliceHT{:=}\top \rangle$

- $s_0 = bobHT{=}\bot \wedge aliceHT{=}\bot \wedge bobWT{=}\bot \wedge aliceWT{=}\bot$
- $s_\star = bobHT{=}\top \wedge aliceHT{=}\top$

In the plan $\pi = \langle announceTea, serveBob, serveAlice \rangle$, $bobWT{=}\top$ is an effect of $announceTea$, and it is a precondition of $serveBob$. Moreover, $aliceHT{=}\top$ is an effect of the later action $serveAlice$. Hence, according to the definition by Govindarajulu and Bringsjord (2017), $bobWT{=}\top$ is a means for $aliceHT{=}\top$. This unintuitive result comes from the fact that the relation between the effects is not correctly captured by the definition. Instead, a counterfactual analysis is necessary: $bobWT{=}\top$ may only count as a means for $aliceHT{=}\top$, if $aliceHT{=}\top$ could not be made true by the plan if, counterfactually, $bobWT{=}\top$ were no effect of $announceTea$.

This analysis suggests that the means relation should be defined using a counterfactual condition that better matches the intuition.

Refined Proposal

Drawing on the analysis in the preceding section, we propose a counterfactual definition of the concept *means to an end*, which relates effects of actions and goals. Let us start with a preliminary definition. An effect $v_m{=}d_m$ of an endogenous or exogenous action a_m in a plan π is called *a means to achieve a fact* $v_e{=}d_e$ (i.e. $s_\star \models v_e{=}d_e$) if and only if removing the effect $v_m{=}d_m$ from action a_m would lead to final state s'_n such that $s'_n \not\models v_e{=}d_e$. Remember that we assume that the original plan π is still considered to be executable even if some of the actions are not executable any more.

This definition gets the intuition of the relation between Bob wanting tea and Alice having tea right. If, counterfactually, the effect $bobWT{=}\top$ were not an effect of $announceTea$, it would still be the case that after $\pi = \langle announceTea, serveBob, serveAlice \rangle$, the goal $aliceHT{=}\top$ would be achieved, but not $bobHT{=}\top$. Hence, according to our preliminary definition, $bobWT{=}\top$ is not a means the end $aliceHT{=}\top$, but it is a means to the end $bobHT{=}\top$.

What is not clear is how to check the *means* relation if an effect appears more than once during the execution of a plan. For instance, assume that in a plan the electric light is switched on, illuminating the room, i.e. $roomIlluminated{=}\top$. Further, a candle is lit, which also illuminates the room. One of the goals is to make an object in the room visible, i.e., $object{=}visible$, which happens, if the room is illuminated. If we now check counterfactually whether the fact $roomIlluminated{=}\top$ is a means to achieve $object{=}visible$, it is not clear, for which action we should delete the fact $roomIlluminated{=}\top$. Moreover, regardless of which effect we delete, the object will still be visible. Only if we delete both effects in the plan, then the object is not any longer visible. So, one could argue that the above definition should be modified by requiring that *all effects* in the plan of the form $v_m{=}d_m$ should be deleted in order to check whether $v_m{=}d_m$ is a means to achieve $v_e{=}d_e$. This requirement appears to be too strict, however. Assume a toggle switch action that has an effect $pressed{=}\top$, which in turn leads through an exogenous action to toggling the light and resetting the pressed status, i.e., $pressed{=}\bot$. Assume two of these actions are executed in a plan. Removing all $pressed{=}\top$ effects will not change the status of the light in the end, but only one removal will change the status of the light in the final state. For these reasons, we argue that we should consider all possible subsets of effect appearances in plan, when the *means to an end* relation is considered, which leads to the following formal definition.

Definition 1 (Means to an End)**.** *For a given plan π with final state s_n, a fact $v_m{=}d_m$ is called a* means to the end *$v_e{=}d_e$ if and only if $s_n \models v_e{=}d_e$ and the plan π' obtained by deleting the effect $v_m{:=}d_m$ from some actions in π does lead to a final state s'_n s.t. $s'_n \not\models v_e{=}d_e$.*

Formalization of Ethical Principles

We can now formalize moral permissibility of action plans according to the ethical principles introduced above. To exemplify each of the principles and to demonstrate how they come to different judgments about the moral permissibility of plans, we first introduce two famous versions of the *trolley problem* (Foot 1967). The classical trolley problem is a thought experiment that asks the listener to imagine it were in the following situation: "A runaway trolley is about to run over and kill five people. If you, as a bystander, throw a switch then the trolley will turn onto a sidetrack, where it will kill only one person." Using SAS$^+$, the problem can be modeled as a planning task $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$, such that:

- $\mathcal{V} = \{man, men, tram, lever\}$
- $A_{endo} = \{pull\}, A_{exo} = \{advance\}$
  - $pull = \langle \top, lever{=}l \triangleright lever{:=}r \wedge lever{=}r \triangleright lever{:=}l \rangle$
  - $advance = \langle \top, tram{=}start \wedge lever{=}r \triangleright tram{:=}r \wedge tram{=}start \wedge lever{=}l \triangleright tram{:=}l \wedge tram{=}r \triangleright men{:=}dead \wedge tram{=}l \triangleright man{:=}dead \rangle$
- $t(advance) = \{1, 2\}$
- $s_0 = man{=}alive \wedge men{=}alive \wedge tram{=}start \wedge lever{=}r$
- $s_\star = men{=}alive$
- $u(pull) = u(lever{=}l) = u(lever{=}r) = u(tram{=}start) = u(tram{=}l) = u(tram{=}r) = 0, u(man{=}alive) = 1, u(men{=}alive) = 5, u(man{=}dead) = -1, u(men{=}dead) = -5$

In this model, the variable men models the state of the five persons on the one track ($dead$ or $alive$), and man models the state of the one person on the other track. The variable $tram$ tracks the position of the tram ($start$, right track r, left track l), and the variable $lever$ represents the state of the lever (left position l or right position r). There is one endogenous action $pull$ available to the bystander. The action switches the state of the lever. The timed exogenous action $advance$ changes the position of the tram at time points 1 and 2. Deaths are considered morally bad and hence

they have negative utility, and survival facts are considered morally good and hence have positive utility. All other facts and actions are considered morally neutral. Depending on the state of the lever, at time point 1, the tram will move from its start position either to the left track or to the right track. At time point 2, if it is on the left track, the tram will hit the one man, and if it is on the right track, it will hit the five men. So, if the bystander's goal was to save the five men, her only chance is to execute $pull$ at time point 0.

The classical trolley problem is often contrasted with the *footbridge trolley problem*, which reads: "A trolley has gone out of control and now threatens to kill five people working on the track. The only way to save the five workers is to push a big man currently standing on the footbridge above the track. The big man will fall onto the track thereby stopping the tram. He will die, but the five other people will survive." Like the classical trolley problem, also the footbridge trolley problem involves a decision between one death and five deaths. But the intuition about what is morally permissible to do turns out very different. The SAS$^+$model of this scenario is given by a planning task $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$, such that:

- $\mathcal{V} = \{man, men\}$
- $A_{endo} = \{push\}, A_{exo} = \{advance\}$
  - $push = \langle man{=}onBridge, man{:=}deadOnTrack \rangle$
  - $advance = \langle \top, man{=}onBridge \triangleright men{:=}dead \rangle$
- $t(advance) = \{1\}$
- $s_0 = man{=}onBridge \wedge men{=}alive$
- $s_\star = men{=}alive$
- $u(push) = -1, u(man{=}onBridge) = 1, u(man{=}deadOnTrack) = -1, u(men{=}dead) = -5, u(men{=}alive) = 5$

The variable man represents the state of the big man on the footbridge (either $onBridge$ or $deadOnTrack$), and the variable men represents the state of the five people on the track (either $dead$ or $alive$). The endogenous action $push$ is available to the decision-making agent, who reasons about whether or not to push the big man off the bridge. The timed exogenous action $advance$ changes the state of the tram. Depending on whether or not the big man is on the track, the tram will stop at time point 1 due to its collision with the big man, or it will hit the other five men. We assume that pushing is inherently morally bad, that the fact that the big man is lying dead on the track is morally bad and that him surviving on the bridge is morally good, and that the death of the five men also is morally bad but their survival is morally good.

So, one reasoning task of interest is to check possible plans for moral permissibility. To do so, we define moral permissibility of several ethical principles: Deontology, utilitarianism, do-no-harm principle, do-no-instrumental-harm principle, and the principle of double effect.

The definition of the deontological principle (Def. 2) requires that all actions in a plan are intrinsically morally good or neutral.

Definition 2. *A plan $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is* morally permissible according to the deontological principle *if and only if for all actions a_i, $u(a_i) \geq 0$.*

Consider the plans $\pi_1 = \langle pull \rangle$ for the classical trolley problem and $\pi_2 = \langle push \rangle$ for the footbridge trolley problem. Plan π_1 does not contain any intrinsically bad action, whereas π_2 does. Therefore, according to the deontological principle, π_1 is morally permissible, because it does not contain any morally bad action, and π_2 is morally impermissible, because it does contain a morally bad action.

Consequentialists argue that the moral value of actions is determined by their consequences rather than by some intrinsic value. One such consequentialist ethical principle is utilitarianism, which requires an agent to always do what is morally optimal. In the context of action plans, we call a plan morally permissible according to the utilitarian principle iff the final state of the plan is among the morally optimal states.

Definition 3. *A plan $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is* morally permissible according to the utilitarian principle *if and only if $u(s_n) \geq u(s')$ for all reachable states s', where s_n is the final state reached by π.*

Given that the $advance$ actions will be executed anyway, the set of reachable states in both the trolley problems boil down to the states reached by acting at time point 0 or refraining from action. In the classical trolley problem, the two reachable states differ in the number of people dead. In our version of utilitarianism, the number of people harmed is morally relevant. Thus, the plan $\langle pull \rangle$ is morally permissible, but the empty plan is not. Also for the footbridge trolley problem, pushing the big man off the bridge, $\langle push \rangle$, is morally permissible but the empty plan is not.

While utilitarianism allows for harm for the greater good, it has been argued that a moral agent should avoid harm at all (for instance, the first Law of Robotics by Asimov contains such a do-no-harm clause). Thus, definition 4 renders an action plan morally permissible only if no part of the plan produces avoidable harm.

Definition 4. *A plan $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is* morally permissible according to the do-no-harm principle *if and only if for all facts $v{=}d$, if $s_n \models v{=}d$ and $u(v{=}d) < 0$, then for all plans π' obtained by deleting a subset of actions in π, $v{=}d$ still holds in the final state of π'.*

According to this definition, the plan $\langle pull \rangle$ for the classical trolley problem is morally impermissible. This is because it makes the morally bad fact $man{=}deadOnTrack$ true, which is false if $pull$ is deleted from the plan. For the analog reason, the plan $\langle push \rangle$ for the footbridge trolley problem is impermissible, as well. Note that, although the deontological principle and the do-no-harm principle agree on the judgment of the plan $\langle push \rangle$, they do so for different reasons: The deontological reasoner argues that the plan is impermissible, because pushing someone is wrong, whereas under the do-no-harm principle, the reasoner argues that the plan is impermissible, because pushing the man actively brings about a morally bad consequence. Thus, different principles give rise to different explanations even though they may come to similar judgments.

While the principle is clear as long we consider only one action or talk about executing the plan in full or not all, the judgment appears to be more difficult when one deliberates about leaving out arbitrary parts of the original plan. If, for example, we have two actions in the plan, one deleting a morally bad effect, which is true in the initial situation, and the second action reinstantiates the morally bad effect, then we have not lost anything compared with the initial situation. However, when executing the plan we reach a state which state from which executing the second action leads to some harm. For this reason, we consider a plan only as acceptable when we can guarantee that by deleting arbitrary parts we never reach a less harmful state.

An attractive extension of the do-no-harm principle is the do-no-instrumental-harm principle defined in Def. 5. The idea is that harm is permissible in case it is not committed as a means to one's end but only occurs as side effect.

Definition 5. *A plan $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is* morally permissible according to the do-no-instrumental-harm principle *if and only if for all moral facts $v{=}d$, if $s_n \models v{=}d$ and $u(v{=}d) < 0$, then $v{=}d$ is not a means to an end (see Def. 1).*

According to the definition of the do-no-instrumental harm principle, the plan $\langle pull \rangle$ is permissible. This is because the bad effect $man{=}dead$ is not a means to the end $men{=}alive$: If, counterfactually, $man{=}dead$ was no effect the actions in the plan, then still $men{=}alive$ would finally hold. Contrarily, in the footbridge trolley problem, if, counterfactually, $man{=}deadOnTrack$ was no effect of $push$, the the goal $men{=}alive$ would not finally hold. Hence, the plan $\langle pull \rangle$ is morally permissible according to the do-no-instrumental harm principle, and $\langle push \rangle$ is not. These judgments correspond to the judgments made by the deontological principle. But again, the judgments are made for different reasons: Pulling is permissible not because it is intrinsically permissible, but because no harm is done as a means to the end. Also, pushing the big man off the bridge is impermissible, not because pushing is morally bad or because harm is done. Rather, the do-no-instrumental-harm reasoner would argue that the plan is morally impermissible, because the harm produced was brought about as a means to the end.

Finally, we define the principle of double effect in Def. 6, which contains many of the above principles.

Definition 6. *A plan $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is* morally permissible according to the double-effect principle *if and only if all of the following conditions are satisfied:*

1. *The plan π is morally permissible according to the deontological principle.*
2. *At least one goal fact $v{=}d$ satisfies $u(v{=}d) > 0$.*
3. *No goal fact $v{=}d$ satisfies $u(v{=}d) < 0$.*
4. *The plan π is morally permissible according to the do-no-instrumental-harm principle.*
5. *$u(s_n) > 0$, where s_n is the goal state reached by π.*

As can be seen from the definition, the principle of double effect contains the deontological principle as its first condition and the do-no-instrumental-harm principle as the fourth condition. The second and third conditions are constraints on the goal of the planning agent: She is not allowed to have morally bad goals, and the goal should contain something morally good. The last condition is a weaker form of utilitarianism, which requires that all in all the plan brings about more good facts than bad facts—but unlike utilitarianism, it does not require the plan's final state to be among the optimal states.

As we already know, in case of the footbridge trolley problem, the first condition renders pushing the man off the bridge impermissible. However, the second and third conditions are fulfilled, because the goal of the agent only consists of one fact, viz., $men{=}alive$, and this fact is morally good. The fourth condition also is violated as we have already discussed above. The fifth condition is fulfilled, because, all in all, the good consequences yield more positive utility than the negative consequence add negative utility. Hence, using the principle of double effect, the reasoner can explain that there are two reasons why the plan $\langle push \rangle$ is morally impermissible: Because pushing is morally bad, and because the death of the big man is used as a means. For the case of the classical trolley problem, the principle of double effect comes to the conclusion that the plan $\langle pull \rangle$ is morally permissible: Pulling is not intrinsically bad, the goal is of the agent is morally good, the bad effect is not used as a means, and overall, the balance of positive and negative utility of the consequences is positive.

Ethical Validation of Action Plans

The output of a planning algorithm is a sequence of actions $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ and a final state s_n. Our goal is to ethically evaluate a given action plan. To this end, we here describe procedures that take a planning task $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$, the utility function u, a plan π, its final state s_n, and one of the introduced ethical principles as the input and decide whether or not the principle renders the plan as morally permissible.

To check whether or not a given plan π is morally permissible according to the deontic principle (Def. 2), it needs to be checked if some of the actions in π are intrinsically bad, i.e., if for one of the action a_i in π, we have $u(a_i) < 0$. This can be apparently done in time linear in the length of π.

Proposition 1 (Deontic Validation)**.** *Deciding whether a plan is morally permissible according to the deontic principle can be done in polynomial time.*

A procedure for verifying that π is morally permissible according to the utilitarian principle (Def. 3) is much more involved then checking deontological permissibility. Recall that the utilitarian principle only permits plans that lead to reachable states with maximum utility. In so far, this is very similar to over-subscription planning (Smith 2004). Based on that, we can formulate a non-deterministic procedure for deciding the complement of the permissibility problem as follows. Compute the overall utility of s_n. Then guess another complete state s' with utility that is larger than the utility of s_n. Finally generate (non-deterministically) a plan π' to achieve s'. If successful, it demonstrates that π is not morally permissible. That this is indeed an (asymptotically) optimal procedure is shown by the following theorem.

Theorem 1 (Utilitarian Validation)**.** *Deciding whether a plan is morally permissible according to the utilitarian principle is PSPACE-complete.*

Proof. PSPACE membership follows from the arguments above, and the facts that PSPACE is closed under complement and non-determinism and that deciding plan existence is in PSPACE. PSPACE-hardness follows straight-forwardly from a reduction of plan existence in SAS^+ planning. Given a SAS^+ planning task Π, generate a new task Π' by extending the set of variables by two Boolean variables g_1 and g_2, which are both assumed to be false in s_0. Extend the set of actions by two new endogenous actions: $a_1 = \langle \top, g_1{:=}\top \rangle$ and $a_2 = \langle s_\star, g_2{:=}\top \rangle$. The new goal description of Π' is $s_\star = g_1{=}\top$. The utility function is identical to zero on all actions and facts except for g_1 and g_2, where it evaluates to 1. Clearly, the only possible plan is $\langle a_1 \rangle$ leading to state s with $u(s) = 1$. This plan is impermissible according to the utilitarian principle iff there exists a plan for the original task Π because in this case we could reach a state s' for Π' such that $u(s') = 2$. □

To check whether a given plan π is morally permissible according to the do-no-harm principle (Def. 4), we have to verify that no parts of the plan lead to avoidable harm. A non-deterministic algorithm for deciding impermissibility could be: We guess one fact $v_b{=}d_b$ with $u(v_b{=}d_b) < 0$ and a subplan π' of π leading to s' and then verify that $s_n \models v_b{=}d_b$ but $s' \not\models v_b{=}d_b$.

Theorem 2 (Do-No-Harm Validation)**.** *Deciding whether a plan is morally permissible according to the do-no-harm principle is co-NP-complete.*

Proof. The sketched non-deterministic algorithm demonstrates membership in co-NP. In order to show hardness, we use a reduction from 3SAT to the impermissibility problem. Assume a 3SAT problem over the variables $v_1, \ldots, v_n$ and clauses $c_1, \ldots, c_m$, where each clause consists of 3 literals l_{j1}, l_{j2}, l_{j3}. We now construct a planning task $\Pi = \langle \mathcal{V}, A, s_0, s_\star \rangle$, where $\mathcal{V} = \{b, g, v_1, \ldots, v_n, c_1, \ldots, c_m\}$, $A = \{V_1, \ldots, V_n, C_1, \ldots, C_m, G, B\}$, $s_0 = \{v{=}\bot \mid v \in \mathcal{V}\}$, and $s_\star = \{g\}$. The actions are defined as follows: $V_i = \langle \top, v_i{:=}\top \rangle$, $C_j = \langle \top, \bigwedge_{k=1}^{3} (l_{jk} \rhd c_j) \rangle$, where $l_{jk} \equiv v_{jk}{=}\top$ if the literal l_{jk} in the original SAT problem is positive, otherwise, $l_{jk} \equiv v_{jk}{=}\bot$. Further, $G = \langle \top, g{:=}\top \wedge (\bigwedge_{j=1}^{m} c_j \rhd b{:=}\bot) \rangle$, $B = \langle \top, b{:=}\bot \rangle$. All facts have zero utility except for $b{=}\bot$, which is valued -1. The plan, we want to check is $\pi = \langle V_1, \ldots, V_n, C_1, \ldots, C_m, G, B \rangle$. This plan obviously achieves the goal and the final state contains some harm. Moreover, the only way to avoid this harm is to delete action B. However, even without this action, we still may have harm. This harm can be avoided, if and only if we can delete a (perhaps empty) subset of the V_i actions corresponding to a variable assignment of the 3SAT problems that satisfies the original 3SAT formula, which demonstrates that impermissibility is co-NP-hard. □

For the do-no-instrumental-harm principle (Def. 5), we can use a very similar method. Instead of deleting subsets of actions, we have to delete subsets of effect occurrences in the plan. Hence, checking this principle for a given plan has the same computational complexity.

Theorem 3. *Deciding whether a plan is morally permissible according to the do-not-instrumental-harm principle is co-NP-complete.*

Proof. One can use obviously the same non-deterministic algorithm as for the do-no-harm principle, demonstrating that deciding permissibility of plan for this principle is again in co-NP. For hardness, we can use a reduction very similar to the one in the last theorem. Instead of deleting actions we would delete effects, which are used to enable the execution of exogenous actions that regulate the assignment of the variables. □

Finally, let us consider the double-effect principle. Except for the fourth condition, everything can obviously be checked in polynomial time. The fourth condition is just the do-not-instrumental-harm principle. In other words, deciding permissibility for this principle is in co-NP.

Theorem 4. *Deciding whether a plan is morally permissible according to the double-effect principle is co-NP-complete.*

Proof. Membership is obvious. Hardness follows with the same proof as above by setting $u(g) = 2$. □

Related Work

While there exists a number of papers on machine ethics, papers that focus on generating and/or validating plans according to ethical principles are scarce.

Dennis et al. (2016) propose to establish ethical principles and ethical rules that judge the severity of violation an ethical principle, whereby an ethical principle could be not to harm a human. Plans can then be ordered by comparing the worst violations of these plans. While this has an deontological flavor, in fact, plans are judged according to their ultimate consequences, and hence this appears to be a consequentialist approach. The authors do not consider the distinction between causing harm and causing instrumental harm.

Pereira and Saptawijaya (2017) showed how to use abductive logic programming in order to specify the principle of double effect and to evaluate some of the trolley scenarios. Berreby et al. (2015) similarly use logic programming (in this case ASP) in order to specify the principle of double effects and evaluate on trolley scenarios described using the event calculus. In this case, however, they do not use counterfactual reasoning to judge causality, but they use simple syntactical means to determine what is a cause of an effect. Finally, Govindarajulu and Bringsjord (2017) propose a general framework to create or verify that an autonomous system is compliant to the double doctrine principle. For this purpose they introduce a very powerful logical formalism called *deontic cognitive event calculus*. In particular, they propose a formalization of the notion of *means to an end* in a STRIPS framework, which we criticized earlier in this paper.

Interestingly, all papers mentioned above do not address the issue that evaluating the moral permissibility might lead to a counterfactual analysis that is combinatorial in nature.

Conclusions

In this paper, we formalized five ethical principles (utilitarianism, deontology, the do-no-harm and instrumental do-no-harm principles, and the doctrine of double effect) in the context of *action sequences*, as opposed to the more usual way of studying them in the context of *individual actions*. Only in this way we can analyze moral permissibility of entire plans, since it is not sufficient to judge the moral permissibility of each action in isolation, but also in the context of the whole plan.

We exemplified and explained our formalizations using classical moral dilemmas such as the trolley problem, and identified how and for which reasons different principles may arrive at different (or the same) conclusions. Furthermore, we studied the computational complexity of verifying whether a given plan is permissible with respect to each of the five investigated principles. We saw that, with respect to our formalization, verification is PSPACE-complete for utilitarianism, co-NP-complete for do-no-harm, for do-no-instrumental-harm, and for the doctrine of double effect, and that it is polynomial-time for deontology. It turned out that verifying the do-no-harm principles involves a combinatorial reasoning over possible *sets* of actions that lead to harm or that may be instrumental towards achieving a goal condition, which makes verifying those ethical principles surprisingly hard.

We believe that our work has the potential of being useful in making autonomous systems ethically competent by providing them with the capability of coming up with morally permissible plans or at least being able to judge ethical permissibility of given plans. Based on the framework developed in this paper, a planning system will be able to *explain* to a human user why it preferred one plan over another, if the reason for this preference is that the less preferred plan is morally problematic with respect to one or more of the five ethical principles we formalized.

References

Anderson, M., and Anderson, S. L., eds. 2011. *Machine Ethics*. Cambridge, UK: Cambridge University Press.

Anderson, M.; Anderson, S. L.; and Armen, C. 2005. Machine Ethics: Papers from the AAAI Fall Symposium. Technical report, AAAI Press.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

Berreby, F.; Bourgne, G.; and Ganascia, J. 2015. Modelling moral reasoning and ethical responsibility with logic programming. In *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, 532–548.

Cresswell, S. N., and Coddington, A. M. 2003. Planning with timed literals and deadlines. In *Proceedings of the 21st Workshop of the UK Planning and Scheduling SIG*, 22–35.

Cushman, F.; Young, L.; and Hauser, M. 2006. The role of conscious reasoning and intuition in moral judgment: Testing three principles of harm. *Psychological science* 17(12):1082–1089.

Dennis, L. A.; Fisher, M.; Slavkovik, M.; and Webster, M. 2016. Formal verification of ethical choices in autonomous systems. *Robotics and Autonomous Systems* 77:1–14.

Driver, J. 2006. *Ethics: The Fundamentals*. Hoboken, NJ: Wiley-Blackwell.

Feldman, F. 1978. *Introductory Ethics*. Prentice-Hall.

Foot, P. 1967. The problem of abortion and the doctrine of double effect. *Oxford Review*.

Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, 1151–1156.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Govindarajulu, N. S., and Bringsjord, S. 2017. On automating the doctrine of double effect. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 4722–4730.

Lindner, F.; Bentzen, M. M.; and Nebel, B. 2017. The HERA approach to morally competent robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, 6991–6997.

Pereira, L. M., and Saptawijaya, A. 2017. Agent morality via counterfactuals in logic programming. In *Proceedings of the Workshop on Bridging the Gap between Human and Automated Reasoning - Is Logic and Automated Reasoning a Foundation for Human Reasoning? co-located with 39th Annual Meeting of the Cognitive Science Society (CogSci 2017), London, UK, July 26, 2017.*, 39–53.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003)*, 185–194.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, 393–401.

Why Bad Coffee? Explaining Agent Plans with Valuings

Michael Winikoff
University of Otago
New Zealand

Virginia Dignum
Delft University of Technology
The Netherlands

Frank Dignum
Utrecht University
The Netherlands

Abstract

An important issue in deploying an autonomous system is how to enable human users and stakeholders to develop an appropriate level of trust in the system. It has been argued that a crucial mechanism to enable appropriate trust is the ability of a system to explain its behaviour. Obviously, such explanations need to be comprehensible to humans. We argue that it makes sense to build on the results of extensive research in social sciences that explores how humans explain their behaviour. Using similar concepts for explanation is argued to help with comprehensibility, since the concepts are familiar. Following work in the social sciences, we propose the use of a commonsense-psychology model that utilises beliefs, desires, and "valuings". We propose a formal framework for constructing explanations of the behaviour of an autonomous system, present an (implemented) algorithm for giving explanations, and present evaluation results.

Introduction

This paper addresses the problem of how an autonomous system can explain itself by developing a computational mechanism that provides explanations for why a particular action was performed. It has been argued (EU 2016; Winikoff 2017; Gunning 2018) that in a range of domains, a key factor in humans being willing to trust autonomous systems is that the systems need to be able to *explain* why they performed a certain course of action.

The problem this paper therefore addresses is how an autonomous system can provide an explanation for why it chose a particular course of action. Specifically, we develop a computational mechanism that provides explanations for why a particular action was performed.

In developing such an explanation mechanism, it is important to be mindful that the explanations have to be comprehensible, and useful, to a human, and therefore we should consider relevant social sciences literature (Miller 2017). According to Miller (2017) explanations should be *contrastive* i.e. answer questions of the form "why did you do X ... instead of Y?"; *selected*, i.e. select relevant factors and present those; and, *social*, i.e. presented relative to what the explainer believes the listener[1] (i.e. explainee) knows. That is, explanations, being in fact conversations, should follow Grice's maxims of quality, quantity, manner and relevance (Grice 1975).

In our work we consider in particular the work of Malle (2004), which argues that humans use commonsense psychological constructs (e.g. beliefs, desires) to explain behaviour. This leads us to adopt an agent model that includes desires and beliefs, specifically the well-known BDI model (Rao and Georgeff 1992; Bratman, Israel, and Pollack 1988; Bratman 1987). Malle identifies three types of reasons in explaining behaviour: desires, beliefs, and what he terms *valuings*, defined as things that "*directly indicate the positive or negative affect toward the action or its outcome*". We therefore extend the BDI model with valuings, following recent work by Cranefield *et al.* (2017) (cf. Section "Running Example"). This framework (described below) includes beliefs, desires, and valuings, and is the setting for our explanation algorithm. We contend that providing explanations in terms of the same concepts used in human-to-human explanations will help enable explanations to be comprehensible.

Formal Setting

In this paper, we assume that the listener assumes a goal tree model as the deliberation mechanism of the BDI agent.

A *goal tree*[2] is a tuple (N,G) of a name N, and either an action[3] (A, with associated pre and post conditions), or a combination of sub-goals (N_i, G_i), which can be in sequence (SEQ), unspecified order (AND), or a choice (OR) between options O_i, where each option $O_i = (C_i, (N_i, G_i))$ has a sub-goal and a condition C_i indicating in which situations that sub-goal can be selected to realise the parent goal. We write $(G_{1\text{-}n})$ (resp. $(O_{1\text{-}n})$) to abbreviate $((N_1, G_1), \ldots, (N_n, G_n))$ (resp. $(O_1, \ldots, O_n)$). We also sometimes abbreviate (N, G) to G_N for readability, and, where the name is not important, just write G for G_N.

Formally we define a goal tree G as:

$$G ::= A \mid \textsc{Seq}(G_{1\text{-}n}) \mid \textsc{And}(G_{1\text{-}n}) \mid \textsc{Or}(O_{1\text{-}n})$$

[1] In the remainder of the paper, we use the term listener to refer to the one who is given the explanation.

[2] We use "goal" to be consistent with the literature. Space precludes a discussion of the subtle distinction between "desire" and "goal".

[3] For actions we assume that the name of the goal tree node and the name of the action coincide, i.e. that $A = N$.

Intuitively, a goal tree is executed as follows. If the tree is simply an action, then the action is performed (assuming its preconditions hold). If the tree is an AND or SEQ decomposition, then all of the sub-goals are executed, either in the specified sequential order (SEQ), or in some, unspecified, order. Finally, if the tree is an OR decomposition, then an applicable option (i.e. one whose condition C_i is believed to hold in the current situation) is selected and executed. Many BDI platforms provide a way to handle failure, which we discuss later in the paper.

The formal semantics of a goal tree is obtained by mapping it to a sequence of actions. A goal tree can yield multiple such sequences, so formally $[\![(N, G)]\!]$ is a set of sequences of actions with the expectation that each sequence of actions achieves the goal.

We extend goal trees with *valuings*. The semantics of valuings is based on the theory of values as put forward by Schwartz (2012). In Weide (2011) it is shown how these abstract values can be connected to concrete aspects of action decision. Following Cranefield *et al.* (2017) we incorporate them by annotating nodes in the goal tree with an abstract evaluation of key aspects of their effects. By "key aspects" we mean those that are relevant to the agent evaluating which options it prefers, that is, its valuings. The agent's valuings, i.e. which options it appreciates more or less, are specific to a given situation. They are founded on the agent's values, which are the underlying drivers. In Cranefield *et al.* (2017) it is shown how these valuings can be kept consistent and work for large goal-plan trees. Due to space restrictions we do not repeat that part in this paper, but just assume the valuings to be present and indicating consistent preferences over alternatives.

For example, an agent might value good coffee, saving money, and saving time. These aspects are the measurable criteria indicating whether a certain value is promoted by a course of action. However, as already can be seen by the fact that we have multiple aspects (thus creating a kind of multi-criteria optimization), they do not completely determine the agent's valuing. E.g. an agent might prefer good coffee over bad coffee, but decide to get bad coffee for free at the end of the month when his salary runs out and get good coffee once his salary is in. So, the weighting of the different aspects and thus the resulting valuings might not be fixed, but depend on the context. Also he might prefer the best coffee from the shop, but not want to spend much time to get it when he is finishing a paper for a deadline. Thus, in general a valuing (or preference) for an option is based on the values, but also on the current situation and practical considerations.

Finally, we note that not all of the information that we use would necessarily need to provided by the designer. For instance, action postconditions and preconditions could perhaps be learned from observation. By defining actions in terms of their pre and post conditions we can view them basically as black boxes.

Running Example: Getting Coffee

Jo is an academic visiting colleagues at another university. Like many academics, he requires coffee. There are a number of possible sources of coffee: The little kitchen near Ann's office has coffee-like-substance freely available, but this machine requires a staff card to operate. Ann has in her office a coffee machine which converts pods into nice coffee. There is also a coffee shop a few buildings away, where good coffee can be obtained, at a (financial) cost. Jo prefers coffee to coffee-like substances, which is the over-riding preference. Less-important preferences are to save money, and to use the nearest coffee source. Therefore the three relevant quality attributes are (in order): quality (coffee preferred to coffee-like), money (free preferred to expensive), and location (smallest distance from starting location). We assume that an observer, not aware of Jo's preferences, will require an explanation concerning Jo's actual choice.

As noted earlier, we follow (Cranefield et al. 2017) in capturing valuings as annotations. In this case each annotation V_i is of the form (coffee quality, cost, distance), respectively drawn from {veryGood, good, bad}, {none, low, high}, and {none, low, medium, high} where the office and kitchen are close to each other ("low" distance), and the shop is far from both kitchen and office ("high" distance).

Figure 1 shows a goal tree for obtaining coffee in the setting of this running example. The figure also shows the pre- and post-conditions, and the valuing annotations. We use $dist(L_1, L_2)$ to denote the distance between locations L_1 and L_2.

Generating Explanations

As discussed in the introduction, an explanation is given in terms of reasons which can be desires (goals), beliefs, or valuings. More precisely, an explanation is either $\bot$ (representing that the question does not make sense, e.g. "why did you do X?" when X was not done), or a set of explanatory factors. Factors can be beliefs that were held (i.e. logical formulae, $Condition$), desires that were pursued, and valuings. Valuings are explained as "I preferred V to $\{V_1, \ldots, V_n\}$". We also have forward-looking explanatory factors of the form "I did N_1 in order to be able to later do N_2" ($N_1 \mapsto N_2$). Finally, as discussed towards the end of this section, one possible type of explanatory factor is an indication that a particular option was attempted but failed. For example, "I chose to get coffee from the kitchen because I tried to buy it from the shop but failed" (e.g. shop was closed). Finally, we also define $\top$ to be an explanation that carries no information. Clearly, $\top$ is not a useful explanation to a user, but it is used in the formal definitions below where some parts of the process do not provide any useful information. Formally an explanation X can be defined as:

$$
\begin{array}{rcl}
X & ::= & \bot \mid \{X'_1, \ldots, X'_n\} \\
X' & ::= & Condition \mid \{V_1, \ldots, V_n\} \prec V \mid \mathsf{Desire}\ N \\
 & & \mid N_1 \mapsto N_2 \mid \mathsf{Tried}\ N \mid \top
\end{array}
$$

We now define an explanation function which explains why a particular action was done. The definition of the explanation function E is with respect to the goal-tree and observed behaviour trace. Specifically, $E^T_N(G_{N'})$ is "explain N using the tree (N', G) and trace (T)". We define $n(G)$ as denoting the set of all node names occurring in the tree rooted at G. We define $T^{\prec N}$ to be the part of the trace T

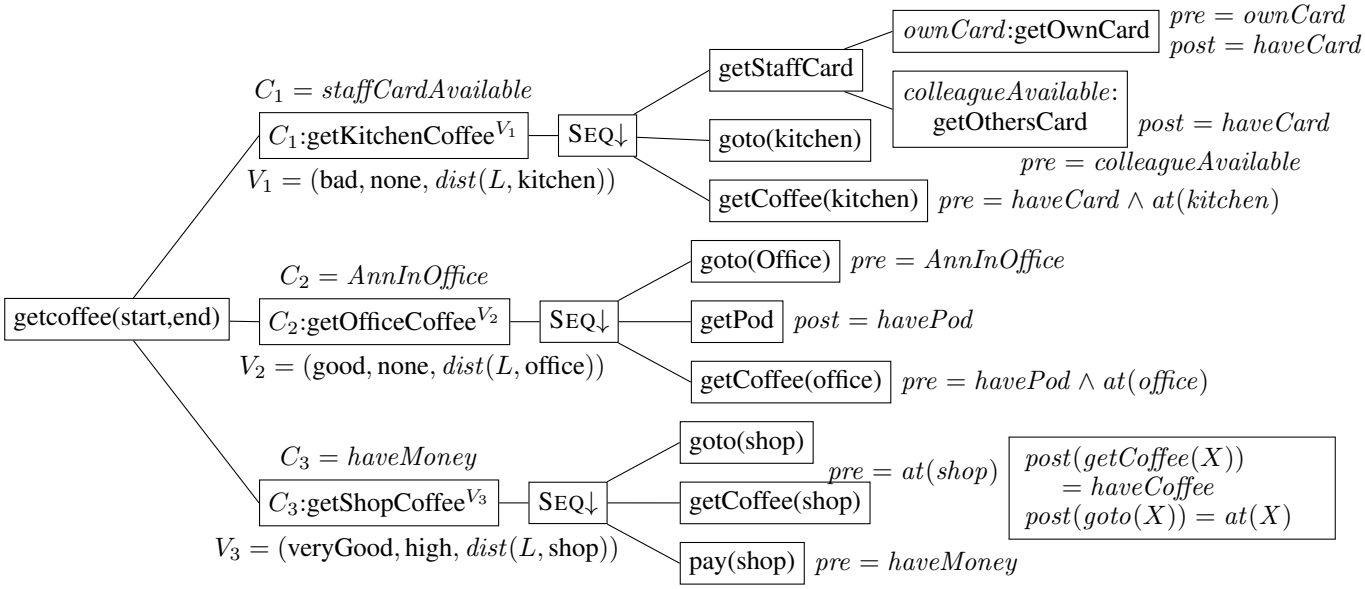


Figure 1: Running Example

that occurs before N, and we overload set membership to operate on the trace T which is a sequence of names. Note that if $N \notin T$ then we simply define $E_N^T(G) = \bot$, otherwise the rest of the definitions below apply.

$$
\begin{aligned}
E_N^T(G_{N'}) &= \bot \text{, if } N \notin T \\
E_N^T(A_{N'}) &= \begin{cases} \{\} & \text{if } pre(A) = \top \\ \{pre(A)\} & \text{otherwise} \end{cases} \\
E_N^T(\textsc{And}(G_i)_{N'}) &= \Omega \\
E_N^T(\textsc{Seq}(G_i)_{N'}) &= \Omega \\
E_N^T(\textsc{Or}(O_i)_{N'}) &= \begin{cases} \Omega \cup \Theta & \text{if } N \in n(G_i) \\ \Omega & \text{otherwise} \end{cases} \\
&\text{where } \Omega = \bigcup_{G_i : n(G_i) \cap T^{\prec N} \neq \emptyset} E_N^T((N_i, G_i)) \\
&\text{and } \Theta = pref(O_i, \{O_1, \ldots, O_n\})
\end{aligned}
$$

The function E collects explanation factors by traversing the relevant parts of the goal tree. A part of the goal tree is relevant if it occurs in the execution trace before beginning the process of executing the node N that is being explained. Simply, if something occurs before N, then it can affect N. This relevance condition is checked in the definition of Ω: $G_i : n(G_i) \cap T^{\prec N} \neq \emptyset$ finds all sub-goals G_i which contain at least some node that appears in the prefix of the trace T before N.

In the case of an action A the explanation collected is the action's precondition. This is because whether the precondition holds or not affects the execution of the action, and consequently, of whatever comes after it.

In the case for SEQ and AND the explanation collected is simply the explanation associated with the sub-goals.

In the case for OR there is an additional explanation relating to why the particular option taken was chosen. This is defined by the function $pref$ which provides an explanation for why the selected option, G_i, is preferred to the other options. The definition of $pref$ is complex. Intuitively, given a choice-point $(N, \textsc{Or}(O_1, \ldots, O_n))$, where G_i was selected the explanation consists of three parts (recall that $O_j = (C_j, G_j)$ where C_j is a condition):

1. the condition of the selected sub-goal being true ("C_i");
2. for each condition C_j ($j \neq i$) that is false at the decision point[4], the explanation includes that the condition was false:
$$\bigcup_{C_j : \mathcal{B}(N) \not\models C_j} \neg C_j$$
3. for each condition C_j ($j \neq i$) that is true at the decision point, the annotations of those sub-goals, and an indication that the selected sub-goal was preferred to the other available sub-goals[5] in the current situation:
$$\{V_j \mid j \neq i \wedge \mathcal{B}(N) \models C_j\} \prec V_i$$

We then define $pref(O_i, \{O_1, \ldots, O_n\})$ as the union of the above three parts: $pref(O_i, \{O_1, \ldots, O_n\}) = \{C_i\} \cup \left(\bigcup_{C_j : \mathcal{B}(N) \not\models C_j} \{\neg C_j\}\right) \cup \{\{V_j \mid j \neq i \wedge \mathcal{B}(N) \models C_j\} \prec$

[4] $\mathcal{B}(N)$ stands for the beliefs of the agent just *prior* to the execution of the goal tree (N, G). Note that recording, and retrieving, a history of beliefs during execution can be done efficiently (Koeman, Hindriks, and Jonker 2017).

[5] This is one place where the details of the BDI model matter. If the children of OR nodes are considered in a particular order and the BDI model is being used post-hoc, then instead of considering $j \neq i$ we only consider options that appear earlier, i.e. $j < i$. If the BDI model is being used to both generate and explain behaviour then, assuming the implementation captures which options were considered when O_i was selected, we would only consider those options that were considered in the explanation.

$V_i\}$ where we define $\{\} \prec V_i$ to be equivalent to $\top$, in other words, we do not generate that part of the explanation if there actually was no alternative option.

Consider as an example the situation in which C_2 is false, and the other C_i are true. Then the preference explanation for why C_3 was chosen is: $\{C_3, \neg C_2, \{V_1\} \prec V_3\}$. in other words: "I chose to get coffee from the shop because I had money, and Ann was not in her office, and I prefer V_3 to V_1 in this situation". On the other hand, in a situation where all C_i are true and C_3 is selected, the explanation would take the form: $\{C_3, \{V_1, V_2\} \prec V_3\}$, in other words: "I chose to get coffee from the shop because I had money, and I prefer V_3 to both V_1 and V_2 in this situation".

Note that these explanations just present the set of annotations, indicating an overall preference between them. However, we could provide more precise explanations by taking into account the known priorities of factors, e.g. that coffee quality is the overriding factor, followed by money, then distance. So, for example, for the first example above, we could explain more precisely that the reason why V_3 was preferred to V_1 is that it yields better quality coffee. Similarly, for the second example, we could explain that V_3 was preferred to both V_1 and V_2 because the coffee quality was better (despite V_2 being good coffee and cheaper than V_3).

On the other hand, suppose that the second option (office coffee) was selected, even though all three C_i were true. In this situation, in order to explain why $\{V_1, V_3\} \prec V_2$ we would need to use two factors. We could note that V_2 was preferred to V_1 because it had better coffee, and, perhaps, that it was preferred to V_3 because cost was a factor at this point in time.

Adding Preparatory Actions

We now extend the definition to also include preparatory actions. For example, an explanation for "why did you go to the kitchen?" could also be "because I need to be in the kitchen in order to get coffee". This is where an action's post condition is (part of) the precondition of a future action. Specifically, a preparatory reason applies to explain an action A when (i) the post-condition of A is required in order for the pre-condition of another action A' to hold, and (ii) A occurs before A'. We now need to formalise these two conditions.

For the first condition, i.e. that the post-condition of A is required for the pre-condition of A' to hold, an obvious formalisation is simply $post(A) \rightarrow pre(A')$. But A's post condition may be only *part* of the pre-condition. For example, the action getPod only achieves havePod, so $post(\text{getPod}) \not\rightarrow pre(\text{getCoffee(office)})$. We therefore formalise "required" as "without it, things don't work", i.e. if A's post-condition fails to hold, then the pre-condition of A' also must fail to hold: $(\neg post(A)) \rightarrow (\neg pre(A'))$. This assumes that $post(A) \neq \top$. In our setting, where pre and post conditions are conjunctions of positive atoms, this is equivalent (viewing the conjunctions as sets) to $post(A) \neq \emptyset \wedge post(A) \subseteq pre(A')$.

The second condition (A before A') holds exactly when A and A' have a common ancestor that is a SEQ node, where the sub-tree containing A occurs before the sub-tree containing A'. Formally:

$$\begin{aligned} before(A, A') \quad \equiv \quad & \exists N = \text{SEQ}(G_{1-n}) : \\ & A \in n(G_i) \wedge A' \in n(G_j) \wedge i < j \end{aligned}$$

Combining, we therefore have:

$$\begin{aligned} link(A, A') \quad \equiv \quad & before(A, A') \wedge \\ & post(A) \neq \emptyset \wedge post(A) \subseteq pre(A') \end{aligned}$$

We then extend the explanation with preparatory action explanations: when explaining an action A given goal tree G and trace T, we add to the explanation the set of links $A \mapsto A'$ where $A' \in n(G) \wedge link(A, A')$. So, for example, an alternative explanation for why the agent performed the action getPod is that it was required for the subsequent getCoffee(office) action. Finally, in order to consider preparatory actions between *goals*, we follow previous work on summary information (Thangarajah, Padgham, and Winikoff 2003a; 2003b; Visser et al. 2016), and extend pre and post conditions to intermediate goals, inferring them (details omitted due to space).

Adding Motivations

Finally, we also add explanations in terms of parent goals: these are desires that explain why the current course of action is being pursued.

This factor is simple: we also include in the explanation all the ancestors of the node being explained. However, we do not include ancestors that are OR refined, since these are not helpful. In explaining why a particular option was done, for instance why getOwnCard was done, it is not helpful to refer to the parent, getStaffCard.

Pulling all the pieces together, the overall explanation function is then:

$$\begin{aligned} \mathcal{E}^T_N(G_{N'}) \quad = \quad & E^T_N(G_{N'}) \cup \\ & \{N \mapsto N'' \mid N'' \in n(G) \wedge link(N, N'')\} \\ & \cup \{\textsf{Desire}(N''') \mid ancestor(N''', N) \\ & \qquad \wedge \neg isOR(N''')\} \end{aligned}$$

For example, given the scenario described, in a situation where C_1 and C_3 hold, but not C_2, the possible factors that could be used to explain why the agent did goto(shop) are (assuming a starting location at the shop): {haveMoney, ¬AnnInOffice, $\{\langle$bad, none, high$\rangle\} \prec \langle$ veryGood,high, none$\rangle$, goto(shop) $\mapsto$ getCoffee(shop), Desire: getShopCoffee}. In English, these are: I had money, Ann was not in her office, I preferred V_3 to V_1 (perhaps because it yields better quality coffee), I needed to go to the shop in order to do getCoffee(shop), and I desired to getShopCoffee.

Adding Failure Handling

We now extend the explanation mechanism to handle failure handling. Informally[6], actions can fail, and the failure of a node is handled by considering its parent. If the parent is a SEQ or a AND then it too is considered to be failed, and

[6]Space precludes a full formal definition.

failure handling moves to consider that node's parent. When an OR node is reached, failure is handled by trying an alternative plan (if one exists, otherwise the OR node is deemed to have failed). We assume that we know which actions in the trace are failed (denoted $failed^T(A)$). Then the condition under which a non-leaf node is considered to be failed is defined as:

$$failed^T(\text{AND}(G_{1-n})) = \bigvee_{1 \leq i \leq n} failed^T(G_i)$$

$$failed^T(\text{SEQ}(G_{1-n})) = \bigvee_{1 \leq i \leq n} failed^T(G_i)$$

$$failed^T(\text{OR}(O_{1-n})) = \bigwedge_{1 \leq i \leq n} failed^T(G_i)$$

Extending the explanation to account for the possibility of previous failures is done by defining an extended *pref* function. Note that the definition of the explanation function E is unchanged, except that in the definition of the recursive call, Ω, we exclude failed nodes:

$$\Omega = \bigcup_{G_i : n(G_i) \cap T^{\prec_N} \neq \emptyset \wedge \neg failed^T(G_i)} E_N^T((N_i, G_i))$$

Turning to *pref* recall that the definition of *pref* has three components: the condition of the selected sub-goal being true, the conditions of those (other) sub-goals that are false, and, for those other sub-goals that have true conditions, a preference indication.

We modify the second and third components by only considering those sub-goals that have not yet been attempted. So, instead of the second component being $\bigcup_{C_j : \mathcal{B}(N) \not\models C_j} \neg C_j$ we modify it to $\bigcup_{C_j : \mathcal{B}(N) \not\models C_j \wedge \neg failed^T(G_j)} \neg C_j$ Similarly, we modify the third component to:

$$\{V_j \mid j \neq i \wedge \mathcal{B}(N) \models C_j \wedge \neg failed^T(G_j)\} \prec V_i$$

Finally, we add a fourth component that explains those things that have been previously attempted. Intuitively, this is of the form "... and I already unsuccessfully tried doing G_j". Formally we have:

$$\{\mathsf{Tried}(G_j) \mid j \neq i \wedge failed^T(G_j)\}$$

To illustrate this definition, consider a situation where Jo has decided to getOfficeCoffee, but by the time he reaches Ann's office, Ann has had to leave for a meeting. The plan therefore fails, and Jo then recovers by electing to go to the shop. In response to the query "why did you getShopCoffee?" the explanation given is "{haveMoney, {⟨bad, none, low⟩} ≺ ⟨ veryGood, high, high⟩, Tried:getOfficeCoffee}" which can be rendered in English as "because I have money, I prefer good coffee to bad coffee, and because I tried (and failed) to get pod coffee".

Evaluation

There are two broad questions that concern evaluation of this work. The first is whether the explanations provided are comprehensible and *useful* to a human user. The second is whether the approach is sufficiently *efficient*.

Evaluation of Usability

In order to assess the comprehensibility and usability of the explanations generated, as well as provide guidance to future work on selecting explanations, we conducted a preliminary human participant evaluation.

However, before proceeding to present the evaluation, we make three observations. Firstly, and most importantly, our mechanism is informed by extensive research in social sciences, and uses concepts that we know humans use when explaining behaviour. That our explanations are couched in terms of familiar concepts suggests that the explanations are likely to be comprehensible. Secondly, we gave earlier a number of explanations generated for the running example. We argue that these explanations are both comprehensible and useful. In particular, we note that explanations are not excessively long, and, furthermore, that explanations are not excessively complex. What these examples do not, and cannot, show, is the extent to which explanations remain compact and comprehensible for larger examples. However, we note that we have not yet developed a selection mechanism. We know (Miller 2017) that humans give selected explanations, rather than complete explanations. Providing a selection mechanism would help ensure that explanations remain compact even with larger domains. Thirdly, as it can be seen in the evaluation described below, we use natural language sentences. The implementation of the algorithm described in the previous section was used to generate the complete explanation, including all factors, (beliefs, valuings, goals). These were manually converted to natural language. The focus of current work is the generation of formalized explanations from a goal-plan tree, and not generation of the corresponding sentences in natural language. Given the formal syntax of the explanations generated by the algorithm, it is straightforward to use patterns to map explanations to natural language.

Our evaluation took the coffee scenario described and administered a survey. Note that we focused on evaluating E_N^T, and did not include in the explanations either preparatory actions (links) or parent goals (except for the fifth explanation - see below).

Participants were recruited on Mechanical Turk and paid US$0.50 for an estimated 5 minute survey. Each participant was provided with a brief description of the scenario and an indication of what behaviour was observed. Participants were divided into three cohorts, each of which was given a different observed behaviour. The allocation to cohorts was random.

We obtained 109 responses, comprising 42 in Cohort 1, 37 in Cohort 2, and 30 in Cohort 3. Participants were 28 females and 81 males. Their highest level of education was high school (22), bachelors (63), and master / graduate degree (21). One person had not completed high school and two respondents had PhDs. Finally, around 35% had some experience with programming (38 out of 109).

Each cohort was given five possible explanations for the observed behaviour. The first explanation combined valuings and beliefs, and corresponds to the E_N^T function defined earlier (indicated with "V+B" below). The second and third explanations are solely in terms of valuings: one is abstract

	Cohort 1						Cohort 2						Cohort 3					
	Believ.		Accept.		Compr.		Believ.		Accept.		Compr.		Believ.		Accept.		Compr.	
E1	3.929	3	4.071	2	3.976	2	3.649	3	3.811	2	4.027	3	3.933	1	4.200	1	3.867	2
E2	3.095	5	3.286	5	3.714	4	3.892	1	3.892	1	4.054	2	2.500	5	2.767	5	3.200	5
E3	4.190	1	4.238	1	4.286	1	3.865	2	3.811	2	4.243	1	3.933	1	4.167	2	4.167	1
E4	3.857	4	3.500	4	3.690	5	2.973	5	3.000	5	3.108	5	3.567	4	3.600	4	3.567	3
E5	3.976	2	3.857	3	3.976	2	3.541	4	3.595	4	3.568	4	3.600	3	3.733	3	3.533	4
$p =$	0.00026		0.000013		0.038		0.0013		0.0047		0.00013		.000029		0.000025		0.034	

Figure 2: Believability, Acceptability, and Comprehensibility average scores (1=very bad, 5=excellent).

(AV), just saying "*This is the best possible coffee available*", and the second is concrete (V), with a specific explanation (see below). The fourth candidate explanation provides only relevant beliefs (B). The fifth candidate explanation gives the goal, and the beliefs that enabled the specific behaviour that was selected, which is the explanation mechanism proposed by Harbers (Harbers 2011) (G+B).

For example, in the case where the colleague's machine was selected (Cohort 1), the five explanations given are:

(E1) "This is the best possible coffee available; I had no money." (V+B)

(E2) "This is the best possible coffee available." (AV)

(E3) "This coffee is better than the kitchen and cheaper than in the shop." (V)

(E4) "I've no money; Ann was in her room." (B)

(E5) "I wanted coffee; Ann was in her room." (G+B)

In the case where the coffee shop is selected (Cohort 2), the five explanations are:

(E1) "This is the best possible coffee available; I had money." (V+B)

(E2) "This is the best possible coffee available." (AV)

(E3) "This is the best quality coffee." (V)

(E4) "I've money; Ann was away." (B)

(E5) "I wanted coffee; I had money." (G+B)

In the case where the kitchen is selected (Cohort 3), the five explanations are:

(E1) "This is the cheapest coffee; Ann was away." (V+B)

(E2) "This is the best possible coffee available." (AV)

(E3) "This coffee is cheaper than in the shop." (V)

(E4) "I've a card; Ann was away." (B)

(E5) "I wanted coffee; I've a card." (G+B)

For each possible explanation, the participants were asked to *score* the explanation in terms of three criteria: *believability* ("I can imagine someone giving this answer"), *acceptability* ("This is a valid explanation of Jo's choice"), and *comprehension* ("I understand the text of this explanation"). Each score was on a five-point Likert scale from "very bad" (1) to "excellent" (5). Participants were also asked to *rank* the five candidate explanations by order of preference, from most preferred (rank 1) to least preferred (rank 5). Finally, participants were also asked whether they felt that further explanation was required, and, if so, what form it should take (e.g. providing source code, entering a dialogue with the system). Finally, we also collected demographic information.

Explanation	**Average Ranking and Implied Collective Ranking**					
	Cohort 1		**Cohort 2**		**Cohort 3**	
E1	2.7857	2	2.5135	1	2.0667	1
E2	3.5714	5	2.5676	2	3.7333	5
E3	2.5238	1	2.7297	3	2.5667	2
E4	3.1429	4	4.1622	5	3.1667	3
E5	2.9762	3	3.027	4	3.4667	4
$p =$	0.011		0.00000074		0.000016	

Figure 3: Rankings for the three Cohorts and five Explanations.

Figure 2 shows for each cohort and each explanation the average score for each of the three criteria. The figure also shows the implied ranking. For example, for Cohort 1 and Believability, the third explanation (E3) had the best (highest[7]) average score, and therefore collectively E3 is ranked best for Believability by this cohort. For each of the three criteria and three cohorts a statistical test[8] confirms there is a difference amongst the explanations for that cohort[9].

Figure 3 shows for each explanation (E1 to E5) and for each cohort the *average ranking*, which is the average of each explanation's ranking. Note that the most preferred rank is 1, and the least preferred is 5, so a *lower* average ranking is a *more* preferred explanation. The table also shows for each explanation and cohort the preferred order of explanations implied by the average ranking (the implied collective ranking). For example, for cohort 1, explanation 3 had the lowest average ranking, and is therefore the most preferred explanation. A statistical test confirms that there are differences between the explanations' scores for each of the cohorts (as before, all p values are < 0.05). Post-hoc tests (Mann-Whitney, with Holm correction), find that the ranking differences are significant between E1-E2, E2-E3

[7] Recall that 1 = very bad, and 5 = excellent.

[8] Kruskal-Wallis, since data is not expected to be normally distributed.

[9] All p values are < 0.05 and hence significant, space precludes presentation of the post-hoc tests, a sequence of pair-wise Mann-Whitney tests, with Holm adjustment to reduce Type I errors, which find that some of the pairwise differences are significant.

(Cohort 1), E4 and all other explanations (Cohort 2), and between E1-E2, E1-E4, E1-E5, E2-E3, E3-E5 (Cohort 3). Space precludes detailed discussion.

Considering the question of whether the explanation given would be adequate, or whether additional information would be desired, 69% of Cohort 1, indicated that no further explanation would be required (with the remaining responses asking for a dialogue (19%) or source code (12%)). For Cohort 2 these figures were respectively 54% (no further explanation), 22% (dialogue), 19% (source code), and for Cohort 3 they were 63%, 20% and 17%.

Overall, explanations 1 and 3 were considered as being better than the other explanations, and that, except for Cohort 2, explanation 2 was seen as being the worst. Since explanations 1 and 3 both include valuings, this finding supports the key thesis of this paper, that valuings are important to provide useful explanations. Furthermore, for Cohorts 2 and 3, E1 was preferred to E3, indicating that valuings alone were not sufficient.

Evaluation of Efficiency

We now turn to efficiency. We observe that the explanation has three components: the reasons calculated by the function $E_N^T(G)$, the links between nodes, and parent goals. The last is simple to compute, involving merely traversing the tree upwards from the node being queried (i.e. $O(\log N)$ where N is the number of nodes in the goal tree). The second, the links, only depend on the static structure of the tree (i.e. which nodes precede other nodes), and on the pre and post conditions, and therefore can be computed ahead of time. This does assume that pre and post conditions are specified ahead of runtime. If this is not the case, then a runtime calculation is required, which involves checking pre and post conditions for every pair of nodes that precede each other. Given a tree with N nodes, there are obviously at most $O(N^2)$ such pairs, and the check is $O(1)$ (we assume that each node's pre and post conditions do not become longer as the tree grows).

Turning now to the explanation function E, we observe that the function basically traverses the tree from root to leaves. For each non-leaf node it checks which of the child nodes contain at least one node that is in the trace prefix ($n(G_i) \cap T^{\prec N} \neq \emptyset$). This check could be implemented by first traversing the tree upwards, tagging each node G_i with its $n(G_i)$, and then checking for intersection between $n(G_i)$ and $T^{\prec N}$. Since for each node the size of $n(G_i)$ is a function of the number of nodes beneath it, i.e. $O(N)$, computing the intersection (assuming indexing on $T^{\prec N}$) for a single node is $O(N)$, and for the whole tree it would be[10] $O(N^2)$. Finally, for each OR node, there is an additional calculation of *pref* which is proportional to the number of children and the size of conditions, both of which we assume is effectively a constant, i.e. does not grow with N. Therefore calculating E_N^T is $O(N^2)$.

In order to empirically assess the actual runtime required, and the algorithm's scalability, we have conducted an experimental evaluation on generated trees. The generated trees have the following structure: $T^0 = A$ and $T^{d+1} = \text{OR}_N(O_{1-j})$ where $O_i = (c, \text{SEQ}_{N_i}(T^d_{1-k}))$. In other words, a generated tree of depth 0, denoted T^0, is just an action A (with a new unique name), and a generated tree of depth $d+1$ is a disjunction of j options, where each option O_i has the same fixed condition c, and a sequential composition of k trees of depth d. All nodes have unique names. Note that the number of nodes in a tree with branching factors j and k and depth d can be calculated as: $n(j,k,0) = 1$ and $n(j,k,(d+1)) = 1 + j + (j \times k \times n(j,k,d))$.

For the efficiency evaluation various values of j, k and d were systematically generated, and the number of nodes in the tree and the time taken to compute E_N^T were recorded. The experiments were done using the GHC Haskell implementation (version 8.2.1) running on a 3.2 GHz Intel Core i5 iMac with 16 GB RAM running OSX 10.10.3.

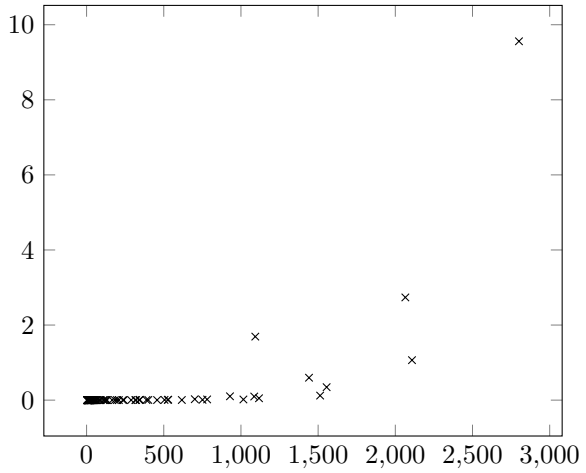


Figure 4: Time in seconds (Y) vs. number of nodes (X)

Figure 4 shows a scatter plot[11] of runtime (Y axis, in seconds) against the number of nodes in the tree (X axis). It is worth emphasising that the core of the Haskell implementation is a direct transliteration of the equations earlier in this paper. While this ensures that the implementation matches the paper, there are clear, and substantial, opportunities to improve efficiency.

As can be seen, for relatively small trees (fewer than 1000 nodes) the explanation generation, even with an unoptimised Haskell prototype, is clearly fast enough to be practical. It is worth noting that real goal trees are not necessarily large. For instance, the (real-world) application described by Burmeister *et al.* (Burmeister et al. 2008) has 57 nodes in its goal tree.

[10] However, the prototype implementation does not tag nodes, so it recomputes $n(G_i)$, leading to higher computational complexity.

[11] The time taken is affected not just by the number of nodes, but also by the shape of the tree, so there can be a number of trees with the same number of nodes and different time taken to compute the explanation.

Related Work

Miller (2017) surveys a broad range of work in the social sciences. His key argument is that work on explainable AI should take account of the work that has been done on human explanation of behaviour. We have already, in the introduction, summarised his three key findings (that explanations are contrastive, selected, and social). However, the paper does not itself propose any specific explanation mechanisms, instead it provides a broad survey. In doing so, it raises questions, poses challenges to the field of explainable AI, and identifies various factors that affect the design of explanation mechanisms. For example, that humans give explanations of the behaviour of intentional entities in terms of folk psychological constructs. Another example is the importance of abnormality in explanations: a factor that is unexpected, or abnormal, is more important in an explanation, even though other explanatory factors might be closer in time.

Harbers (2011), like us, assumes that a goal tree is given, and defines a number of templates that can be used to explain observed behaviour. Harbers' templates provide a range of possible explanations, e.g. explaining an action in terms of its parent goal, or its grand-parent goal, or the *next* action (corresponding to our links). She also conducted an empirical evaluation of the different templates in a given scenario. It is worth noting that our approach strictly generalises Harbers' approach, in that we include links, ancestor goals, and relevant beliefs. In other words, every factor that is included in explanations generated using Harbers' templates is included in $\mathcal{E}$. Furthermore, Harbers does not take into account other available alternatives, i.e. it explains why X was done solely in terms of what allowed X to be done, and does not mention anything relating to other available options. Finally, we note that whereas Harbers just outlines the rules as brief templates, we provide full formal definitions that have been implemented.

An approach closer to our work is that of explanation as model reconciliation, where the assumption is that in realistic scenarios humans have domain and task models that differ significantly from that used by the agent (Chakraborti et al. 2017). This assumption is supported by psychological studies that observed that explanations are *"typically contrastive... the contrast provides a constraint on what should figure in a selected explanation..."* (Lombrozo 2012). However, this approach does not link to the values/valuings, beliefs and desires of the human in the loop and is therefore less adequate to connect to the reasons behind the decisions taken in the process. Additionally, it assumes that we *know* the human's mental model, which is a fairly strong assumption, and one that we do not make.

Finally, it is worth noting that the term "explaining" can be applied to other things. In this paper we tackle the problem of an autonomous system explaining its *behaviour* to a human observer. There is also work (e.g. (Milliez et al. 2016)) that considers an autonomous system (specifically a robot) explaining a *plan* with a human collaborator. The problem being addressed is different. In the work of Milliez *et al.* there is a plan, and the explanation is of the next steps that the human needs to perform. In our work the explanation is of the observed behaviour, i.e. the past actions of the software, rather than the immediate future actions of the human.

Discussion

We have argued that explaining the behaviour of autonomous software could be done using the same concepts as are used by humans when explaining their behaviour. Specifically, we have followed the findings of Malle that "*Among the mental states that function as reasons, beliefs and desires are most common, and there is a third class that we might call* valuings" (Malle 2004, Section 4.2.4).

This paper has proposed a formal framework, using BDI-style goal-trees, augmented with value annotations. This formal framework is then used to define an explanation function, which has been implemented. A human subject evaluation has highlighted that, as expected based on the literature, valuings are seen as being of value in explaining behaviour, and, indeed, the option that corresponds to (part of) our explanation function was collectively ranked as the best explanation by two of the evaluation cohorts, and as second-best by the remaining cohort.

However, there is scope for further empirical evaluation. This would include using more scenarios, including the other explanatory factors, and assessing not just the believability, acceptability and comprehensibility of explanations, but more broadly assessing their effect on trust in the autonomous system.

Stepping back to consider the bigger picture, we have provided a mechanism for *generating* explanatory factors. However, this is only part of the solution to the problem of explaining behaviour. We know that humans select parts of the explanation (Miller 2017). The next step in this research is to define means for *selecting* parts of the possible explanation. This would need to take into account what information is available about the human asking for the explanation, for instance, when a contrastive question of the form "why did you do X rather than Y?" is asked, we can interpret Y as the expected behaviour, and look for factors that specifically explain the difference between the implied expected behaviour of Y and the actual behaviour of X.

There is also scope to extend the representation (goal-plan trees) in various ways, and to consider related notations such as Behaviour Trees (Colledanchise and Ögren 2017), Hierarchical Task Networks (Alford et al. 2016) (which extend Hierarchical Goal Networks (Shivashankar 2015)), or Geib & Goldman's Plan Trees (Geib and Goldman 2009).

Finally, note that in our work we have assumed that explanations are *honest*, in that they attempt to provide an accurate representation of the reasons for the observed behaviour. An alternative, which we eschew, is to consider *dishonest* explanations that seek to present the agent in the best possible light.

References

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016. Hierarchical planning: Relating task and goal decomposition with task sharing. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 3022–3029. IJCAI/AAAI Press.

Bratman, M. E.; Israel, D. J.; and Pollack, M. E. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4:349–355.

Bratman, M. E. 1987. *Intentions, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Burmeister, B.; Arnold, M.; Copaciu, F.; and Rimassa, G. 2008. BDI-agents for agile goal-oriented business processes. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS) [Industry Track]*, 37–44. IFAAMAS.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 156–163.

Colledanchise, M., and Ögren, P. 2017. Behavior trees in robotics and AI: an introduction. *CoRR* abs/1709.00084.

Cranefield, S.; Winikoff, M.; Dignum, V.; and Dignum, F. 2017. No Pizza for You: Value-based Plan Selection in BDI Agents. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 178–184.

EU. 2016. EU General Data Protection Regulation. http://tinyurl.com/GDPREU2016 (see articles 13-15 and 22).

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artif. Intell.* 173(11):1101–1132.

Grice, H. P. 1975. *Logic and conversation*. Academic Press, New York.

Gunning, D. 2018. Explainable Artificial Intelligence (XAI). https://www.darpa.mil/program/explainable-artificial-intelligence.

Harbers, M. 2011. *Explaining Agent Behavior in Virtual Training*. SIKS dissertation series no. 2011-35, SIKS (Dutch Research School for Information and Knowledge Systems).

Koeman, V. J.; Hindriks, K. V.; and Jonker, C. M. 2017. Omniscient debugging for cognitive agent programs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 265–272.

Lombrozo, T. 2012. Explanation and abductive inference. *Oxford handbook of thinking and reasoning* 260–276.

Malle, B. F. 2004. *How the Mind Explains Behavior: Folk Explanations, Meaning, and Social Interaction*. The MIT Press. ISBN 0-262-13445-4.

Miller, T. 2017. Explanation in artificial intelligence: Insights from the social sciences. *CoRR* abs/1706.07269.

Milliez, G.; Lallement, R.; Fiore, M.; and Alami, R. 2016. Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, HRI '16, 43–50. Piscataway, NJ, USA: IEEE Press.

Rao, A. S., and Georgeff, M. P. 1992. An abstract architecture for rational agents. In Rich, C.; Swartout, W.; and Nebel, B., eds., *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 439–449. San Mateo, CA: Morgan Kaufmann Publishers.

Schwartz, S. 2012. An Overview of the Schwartz Theory of Basic Values. *Online Readings in Psychology and Culture* 2(1).

Shivashankar, V. 2015. *Hierarchical Goal Networks: Formalisms and Algorithms for Planning and Acting*. Ph.D. Dissertation, University of Maryland, College Park, MD, USA.

Thangarajah, J.; Padgham, L.; and Winikoff, M. 2003a. Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 721–726.

Thangarajah, J.; Padgham, L.; and Winikoff, M. 2003b. Detecting and exploiting positive goal interaction in intelligent agents. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 401–408. ACM Press.

van der Weide, T. 2011. Arguing to motivate decisions. Dissertation, Utrecht University Repository https://dspace.library.uu.nl/handle/1874/210788.

Visser, S.; Thangarajah, J.; Harland, J.; and Dignum, F. 2016. Preference-based reasoning in BDI agent systems. *Autonomous Agents and Multi-Agent Systems* 30(2):291–330.

Winikoff, M. 2017. Towards Trusting Autonomous Systems. In *Fifth Workshop on Engineering Multi-Agent Systems (EMAS)*.

Explicable Planning as Minimizing Distance from Expected Behavior

Anagha Kulkarni, Yantian Zha, Tathagata Chakraborti,
Satya Gautam Vadlamudi, Yu Zhang and Subbarao Kambhampati
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ 85281 USA
{akulka16, yzha3, tchakra2, yzhan442, rao} @ asu.edu and gautam.vadlamudi@creditvidya.com

Abstract

In order to have effective human-AI collaboration, it is necessary to address how the AI agent's behavior is being perceived by the humans-in-the-loop. When the agent's task plans are generated without such considerations, they may often demonstrate *inexplicable behavior* from the human's point of view. This problem may arise due to the human's partial or inaccurate understanding of the agent's planning model. This may have serious implications from increased cognitive load to more serious concerns of safety around a physical agent. In this paper, we address this issue by modeling plan explicability as a function of the distance between a plan that agent makes and the plan that human expects it to make. We learn a regression model for mapping the plan distances to explicability scores of plans and develop an anytime search algorithm that can use this model as a heuristic to come up with progressively explicable plans. We evaluate the effectiveness of our approach in a simulated autonomous car domain and a physical robot domain.

1 Introduction

There is growing interest in applications that involve human-robot collaboration. An important challenge in such human-in-the-loop scenarios is to ensure that agent's behavior is not just optimal with respect to its own model, but is also explicable and comprehensible to the humans in the loop. Without it, the robot runs the risk of increasing the cognitive load of humans which can result in reduced productivity, safety, and trust (Fan et al. 2008). This mismatch between the robot's plans and the human's expectations of the robot's plans may arise because of the difference in the actual robot model $\mathcal{M}^{\mathcal{R}}$, and the *human's expectation of the robot model* $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. For example, consider a scenario with an autonomous car switching lanes on a highway. The autonomous car, in order to switch the lane, may make sharp and calculated moves, as opposed to gradually moving towards the other lane. These moves may well be optimal for the car and backed by the car's superior sensing and steering capabilities. Nevertheless, a human passenger sitting inside may perceive this as dangerous and reckless behavior, in as much as they might be ascribing the car the sort of driving abilities they themselves have.

In order to avoid inexplicable behavior, the robot has to take the human mental model into account and compute the plans that align with this model. As long as the robot's behavior is aligned with the human mental model, the human can make sense of it. Therefore, the objective of explicable planning is to generate robot plans that not only minimize the cost of the plan, but also the distance between $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. Of course, an immediate question is, if $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$ is available to the robot, why is $\mathcal{M}^{\mathcal{R}}$ required in the plan generation process at all? We note that this is necessary since the human mental model might entail plans that are not even feasible for the robot or are prohibitively expensive, and can thus at best serve as a guide, and not an oracle, to the explicable plan generation process. In such scenarios, the robot can choose to provide explanations about its inexplicable behavior which help to communicate the model discrepancies to the perplexed human (Chakraborti et al. 2017; Langley 2016).

An important consideration in the computation of explicable behavior is access to the human mental model, $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. In many domains, such as in household or factory scenarios), there is generally a clear expectation of how a task should be

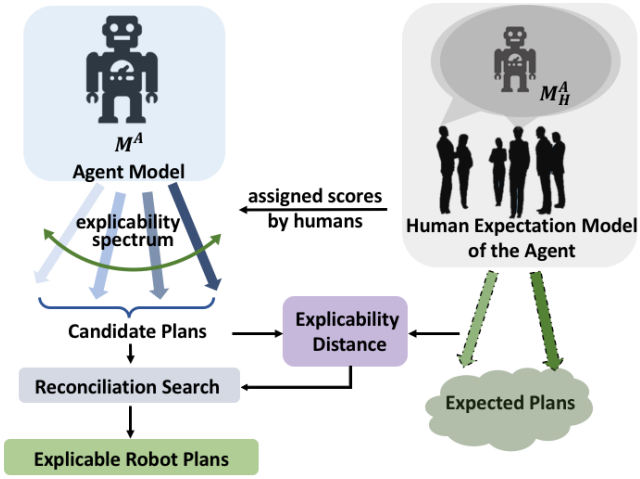


Figure 1: Schematic diagram of the first setting: Here a regression model called explicability distance is learned to fit plan scores assigned by humans to plan distances between robot's and human's expected plans. This gives heuristic for computing explicable plans.

performed. In such cases, $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$ can be constructed following the norms or protocols that are relevant to that domain. Most deployed products make use of inbuilt models of user expectations in some form or the other. In the first setting, we hypothesize that the plan distances (Srivastava et al. 2007; Nguyen et al. 2012) can quantify the distance between the robot plan $\pi_{\mathcal{M}^{\mathcal{R}}}$ and the expected plans $\pi_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$ from $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. The domain modeler constructs $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$, which is then used to generate expected plans. Then we compute the distance between plans from $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. The test subjects provide explicability assessments (scores reflecting explicability) for robot plans. These scores are mapped to the precomputed distances to derive an explicability distance. Then the plan generation process uses explicability distance as a heuristic to guide the search. This approach is illustrated in Figure 1.

In the following sections, we provide detailed methodologies to compute explicable plans in each of the settings. We also present results on each setting using physical robot-based domains.

2 Explicable Planning Problem

2.1 Classical Planning

A classical planning problem can be defined as a tuple $\mathcal{P} = \langle \mathcal{M}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{M} = \langle \mathcal{F}, \mathcal{A} \rangle$ is the domain model (that consists of a finite set $\mathcal{F}$ of fluents that define the state of the world and a set of operators or actions $\mathcal{A}$), and $\mathcal{I} \subseteq \mathcal{F}$ and $\mathcal{G} \subseteq \mathcal{F}$ are the initial and goal states of the problem respectively. Each action $a \in \mathcal{A}$ is a tuple of the form $\langle pre(a), eff(a), c(a) \rangle$ where $c(a)$ denotes the cost of an action, $pre(a) \subseteq \mathcal{F}$ is the set of preconditions for the action a and $eff(a) \subseteq \mathcal{F}$ is the set of the effects. The solution to the planning problem is a *plan* or a sequence of actions $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that starting from the initial state, sequentially executing the actions lands the agent in the goal state, i.e. $\Gamma_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$ where $\Gamma_{\mathcal{M}}(\cdot)$ is the transition function defined for the domain. The cost of the plan, denoted as $c(\pi)$, is given by the summation of the cost of all the actions in the plan π, $c(\pi) = \sum_{a_i \in \pi} c(a_i)$.

2.2 Problem Definition

The problem of explicable planning arises when the robot plan, $\pi_{\mathcal{M}^{\mathcal{R}}}$, deviates from the human's expectation of that plan, $\pi_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$. Here $\pi_{\mathcal{M}^{\mathcal{R}}}$ is the robot's plan solution to the planning problem, $\mathcal{P}^{\mathcal{R}} = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$, where $\mathcal{M}^{\mathcal{R}} = \langle \mathcal{F}^{\mathcal{R}}, \mathcal{A}^{\mathcal{R}} \rangle$; whereas, $\pi_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$ is the plan solution considering the human mental model, such that, $\mathcal{P}^{\mathcal{R}}_{\mathcal{H}} = \langle \mathcal{M}^{\mathcal{R}}_{\mathcal{H}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$, where $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}} = \langle \mathcal{F}^{\mathcal{R}}, \mathcal{A}^{\mathcal{R}}_{\mathcal{H}} \rangle$. The differences in the human mental model can lead to different plan solutions. We now formally define the explicable planning problem and the solution to it.

Definition 1. *The **explicable planning problem** is defined as a tuple $\mathcal{P}_{EPP} = \langle \mathcal{M}^{\mathcal{R}}, \mathcal{M}^{\mathcal{R}}_{\mathcal{H}}, \mathcal{I}^{\mathcal{R}}, \mathcal{G}^{\mathcal{R}} \rangle$, where,*

- *$\mathcal{M}^{\mathcal{R}} = \langle \mathcal{F}^{\mathcal{R}}, \mathcal{A}^{\mathcal{R}} \rangle$ is the robot's domain model where $\mathcal{F}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{R}}$ represent the actual set of fluents and actions available to the robot*
- *$\mathcal{M}^{\mathcal{R}}_{\mathcal{H}} = \langle \mathcal{F}^{\mathcal{R}}, \mathcal{A}^{\mathcal{R}}_{\mathcal{H}} \rangle$ is the human's mental model of the robot model where $\mathcal{F}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{R}}_{\mathcal{H}}$ represent the set of fluents and actions that the human thinks are available to the robot*
- *$\mathcal{I}^{\mathcal{R}}$ is the initial state of the robot*
- *$\mathcal{G}^{\mathcal{R}}$ is the goal state of the robot*

Here $\mathcal{A}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{R}}_{\mathcal{H}}$ represent that the action names, preconditions, effects and costs of the actions can be different. The initial state and the goal state are known to the human-in-the-loop. In order to compute the difference between the robot plan and the plan expected by a human, we need an evaluation function that computes the distance in terms of various aspects of the plan, like the action sequence, state sequence, etc. Let's denote such an evaluation function as δ^*. The solution to an explicable planning problem is an explicable plan that achieves the goal and minimizes the model differences while also minimizing the cost of the plan. We now formally define it as follows:

Definition 2. *(Zhang et al. 2017) A plan is an **explicable plan**, $\pi^*_{\mathcal{M}^{\mathcal{R}}}$, starting at $\mathcal{I}^{\mathcal{R}}$ that achieves the goal $\mathcal{G}^{\mathcal{R}}$, such that, $\arg\min_{\pi_{\mathcal{M}^{\mathcal{R}}}} c(\pi_{\mathcal{M}^{\mathcal{R}}}) + \delta^*(\pi_{\mathcal{M}^{\mathcal{R}}}, \pi_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}})$, where δ^* is an evaluation function.*

Since the evaluation function is not directly available to us, we learn an approximation of it using a combination of three plan distances measures. In the following section, we discuss our approach for learning the evaluation function and computing explicable plans.

3 Proposed Methodology

In this section, we quantify the explicability of the robot plans in terms of the *plan distance* between the robot plan $\pi_{\mathcal{M}^{\mathcal{R}}}$ and candidate expected plans $\pi_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$ from $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. The outline of our approach is illustrated in Figure 1. Our first aim is to compute the plan distances between the robot plans and the expected plans using plan distance measures. Then we map the human explicability assessments (scores reflecting explicability) of candidate robot plans to the plan distance measures in form of regression model called explicability distance. The synthesis of explicable plans is achieved by modifying the `Fast-Downward` (Helmert 2006) planner to incorporate an anytime search with explicability distance as the heuristic. This process results in incrementally more explicable plans.

3.1 Background on Plan Distance Measures

We now briefly look at the three plan distance measures introduced and refined in (Srivastava et al. 2007; Nguyen et al. 2012), namely action, causal link and state sequence distances.

Action distance We denote the set of unique actions in a plan π as $A(\pi) = \{a \mid a \in \pi\}$. Given the action sets $A(\pi_{\mathcal{M}^{\mathcal{R}}})$ and $A(\pi^*_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}})$ of two plans $\pi_{\mathcal{M}^{\mathcal{R}}}$ and $\pi^*_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$ respectively, the action distance is, $\delta_A(\pi_{\mathcal{M}^{\mathcal{R}}}, \pi^*_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}) = 1 -$

$\frac{|A(\pi_{\mathcal{M}^\mathcal{R}}) \cap A(\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}})|}{|A(\pi_{\mathcal{M}^\mathcal{R}}) \cup A(\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}})|}$. Here, two plans are similar (and hence their distance measure is smaller) if they contain same actions. Note that it does not consider the ordering of actions.

Causal link distance A causal link represents a tuple of the form $\langle a_i, p_i, a_{i+1} \rangle$, where p_i is a predicate variable that is produced as an effect of action a_i and used as a precondition for the next action a_{i+1}. The causal link distance measure is represented using the causal link sets $Cl(\pi_{\mathcal{M}^\mathcal{R}})$ and $Cl(\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}})$, $\delta_C(\pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) = 1 - \frac{|Cl(\pi_{\mathcal{M}^\mathcal{R}}) \cap Cl(\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}})|}{|Cl(\pi_{\mathcal{M}^\mathcal{R}}) \cup Cl(\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}})|}$.

State sequence distance This distance measure finds the difference between sequences of the states. Given two state sequences $(s^R_0, \ldots, s^R_n)$ and $(s^H_0, \ldots, s^H_{n'})$ for $\pi_{\mathcal{M}^\mathcal{R}}$ and $\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}$ respectively, where $n \geq n'$ are the lengths of the plans, the state sequence distance is, $\delta_S(\pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) = \frac{1}{n}\Big[\sum_{k=1}^{n'} d(s^R_k, s^H_k) + n - n'\Big]$, where $d(s^R_k, s^H_k) = 1 - \frac{|s^R_k \cap s^H_k|}{|s^R_k \cup s^H_k|}$ represents the distance between two states (where s^R_k is overloaded to denote the set of predicate variables in state s^R_k). The first term measures the normalized difference between states up to the end of the shortest plan, while the second term, in the absence of a state to compare to, assigns maximum difference possible.

3.2 Explicability Distance

We start with a general formulation for capturing a measure of explicability of the robot's plans using plan distances. A set of robot plans are scored by humans such that each action that follows the human's expectation in the context of the plan is scored 1 if explicable (0 otherwise). The plan score is then computed as the ratio of the number of explicable actions to the total plan length.

Definition 3. *A set of plans is a **set of expected plans**, $\mathcal{E}(\mathcal{P}^\mathcal{R}_\mathcal{H})$, for the planning problem $\mathcal{P}^\mathcal{R}_\mathcal{H} = \langle \mathcal{M}^\mathcal{R}_\mathcal{H}, \mathcal{I}^\mathcal{R}, \mathcal{G}^\mathcal{R} \rangle$, is a set of optimal cost plans that solve $\mathcal{P}^\mathcal{R}_\mathcal{H}$, $\mathcal{E}(\mathcal{P}^\mathcal{R}_\mathcal{H}) = \{\pi^{(i)}_{\mathcal{M}^\mathcal{R}_\mathcal{H}} | i = 1, \ldots, n\}$.*

This set of expected plans consists of the plan solutions that the human expects the robot to compute. But these plans are not necessarily feasible in the robot model, $\mathcal{M}^\mathcal{R}$.

Definition 4. *A **composite distance**, δ_{exp} is a distance between pair of two plans $\langle \pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}} \rangle$, such that, $\delta_{exp}(\pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) = ||\delta_A(\pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) + \delta_C(\pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) + \delta_S(\pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}})||^2$.*

Among the set of expected plans, we compute a distance minimizing plan as follows:

Definition 5. *A **distance minimizing plan**, $\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}$, is a plan in $\mathcal{E}(\mathcal{P}^\mathcal{R}_\mathcal{H})$, such that for a robot plan, $\pi_{\mathcal{M}^\mathcal{R}}$, the composite distance is minimized, $\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}} = \{\pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}} | \arg\min_{\pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}}} \delta_{exp}(\pi_{\mathcal{M}^\mathcal{R}}, \pi_{\mathcal{M}^\mathcal{R}_\mathcal{H}})\}$.*

Algorithm 1 Reconciliation Search

Input: $\mathcal{P}_{\mathcal{EPP}} = \langle \mathcal{M}^\mathcal{R}, \mathcal{M}^\mathcal{R}_\mathcal{H}, \mathcal{I}^\mathcal{R}, \mathcal{G}^\mathcal{R} \rangle$, max_cost, and explicability distance function *Exp*(. , .)
Output: $\mathcal{E}_{\mathcal{EPP}}$

$\mathcal{E}_{\mathcal{EPP}} \leftarrow \emptyset$ ▷ Explicable plan set
open $\leftarrow \emptyset$ ▷ Open list
closed $\leftarrow \emptyset$ ▷ Closed list
open.insert($\mathcal{I}, 0, \inf$)
while *open* $\neq \emptyset$ **do**
  $n \leftarrow$ *open*.remove() ▷ Node with highest $h(\cdot)$
  if $n \models \mathcal{G}$ **then**
    $\mathcal{E}_{\mathcal{EPP}}$.insert($\pi$ s.t. $\Gamma_{\mathcal{M}^\mathcal{R}}(\mathcal{I}, \pi) \models n$)
  end if
  closed.insert(*n*)
  for each $v \in$ successors(*n*) **do**
    if $v \notin$ *closed* **then**
      if *g(n)* + *cost*$(n, v) \leq$ *max_cost* **then**
        open.insert(v, *g(n)* + *cost*(n, v), *h(v)*)
      end if
    else
      if *h(n)* < *h(v)* **then**
        closed.remove(v)
        open.insert(v, *g(n)* + *cost(n, v)*, *h(v)*)
      end if
    end if
  end for
end while
return $\mathcal{E}_{\mathcal{EPP}}$

Definition 6. *An **explicability feature vector**, Δ, is a three-dimensional vector, which is given with respect to a distance minimizing plan pair, $\langle \pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}} \rangle$, such that, $\Delta = \langle \delta_A(\pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}), \delta_C(\pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}), \delta_S(\pi_{\mathcal{M}^\mathcal{R}}, \pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}) \rangle^T$.*

We approximate the evaluation function, δ^*, using a combination of the three plan distance measures as follows:

Definition 7. *The **explicability distance function** Exp($\pi_{\mathcal{M}^\mathcal{R}}$ / $\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}$), is a regression function, f, that fits the three plan distances to the total plan scores, with b as the parameter vector, and Δ as the explicability feature vector, such that, Exp($\pi_{\mathcal{M}^\mathcal{R}}$ / $\pi^*_{\mathcal{M}^\mathcal{R}_\mathcal{H}}$) $\approx f(\Delta, b)$.*

A regression model is trained to learn the explicability assessment (total plan scores) of the users by mapping this assessment to the explicability feature vector which consists of plan distances for corresponding plans.

3.3 Plan Generation

In this section, we present the details of our plan generation phase. We use the learned explicability distance function as a heuristic to guide our search towards explicable plans.

Reconciliation Search

Given an explicable planning problem $\mathcal{P}_{\mathcal{EPP}} = \langle \mathcal{M}^\mathcal{R}, \mathcal{M}^\mathcal{R}_\mathcal{H}, \mathcal{I}^\mathcal{R}, \mathcal{G}^\mathcal{R} \rangle$, the set of explicable plans in $\mathcal{M}^\mathcal{R}$ are defined as, $\mathcal{E}_{\mathcal{EPP}} = \{\pi^{*(i)}_{\mathcal{M}^\mathcal{R}} | i = 1, \ldots, m\}$, such

that, each $\pi^{*(i)}_{\mathcal{M}^{\mathcal{R}}}$ is found by performing reconciliation search. Here each $\pi^{*(i)}_{\mathcal{M}^{\mathcal{R}}}$ may have different degree of explicability.

Non-Monotonicity Since plan score is the fraction of explicable actions in a plan, it exhibits non-monotonicity. As a partial plan grows, a new action may contribute either positively or negatively to the plan score, thus making the explicability distance function non-monotonic. Consider that the goal of an autonomous car is to park itself in a parking spot on its left side. The car takes the left turn, parks and turns on its left indicator. Here the turning on of the left tail light after having parked is an inexplicable action. The first two actions are explicable to the human drivers and contribute positively to the explicability score of the plan but the last action has a negative impact and decreases the score.

Due to non-monotonic nature of explicability distance, we cannot stop the search process after finding the first solution. Consider the following: if e_1 is explicability distance of the first plan, then a node may exist in open list (set of unexpanded nodes) whose explicability distance is less than e_1, which when expanded may result in a solution plan with explicability distance higher than e_1. A greedy method that expands a node with the highest explicability score of the corresponding partial plan at each step is not guaranteed to find an optimal explicable plan (one of the plans with the highest explicability score) as its first solution. Therefore, to handle the non-monotonic nature, we present a cost-bounded anytime greedy search algorithm called *reconciliation search* that generates all the valid loopless candidate plans up to a given cost bound, and then progressively searches for plans with better explicability scores. The value of the heuristic *h(v)* in a particular state *v* encountered during search is based entirely on the explicability distance of the agent plan prefix $\pi_{\mathcal{M}^{\mathcal{R}}}$ up to that state, $h(v) = Exp(\pi_{\mathcal{M}^{\mathcal{R}}}/\pi'_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}})$ s.t. $\Gamma_{\mathcal{M}^{\mathcal{R}}}(\mathcal{I}, \pi_{\mathcal{M}^{\mathcal{R}}}) \models v$ and $\Gamma_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}(\mathcal{I}, \pi'_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}) \models v$.

We implement this search in the `Fast-Downward` planner. The approach is described in detail in Algorithm 1. At each iteration of the algorithm, the plan prefix of the agent model is compared with the explicable trace $\pi'_{\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}}$ (these are the plans generated using $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$ up to the current state in the search process) for the given problem. Using the computed distances, the explicability score for the candidate agent plans is predicted. The search algorithm then makes a locally optimal choice of states. We do not stop the search after generating the first solution, but instead, continue to find all the valid loopless candidate solutions within the given cost bound or until the state space is completely explored.

4 Experimental Analysis

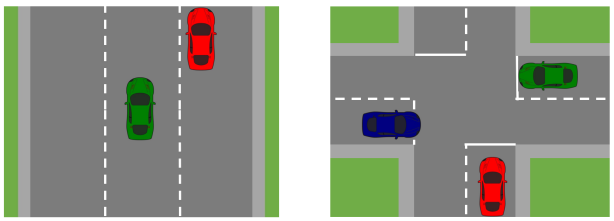
Here we present the evaluation of our system in simulated autonomous car domain and physical robot delivery domain.

4.1 Autonomous Car Domain

Domain model There are a lot of social norms followed by human drivers which are usually above explicit laws. This can include normative behavior while changing lanes or during turn-taking at intersections. As such this car domain has emerged as a vibrant testbed for research (Sadigh et al. 2016; Kyle Hollins Wray 2017) in the HRI/AI community in recent times. In this work, we explore the topic of explicable behavior of an autonomous car in this domain, especially as it related to mental modeling of the humans in the loop.

No.	Questions	Yes	No
1	Autonomous car should always maintain itself in the center of the lane unless it is changing lanes	16	4
2	The car should automatically turn signal lights on before lane change.	20	0
3	The car should automatically turn signal lights on when the car starts swerving away from the center of the lane.	9	11
4	At a four-way stop, it should automatically provide turn signal lights when it is taking left or right turns.	18	2
5	It should slow down automatically when it is swerving off the center of the lane.	15	5
6	It should slow down automatically when there is an obstacle (pedestrian or parked vehicle) ahead in the lane.	17	3
7	Check one: When an emergency vehicle is parked on the rightmost lane, it should (1) automatically follow the move over maneuver, (2) whenever possible follow the move over maneuver.	9	11
8	Check one: At a four-way stop, it should (1) wait for the intersection to be clear and to be extra safe. (2) wait for the other cars unless it is its turn to cross over.	15	5

Table 1: The questionnaire used in the human study for Car Domain, and the tally of answers given by participants. For the last two questions, the participants were asked to choose one of the two options, and the "yes" tally corresponds to the first answer, "no" to the second.



(a) (b)

Figure 2: Simulated Autonomous Car Domain. Here only the red car is autonomous. (a) It is having difficulty merging to the middle lane and is confusing the human driver with its signals. (b) It is waiting at a 4-way stop even though it is its turn to cross.

Algorithm	Car R^2 %	Delivery R^2 %
Ridge Regression	53.66	31.42
AdaBoost Regression	61.31	66.99
Decision Tree Regression	74.79	39.61
Random Forest Regression	90.45	75.29

Table 2: Accuracy for car and delivery domain.

In our autonomous car domain (modeled in PDDL), the autonomous car model $\mathcal{M}^{\mathcal{A}}$ consists of lane and car objects as shown in Figure 2. The red car is the autonomous car in the experiments and all other cars are assumed to have human drivers. The car objects are associated with predicates defining the location of the car on a lane segment, status of left and right turn lights, whether the car is within the speed limit, the presence of a parked police car, and so on. The actions possible in the domain are with respect to the autonomous car. These actions are *Accelerate*, *Decelerate*, *LeftSqueeze*, *RightSqueeze*, *LeftLight* {*On, Off*}, *RightLight* {*On, Off*}, *SlowDown* and *WaitAtStopSign*. To change a lane, three consecutive actions of {*Left, Right*} *Squeeze* are required.

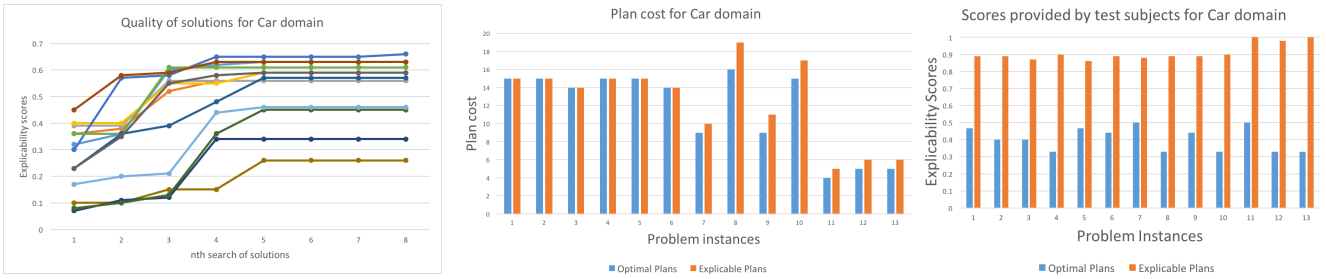
From $\mathcal{M}^{\mathcal{A}}$ we generated a total of 40 plans (consisting of both explicable and inexplicable behaviors) for 16 different planning problems. These plans were assessed by 20 human subjects, with each subject evaluating 8 plans (apart from 1 subject who evaluated 7 plans). Also, each plan was evaluated by 4 different subjects. The overall number of training samples was 159. The test subjects were required to have a state driving license. The subjects were provided with the initial state and goal of the car. After seeing the simulation the plan, they had to record whether they found each action explicable or not. The assessment had two parts: one part involved scoring each autonomous car action with 1, if explicable, and 0 otherwise (plan score was calculated as the fraction of actions in the plan that were labeled as explicable); the other part involved answering a questionnaire on the preconditions, effects of the agent actions. It consisted of 8 questions with *yes/no* answers. The questions used for constructing the domain and the corresponding answers are provided in Table 1. For each question, the answers with votes were used by the domain modeler to construct $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$. The questions with divided opinions (3 and 7) were not included in the models as some found that behavior explicable while some others did not. The $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$ domain consists of the same state predicates but ended up with different action definitions with respect to preconditions, effects, and action-costs.

Following are two examples of how the feedback was interpreted by the domain modeler. For question 1, the majority of answers agreed with the statement. Therefore for actions Accelerate and SlowDown, additional preconditions like *(not (squeezingLeft ?x)), (not (squeezingLeft2 ?x)), (not (squeezingRight ?x)), (not (squeezingRight2 ?x))* were added, where x stands for car. For question 8, since the answers agreed with choice 2, actions *waitAtStopSign1*, *waitAtStopSign2*, *waitAtStopSign3* were replaced by a new general action *waitAtStopSign*. This action removed predicates *waiting1, waiting2, waiting3* from the action definition. Also, actions *atStopSignAccelerate*, *atStopSignLeft*, *atStopSignRight* were changed to remove the precondition *waiting3* (these actions thus had two definitions in $\mathcal{M}^{\mathcal{A}}$ to allow for explicable behavior, one with higher cost).

Defining the explicability distance For the training problems, explicable plans were generated using the model $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$. Since some actions names were not common to both the domains, an explicit mapping was defined between the actions over the two domains. This mapping was done in order to support plan distance operations performed between plans in the two domains (for the plan distances to be used effectively common action names are required).

As noted in Definition 6, features of the regression model are the three plan distances and the target is the score associated with the plans. We tune the hyperparameters by performing grid search over parameters like the number of trees, depth of tree and the minimum number of nodes to perform sample split. The results for different learning models are as shown in Table 2. We tried several ensemble learning algorithms to improve the accuracy of our model, out of which random forest regression gave the best performance. Random forests allow selection of a random subset of features while splitting the decision node. We evaluate the goodness of fit of the model, using the coefficient of determination or R^2. After training process, the new regression model was found to have 0.9045 R^2 value. That is to say, 90% of the variations in the features can be explained by our model. Our model predicts the explicability distance between the agent plans and human mental model plans, with a high accuracy.

Evaluation We evaluated our approach on 13 different planning problems. We ran the algorithm with a high cost bound, in order to cover the most explicable candidate plans for all the problems. The Figure 3a reports the explicability scores of the first 8 solutions generated by our algorithm for 13 test problems. From this graph, we note that the reconciliation search is able to develop plans with incrementally better explicability scores. These are the internal explicability scores (produced by the explicability distance function). From Figure 3b, we see for the last 7 problems, our planner generated explicable plans with a cost higher than that of optimal plans; a planner insensitive to explicability would not have been able to find these expensive but explicable plans. This additional cost can be seen as the price the agent pays to make its behavior explicable to the human. For the first 6 problems, even though the cost is same as that of the optimal plans, a planner insensitive to explicability cannot guarantee the generation of explicable plans. From Figure 3c, the test subjects provided higher explicability scores for the explicable plans than the optimal plans for all 13 problems. The plan scores given by 10 test subjects were computed as the ratio of explicable actions in the plans to the total plan length. Testing phase protocol was same as that



(a) (b) (c)

Figure 3: Car domain: (a) The graph shows how the search process finds plans with incrementally better explicable scores. Each color line represents one of the 13 different test problems. The markers on the lines represent a plan solution for that problem. The y-axis gives the explicability scores of the plans and the x-axis gives the solution number. The optimal plans generated using the `Fast-Downward` planner were compared for (b) plan costs (c) explicability scores provided by test subjects.

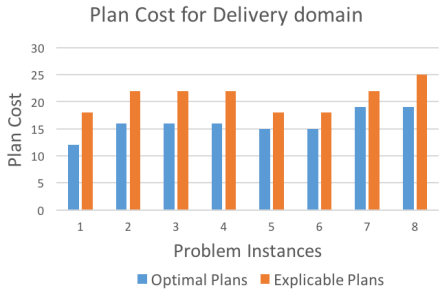


Figure 4: For delivery domain test problem instances, the optimal and explicable plans were compared for their plan costs.

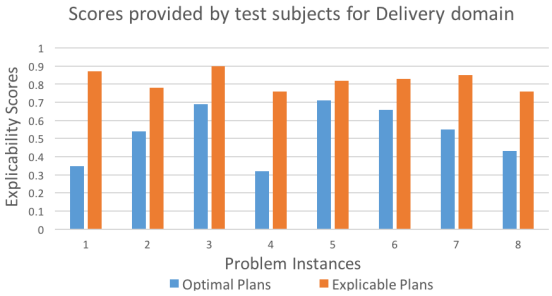


Figure 5: For delivery domain test problem instances, the optimal and explicable plans were compared for explicability scores provided by test subjects.

of the training phase, except for the questionnaire. The plan scores were averaged over the number of test subjects. In this domain, an inexplicable plan for changing lanes from *l2* to *l1* can be *LeftSqueeze-l2-l1, LeftSqueeze-l2-l1, LeftSqueeze-l2-l1, LeftLightOn* whereas an explicable plan would be *LeftLightOn, LeftSqueeze-l2-l1, LeftSqueeze-l2-l1, LeftSqueeze-l2-l1*. The ordering of *LeftLightOn* action decides whether the plan is explicable.

4.2 Robot based Delivery Domain

This domain is designed to demonstrate inexplicable behaviors of a delivery robot. The robot can deliver parcels/electronic devices and serve beverages to the humans. It has access to three regions namely kitchen, reception and employee desk. For the evaluation, we used a Fetch robot, which has a single arm with parallel grippers for grasping objects. It delivers beverages, parcels, and devices using a tray. Whenever the robot carries the beverage cup there is some risk that the cup may tip over and spill the contents all over the electronic items on the tray. Here the robot has to learn the context of carrying devices and beverages separately even if it results in an expensive plan (in terms of cost or time) for it. A sample plan in this domain with explicable and inexplicable plans is illustrated in Figure 6. Here, in the inexplicable version, the robot delivers device and beverage together. Although it optimizes the plan cost, the robot may tip the beverage over the device. Whereas, in the explicable version robot delivers the device and beverage cup separately, resulting in an expensive plan due to multiple trips back and forth. [1].

Domain and explicability distance This domain is also represented in PDDL. Here both the models were provided by the domain expert. The robot model has the following actions available: `pickup`, `putdown`, `stack`, `unstack` and `move`. The domain modeler provided $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$ based on usual expectations of robots with a similar form factor. For example, in $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$, certain actions which could be perceived riskier (like, carrying the device and cup in the same tray) had a higher cost due to the possibility of damaging the items. Thus $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$ incentivizes the planner to choose safer actions. Both models have same state space and action representation. The model differences lie in the action-costs as well as preconditions and effects of actions. There were 20 plan instances created for each of the 13 planning problems. Each of the plans was labeled by 2 human subjects, which resulted in 40 different training samples (some problems have multiple solution plans). The performance of different ensemble learning techniques is as shown in Table 2. We again

[1] A video demonstration can be viewed at `https://bit.ly/2JweeYk`

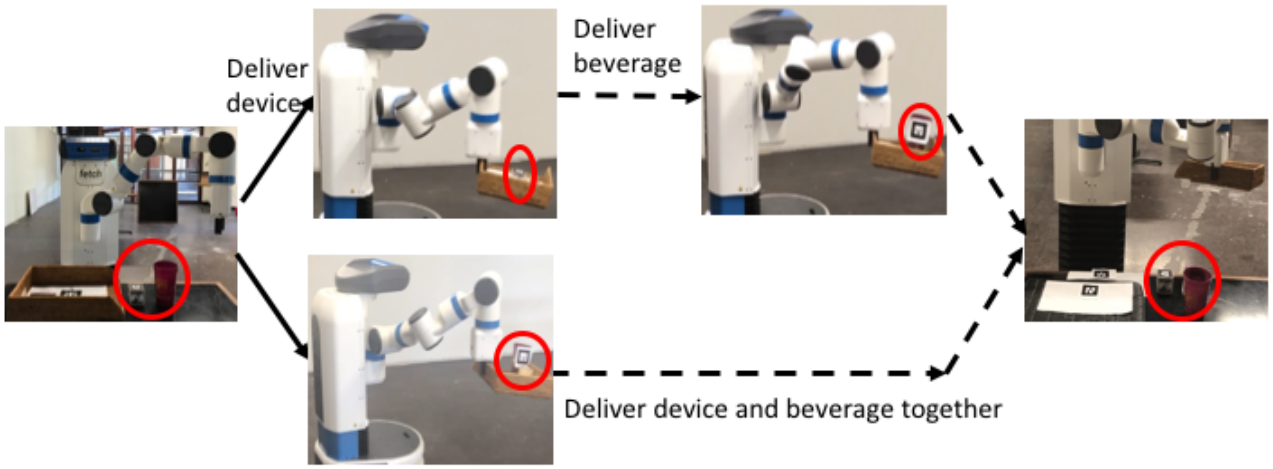


Figure 6: The goal of the agent is to deliver the device and beverage cup to the destination. In the cost-optimal plan, agent delivers both the items together, whereas in the explicable plan the agent delivers the items separately. A video demonstration can be viewed at `https://bit.ly/2JweeYk`

use the random forest regression model with an accuracy of 75%.

Evaluation For evaluation of this domain, 8 new planning problems (similar to the one shown in Figure 6) were used. These plan instances with a pictorial representation of intermediate behavioral states were labeled by 9 test subjects. For testing phase, the same protocol was followed as that in training phase. Figure 4 shows the comparison between the plan costs. Whenever the items consist of beverage cups, the robot has to do multiple trips, therefore all the explicable plans are more expensive than the optimal plans. For such scenarios, if a robot uses a cost-optimal planner, the plans chosen will always be inexplicable with respect to the plan context. In Figure 5, we compare the explicability scores provided by test subjects. The explicability scores provided by the subjects are higher for explicable plans. Some plans involved the robot stacking cups over devices to generate cost-optimal plans. These plans ended up receiving least scores.

5 Related Work

An important requirement for an AI agent collaborating with a human is the ability to infer human's goals and intents and use this information to inform its planning process. There have been various efforts in the direction of human aware planning to account for human's beliefs and goals (Sisbot et al. 2007; Cirillo, Karlsson, and Saffiotti 2010; Mainprice et al. 2011; Chakraborti et al. 2015; 2016) to encourage human agent interactions. Human-AI collaboration can involve both purely virtual tasks (Allen 1994; Ai-Chang et al. 2004; Sengupta et al. 2017), purely physical tasks (Alami et al. 2005; Montreuil et al. 2007) or their combination.

HRI community has focused on some related issues in the context of the human-AI collaboration, but they have mainly considered purely physical tasks involving manipulation and motion. Our multi-model explicability framework can be adapted to the HRI scenarios too. Indeed, there has been some work on explicability of motion planning that goes under the name of *predictability* (Dragan and Srinivasa 2013; Fisac et al. 2016; Knepper et al. 2017). Predictability assumes the goal is known and deals with generating likely (straightforward) motion towards it (when the goal is unknown, legibility is defined in this context, which conveys with a motion which goal the robot is reaching for). These can be understood in terms of $\mathcal{M}^{\mathcal{A}}$ - $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$ models. The difference is that for motion tasks, the advantage is that humans have well understood intuitive/common-sense models (e.g. move in straight lines as much as possible). This reduces the difficulty of generating explicable plans considerably, as the human model is fixed, and the *distance* between the expected and observed trajectories can be easily measured. However, in many general domains that may involve both physical and virtual tasks (or purely virtual tasks), there are no such widely accepted intuitive human models. In such cases, the full generality of our approach will be necessary.

Previously, (Zhang et al. 2017) showed the generation of explicable plans by approximating $\mathcal{M}^{\mathcal{A}}_{\mathcal{H}}$ to a labeling scheme given that was learned using conditional random fields. In this work, we study how the problem changes when the human model is known and is represented in terms of an action model similar in representation to the agent's model. In many domains (household/factory scenarios), there is a clear expectation of how a task should be performed and $M^{\mathcal{A}}_{H}$ can be constructed using relevant norms. Most deployed products make use of inbuilt models of user expectations. This allows for a more informed plan generation process, which we refer to as *reconciliation search*. This assumption on model representation also allows us to investigate the relationship between the traditional measures of plan distances studied in the planning community and *explicability measure* im-

plicitly used by the humans to compare the agent plans with the expected plans from their model.

6 Conclusion

In conclusion, we showed how the problem of explicable plan generation can be modeled in terms of plan distance measures studied in existing literature. We also demonstrated how a regression model on these distance measures can be learned from human feedback and used to guide an agent's plan generation process to produce increasingly explicable plans in the course of an anytime search process. We demonstrated the effectiveness of this approach in a simulated car domain and a physical robot delivery domain.

Going forward, two interesting avenues of future work include (1) learning to model plan similarities when domains are represented at different levels of abstraction (e.g. with experts versus non-experts in the loop); and (2) exploring interesting modes of behavior that can build upon the developed understanding of human expectations (e.g. by being inexplicable to draw attention to an event or, in the extreme case, being deceptive by conforming to or reinforcing wrong human expectations). The former may require a rethink of existing plan distance metrics to account for the effects of model difference in the presence of lossy abstractions.

Acknowledgements

This research is supported in part by the ONR grants N00014161-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, and the NASA grant NNX17AD06G.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.-J.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; et al. 2004. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems* 19(1):8–12.

Alami, R.; Clodic, A.; Montreuil, V.; Sisbot, E. A.; and Chatila, R. 2005. Task planning for human-robot interaction. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, 81–85. ACM.

Allen, J. F. 1994. Mixed initiative planning: Position paper. In *ARPA/Rome Labs Planning Initiative Workshop*.

Chakraborti, T.; Briggs, G.; Talamadupula, K.; Zhang, Y.; Scheutz, M.; Smith, D.; and Kambhampati, S. 2015. Planning for serendipity. In *IROS*.

Chakraborti, T.; Zhang, Y.; Smith, D. E.; and Kambhampati, S. 2016. Planning with resource conflicts in human-robot cohabitation. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*.

Cirillo, M.; Karlsson, L.; and Saffiotti, A. 2010. Human-aware task planning: An application to mobile robots. *ACM Transactions on Intelligent Systems and Technology (TIST)* 1(2):15.

Dragan, A., and Srinivasa, S. 2013. Generating legible motion. In *Proceedings of Robotics: Science and Systems*.

Fan, X.; Oh, S.; McNeese, M.; Yen, J.; Cuevas, H.; Strater, L.; and Endsley, M. R. 2008. The influence of agent reliability on trust in human-agent collaboration. In *Proceedings of the 15th European conference on Cognitive ergonomics: the ergonomics of cool interaction*, 7. ACM.

Fisac, J.; Liu, C.; Hamrick, J. B.; Sastry, S.; Hedrick, J. K.; Griffiths, T. L.; and Dragan, A. D. 2016. Generating plans that predict themselves. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)* 26:191–246.

Knepper, R. A.; Mavrogiannis, C. I.; Proft, J.; and Liang, C. 2017. Implicit communication in a joint action. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 283–292. ACM.

Kyle Hollins Wray, Stefan J. Witwicki, S. Z. 2017. Online decision-making for scalable autonomous systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4768–4774.

Langley, P. 2016. Explainable agency in human-robot interaction. In *AAAI Fall Symposium Series*.

Mainprice, J.; Sisbot, E. A.; Jaillet, L.; Cortés, J.; Alami, R.; and Siméon, T. 2011. Planning human-aware motions using a sampling-based costmap planner. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 5012–5017. IEEE.

Montreuil, V.; Clodic, A.; Ransan, M.; and Alami, R. 2007. Planning human centered robot activities. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2618–2623. IEEE.

Nguyen, T. A.; Do, M.; Gerevini, A. E.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence* 190(0):1 – 31.

Sadigh, D.; Sastry, S.; Seshia, S. A.; and Dragan, A. D. 2016. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; Vadlamudi, S. G.; and Kambhampati, S. 2017. Radar-a proactive decision support system for human-in-the-loop planning. *AAAI Fall Symposium on Human-Agent Groups: Studies, Algorithms and Challenges*.

Sisbot, E. A.; Marin-Urias, L. F.; Alami, R.; and Simeon, T. 2007. A human aware mobile robot motion planner. *IEEE Transactions on Robotics* 23(5):874–883.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *IJCAI*, 2016–2022.

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan explicability and predictability for robot task planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*.

Generating Explanations for Mathematical Optimisation: Solution Framework and Case Study

Christina Burt and **Katerina Klimova**
Satalia, London
{christina,katerina}@satalia.com

Bernhard Primas
School of Computing, University of Leeds
scbjp@leeds.ac.uk

Abstract

In this paper, we address the problem of generating explanations automatically for mathematical optimisation. Explanations can improve the way users interact with optimisation tools and help them trust the solutions. One of the challenges of generating explanations for mathematical optimisation is to reconstruct meaning from abstract mathematical expressions. We present a general framework in which we exploit problem diversity exploration in order to infer meaning from algorithm results, and present an automatic sentence generator that works with this framework. Finally, we describe an industrial project where we applied these algorithms.

Introduction

In the field of mathematical optimisation, and particularly in academia, we seek simplified expressions of a problem class so that we can study its structure in the purest sense. However, real-world problems are often best expressed using both hard and soft constraints—meaning that some restrictions on the solution space are simply *preferences*. Additionally, it is common to find that real-world problems have multiple, typically competing, objective functions.

Finding a high-quality or even optimal solution is not the most important aspect for every real-world optimisation application. For some applications it is desirable to obtain several diverse solutions that come with explanations of their advantages and drawbacks such that they are understandable for a human user. It is in this spirit that we define the following general optimisation problem:

> *Given a set of problem inputs, determine the solution(s) that optimise a set of objectives such that a set of hard and soft constraints is satisfied.*

In the context of explainability, we can extend the above problem as follows:

> *The output is a set of solutions with explanations.*

In this paper, we study the problem of solving an optimisation problem where diverse solutions are useful, and exploit the diversity in order to obtain human-readable explanations for the user.

State-of-the-art mathematical optimisation algorithms fall short when it comes to providing adequate explanations to users. These algorithms have traditionally focused on finding a single solution with an optimal objective function value. Once such a solution is found, the algorithm returns the solution and terminates. The users are not provided with information to help them understand why the returned solution was considered 'best', or even if it is the *only* 'best' solution. Therefore, the optimisation problems we consider in this paper require extensions to existing technology.

The development of an algorithm with explanations is challenging for a number of reasons. First, explanations can be highly domain-specific. Mathematical optimisation algorithms, through the manner of encoding, tend to abstract away domain-specific information. This leaves a difficult task to reconstruct the relationship between constraints, variables and objectives in a meaningful way. Secondly, providing an explanation requires supportive and easily understandable sentence construction so that human users can understand them. This incorporates various difficulties including understanding what an explanation should and should not contain, and the fact that even solutions obtained by entirely theoretically well-understood algorithms may be hard to explain. As a consequence, a structural approach for finding diverse solutions is required to enable our framework to provide sensible and supportive explanations.

The contributions of this paper are as follows:

- *General explanation algorithm development*. We present a framework for eliciting explanation information from a diverse search space for mathematical optimisation methodologies. The algorithm aims towards providing rich explanation to users.
- *Implementation and illustration as a case study*. We illustrate the use of our framework for an industrial project as an anonymised case study.
- *Bridging the gap between mathematical optimisation and explainable AI*. Explainability is not well explored in the field of mathematical optimisation. This paper brings explainable AI to this field.

Background

For every application in the context of explainable AI, there is a set of functional abilities an autonomous agent should be able to provide. Some functional abilities may differ for different types of applications, whereas other abilities should be provided for an entire class of applications (Langley et al. 2017). For our application, we have identified three functional abilities that should be provided:

- *Rich explanation*. The framework should provide explanations for (a) the solutions chosen, (b) why the solutions were selected, (c) how many constraints and user preferences are satisfied or violated and (d) how good each solution is, i.e. what the objective function values are. If no solution is found, this should also be explained.
- *Understandable explanation*. Explanations provided should come in the form of full sentences understandable for a non-scientist that was not involved in the development process of the framework.
- *Interactivity*. The framework should be designed to allow the user to interact with the exploration of diverse solutions. This should enable the prioritisation of certain restrictions or objectives in a natural way.

(Fox, Long, and Magazzeni 2017) explain the need for explainable AI. Three key reasons have been identified that drive the motivation for explainable AI: trust, interaction and transparency. We explain their importance for our application as follows. If a scheduler considers using our framework in order to create a timetable, the scheduler needs rich explanation for every timetable suggested in order to *trust* the quality and meaningfulness of the suggested timetable. Creating a timetable is a step-by-step process, in which the user typically wants to fix the position of certain items first as the user considers them to be of high priority. This requires our framework to be designed such that it enables *interaction* with users. Finally, *transparency* is required for situations in which our framework makes a decision which is wrong or unexpected for the user so that the user can react accordingly.

(Smith 2012) discusses difficulties standing in the way of a more automated approach for which only minimal or no work at all is performed by a human. Examples include *oversubscription* (not all hard and soft constraints are satisfiable simultaneously), preferences (some goals are prioritised over others) or uncertainty (the input data is only partially known at the time of scheduling).

In related work, (Horiguchi, Tomoto, and Hirashima 2015) present a classification approach to components of a model in order to help students learn the appropriate application of the models. They create *model fragments* and use laws of physics to connect them. These connections lead naturally to explanations, which in turn can help students understand the processes better.

In our work, we will focus on reconstructing the connections between components of an optimisation model. The model itself can be formulated by

$$\max \{c^T x | Ax \leq b, x \geq 0, x \in \Omega\},$$

where $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are vectors, $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix and $x \in \mathbb{R}^n$ is the vector containing the variables. Additionally to constraints $Ax \leq b$ and $x \geq 0$, the set Ω is used to impose further restrictions on x. If no further restrictions are required, i.e. $\Omega = \mathbb{R}^n$, the problem is called a *linear program*. For the special case in which only integer values are allowed, i.e. $\Omega = \mathbb{Z}^n$, the problem is called an *integer linear program*. If only some (but not all) variables are required to be integer, the problem is called a *mixed-integer linear program*.

In the context of linear programming, sensitivity analysis is a methodology that can be used to generate rich explanations. Using this methodology, one can automatically generate information such as (Winston and Goldberg 2004):

- whether the optimal solution is unique or not;
- how much the coefficient vector c in the objective function can vary before a different solution becomes optimal;
- how much the right-hand-side vector b can vary before a different solution becomes optimal.

The theory of duality provides these results for linear programming. However, for mixed-integer programming, no such duality theory, and explanation pathway, exists. The closest concept available for understanding infeasibility is the one of Irreducible Infeasible Sets (Chinneck 2008). An IIS represents a smallest set of constraints that cannot be satisfied simultaneously, i.e. removing any of these constraints would make the remaining set of constraints satisfiable. Only for linear programming, there exist exact methods for determining the IIS. However, for mixed-integer programming heuristics can be used such as, simply, removing constraints one at a time in order to discover a feasible set.

General Approach

In order to solve the problem presented in the previous section, we follow an intuitive and effective heuristic in Algorithm 1. We provide an illustration of the algorithm in flow chart form in Figure 1.

Algorithm 1 Detecting diverse solutions.

```
1: H ... set of hard constraints
2: S ... list of soft constraints
3: O ... set of objective functions
4: C ← S ∪ H
5: for i = 0 : (|S| − 1) do
6:     Check if constraints in C are satisfiable
7:     if C is not satisfiable then
8:         Record explanation (1)
9:     else if C is satisfiable then
10:        for o ∈ O do
11:            Solve, using constraints C and objective o
12:            Record explanation (2)
13:    Remove S[i] from C
```

Explanation (1), in its simplest form, is *"No solution exists, constraint set $\mathcal{C}$ is too restrictive"*. Explanation (2) is of the form *"An optimal solution exists for objective o, all constraints are satisfied from constraint set $\mathcal{C}$"*.

Removing constraints one at a time allows us to logically infer causality. Of course, it is possible to take this further. In the decision hierarchy described, only the last constraint that created infeasibility provably causes the infeasibility. Therefore, we could allow those constraints eliminated earlier in the process to re-enter the constraint set. The usefulness of such an exhaustive analysis depends on the application of the explanation generator. The richness of the 'infeasible' explanation can be enhanced by starting with the most relaxed constraint set and adding constraints one at a time, in-

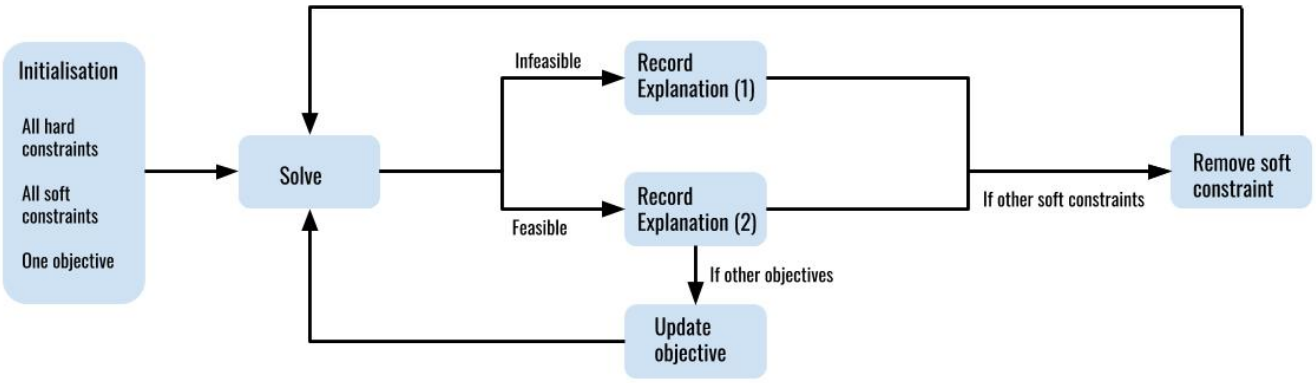


Figure 1: A flow diagram illustrating the search strategy in Algorithm 1

stead, as it allows us to pinpoint the precise constraints that created the present infeasibility in the system.

The incorporation of soft constraints can be used to achieve particular goals. To this end, an order is assigned to the soft constraints. For example, starting with soft constraints we expect to be most restrictive allows us to find infeasible solutions sooner, and therefore eliminate those constraints that lead to infeasibility. On the other hand, starting with soft constraints we expect to be least restrictive allows us to potentially satisfy more soft constraints and therefore achieve more preferences for the user. The implication of ordering soft constraints can lead to richer explanations if the context of the constraints is well understood. In mixed-integer programming, this cannot be automatically derived from a model, but can be achieved with domain knowledge.

Since there is no sufficient sensitivity analysis theory for mixed-integer programs, it is challenging to obtain relationships between constraints, variables and the solution space. What we achieve in Algorithm 1 is a mimicking of sensitivity analysis. We consider the set of hard constraints and start by adding all soft constraints. In most cases, it is not feasible to satisfy all constraints simultaneously. Armed with this information, we explore the ways in which soft constraints interact with the feasible region, and seek diversity in our solutions by trialling different combinations of objectives.

Achieving diversity in this controlled manner is an important part of deriving explanations. Since we can usually measure the quality of solutions and compare them against one another, it is useful to not only have diverse solutions but also to have the precise reasons for this diversity. This construction of diversity leads directly to richer explanations.

Allowing the user to select preferences, i.e. soft constraints, is also important. Often, the user is the expert and will pre-filter the constraints to a more interesting sub-set within the context. In this way, the unsophisticated and most uninteresting part of the exhaustive search is replaced by inference from the user.

Automatically Generating Sentences

Algorithm 2 presents an illustration of how we achieved explanations in sentence form. The main arguments are:

- P is a `boolean` that denotes whether there exists a solution without any constraint relaxation.
- If P is `true`, each constraint in the constraint set $\mathcal{C}$ has assigned a boolean indicating if it was applied in the solving round or not.
- *objective* is a `string` stating the objective function currently under consideration.
- Finally *item* contains information about the type of the object we are creating the solution for.

The algorithm starts with a check whether the perfect solution has been found. In that case, the explanation message states that the solution fulfils all constraints and is the best among others. In the case that there is no solution for the full set of constraints the algorithm starts the explanation message based on the type of objective function. It then continues with the information that this solution does not satisfy some of the constraints, and names these constraints in a user-friendly way. The list of unsatisfied constraints is then added and the explanation message is returned.

Case study

In this section we describe a pilot client project, for which we created an interactive scheduling tool. To protect our client's identity, we describe the project using an analogue. The essential features of the problem and solution, with respect to the topic of this paper, are intact.

Our study case is based on the problem of scheduling different free time activities in a leisure centre. The centre offers a range of different activities for all age categories, starting with preschool kids and ending with courses for adults. Each of the age categories has a specific time range during the day and also different activities. In our project, we focused on scheduling activities for the two youngest age groups: preschool kids and primary-school kids.

The preschool group has lessons planned between 1 pm and 5 pm and a primary-school group from 1 pm to 7 pm,

Algorithm 2 Automatically generating explanations

Require: $\mathcal{C}$, P, $item$, $objective$

if not P **then**
  explanation ← "There are no $<$ *item.type* $>$ that account for $<$ *all constraints from* $\mathcal{C}$ $>$."
else
  explanation ← "This $<$*item.type*$>$ is the most recommendable in terms of $<$ *objective* $>$ and $<$*full set of hard constraints*$>$"
  for constraint $\in \mathcal{C}$ **do**
    if not constraint.applied **then**
      nonActiveList ← constraint
  for constraint ∈ nonActiveList **do**
    explanation ← ", but may not satisfy"
    if first constraint **then**
      explanation ← " $<$*constraint*$>$"
    else if last constraint **then**
      explanation ← " and $<$*constraint*$>$"
    else
      explanation ← ", $<$*constraint*$>$"
    explanation ← " restrictions."
return explanation

from Monday to Friday. These hours have been set by the leisure centre with regards to kids age and school attendance.

The leisure centre offers three types of lessons. The first category accommodates all physical activities such as swimming, dancing and yoga. The second category contains all the creative activities starting with drawing classes and ceramics and ending with musical education. The third category is formed by educative courses strongly represented by language classes of different levels. Some of these activities are accessible for kids of all ages, but some, such as playing the accordion, are only for 6 years and older, and vice versa.

Some of the classes are fixed in the schedule from the beginning due to the specific requirements. The rest of the classes are then planned with respect to many constraints. The constraints include activity continuity for the instructors when the leisure centre aims to plan the activities with same requirements for different levels in a block. This continuity feature is one of our soft constraints. Another soft constraint is then the aspiration for balanced schedules and balanced offers from all activity types. Hard constraints were mainly driven by the target age and facility availability.

Our project was split into three types of scenarios. The first scenario is to recommend the activity for a chosen time slot. The idea behind this scenario is to help the scheduler choose the right activity with respect to all or as many as possible fulfilled constraints. The second scenario was then to recommend the best slot for given activity. This describes the situation when some of the lectures are for example sponsored by a council, therefore, the scheduler needs to schedule these activities and our tool recommends best slots and explains why these sets are the most preferable. The last scenario is then focused on the themed weeks. These weeks are not necessarily only pre-Christmas week, pre-Easter week and other holiday-specific weeks, but also weeks with a general theme like 'sea life'. In these weeks the leisure centre plans all lessons which are related to the theme such as drawing sharks or saying "fish", "boat" or "sand" in other languages. In this scenario, the tool recommends the whole sequence of classes related to the theme.

Implementation

The implementation of the problem is composed of two parts. The first is a wrapper which we built around an optimisation solver to get results efficiently and with the explanation. This part follows Algorithm 1 and is described further below. The second consists of Satalia's SolveEngine solver and supporting functions. After solving is completed, the results are passed back to the wrapping function which evaluates the results and takes one of the following actions:

- If solver did not find any solution, one of the active constraints is deactivated and the problem is passed again to the solver.
- If the solver found a solution but more variants of the solution are needed, the chosen activity or time slot is removed and the problem is passed back to the solver.
- If there are enough diverse solutions or no new solution can be found and no further constraint relaxation is possible, the search is terminated.

After we have all solutions which we need or can obtain, we collect all explanations created for each solution using Algorithm 2 and return them as a list of tuples containing recommended activity or time slot and explanation. In the project, we were given different *item.types* so that we extended the algorithm accordingly in order to keep the explanation meaningful.

We added additional functionality to the tool, such as automatically generation of tweets to announce new activities. The users interacted with the pilot recommendation tool via an interface. Our feedback was very positive for the use of real sentences and explanations, as these were features the users could readily understand. In fact, they appreciated these elements far more than the optimisation algorithm under the hood, which highlights the importance of explanations of abstract algorithmic results for users.

Conclusion

We have presented a framework for automatically generating explanations about the search space for the user to interpret. In a case study, we communicated this information to the users by automatically generated sentences. Our approach relies on the user to interpret the relationship between model components from the information we present as an explanation. By utilising this domain knowledge of the user, interpretations of the model components can be automatically encoded on an application basis. We continue to explore this aspect in our ongoing projects.

Acknowledgement

The tool for the case study was created by a larger project team at Satalia including Harry Edmonds, Yohann Pitrey, Alex Lilburn, Vito Tumas and Daniel Hulme.

References

Chinneck, J. W. 2008. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer US.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*, 24–30.

Horiguchi, T.; Tomoto, T.; and Hirashima, T. 2015. A framework of generating explanation for conceptual understanding based on "semantics of constraints". *Research and Practice in Technology Enhanced Learning* 10(1):1–21.

Langley, P.; Meadows, B.; Sridharan, M.; and Choi, D. 2017. Explainable Agency for Intelligent Autonomous Systems. In *Twenty-Ninth AAAI Conference on Innovative Applications (IAAI-17)*, 4762–4763.

Smith, D. E. 2012. Planning as an Iterative Process. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2180–2185.

Winston, W. L., and Goldberg, J. B. 2004. *Operations Research: Applications and Algorithms*. Thomson Brooks/Cole.

What *was* I planning to do?

Mark Roberts[1] and **Isaac Monteath**[2] and **Raymond Sheh**[2] and **David W. Aha**[1]
Piyabutra Jampathom[1] and **Keith Akins**[1] and **Eric Sydow**[1]
Vikas Shivashankar[3] and **Claude Sammut**[4]
[1]The U.S. Naval Research Laboratory; Washington, DC, USA | {first.last}@nrl.navy.mil
[2]Curtin University; Bentley, WA, Australia | isaac.monteath@postgrad.curtin.edu.au, raymond.sheh@curtin.edu.au
[3]Amazon Robotics; North Reading, MA, USA | vikshiv@amazon.com
[4]The University of New South Wales; Sydney, NSW, Australia | claude@cse.unsw.edu.au

Abstract

Adjusting commitments to ongoing plans can occur frequently when executing these plans in a dynamic environment. Often, an agent will repair its plan or replan vis-a-vis such change, which is a type of planning-specific adjustment. However, adjusting commitments to the goals by regoaling, transforming goals, or deferring goals may also be needed. As discussed in the literature on plan explanation, an agent may be asked to account for plan adjustments. In this position paper, we advocate for considering the full suite of possible adjustments in explanation. This includes plan repair, replanning, deferring, regoaling, and abandoning goals. Using an example from the RoboCup Rescue Agent Simulator (`Roborescue`), we leverage a goal lifecycle, extended with time-based Chronicles and transitions goals, for explanation. We propose an explanation taxonomy that spans three dimensions and illustrate the use of this taxonomy for many possible explanations in `Roborescue`. Finally, we highlight several possible user interfaces we intend to build.

1 Introduction

Autonomous agents will be increasingly called upon to react to ever more challenging, dynamic environments. Execution rarely proceeds according to plan, resulting in the need of an agent to adjust its commitments. When called upon to explain these changes in commitments, the agent needs to account for the reasons behind the change.

Following the approach of Langley et al. (2017), we maintain that explaining plans requires more than only explaining the choices of the planner. In many applications, deliberation – commonly referred to as planning, problem solving, or reasoning – occurs at many levels of a situated autonomous system (Ghallab, Nau, and Traverso 2016). Pollack and Horty (1999) argue that there is more to planning than simply making plans, noting that plans are meant to be executed and a variety of interesting problems occur when plans fail. In order to explain their choices, agents must be able to recreate the context and history of the moment for each commitment.

Accordingly, we argue that a wider perspective, one that considers the role of the planning system integrated *within* a situated system, as necessary for effective explanation of plans and the planning systems. Much of the literature around plan revision focuses on plan repair or replanning, and much discussion about explaining plans and planner choices emphasizes repairing or replanning. A similar argument is echoed in the literature on explanation (Miller 2017; Abdul et al. 2018). However, we offer a distinct perspective and solution for solving the problem. In the broader sense, an agent may deliberate about its goals, performing what we call goal reasoning (e.g., (Hawes 2011; Vattam et al. 2013) (Ghallab, Nau, and Traverso 2016, Section 7.2.3), (Aha 2018)).

We describe a research plan that will integrate explanation in a goal reasoning context, expanding plan explanation to encompass the agent formulating new goals, considering alternative plans (before and during execution), repairing plans, replanning from scratch, deferring or abandoning goals, switching between multiple goals, reformulating revised goals, and learning when to apply each of these possible transitions. Our proposed approach complements this literature by embedding explanations within a goal lifecycle that makes specific commitments to hierarchical, temporal, and numeric planning and characterizing explanations within to a 3-axis taxonomy. The contributions of this paper include:

- An extension of the goal lifecycle by Roberts et al. (2015; 2016) to include Chronicles from recent developments in planning and acting (Ghallab, Nau, and Traverso 2016);
- an informal description of how the extended goal lifecycle facilitates explanation using *transition goals*;
- a 3-axis taxonomy for explanations that considers their source, scope, and depth;
- an exploration of using the taxonomy to categorize various explanations; and
- a proposed extension to a user interface, the Virtual Mission Operation Console, to support explanations.

We begin by describing our proposed application, `Roborescue`, for exploring explanation (Section 2), which we follow with the assumptions for our solution (Section 3). Based on these assumptions, we introduce the goal lifecycle (Section 4), extend it to incorporate the above assumptions (Section 4.1), and demonstrate how it provides a natural mechanism for supporting anticipatory, ad-hoc, and post-hoc explanations (Section 4.2). We introduce a 3-axis taxonomy of explanations and demonstrate how explanations can be generated (Sections 5 and 6). These assumptions and commitments are not unique and much research has advanced one or more of these areas, as is elaborated in our discussion of related work (Section 7).

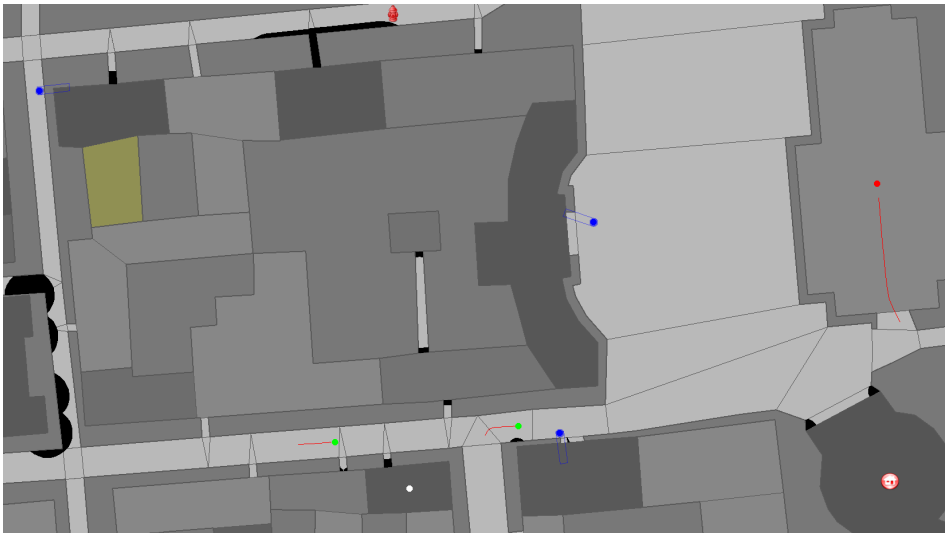


Figure 1: A single city block of the Agent Simulator running on the Berlin scenario.

2 `Roborescue`: A testbed for explanation

The RoboCup Rescue Agent Simulator, or `Roborescue`, models a situation immediately following a natural disaster (Sheh, Schwertfeger, and Visser 2016). For example, Figure 1 displays a `Roborescue` scenario using the 2017 simulator. It is the basis of the RoboCup Rescue Simulation League (Akin et al. 2012), an international competition for collaborative AI agents that has been running since 2000[1]. Inspired by the events of the 1995 Great Hanshin earthquake in Kobe, it aims to provide a venue where the performance of different algorithms for coordinating and controlling teams of simulated agents can be compared. The competition is held annually with an international championship fed by numerous regional opens. The Simulation League, which focuses on environments the size of a few city blocks, is just one of three competitions under the RoboCup Rescue umbrella. The Virtual Robot League is a competition based on a more detailed, high fidelity physics simulator within the confines of one city block. In the Robot League, physical robots compete within an arena of standard test methods for response robots.

In the Simulation League (cf. Figure 1), specific agents work together to help civilians (green dots) move to a safe area (the lower-right building). Civilians will move along roads (light gray) if they are able but may become trapped under or behind rubble (solid black). Fires (yellow building on the left) can cause further harm and produce rubble. If a civilian is trapped for too long or suffers too much damage then they turn black, indicating that they have died. Agents must communicate and cooperate to transport as many victims as possible to safety while also gathering information about the disaster area. Each agent may have its own objectives that can be interrupted by events such as requests to help other agents, discovering a victim, or coming across rubble in its path. The challenge is responding in a manner that best assists the entire team and communicating those responses.

[1] `http://wiki.robocup.org/Rescue_Simulation_League`

Controllable agents consist of police (blue dots), ambulance (white dots), and fire patrols (red dots). All three agents can `scout` an area, but only police can `clear` rubble, ambulances can `transport` a single civilian, and fire patrols can `douse` fires using water from hydrants (top). Not shown are centralized agents that serve as dispatchers each type. In this figure, two healthy civilians (bottom middle) are moving to the safe area. To their right, two police clearing rubble. Below them, an ambulance is stuck behind rubble.

2.1 Running Example

We illustrate the core explanation challenges resulting from execution failures and opportunities for an ambulance agent, although similar situations could occur for a police or fire agents. Failures can occur when a planned task is blocked from completing, a task takes longer than anticipated, a maintenance goal is violated, or a resource is blocked. Opportunities may appear when an agent discovers a previously infeasible goal is possible, through the arrival of new tasks, or while merging plans to complete multiple goals.

Figure 2 demonstrates the goal network of an ambulance agent for the following simplified scenario; we will formalize goal networks in Section 4. Suppose an ambulance agent, who can only scout or carry civilians, is commanded to scout a 'far away district' for potential civilians. After generating three alternative paths, A, B, and C, to the district, it starts down path A (solid boxes) but comes across rubble that must be cleared. It could here switch to an alternative route or ask the police to help clear the rubble. Suppose it selects path B (gray boxes) but cannot communicate this change to its dispatcher because communication is blocked by the buildings around it.

While traversing path B it identifies a civilian needing assistance. It could ignore the civilian and continue its current goal or transport the civilian to a shelter. Suppose that it chooses to defer its original goal (i.e., scout 'far away') to transport the civilian, generates a plan to the nearest shelter,

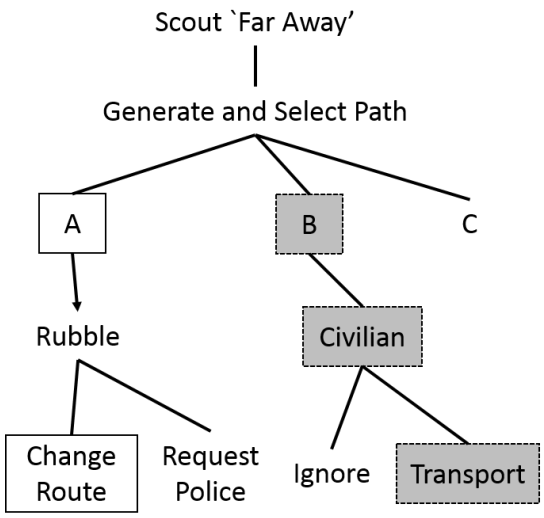


Figure 2: A goal network for the ambulance example.

Shelter 1, and picks up the civilian; again it cannot communicate this change because its radio is blocked. Upon reaching the Shelter 1, the agent discovers the shelter is full and must transport the civilian to Shelter 2, which it proceeds to do. Along the way, it identifies several more civilians needing assistance. Once the transport to Shelter 2 has completed, it could determine whether to return to the civilians it found, abandon path B, select the other alternative path C, or replan from the shelter.

2.2 Challenges in Explanation

Even for the above small example, the agent may be asked to explain its choices. Miller (2017), in surveying the relevant literature on explanation, suggests that explanations are contrastive, selected, and social. In line with this view, we imagine some possible kinds of explanations that may surface during or after the above example for the ambulance agent, which made many decisions, modified or abandoned some commitments, deferred others, and generated new goals in response to unexpected events. How does the agent explain its choice to change to a different path, switch goals by locating an agent who can clear the rubble, or attend to the civilian it just discovered? What if the agent is asked to explain which alternatives it considered when it found the first civilian? Suppose the agent did not know that a better path to 'far away' has become available and the dispatcher conveys that information (because the radios work at shelters); what if the agent sticks to its plan for path B and is later asked why it did not take the more direct route?

When appearing at the Shelter 1 with the civilian, suppose the agent is asked to report the damage exists in the 'far away', a type of *selected* explanation, which it will not know but could probably have anticipated. In this example, we imagined the agent tried to communicate changes to its plan but failed. Suppose the agent could communicate. From a *social* perspective, when should the agent proactively explain itself? Too many plan updates would likely be overwhelming but too few will result in reduced situational awareness for the entire team. Even if the agent waits for a request to explain itself, it must still anticipate when an explanation may be needed and facilitate such explanations by storing not only the plan trace but also the context that lead to changes in its commitments. Additional deliberation may be required to recreate the context of the decision if, for example, the agent is asked to consider other ways it could have solved the problem (e.g., during a post-hoc debriefing), which strengthens the important role of automated planning for explanation.

Learning can also play a role even for planning systems and plan explanation. Suppose the agent had applied machine learning to make some of its prior decisions, then it may also need to explain the learned model it used to prioritize its decisions (e.g., to select one goal over another) without being explicitly commanded to do so. For example, it may have learned that seriously wounded civilians in this situation do not live as long as usual compared to an expected model, leading it to choose to transport a critically injured civilian because it reasoned that no other agent could help in time.

Finally, consider an even more abstract view of a commander managing the entire disaster, who must communicate with dispatchers as well as agents. The commander may desire answers to *contrastive* questions such: *Why doesn't this task fit in the current solution?* or *What change is needed (or which constraint should I relax) to fit this task into the solution?* The commander may need to consider unit placement if new resources become available. For example, suppose the commander receives an offer of 5 ambulance teams from a neighboring city, a natural question question would be *Where should I place these units?* And lastly, the commander may want to anticipate possible future situations based on past data (where are fires most likely to start, for example), which is a variant of anticipatory planning.

3 Solution Assumptions

We make a number of assumptions in our proposed solution to help make solving the problem tractable. We assume agents interact with other agents and the environment in a distributed manner. Each agent generates their own plan(s) according the local information available – that is, we don't assume a globally optimal criteria. An agent may modify its goals in response to notable events that impact goal processing and can occur as a result of the environment or other agents.

The situation is oversubscribed as there will typically be more goals than resources to complete them, so an agent seeks to achieve or maintain high-quality, satisficing solutions. Goals can be hard goals (i.e., requirements), maintenance goals, and soft goals (i.e., preferences). Agents may consider a set of criteria when evaluating whether to pursue a goal. This means an agent may need to generate alternatives, consider tradeoffs, and commit to plans; such reasoning requires what is often called numeric planning or planning with metric quantities, which has been examined in the automated planning literature (e.g., (Fox and Long 2003)). However, the prevalent modeling approach of using action-costs prohibits more than one metric; this has led to benchmark problems with relatively "flat" objective spaces that often correlate with

plan length. (Radzi 2011). Although some planners generate alternatives according to multiple metrics, more work is needed to incorporate trade-off analysis between competing metrics. The agent may desire to maintain commitments it has already made to itself, to future projected events, or to other agents.

We commit to hierarchical structures to facilitate memory, planning, and retrieval. This commitment to a hierarchical approach provides for the use of domain-dependent knowledge to guide the search process in selecting and planning for goals; we leave it to future work to generalize the approach to domain-independent planning.

Agents maintain a history (or future, as appropriate) of: relevant world events, attempts to achieve goals, past experiences, other agents' actions, and commitments to other agents. This assumption requires temporal reasoning for the past, current, and future. The stored history may be compressed through learning, which we plan to address in future work. Explanations may be required before, during, and after completion of one or more goals. This requires an agent to perform introspection about its own history.

4 Goal Networks and the Goal Lifecycle

Our agents reason over goal networks, which are partially ordered sets of goals. For historical reasons, goal-networks are called Hierarchical Goal Networks (HGNs) (Shivashankar et al. 2013a; 2013b), although the decomposition of a goal network need not be strictly hierarchical. Alford et al. (2016) showed that HGN planning has the same expressiveness and complexity class, semi-decidable, as task-network planning, commonly referred to as hierarchical-task networks (HTNs). Further, their work proposed a hybrid formalism, goal-task networks (GTNs), which unified goal-network and task-network planning under sharing or insertion semantics.

Roberts et al. (2016) extended goal-task networks to model the goal lifecycle for a goal-reasoning agent. We restrict our attention to goal networks. Figure 3 (left) shows a subset of that goal lifecycle, which models the main decision points of goal reasoning (Hawes 2011; Vattam et al. 2013; Aha 2018). Goals transition between modes (rounded boxes) via strategies (arcs); colored strategies highlight the focus of this study. Figure 3 (right) shows a timeline of plan revision with the same colors. Goals are FORMULATED, SELECTED, and then EXPANDED (i.e., planned) for possible ways of achievement. Alternative plans are EXPANDED (magenta), and the system commits to a single plan (black line), which is then DISPATCHED for online execution. An objective might fail (hollow circle) due to events such as cloud cover, which may force the system to EVALUATE (wide gold line) revision of prior strategies. To move the failed task (hollow circle) to a new slot (solid circles), the system might: CONTINUE by ignoring the problem or falling back to a secondary task (red, dotted); REPAIR the plan by reusing a previous alternative (purple, dashed); REPLAN by expanding multiple alternatives (green, dot-dashed); or DEFER the objective (gray, long-dashed).

A goal transitions through the lifecycle as a *goal node* $\check{g} = (g_i, N, C, o, X, x, q)$ where: g_i is the goal to be achieved; $N = (s, gtn)$ is a gtn-node for g_i; C is the set of constraints on g_i and gtn; o is the current mode of g_i, defined below; X is the set of expansions that could achieve g_i, defined below; $x \in X$ is the committed expansion along with any applicable execution status; and q is a vector of quality metrics. Metrics could be domain-dependent (e.g., priority, cost, value, risk, reward) and are associated with achieving g_i. We used $\mathcal{N}$ in prior work but find $\check{g}$ more natural.

Figure 3 (left) displays the possible refinement strategies, where an actor's decisions consist of applying one or more refinements from R (the arcs) to transition $\check{g}$ between modes (rounded boxes). Strategies are denoted using small caps (e.g., FORMULATE) with the modes in monospace (e.g., FORMULATED). Execution may require adjustments, via the resolve strategy, to prior commitments. The degree of revision determines how far back in the goal lifecycle the goal must transition, and it is not always clear which resolution is best.

A *goal memory* $M = \{\check{g}_1, \check{g}_2, .., \check{g}_m\}$ for $m \geq 0$ holds the active goal nodes for an agent. Most refinements modify the goal memory by modifying a node within memory, in which case we write $M \rightarrow_R M'$ for a single strategy application of R resulting in M' and $M \rightarrow_R^* M''$ for a sequence of applied strategies resulting in M''.

4.1 Extending the Goal Lifecycle

Following the assumptions from Section 3, we make a specific commitment to temporal reasoning with timelines and discrete resources. We accomplish this with a hybrid planning approach called Constraint-Based Interval (CBI) Planning, which is commonly used in NASA (e.g., systems such as Europa or Aspen) and MBARI missions (e.g, (Smith, Frank, and Jónsson 2000; Rajan, Py, and Barreiro 2013)). Central representations in CBI planning include a variable, often called a token, and temporal constraints, called intervals, over the variable. A token is an object or resource in the system that needs to be tracked (e.g., the ambulance status). Tokens can be assigned attributes to indicate state (e.g., transporting, clearing or scouting). Constraints restrict the set of valid solutions by limiting resource usage (e.g,. keep health above 74 units), temporal extent (e.g., duration of 90 minutes), or ordering (e.g., scout before transport). To incorporate temporal constraints, we will extend the goal node with Chronicles as described by Ghallab et al. (2016). Decision points of the agent follow many critical points the goal lifecycle (cf. Figure 3). We next describe how these decision points will support explanation.

4.2 Facilitating Explanation with the Goal Lifecycle

The goal lifecycle with temporal constraints provides a natural mechanism for tracking changes during execution, such as those highlighted in the example from Section 2.1. We partition the goal memory into the past, present, and future, $M = M_{\text{past}} \cup M_{\text{present}} \cup M_{\text{future}}$, segmented by time: $t_0 < M_{\text{past}} < t_{\text{now}} \leq M_{\text{present}} < t_{\text{horizon}} \leq M_{\text{future}} < t_{\text{end}}$, so that t_0 and t_{end} indicate the start and end of time, t_{now} indicates now, t_{horizon} indicates the end of the furthest plan into the future. The system stores the transitions of all goals in the appropriate memories. Goals that interact will have appropriate references to their related (sub)goals.

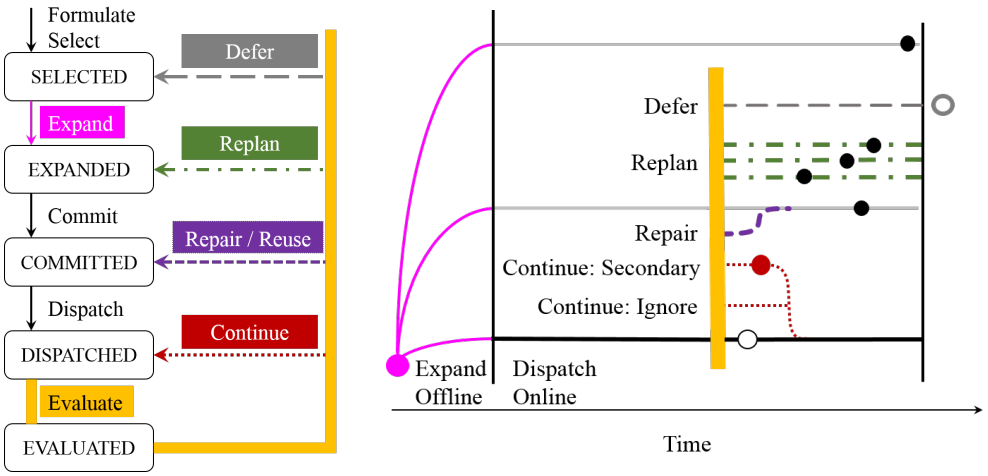


Figure 3: Left: The goal lifecycle of Roberts et al. (Roberts et al. 2016) Right: A plan revision timeline.

Explanations will often hinge on transitions in the goal lifecycle, since these capture changes in commitments. For example, the agent may be later asked *Why did you defer this goal?* or *Which alternatives did you consider at time point t?* The goal lifecycle supports explanation for such situations by creating *transition goals* to achieve each *important* transition, where importance is defined by the scope, depth, and source of expected explanations as defined later in Section 5. The agent will only store transitions that may require future explanations. It may help to see transition goals as "meta" goals, though we do not necessarily make a commitment to meta-reasoning specifically.

Transition Goal Example: Returning to the example of the ambulance agent in Figure 2. When the agent sees rubble along path A, it needs to deliberate about possible alternatives. This places `transit_path(A)` into `EVALUATED` to resolve. Since this decision is one that may require explanation, the agent creates a transition goal `resolved(rubble_blocking_path_A)`, which follows the same lifecycle as any other goal. During the expansion of this new goal, it generates the possibilities (i.e., expansions) shown under the rubble node in Figure 2; (it generates many more but only two are shown). Supposing it resolves the `transit` goal by changing the route, then `resolved` goal is now finished, the `transit_path(A)` goal is dropped, and a new `transit_path(B)` is formulated. In essence, the entire set of "goal" transitions has been captured from the moment the rubble was detected to the point where a new path was chosen. It is easy to see that transition goals can be created for similar situations.

We argue that the transition goal – the `resolved` goal in the above example – is vital for any reasonable explanation. When the system is asked to explain itself, it can simply refer to this goal to elucidate its entire deliberation. When used in this way, the goal lifecycle exposes key constraints that guide solutions and justify changes in commitments to goals, plans, or actions. M stores these transitions for future reference.

A limitation of transition goals is the increased memory needed to store additional information. This is especially true if all transitions are stored. We mentioned earlier that the agent should only store *important* transitions. We now turn our attention to an explanation taxonomy that will aid in determining those important transitions.

5 A Proposed Explanation Taxonomy

A key objective of this work is to develop agents that can perform a given task and explain their decision-making process to humans and other agents in the scenario. Providing effective explanations depends on a variety of factors, such as the requirements of the consumer and the agent's explanatory capabilities. We categorize explanations along three dimensions that identify its source, its depth and its scope. This categorization is inspired by Sheh (2017) and each dimension is currently segmented into a binary split:

- **Source**: *Post-Hoc Rationalization* vs. *Introspective*
- **Depth**: *Attribute (Entity/Use)* vs. *Model*
- **Scope**: *Justification* vs. *Teaching*

Source identifies the type of data used to generate the explanation. The source is *Introspective* if the provided information is based on a full trace of the underlying decision process. This means that the explanation must accurately reflect the actual decision process that was taken to arrive at a particular decision. The source is *Post-Hoc Rationalization* if it is based on observing the external behavior of the agent as a black box. These types of explanation do not necessarily reflect the actual decision process taken by the agent but rather seek to rationalize the result. An explanation's source can be somewhere in between if it has access to intermediate representations or processes but not the full "trace" of the decision-making process. Note that this is defined relative to a useful level of abstraction. After all, any program running on a deterministic computer can be traced, even if it's at the level of individual instructions, but this "execution" explanation is less useful in most applications.

Depth identifies the granularity of the explanation. The depth is called *Attribute Identity* if the explanation is based

on identifying the attributes, or features, that were used to make a decision. For example, saliency maps in object recognition tasks are considered Attribute Identity explanations. It is *Attribute Use* if it also includes information on how these attributes were used, such as decision boundaries and relationships. A Support Vector Machine might report the hyperplanes bordering a region containing the decision instance, for example. It is *Model* if the explanation extends to how the model, on which these decisions are based, was itself generated from the training data and other background information. A Decision Tree might report the training examples that flowed through to a given decision node to explain that node's decision boundary, for example.

Scope identifies the purpose of the explanation. The scope is *Justification* if the purpose is to justify a particular decision or set of decisions. It is *Teaching* if the explanation is about something beyond a decision. In general, teaching explanations aim to teach a user (or another agent) something that the explainer has learned, such as synthesizing new examples or answering hypothetical questions.

6 Generating Explanations

Using the taxonomy just described, we highlight some possible explanations an agent could provide. Explicitly stating the kinds of explanations we expect the system to provide elucidates which information will be required and the type of model that would need to be used to enable the explanation. Returning to the `Roborescue` scenario, consider a single police agent en-route to clear rubble in pile_A that receives a report of another nearby rubble pile_B, which is preventing a fire truck from accessing a burning building. The police agent revises its commitments by deferring its goal of `cleared`(pile_A) in order to achieve `cleared`(pile_B) first. This results in creating a plan with a detour to clear pile_B. We next provide several examples of complex queries and explanations using the taxonomy.

6.1 Justification scope

To assist in the explanation, we will begin with explanations that have Justification scope because the explainer is being asked to explain why it selected one plan over another or provide a reasonable rationale of its decisions.

Why did you decide to switch to a new goal? [Introspective, Attribute] This explanation considers why the agent decided to change its plan. e.g. Another agent is waiting for me to clear the rubble AND the time required for the detour is less than 15 minutes.

What do you think this other agent is likely to do? [Post-hoc, Attribute] The explanation has attribute depth because the agent does not have access to the internal model of the target agent. It has a post-hoc source because the agent lacks internal access to the decision-making of the target agent.

Asking the agent to explain its internal model/rationalizations for how other agents act in the world. For example, a user could ask whether the police agent thinks that the dispatcher is likely to send another police agent to assist. The agent might respond by stating that, in its experience, assistance is in areas that are far from the center of the city. Therefore, it might conclude that the dispatcher is unlikely to send assistance.

This type of explanation could be useful for debugging purposes, especially where the user has introspective access into these other agents. This can assist the user in finding discrepancies between how the agent believes other agents reason and how they actually reason. In the example above, a user may query the dispatcher for an introspective explanation on whether it is likely to dispatch another police agent to the area. If the explanations does not match that of the police agent, it may provide an important insight into the behavior of the police agent.

How confident are you that your plan will work? This explanation involves the agent computing a measure of how confident it is in the selected plan. For example, a human operator might be doubtful about the effectiveness of the police agent's selected plan and request that the agent to report its confidence. This will help the human operator decide whether to trust the agent or override it. The scope of this explanation is Justification because the confidence measure is used to justify the agent's decision. The source is Introspective in most cases as this is the only way to measure the agent's confidence. However, a measure of *reasonableness* of a given plan may be computed externally in a Post-Hoc fashion. The depth can be either Attribute or Model depending on how the confidence metric is calculated.

6.2 Teaching Scope

We now switch to explanations that follow from a teaching scope because the explainer is informing another user (or agent) about other available decisions rather than justifying a particular one. Some questions from a justification scope could be asked in a teaching scope because of a different motivation.

Why did you decide to switch goals? Agents may be able to use their own local knowledge to check for incorrect assumptions in other agents. For example, suppose a police agent informs the dispatcher of switching from clearing rubble for an ambulance to clearing rubble for a fire patrol. The dispatcher may not know the state of the fire patrol and the explanation needs to shift from justifying to teaching scope, where the agent may need to clarify its closeness to the fire.

Which other similar goals are possible? This involves the agent proposing new goals to the user that it believes are similar to its current goal. This may be particularly useful in cases where the agent cannot complete its current goal and believes it should switch to a new goal.

Extending the example, suppose that the police agent reports to the human operator that it has decided to switch goals to assist the fire truck. The human operator may know that there are many alternative routes for the fire truck to take. The operator knows that there are probably more productive goals for the police agent, but doesn't necessarily know what they are. In this case, the police agent can provide a list of goals that it believes to be similar for the human operator to decide from. These goals might include scouting a nearby area or clearing rubble for another agent.

Interventions: Explanations may relate to the use of inter-

ventions. Questions along these lines ask the agent to consider possible alternatives, some hypothetical.

How would the scenario need to change for you to retain your original goal? Given that an agent is proposing to change its goal, this query requires the agent to propose hypothetical scenarios where it would not change goals. For example, Figure 4 shows a police agent that decides to change plans from Plan 1 to Plan 2. It may explain that if the middle house was not on fire or if the distance was 1 kilometer further, it would have continued with Plan 1 instead.

How would the scenario need to change for you to change to this specific goal? This explanation type is the reverse of the previous explanation type. Given that an agent is proposing to retain its existing goal, a human operator may wish to know which scenarios would cause the agent to change to a specific goal. For example, suppose that the police agent decided to retain its Plan 1 goal in Figure 4. The human operator may ask what would need to change about the current situation for the police agent to consider switching to the goal of Plan 2. An example response might be that in order to switch to the goal of Plan 2, there would need to be more houses on fire in the area.

Counterfactuals: Other explanations relate to the use of counterfactuals. Questions along these lines ask the agent to consider alternatives in the face of negative possible worlds.

What do you predict the scenario will look like after a particular action is taken? [Introspective, Model] This explanation makes a comparison of what the model thinks would happen as a result of executing the proposed plan instead of the current plan (Model explanation) (e.g. show me the predicted outcome of executing both plans – the user really just wants to know which plan is likely to prevent the most damage in the least amount of time).

What would you do in this situation if I changed your priorities? This explanation involves the agent responding to a hypothetical scenario in which its own priorities have been altered. For example, a human operator might ask the police agent what action it would take if its highest priority was to minimize civilian casualties. This type of explanation has an Introspective source. The depth depends on the implementation and could be either Attribute or Model.

6.3 Visualizing the schedule changes

We plan to explore two kinds of visualization for explanations of competing objectives and the resulting plans. The first is exemplified by Figure 4. Agents consist of police (blue) or ambulance (red), and roads are blocked by rubble (black). Buildings can be normal (white), burning (yellow), or destroyed (black). The situation is relatively simple, but the information required to generate this explanation is complex.

The second visualization relies on an existing software product called the Virtual Mission Operations Center (VMOC), which has its origins in satellite scheduling. Figure 5 shows a mockup of an existing interface we will extend. VMOC is a space-qualified, U.S. government-owned, satellite mission management and scheduling framework. It is designed to be reusable and extensible, architected to easily incorporate new missions, optimization algorithms, and visualization applications. VMOC employs an HTML5-compliant user interface which provides users with mission management capabilities that include collection requirement (goals) management, payload ahead-of-time and real-time scheduling (planning), and stakeholder situational awareness. VMOC has been providing mission management services since the mid 2000s and currently supports several operational satellites.

VMOCs requirements management applications are graphical user interfaces for viewing, creating, and refining satellite mission tasking and requirements (goals). Users can choose to be guided through requirement generation or take an advanced view where they are able to create, read, update, and delete requirements.

VMOCs scheduling applications are centered around satellite schedule generation (planning), visualization, comparison, and real-time task (goal) execution status. During candidate schedule generation, the scheduling application returns both planned and unplanned activities, the latter including an explanation for why the activity was not scheduled. Real-time changes to schedules (plans) are pushed to the users browsers for situational awareness. Because it supports a long lifecycle for tasks, we plan to extend VMOC to support the kind of plan (and goal) revision described in Figure 3.

7 Related Work

Our proposed focus builds upon replanning and plan repair, which has a long history going back to the 1990s (e.g., (Hammond 1990; Hanks and Weld 1995; Koenig, Furcy, and Baue 2002; Horty and Pollack 2001)). Comparisons of these approaches exist (e.g., (Fox et al. 2006)), and work by Nebel & Koehler (1995) formally analyzed problems showing where reuse can be as difficult as replanning from scratch. In addition to plan repair and replanning, the goal lifecycle represents changes in commitments that defer, reformulate, or abandon goals. This fits well with the theme of planning as an iterative process advocated by Smith (2012), where the objectives shift along with exploration of the possible plans. We plan to more fully explore this distinction in future work.

Several areas of research have considered how to select between alternative objectives or adjusting commitments during execution. The most closely related to our approach is that of partial satisfaction planning by Benton et al. (2009). This approach selects goals in the face of oversubscription or preferences (soft goals) using the net utility, where the agent tries to maximize the utility of achieving goals less the action costs and penalties for unachieved goals. In contrast, our approach separates goal utilities from action costs through the use of distinct quality metrics for each.

Several studies have examined generating multiple alternative plans in the context of multiple criteria (e.g., (Do and Kambhampati 2003; Nguyen et al. 2012; Srivastava et al. 2007; Coman and Munoz-Avila 2011; Roberts, Howe, and Ray 2014). Plans are often generated according to a diversity measure that seeks to generate distinct alternatives. Our approach naturally accounts for generating multiple plans, what we call goal expansion, in the goal lifecycle. While there is work on creating flexible plans for dispatching (e.g., (Conrad

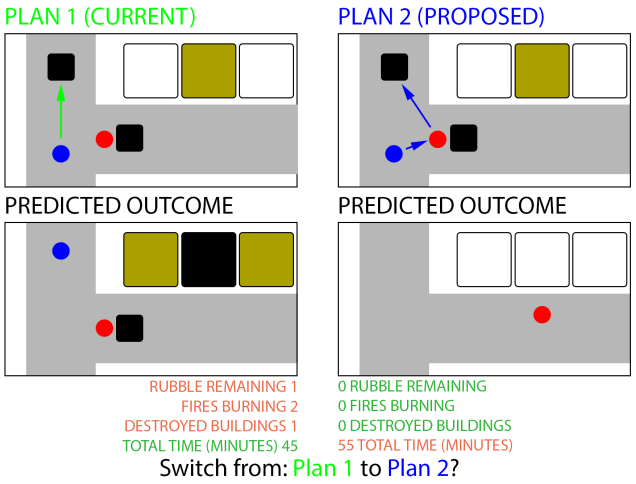


Figure 4: Possible user interface for explaining alternative plans according to multiple objectives.

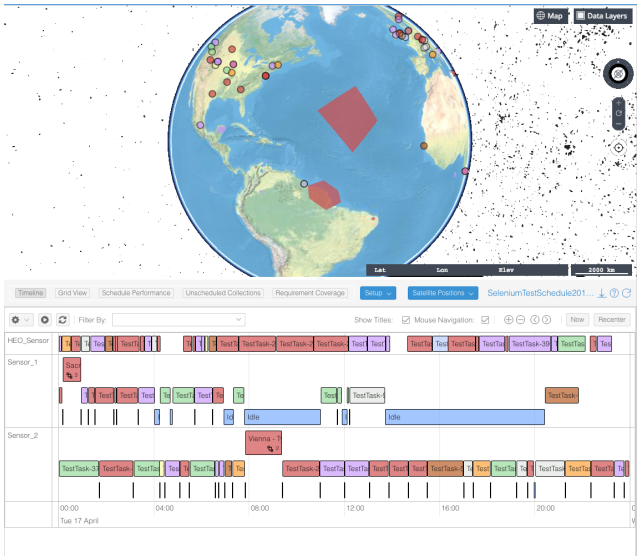


Figure 5: Existing VMOC interface we plan to extend for exploring alternative objectives.

and Williams 2011)), few studies have examined generating diverse alternatives relative to execution flexibility.

Explanations for plans have been studied in the context of generating preferred explanations (Sohrabi, Baier, and McIlraith 2011), explainable agency (Langley et al. 2017), generating trust with humans (Floyd and Aha 2017), and model reconciliation (Chakraborti et al. 2017). Much of this work is concerned with how information is selected and presented to the user. Seegebarth et al.(2012) present a formal model demonstrating how to use the plan structure from a hybrid HTN-POCL planning system to generate explanations; this approach bears the closest resemblance to our effort. Rosenthal, Selvaraj and Veloso (2016) categorize explanations provided by a mobile robot according to three parameters: abstraction (governing vocabulary), locality (governing the relevant portion of the plan) and specificity (governing detail). More recently, Fox et al. (2017) have characterized the kinds of explanations that may be needed in such systems along with initial results and a roadmap advocating for a sustained effort in Explainable Planning.

Work from a cognitive science perspective, such as that of Keil (2003), relate to focusing explanation on what is appropriate for a given audience and application. In contrast, Tolchinsky et al. (2012) focus on the use of explanation to make dialog more convincing. While these categorizations are useful to determine what a user might need, they are less useful at figuring out what techniques, or ”hooks”, need to be present in the underlying AI in order to furnish this information.

Some work has been done along the lines of a ”requirements and capabilities analysis.” Bibal (2016) share this view that existing semantics do not suit this purpose for many applications although they do not propose a unified set of categories that do so. One important distinction is made by Montavon et. al. (2018) where they make the distinction between an explanation (what we call an Attribute Identity explanation) and an interpretation (which aims to make the final output of the AI system more understandable). While this is useful, we feel it may be too narrow. Sometimes an explanation needs to contain detail about the identity of attributes or features, in the interpretable domain, and sometimes an explanation changes based on how it is used (what we call Attribute Use explanations) and why it is used in that manner (what we call Model explanations). Perhaps closest to our work, at least in terms of purpose, is that of Doran et. al. (2017) with four categories: Opaque (no insight), Interpretable (mathematically analyzable), Comprehensible (producing user-interpretable symbols) and Explainable (automated reasoning to produce explanations).

8 Summary

We have advocated a broader perspective when considering explainable planning that includes the spectrum of commitment adjustments that can take place while executing a plan. These adjustments include not just plan repair and replanning but also goal deferment, regoaling, and goal abandonment. We demonstrated that this larger perspective can be captured as part of the goal lifecycle from prior work and showed how the lifecycle can facilitate explanation. We then categorized the kinds of explanation that could take place according to a taxonomy of three dimensions: source, depth, and scope. Using this taxonomy, we examined many kinds of explanations. Finally, we showed two possible visualizations we plan to explore as our project continues.

References

Abdul, A.; Vermeulen, J.; Wang, D.; Lim, B.; and Kankanhalli, M. 2018. Trends and trajectories for explainable, accountable, and intelligible systems: An HCI research agenda. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

Aha, D. W. 2018. Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine* to appear.

Akin, H. L.; Ito, N.; Jacoff, A.; Kleiner, A.; Pellenz, J.; and Visser, A. 2012. Robocup rescue robot and simulation leagues. *AI magazine* 34(1):78.

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. 2016. Hierarchical planning: Relating task and goal decomposition with task sharing. In *Proc. IJCAI*, 3022–3028.

Benton, J.; Do, M.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *AIJ* 173(5-6):562–592.

Bibal, A. 2016. Interpretability of Machine Learning Models and Representations: an Introduction. In *Proc. ESANN*, 77–82.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation. In *Proc. IJCAI*, 156–163.

Coman, A., and Munoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proc. AAAI*, 946–951.

Conrad, P. R., and Williams, B. C. 2011. Drake: An Ecient Executive for Temporal Plans with Choice. *JAIR* 42:607–659.

Do, M. B., and Kambhampati, S. 2003. SAPA: A multi-objective metric temporal planner. *JAIR* 20:155–194.

Doran, D.; Schulz, S.; and Besold, T. R. 2017. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. *arXiv preprint arXiv:1710.00794*.

Floyd, M. W., and Aha, D. W. 2017. Using explanations to provide transparency during trust-guided behavior adaptation. *AI Communications* 30(3-4):281–294.

Fox, M., and Long, D. 2003. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *JAIR* 20:61–124.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proc. ICAPS*, 212–221. AAAI Press.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. In *Working notes of the IJCAI workshop on Explainable AI*.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Hammond, K. J. 1990. Explaining and repairing plans that fail. *Artificial Intelligence* 45(1-2):173–228.

Hanks, S., and Weld, D. S. 1995. A domain independent algorithm for plan adaptation. *JAIR* 2:319–360.

Hawes, N. 2011. A survey of motivation frameworks for intelligent systems. *AIJ* 175(5-6):1020–1036.

Horty, J. F., and Pollack, M. E. 2001. Evaluating new options in the context of existing plans. *AIJ* 127(2):199–220.

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

Koenig, S.; Furcy, D.; and Baue, C. 2002. Heuristic search-based replanning. In *ICAPS*.

Langley, P.; Meadows, B.; Sridharan, M.; and Choi, D. 2017. Explainable Agency for Intelligent Autonomous Systems. In *Proceedings of the Twenty-Ninth Conference on IAAI*, 4762–4764.

Miller, T. 2017. Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*.

Montavon, G.; Samek, W.; and Müller, K. R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing: A Review Journal* 73:1–15.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: a theoretical and empirical analysis. *AIJ* 76(1-2):427–454.

Nguyen, T.; Do, M.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *AIJ* 190:1–31.

Pollack, M., and Horty, J. 1999. There's more to life than making plans: Plan management in dynamic, multiagent environments. *AI Magazine* 20(4):71–83.

Radzi, M. 2011. *Multi-objective planning using linear programming*. Ph.D. Dissertation, Univ. of Strathclyde.

Rajan, K.; Py, F.; and Barreiro, J. 2013. *Towards Deliberative Control in Marine Robotics*. New York, NY: Springer New York. 91–175.

Roberts, M.; Apker, T.; Johnson, B.; Auslander, B.; Wellman, B.; and Aha, D. 2015. Coordinating robot teams for disaster relief. In *Proc. FLAIRS*. Hollywood, FL: AAAI Press.

Roberts, M.; Shivashankar, V.; Alford, R.; Leece, M.; Gupta, S.; and Aha, D. 2016. Goal reasoning, planning, and acting with actorsim, the actor simulator. In *Poster Proceedings of the Fourth Annual Conf. on Advances in Cognitive Systems*.

Roberts, M.; Howe, A.; and Ray, I. 2014. Evaluating diversity in classical planning. In *Proc. ICAPS*. Portsmouth, NH, USA: AAAI Press.

Rosenthal, S.; Selvaraj, S. P.; and Veloso, M. 2016. Verbalization: Narration of Autonomous Robot Experience. In *Proc. IJCAI*, 7.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users – a formal approach for generating sound explanation. In *Proc. ICAPS*, 225–233.

Sheh, R.; Schwertfeger, S.; and Visser, A. 2016. 16 years of robocup rescue. *KI - Künstliche Intelligenz* 30(3):267–277.

Sheh, R. K. 2017. Different XAI For Different HRI. *AAAI Fall Symposium Series*.

Shivashankar, V.; Alford, R.; Kuter, U.; and Nau, D. S. 2013a. The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning. In *IJCAI*, 2380–2386.

Shivashankar, V.; UMD EDU, R. A.; Kuter, U.; and Nau, D. 2013b. Hierarchical goal networks and goal-driven autonomy: Going where ai planning meets goal reasoning. In *Goal Reasoning: Papers from the ACS Workshop*.

Smith, D. E.; Frank, J.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review* 15(1):47–83.

Smith, D. E. 2012. Planning as an iterative process. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, 2180–2185. AAAI Press.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred Explanations: Theory and Generation via Planning. In *Proc. AAAI*, 261–267.

Srivastava, B.; Kambhampati, S.; Nguyen, T.; Do, M.; Gerevini, A.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *IJCAI*, 2016–2022.

Tolchinsky, P.; Modgil, S.; Atkinson, K.; McBurney, P.; and Cortés, U. 2012. Deliberation dialogues for reasoning about safety critical actions. *AAMAS* 25(2):209–259.

Vattam, S.; Klenk, M.; Molineaux, M.; and Aha, D. 2013. Breadth of approaches to goal reasoning: A research survey. In Aha, D.; Cox, M.; and Muoz-Avila, H., eds., *Goal Reasoning: Papers from the ACS Workshop (Technical Report CS-TR-5029). College Park, MD: University of Maryland, Department of Computer Science*, 222–231.

Plan Explanation Through Search in an Abstract Model Space*

Sarath Sreedharan and **Midhun Pookkottil Madhusoodanan**
and **Siddharth Srivastava** and **Subbarao Kambhampati**
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ 85281 USA
{ ssreedh3, mpookkot, siddharths, rao } @ asu.edu

Abstract

There is a growing interest within the AI research community in developing autonomous systems capable of explaining their behavior to users. However, the problem of computing explanations for users of different levels of expertise has received little research attention. We propose an approach for addressing this problem by representing the user's understanding of the task as an abstraction of the domain model that the planner uses. We present algorithms for generating minimal explanations in cases where this abstract human model is not known. We reduce the problem of generating an explanation to a search over the space of abstract models and show that while the complete problem is NP-hard, a greedy algorithm can provide good approximations of the optimal solution. We also empirically show that our approach can efficiently compute explanations for a variety of problems.

1 Introduction

AI systems have the potential to transform society by assisting humans in diverse situations ranging from extraplanetary exploration to assisted living. In order to achieve this potential, however, humans working with such systems need to be able to understand them just as they would understand human team members. This presents a number of challenges because most humans do not understand AI algorithms and their behavior at the same intuitive level that they understand other humans. Recently, there have been attempts to bridge this gap by developing systems capable of explaining their behavior. Most recently (Chakraborti et al. 2017) formulated the problem of explaining plans as that of model reconciliation. Their approach relied on identifying ways of bringing the human model (i.e the explainee model) closer to the robot model so that the plan in question appears optimal in the new model. Their work looked at scenarios in which the human used a model of the domain that was at the same level of fidelity as the one used by the agent to generate the plan. This approach, unfortunately, did not capture scenarios where the human possessed a lower level of expertise and thus used a more "abstract" or coarser representation of the model as compared to the AI agent.

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

For instance, consider a mission-control operator who needs to supervise the activity of an autonomous robot on Mars in the midst of a sandstorm that could present valuable data for analysis. If the robot proposes to go back to the base before going to a vantage point for observing the storm, the operator would naturally be perplexed, and may be motivated to ask, why doesn't the robot go directly to the vantage point?! Similarly, a human team member at a manufacturing plant may be perplexed by a robot's unnecessary detours while assembling an automobile engine. Not only do such situations involve personnel with varying skill levels, they also place a premium on the size of explanations.

A natural interaction would have the robot present an explanation about why the human's counterfactual suggestion would not apply in the current situation. This explanation could involve facts about the environment as well as about the robot's constraints. E.g., "I need to get a new battery pack to observe the sandstorm for at least 30 minutes without interruption". Such explanations need to be attuned to the level of understanding of the human involved. If the operator happens to be the lead designer of the robot's sequential decision-making engine, the robot could provide more specific information, e.g. "I am carrying battery-pack #00920", because this operator knows that some battery packs wouldn't allow it to carry out the full observation.

In this paper we present the **Hierarchical Expertise-Level Modeling** or the **HELM** approach for facilitating such context and user-specific explanations. We assume that the human user's understanding of the task is an abstraction of the model used by the robot. HELM generates the appropriate explanation by searching through a *model lattice*

*This is an extended version of the paper "Hierarchical Expertise Level Modeling for User Specific Contrastive Explanations" that is to appear in IJCAI 2018. The current version includes a demonstration of the proposed approach (Section 5) that was not part of the IJCAI version.

of possible abstractions of the agent's model. Each model within this lattice represents a different level of understanding of the task, with the highest fidelity representation (corresponding to the most detailed understanding of the domain used by the robot) forming the base of the lattice and the model representing the most naive understanding of the task (for example one held by a lay user) forming the highest node. Since the user's level of expertise is unknown to the agent, our algorithm estimates the human model before searching for an explanation. While we assume a closed form model in this paper, we plan to extend our approach to arbitrary generative models or simulations in future work.

Our explanations consist of information that may be absent in the user's abstract model, and show why the foil doesn't apply in the true situation. These explanations will cause the user's model to shift to a more accurate model in the lattice (and ultimately achieve model reconciliation). We will refer to model updates constituting these explanations as *model concretizations*. Our framework can also be extended to situations where a user's understanding is abstract and erroneous. In this paper, we focus on the fundamental aspects of the problem and restrict our attention to settings where the user's understanding is an abstraction of the actual situation.

Readers familiar with counter-example guided model checking (CEGAR) literature (Clarke et al. 2000) or its applications in planning (Seipp and Helmert 2013) will notice that our method of refining models is quite reminiscent of model refinement methods discussed in that literature. The foils we consider in our approach are equivalent to the counter-examples used by CEGAR methods and similar to these methods we too are looking for concretizations that refute these counter-examples. We provide a more detailed comparison between the methods in Section 6.

The rest of this paper is structured as follows. In Section 2, we present our formal framework. Section 3 covers different approaches for generating explanation and Section 4 presents empirical evaluation of these methods on standard IPC domains. Through Section 5, we will discuss the application of this method on a simple task and motion planning problem. In sections 6 and 7, we will discuss the related work and possible future directions.

2 Hierarchical Expertise-Level Models

In this work, we focus on abstractions that form models by projecting out state fluents. While the presentation in the following sections is equally valid for both predicate and propositional abstractions, we will focus on propositional abstractions to keep our formulation clear and concise. We will look at planning models of the form $\mathcal{M} = \langle P, S, A, I, G \rangle$ where P gives the set of state fluents, S the set of possible states, A the set of actions, I the initial state and G the goal. Each state $s \in S$ is uniquely represented by the set of propositions that are true in that state, i.e, $s \subseteq P$.

Each action $a \in A$ is associated with a set of preconditions prec_a that need to hold for the effects (e_a) of that action to be applied to a particular state. Each effect set e_a can be further separated into a set of add effects e_a^+ and a set of delete effects e_a^-. The result of executing an action a on a state s in this setting is defined as follows

$$a(S) = \begin{cases} (S \cup e_a^+) \setminus e_a^-, & \text{if } \text{prec}_a \subseteq S \\ S & \text{otherwise} \end{cases}$$

A plan π is defined as a sequence of actions ($\langle a_1, .., a_n \rangle$, n being the size of the plan), and a plan is said to solve $\mathcal{M}$ (i.e, $\pi(I) \models_{\mathcal{M}} G$) if $\pi(I) \supseteq G$.

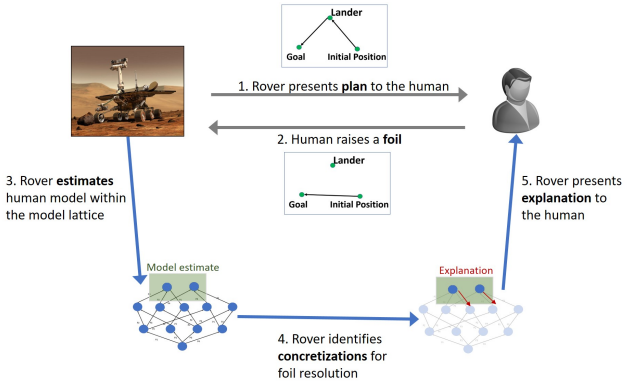


Figure 1: An illustration of the hierarchical explanation process. The human observer who views the task at a higher level of abstraction expects the rover to execute a different plan from the one chosen by the rover. The rover presents the human with an explanation it believes will help resolve the foils in the human's updated model.

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

Definition 1. *A set X is said to be a **propositional abstraction** of a set of states S with respect to some set of propositions Λ, if there exists a surjective mapping $f_\Lambda : S \to X$, such that for every state $s \in S$, there exists a state $f_\Lambda(s) \in X$ where $f_\Lambda(s) = s \setminus \Lambda$.*

For notational convenience we will refer to the set of states obtained by abstracting out the proposition set Λ from some set of states S as $[S]_{f_\Lambda}$.

Definition 2. *For a planning model $\mathcal{M} = \langle P, S, A, I, G \rangle$ with a corresponding transition system $\mathcal{T}$, a model*

*$\mathcal{M}' = \langle P', S', A, I', G' \rangle$ with a transition system $\mathcal{T}'$ is considered an **abstraction of $\mathcal{M}$**, if there exists a set of propositions Λ, such that $P' = P - \Lambda$, $S' = [S]_{f_\Lambda}$, $I' = f_\Lambda(I)$, $G' = f_\Lambda(G)$ and for every transition $s_1 \xrightarrow{a} s_2$ in $\mathcal{T}$ corresponding to an action a, there exists an equivalent transition $(s_1 \setminus \Lambda) \xrightarrow{[a]_{f_\Lambda}} (s_2 \setminus \Lambda)$ in $\mathcal{T}'$, where $[a]_{f_\Lambda}$ is part of the new action set A'.*

As per Definition 2, the abstract model is *complete* in the sense that all plans that were valid in the original model will have an equivalent plan in this new model. We will use the operator $\sqsubset$ to capture the fact that a model $\mathcal{M}$ is less abstract than the model $\mathcal{M}'$, i.e if $\mathcal{M} \sqsubset \mathcal{M}'$ then there exist a set of propositions Λ such that $\mathcal{M} = [\mathcal{M}']_{f_\Lambda}$. With the definition of abstraction and related notations in place, we are now ready to define a model lattice. We will use this lattice to both estimate the human model and to identify explanations.

Definition 3. *For a model $\mathcal{M}^{\#}$, the **model lattice $\mathcal{L}$** is a tuple of the form $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$, where $\mathbb{M}$ is the set of lattice nodes such that $\mathcal{M}^{\#} \in \mathbb{M}$ and $\forall\, \mathcal{M}' \in \mathbb{M}, \mathcal{M}^{\#} \sqsubseteq \mathcal{M}'$, $\mathbb{E}$ is the lattice edges, $\mathbb{P}$ is the superset of propositions considered for abstraction within this lattice and ℓ is a function mapping edges to labels. Additionally, for each edge $e_i = (\mathcal{M}_i, \mathcal{M}_j)$ there exists a proposition $p \in \mathbb{P}$ such that $[\mathcal{M}_i]_{f_p} = \mathcal{M}_j$ and $\ell(\mathcal{M}_i, \mathcal{M}_j) = p$.*

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

Throughout the rest of this work, we will make some assumptions on the structure of the lattice $\mathcal{L}$ and the abstraction methods used by $\mathcal{L}$ to simplify our discussions. In this paper, we will focus on lattices where each node in $\mathbb{M}$ has an incoming edge for every proposition missing from its corresponding model. We will refer to lattices that satisfy this property as **Proposition Conserving** lattices. Additionally, we will call a proposition conserving lattice that contains an abstract node corresponding to each possible subset of $\mathbb{P}$ as the **Complete Lattice** for $\mathcal{M}$ given $\mathbb{P}$.

Formally, a lattice $\mathcal{L}$ is *proposition conserving*, iff for any model $\mathcal{M} \in \mathbb{M}$ and $\forall p \in \mathbb{P}$, if p is not in $P_{\mathcal{M}}$ then there exists a model $\mathcal{M}' \in \mathbb{M}$, such that $(\mathcal{M}', \mathcal{M}) \in \mathbb{E}$ and $\ell(\mathcal{M}', \mathcal{M}) = p$). Notice that enforcing conservation of propositions doesn't require any further assumptions about the human model and can be easily ensured by the agent generating the lattice.

We also assume that all abstraction functions used in generating the models in the lattice are commutative and idempotent, i.e., $[[\mathcal{M}]_{f_{p_1}}]_{f_{p_2}} = [[\mathcal{M}]_{f_{p_2}}]_{f_{p_1}}$ and $[[\mathcal{M}]_{f_{p_1}}]_{f_{p_1}} = [\mathcal{M}]_{f_{p_1}}$. Readers can refer to (Srivastava, Russell, and Pinto 2016) for a comprehensive list of ways to generate abstract models that satisfy these properties.

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

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

The rover knows that the human observer is at most ignorant of its energy requirements and/or storage capabilities. So the model lattice $\mathcal{L}$ needs to consider abstractions corresponding to the following propositions $\mathbb{P}$={battery_level_above_25_perc, battery_level_above_50_perc, battery_level_above_75_perc, full_store}. Figure 2 shows the lattice that the robot would use in this setting. Here we will create each abstract model by dropping a proposition from the more concrete model and by making the effects of action non-deterministic if the dropped predicate appears in the precondition. For example, if the action drop_store1 has effects of the form

$$\{\text{full\_store1},\ \text{store\_of\_store1}\} \rightarrow \{\neg\text{full\_store1},\ \text{empty\_store1}\}$$

Now in an abstract version of this model, if the proposition full_store1 is dropped the effect becomes

$$\{\ \text{store\_of\_store1}\} \rightarrow\ ND\{\ \text{empty\_store1}\}$$

Which now says that the action's effects are non-deterministic and executing drop_store1 may or may not turn the fluent empty_store1 true.

Here the robot presents the plan

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

and the observer responds by proposing the foil set

$$F = \{\langle\ \text{navigate\_w0\_w1, navigate\_lander\_w1, sample\_rock\_store0\_w1}\ \rangle\}$$

If the robot knew that the human was ignorant about all the battery level predicates and nothing else, the robot could help resolve the human confusion by informing them about the fact that action sample rock requires the battery to be above 75% (i.e describing the proposition battery_level_above_75_perc) and in this updated model the

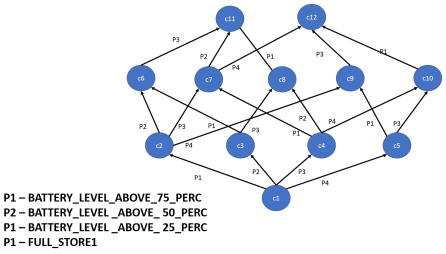


Figure 2: A possible abstraction lattice for the rover domain.

human foil can no longer achieve the goal. We can represent such an explanation using the set of propositions whose concretization is required to refute the given foils.

Definition 4. *An **explanation E** of size n for the human model $\mathcal{M}_H$ and a foil set F can be represented as a set of propositions of the form $E = \{p_1, ..., p_n\}$ such that* $\forall \pi \in F, \pi(I_{\gamma_E(\mathcal{M}_H)}) \not\models_{\gamma_E(\mathcal{M}_H)} G_{\gamma_E(\mathcal{M}_H)}$
Where $\gamma_E(\mathcal{M}_H)$ is the model obtained by applying the concretizations corresponding to E on the model $\mathcal{M}_H$.

In Example 1, the rover would have difficulty coming up with a single explanation as it does not know $\mathcal{M}_H$. One possibility would be to restrict its attention to just the models that are consistent with the foils . In this scenario, this would correspond to $\{c6, c7, c9, c10, c11, c12\}$.

Now we need to find a way of generating explanations given this reduced set of models.

Proposition 1. Let $\mathcal{M}_i$ be some model in $\mathcal{L}$ such that $\mathcal{M}_H \sqsubseteq \mathcal{M}_i$. If E is a valid explanation for $\mathcal{M}_i$ and some foil set F, then E must also explain F for $\mathcal{M}_H$.

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

Definition 5. *Given an the abstraction lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ the* **minimal abstraction set** $\mathbb{M}_{min}$ *is the supremum of all the models that are consistent with the foil set F.* $\mathbb{M}_{min} = \sup\{\mathcal{M}_i | \mathcal{M}_i \in \mathbb{M}, \forall \pi \in F(\pi(I_{\mathcal{M}_i}) \models_{\mathcal{M}_i} G_{\mathcal{M}_i})\}$

In Example 1, the minimal abstract model set will be $\mathbb{M}_{min} = \{c11, c12\}$.

If we can find an explanation that is valid for all the models in $\mathbb{M}_{min}$ then by Proposition 1 it must work for $\mathbb{M}_H$ as well.

Proposition 2. For a given model lattice $\mathcal{L}$, the minimal abstraction set $\mathbb{M}_{min}$ and a set of foils F, there exists an explanation E such that $\forall\, \mathcal{M}' \in \mathbb{M}_{min}$ and $\forall \pi \in F$,
$\pi(I_{\gamma_E(\mathcal{M}')}) \not\models_{\gamma_E(\mathcal{M}')} G_{\gamma_E(\mathcal{M}')}$

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

In most cases, we would prefer to compute the least costly or the shortest explanation (if all concretizations are equally expensive) to the explainee. In the rover example, even if the human is unaware of multiple task details, the robot can easily resolve the explainee's doubts by just explaining the concretizations related to the proposition battery_level_above_75_perc without getting into other details. Describing the details of remaining propositions is unnecessary and in the worst case might leave the human feeling overwhelmed and confused. In this case, the explanation would just include information regarding battery levels and how to identify when the battery level is or above 75% and model updates like
sample_rock-has-precondition-battery_level_above_75_perc
sample_soil-has-precondition-battery_level_above_75_perc
...
Before delving into the optimization version of the problem, let us look at the complexity of the corresponding decision problem

Theorem 1. *Given a minimal abstraction set $\mathbb{M}_{min}$, a plan π_R, the set of propositions being abstracted $\mathbb{P}$ and the set of foils F for a model $\mathcal{M}$, the problem of identifying whether an explanation of size k exists for the complete lattice is **NP-complete**.*

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

3 Generating Minimal Explanations

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

In addition to the actual size, the comprehensibility of the explanations may also depend on factors like human's mental load, the familiarity with the concepts captured by the propositions, etc.. To keep our discussions simple, we will restrict the cost of communicating an explanation to just the number of unique model updates this explanation would bring about in the human model.We will use the symbol C_p to represent the cost of communicating the changes related

to the proposition p and also overload it to work on sets of propositions.

Now our problem is to find the cheapest explanation (represented as E_{min}) for a given set of foils F, and the minimal abstract model set $\mathbb{M}_{min}$. One possibility is to perform an A* search (Hart, Nilsson, and Raphael 1968) over the space of possible propositional concretizations to identify E_{min}. Each search state consists of the minimal set of abstract models for the human model given the current explanation prefix. We will stop the search as soon as we find a state where the foils no longer hold for the current minimal set.

Proposition 3. Let $\mathbb{M}_{min}$ be the minimal abstraction set for a given lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ and foil set F. Then for a proposition p, the set $\widehat{\mathbb{M}}_{min}$ formed by applying the concretization corresponding to p on every element of $\mathbb{M}_{min}$ will be the minimal abstract set for $\widehat{\mathbb{M}}$ formed by applying the concretization γ_p on every element of $\mathbb{M}$ given F.

The above property implies that we don't need to look at the lattice $\mathcal{L}$ to recalculate minimal abstraction set after the application of every concretization function. We can also further simplify our problem by exploiting the fact that a particular propositional concretization resolves a foil (i.e., make the foil no longer valid) when it either adds a precondition (or a new condition for a conditional effect) or a goal fact that can not be satisfied by the foil. To concisely capture this idea we will introduce the concept of a foil resolution set to represent the subset of foils resolved by the concretization of a particular proposition.

Definition 6. *For a set of models $\mathbb{M}'$, a foil set F and a proposition p, the* ***resolution set*** *$\mathcal{R}_F(\mathbb{M}', p)$ gives the subset of foils that no longer holds in the concretized models, i.e $\mathcal{R}_F(\mathbb{M}', p) = \{\pi | \pi \in F \wedge (\forall \mathcal{M}' \in \mathbb{M}'(\pi(I_{\gamma_p(\mathcal{M}')}) \not\models_{\gamma_p(\mathcal{M}')} G_{\gamma_p(\mathcal{M}')} \wedge \pi(I_{\mathcal{M}'}) \models_{\mathcal{M}'} G_{\mathcal{M}'}))\}$.*

We will also use $\mathcal{R}_F$ to represent the set of foils resolved by a sequence of propositions

Proposition 4. For a set of model $\mathbb{M}'$ and a foil set F

$$\mathcal{R}_F(\mathbb{M}', \langle p_1, p_2 \rangle) = \mathcal{R}_F(\mathbb{M}', \langle p_1 \rangle) \cup \mathcal{R}_F(\mathbb{M}', \langle p_2 \rangle)$$

The above property implies that concretizing any n propositions cannot resolve foils that weren't resolved by the individual propositions. The idea of generating resolution sets are again closely related to the idea of resolving counterexamples and can be understood

Proposition 5. For two models $\mathcal{M}_1$, $\mathcal{M}_2$ and a set of foils F, if $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$ then for any proposition p, $\mathcal{R}_F(\{\mathcal{M}_1\}, p) \subseteq \mathcal{R}_F(\{\mathcal{M}_2\}, p)$

The above proposition ensures that if an explanation is the minimal one for $\mathbb{M}_{min}$, then it must be the minimal explanation for $\mathcal{M}_H$ as well.

These propositions will be instrumental in proving the effectiveness of our greedy algorithm described by Algorithm 1. In each iteration of this search, the algorithm greedily chooses the proposition that minimizes $\frac{C_p}{|F' \cap \mathcal{R}_F(\mathbb{M}', p)|}$, where F' is the set of unresolved foils at that iteration and the search ends when all foils are resolved.

Algorithm 1 Greedy Algorithm for Generating $\widehat{E}$

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

Theorem 2. *The explanation $\widehat{E}$ generated by Algorithm 1 for a set of foils F and a lattice $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{P}, \ell \rangle$ is less than or equal to $(\ln k) * C_{E_{min}}$, where $C_{E_{min}}$ is the cost of an optimal explanation and k represents the maximum number of foils that can be resolved by concretizing a single proposition, i.e, $k = \max_p |\mathcal{R}_F(\mathbb{M}_{min}, p)|$.*

Proof (Sketch). We will prove the above theorem by showing that Algorithm 1 corresponds to the greedy search algorithm for a weighted set cover problem. Consider a weighted set cover problem $\langle U, S, W \rangle$ such that the universe set $U = F$, the subcollections set S is defined as $S = \{s_p | p \in \mathbb{P}\}$ where $s_p = \mathcal{R}_F(\mathbb{M}_{min}, p)$ and the cost of each subset s_p is gives as $W(s_p) = C_p$. Proposition 4 ensures that the size of resolution set is a submodular and monotonic function. In this setting, the act of identifying a set of propositions that resolve the foil set is identical to coming up with a set cover for U in the new weighted set cover problem. Furthermore, we can show that the optimal set cover $\mathcal{C}_{opt}$ must correspond to the cheapest explanation E_{min} (We can prove this equivalence using Propositions 1,2 and 4, we are skipping the details of this proof due to space constraints). Algorithm 1 describes a greedy way of identifying the cheapest set cover for this weighted set cover problem and thus the minimal explanation for the original problem. For weighted set cover the above greedy algorithm is guaranteed to generate solutions that are at most $\ln k * W(\mathcal{C}_{opt})$ (Young 2008), where $k = \max_{s \in S} |s|$ and this approximation guarantee will hold for E_{min} as well. □

We can use this algorithm to either generate solutions and or to calculate an inadmissible heuristic for the previously mentioned A* search. For the heuristic generation, we will further simplify the calculations (specifically step 8 in Algorithm 1) by considering an over-approximation of $\mathcal{R}_F$. Instead of considering the set of all foils resolved by concretizing each proposition p, we will consider the set of foils where p appears in the precondition of one of the actions in it. This set should be a superset for $\mathcal{R}_F$ for any proposition.

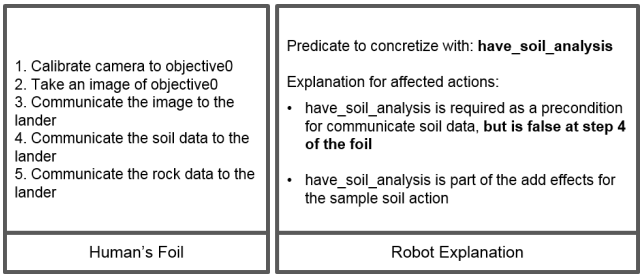


Figure 3: An example explanation generated by our system for rover domain. The human incorrectly believes that the rover can communicate sample information without explicitly collecting any samples. While the abstraction lattice was generated by projecting out upto 12 predicates, the search correctly identifies concretizations related to *(have_soil_analysis ?r - rover ?w - waypoint)* as the cheapest explanation

4 Empirical Evaluations

In our evaluation, we wanted to understand how effective our approaches were in terms of the conciseness of the explanations produced, the solution computation time and the usefulness of approximation. For the approximation, we were interested in identifying the trade-off between decrease in runtime vs. reduction in solution quality. All three explanation methods discussed in this paper (blind, heuristic and greedy) were evaluated on five IPC benchmark domains(International Planning Competition 2011). All the experiments detailed in this section were run on an Ubuntu workstation with 64G RAM.

For each domain, we selected 30 problems from either available test sets or by using standard problem generators (the problems sizes were selected to reflect the size of previous IPC test problems). The lattice for each problem-domain pair was generated by randomly selecting 50% of domain predicates and then generating a fully connected proposition conserving lattice using that set of predicates. Since none of the models contained any conditional effects, we created the abstract models by dropping the propositions to be abstracted from the domain models (which is still complete for these models). The foils were generated by selecting random models from the lattice and creating plans from these models that do not hold in the concrete model. Each search evaluated here, generates the set of proposition whose concretizations can resolve the foils set F. In actual applications, this set of propositions needs to be converted into an explanan (the actual message) by considering how this proposition is used in the robot model. Figure 3 shows the explanation generated by our approach for a problem in Rover domain.

The table in Figure 4 presents the results from our empirical evaluation on the IPC domains. The table shows the average cost/size of each explanation along with the time taken to generate them. Note that by size, we refer to the number of predicates that are part of the explanation while the cost reflects the total number of unique model updates induced by that explanation. We attempted explanation generation for foil set sizes of one, two and four per problem.

Our main conclusion is that heuristic search seems to outperform blind search in almost every problem and generates near-optimal solutions (Blind search always generates the minimal explanation). Further, we saw that greedy search outperformed heuristic search in most cases barring a few exceptions. The greedy search was able to make significant gains especially for higher foil set sizes. This is entirely expected due to the fact that step 8 in Algorithm 1 can be expensive for problems with long plans (but still polynomial). This expensive pre-computation pays off as we move to cases where E_{min} consists of multiple propositions. Additionally, we found out that greedy solutions were quite comparable to the optimal solutions with respect to their costs. For example in $|F| = 4$ for satellite domain, while the greedy solution cost took a penalty of $\sim 1.4\%$ the search time was reduced by $\sim 68\%$. The graph in 4 plots the comparison between the time saved by the greedy search versus any loss in optimality incurred by the greedy search.

5 Robot Demonstration

This section describes a demonstration of our approach on a physical robot for a simple grocery putaway task. Figure 6 presents the basic setup for the task. The goal of the robot here is to put away a bottle of tablets, a can of energy drink and a jar of sugar to proper storage locations. The storage location of each object is decided based on its type. For example, the robot should place the medicine bottle in the medicine cabinet, the sugar jar in the high pantry shelf, while the energy drink needs to be handed over to the human. In addition to these task-level constraints, the robot's operations are restricted by various motion level constraints that limit the possible physical movements that the robot can perform. For example, given the current position of the sugar jar on the table, the robot couldn't come up with any pickup pose that would allow the robot to place the sugar jar on the high shelf. In such cases, the robot could always enlist the help of the human to complete the plan.

In this setting, we will assume that the most concrete robot model consists of action descriptions that include both task-level symbols as well as continuous geometric arguments. Figure 7 presents the definitions for pickup and place_in_high_shelf actions in the most concrete model. In this model, the arguments of type ?pose and ?traj represents the pickup/putdown pose (the position and orientation of the end effector) and motion plans followed by the robot to perform the pickup/putdown.For this demo, we will consider a non proposition conserving model lattice that spans multiple levels of abstractions. Starting out we can convert each of the continuous arguments into geometric symbols (this is similar to the approach used in (Srivastava et al. 2014)). Next, we will further abstract the poses to align with possible regions on the object (i.e., pick up the object from the bottom, middle or top). We will also consider abstractions where we combine the predicates is_putdown_pose and collision_free_trajectory into a single predicate called reachable and also create new models by dropping arguments from the actions (we only drop an argument when none of the predicates use this argument). Figure 8 presents an intermediate model where most of the geometric predicates are already abstracted out. There could also be additional non-geometric

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

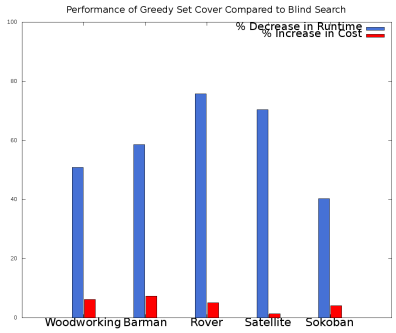


Figure 4: Table showing runtime/cost for explanations generated for standard IPC domains.Column $|\mathbb{P}|$ represents number of predicates that were used in generating the lattice, while $C_{\mathbb{P}}$ represents the cost of an explanation that tries to concretize all propositions in $\mathbb{P}$ and provides an upper bound on explanation cost. The graph on the right side compares the performance of greedy set cover against the optimal blind search for $|F| = 4$. It plots the average time saved by the set cover and the average increase in cost of the solution for each domain.

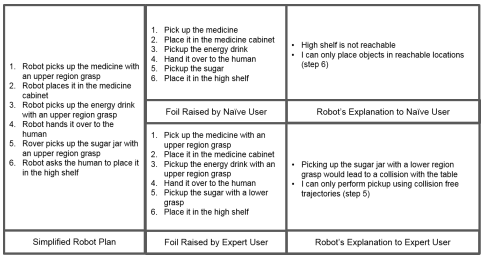


Figure 5: The plan and foils used in the scenario.

Figure 6: The grocery putaway domain setting.

predicates (like the is_condiment_type predicate) that can be abstracted out.

Figure 5 presents a plan and possible foils that could be generated in this domain. The plan involves the robot placing the energy drink and medicine on its own but relying on the human to complete the place action for sugar jar. The naive user asks merely why the robot doesn't finish the plan on its own while the expert user provides specific grasp that she/he believes can help the robot complete the plan. While the explanation to the naive user relies on the high-level predicate reachable, the explanation to the expert is more detailed and relates to the fact that a lower region grasp will result in a collision with the table. We can present such collisions to the human by simulating the trajectories using tools like Rviz (Hershberger, Gossow, and Faust 2016).

The readers can view the demonstration of the scenario implemented on a fetch robot at `https://youtu.be/qUHg8RABjsw`. OpenRAVE (Diankov 2010) was used to compute the trajectories of the robot arm for the pickup and place actions. COLLADA (Arnaud and Barnes 2006) models of the furniture and the items were created and populated in the OpenRAVE environment using AR markers, the transformation of which was obtained using the ar_track_alvar

```
(:action pickup
:parameters (?x - item ?y - storage ?u - pose ?v - traj)
:precondition (and
        (is_pickup_pose ?u ?x)
        (is_collision_free_traj ?x ?y ?u ?v)
        (in ?x ?y) (handempty)
)
:effect (and
        (not (handempty))
        (not (in ?x ?y))
        (holding ?x)
        (increase (total-cost) 1)
)
)

(:action place_in_high_shelf
:parameters (?x - item ?y - storage ?u - pose ?v - traj)
:precondition (and
        (is_putdown_pose ?u ?x)
        (is_collision_free_traj ?x ?y ?u ?v)
        (is_condiment_type ?x)
        (holding ?x)
        (is_high_shelf ?y)
)
:effect (and
        (handempty)
        (in ?x ?y)
        (not (holding ?x))
        (item_putaway ?x)
        (increase (total-cost) 1)
)
)
```

Figure 7: The action definitions for pickup and place_in_high_shelf from the most concrete model.

```
(:action pickup
:parameters (?x - item ?y - storage)
:precondition (and
        (in ?x ?y) (handempty)
        (reachable ?y)
)
:effect (and
        (not (handempty))
        (not (in ?x ?y))
        (holding ?x)
        (increase (total-cost) 1)
)
)

(:action place_in_high_shelf
:parameters (?x - item ?y - storage)
:precondition (and
        (is_condiment_type ?x)
        (holding ?x) (reachable ?y)
        (is_high_shelf ?y)
)
:effect (and
        (handempty)
        (in ?x ?y)
        (not (holding ?x))
        (item_putaway ?x)
        (increase (total-cost) 1)
)
)
```

Figure 8: The action definitions for pickup and place_in_high_shelf from an abstract model.

package available in ROS. Resulting OpenRAVE trajectories were then converted into ROS JointTrajectory messages and executed on the robot.

6 Related Work

There is increasing interest within the automated planning community to solve the problem of generating explanations for plans ((Fox, Long, and Magazzeni 2017; Langley et al. 2017)). Earlier works like (Seegebarth et al. 2012; Bercher et al. 2014; Kambhampati 1990) looked at explanations as a way of describing the effects of plans, while works like (Sohrabi, Baier, and McIlraith 2011; Meadows, Langley, and Emery 2013) looked at plans itself as explanations for a set of observations. Another approach that has received a lot of interest recently is to view explanations as a way of achieving model reconciliation (Chakraborti et al. 2017). Such explanations are seen as a solution to a *model reconciliation problem* (referred to as MRP) and this approach postulates that the goal of an explanation is to update the observer model so they can correctly evaluate the plans in question.

we can also view our explanations as model updates that focuses on a specific type of update, namely model concretization. Unlike MRP we do not make any assumptions about the availability of human model or the human's computational capabilities. The assumption that we have access to foils help us scale to much larger problems as compared to the original MRP approach to generate contrastive explanations. Following the conventions of the original MRP paper, we can see that the explanations studied here are both complete and monotonic.

As noted, our work is closely related to the well studied method of counter-example guided refinement or CEGAR that was originally developed for Model checking. Many planning works have successfully used CEGAR based methods to generate heuristics for plan generation ((Seipp and Helmert 2013; 2014)). The idea of foil resolution set for a given concretization is also closely related to the process of identifying spurious counter examples employed by CEGAR based methods (c.f (Haslum et al. 2012; Keyder et al. 2012; Steinmetz and Hoffmann 2016)). One major difference between our work and standard CEGAR based methods is the fact that in our setting the abstract model producing the foil (or counter-example) is unknown. Since we are exclusively dealing with spurious counter-examples we are also not bound to testing our foils (in other words identifying faults or pivot states) in the most concrete model (which could be quite expensive). Further, traditional CEGAR methods are generally not as focused on identifying the cheapest refinements.

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

7 Conclusion and Discussion

In this paper, we investigated the problem of generating explanations when the explainee understands the task model at a lower levels of abstraction. We looked at how we can use explanations as concretization for such scenarios and proposed algorithms for generating minimal explanations. One unique aspect of our approach is the use of foils as a way of capturing human confusion about the problem. This not only helps us formulate more efficient explanation generation methods but also aligns with the widely held belief that human expect contrastive explanations (c.f. (Lombrozo 2012; 2006)). Moreover, in most real-world scenarios humans usually include the foil in the request for explanations unless the foil is quite apparent from the context. Future directions include extending the methods to handle models that are incorrect in addition to being imprecise and looking at other possible methods for abstraction. We also plan to perform human factors studies on this explanation paradigm to evaluate its effectiveness.

Acknowledgments

We thank Dan Weld for helpful comments on a previous draft. This research is supported in part by the AFOSR grant FA9550-18-1-0067, ONR grants N00014161-2892, N00014-13-1-0176, N00014- 13-1-0519, N00014-15-1-2027, and the NASA grant NNX17AD06G.

References

Arnaud, R., and Barnes, M. C. 2006. *COLLADA: sailing the gulf of 3D digital content creation*. CRC Press.

Backstrom, C., and Jonsson, P. 2013. Bridging the gap between refinement and heuristics in abstraction. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Bercher, P.; Biundo, S.; Geier, T.; Hoernle, T.; Nothdurft, F.; Richter, F.; and Schattenberg, B. 2014. Plan, repair, execute, explain-how planning helps to assemble your home theater. In *ICAPS*.

Bernhard, K., and Vygen, J. 2008. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*.

Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, 154–169. Springer.

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

Diankov, R. 2010. *Automated Construction of Robotic Manipulation Programs*. Ph.D. Dissertation, Carnegie Mellon University, Robotics Institute.

Edelkamp, S. 2000. Planning with pattern databases. In *ECP*.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *IJCAI XAI Workshop*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Haslum, P.; Slaney, J.; Thiébaux, S.; et al. 2012. Incremental lower bounds for additive cost planning problems. In *ICAPS*, volume 12, 74–82.

Hershberger, D.; Gossow, D.; and Faust, J. 2016. Rviz. `http://wiki.ros.org/rviz`.

International Planning Competition. 2011. IPC Competition Domains. `https://goo.gl/i35bxc`.

Kambhampati, S. 1990. A classification of plan modification strategies based on coverage and information requirements. In *AAAI 1990 Spring Symposium on Case Based Reasoning*. Citeseer.

Keyder, E. R.; Hoffmann, J.; Haslum, P.; et al. 2012. Semi-relaxed plan heuristics. In *ICAPS*.

Langley, P.; Meadows, B.; Sridharan, M.; and Choi, D. 2017. Explainable Agency for Intelligent Autonomous Systems. In *AAAI/IAAI*.

Lombrozo, T. 2006. The structure and function of explanations. *Trends in Cognitive Sciences* 10(10):464 – 470.

Lombrozo, T. 2012. Explanation and abductive inference. *Oxford handbook of thinking and reasoning* 260–276.

Marthi, B.; Russell, S. J.; and Wolfe, J. A. 2007. Angelic semantics for high-level actions. In *ICAPS*, 232–239.

Meadows, B. L.; Langley, P.; and Emery, M. J. 2013. Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In *AAAI Workshop: Plan, Activity, and Intent Recognition*, volume 13, 13.

Miller, T. 2017. Explanation in artificial intelligence: Insights from the social sciences. *CoRR* abs/1706.07269.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial intelligence* 5(2):115–135.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Seipp, J., and Helmert, M. 2013. Counterexample-guided cartesian abstraction refinement. In *ICAPS*.

Seipp, J., and Helmert, M. 2014. Diverse and additive cartesian abstraction heuristics. In *ICAPS*.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *AAAI*.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 639–646. IEEE.

Srivastava, S.; Russell, S. J.; and Pinto, A. 2016. Metaphysics of planning domain descriptions. In *AAAI*, 1074–1080.

Steinmetz, M., and Hoffmann, J. 2016. Towards clause-learning state space search: Learning to recognize dead-ends. In *AAAI*, 760–768.

Young, N. E. 2008. Greedy set-cover algorithms. In *Encyclopedia of algorithms*. Springer. 1–99.

Challenges in Explainable Planning for Space Operations

Simone Fratini and **Nicola Policella**
Solenix Deutschland GmbH
Darmstadt, Germany
name.surname@solenix.de

Abstract

This paper discuss the role of explanation and the solutions adopted in the development of planning and scheduling tools for the support of space mission operations at the European Space Agency (ESA). A key point to strengthen the effectiveness and success of fielding planning and scheduling applications in space is the capability of providing an explanation of the solutions as well as of the solving process to the end users. The approach has been consolidated over the last decade while developing several tools and functionality to support explanation, but the challenge is still open and far away for being properly resolved.

Even though the approach has been utilized only in the space operations domain, we think that the concepts and problems highlighted are general enough to be applied to other domains and/or planning algorithms.

Introduction

The success of "deep learning" methods in the 2010s has raised the attention on explaining the decisions taken by an AI based system. In fact these methods are naturally opaque, that is, their behavior cannot in general easily be understood by humans. In order to foster research in the area of Explanable AI (XAI), agencies like DARPA have recently launched ad-hoc programs (DARPA 2016).

AI Planning seems instead to be in a better position (with respect to Machine Learning for instance) to provide a correct and complete explanation of its behavior and to result more transparent to a not-expert end-user (Fox *et al.* 2017).

Even though the nature of AI planning is more adequate to achieve transparent systems, explaining the behavior and the solutions of planning algorithms is still far to be trivial. In this paper we discuss the general approaches we use to facilitate the explanation of the automated solving process. We also make use of two recent applications where explanation is playing a crucial role.

Motivation

A key point to strengthen the effectiveness and success of fielding planning and scheduling applications in space is the capability of providing an exact explanation of the solutions as well as of the solving process to the end users.

This is very important for at least three reasons. First of all the planning tool is usually introduced in an already running, operational environment in order to optimize and/or automate some steps of the process. In this scenario, the users already have been generating plans for some time, and when the system provides solutions that they do not "recognize" it becomes of primary importance to be able to explain how they have been generated and justify differences they might find in the solutions generated by the "new tool".

Second, the planning tool will rarely work as "black-box" or pure plan generator. Usually a "man-in-the loop" approach is the preferred, and the planning technologies shall provide assistance to the human operators in generating or updating plans. In this interactive scenarios, it is necessary to provide explanations and to justify what is being done to put the users in the position to properly interact with the system.

Finally, since in operation usually the planning activity is shared among different groups, each one responsible for a sub-set of the whole plan, a proper explanation is also needed to have effective iterations between the different teams involved.

The remainder of the section provides a few examples of the different types of explanation we encountered in our experience in developing Planning & Scheduling solutions to support space operations.

Mexar2 experience

Our first project done in ESA-ESOC to introduce automated planning was the MEXAR2 system which is responsible for the generation of the activity plan used to download the satellite data from the on board memory to ground (Cesta *et al.* 2007).

The tool enables MARS-EXPRESS mission planners to rapidly produce a data dumping plan as well as to explore alternative solutions, providing the means to choose the more robust plan for execution. Additionally, the user is currently able to analyze the dumping problem over multiple days and identify payload overcommitments that cause resource bottlenecks and high risk of data losses. As a consequence MEXAR2 has allowed a significant increase of data return over the whole mission duration. These features have effectively made the system a fundamental work companion for the mission planners.

From this first experience was already clear the crucial role of being able to explain automatically the steps taken by the planning algorithm. For instance, at first was not clear why the solver was creating plans with many small activities. In fact such a fragmentation was the resulting of the optimization process which was trying to use at best all the available resources. Optimization was necessary to take into account different priorities for different packet stores and to provide robust solutions[1].

A fundamental aspect for users acceptance of the tool was to keep them into the loop. Mission planners in charge of spacecraft operations must be capable of understanding in detail any step of the solution and maintaining authority on any decision. This important requirement encouraged us to develop an interaction tool to foster the collaboration between the mission planners and the automated algorithms. In parallel we also modified the original algorithm in a way to make possible to the user to better control through specific tuning parameters the solving process.

Whereas at the time the approach followed was very specific to tool and the users, several lessons learned were collected and then drove the design of the generic planning and scheduling platform. This also applied to the explanation features.

Alphasat Payload Planner

Since 2013 the TECO planning system (just TECO below) is managing in an completely autonomous way the different payload experiments for the Alphasat satellite (Policella *et al.* 2013). The satellite carries four technology demonstration payloads. While each payload has a dedicated operations center responsible for defining and requesting the different experiments, the spacecraft resources (e.g., power, thermal constraints), downlink data, and telemetry budget are shared among the four payloads. This required a coordination and selection of the different experiments.

By exploiting advanced planning and scheduling technologies, the TECO planning system has been designed to have a high degree of autonomy where the final activity plans are generated automatically (no user intervention). Moreover the generation of the plan is done as an iterative process between the different payload centers on one side and the TECO planner on the other side. The goal of this iterative approach is to maximize the scientific return of the technology demonstration by helping to highlight the possible conflicts among the different experiment requests.

A relevant aspect of the output generation process is the need of providing sufficient explanation about the solving process. This is crucial in our case as the system has been designed to be completely autonomous. Our objective is to provide the system users with the necessary information to understand the decisions taken during the solving process and the final activity plan. A proper explanation is also needed to have effective iterations between the different entities involved. Considering that the time available to provide a new set of task requests is limited, it becomes fundamental to provide the right explanation on, for instance, why a task was not allocated.

Cluster-II Pass Planner

ESAs four Cluster-II satellites conduct three dimensional insitu measurements of the Earths magnetosphere. The mission is operated from the European Space Operations Centre in Darmstadt, where the FCT (Flight Control Team) prepares the routine operations for the spacecraft. A major aspect of the mission operation is the planning of ground station passes. Aside from the input of the science operation planning, the ground station plan also depends on the input of the Flight Dynamics team and on ground station availabilities. Ground station passes enable the FCT to download the scientific data from the spacecraft and check the overall health of the satellites. Given the large number of constraints involved in pass scheduling and the complexity of the problem, a tool based on AI technologies, TIAGO, Tool for Intelligent Allocation of Ground Operations, has been developed to ease such a process (Fratini *et al.* 2017).

A key operational aspect in this scenario is the capability of providing explanation on the choice of the ground station, as well as pass duration and temporal allocation. Users inspect visually the solutions and want to understand why a ground station has been chosen, for instance in place of a different one that might looks better or more suitable. Or why dumps are delayed (according to user's expectation), causing a dangerous filling of the on board memory, or, on the contrary, anticipated, resulting in an expensive over use of the ground stations that looks not justified by the filling level of the on board memory.

Explanation Approach

In pursuing a "process driven" approach to support operations at ESA[2] we aim at general purpose tools for facilitating the design and synthesis of new products. The general idea is the one of improving the "process" of tool development, taking advantage of the state of the art planning and scheduling technology.

In this view, the APSI Framework (APSI 2017) is a Java architecture for rapid prototyping of planning and scheduling applications currently in use at ESA for deploying new tool to support mission operations. The platform is designed for constraint-based temporal planning and scheduling. Constraint-based temporal planning, often referred to as "timeline-based planning", is an approach to temporal planning which has been applied to the solution of several space planning problems – e.g., (Muscettola 1994; Jonsson *et al.* 2000; Smith *et al.* 2000; Frank and Jonsson 2003; Chien *et al.* 2010; Cesta *et al.* 2011). This approach pursues the general idea that planning and scheduling for controlling complex physical systems consist in the synthesis of a set of desired temporal behaviors, named *timelines*, for system

[1] In particular, the idea for robustness in this scenario is to maintain a non-critical level of memory bank usage so as to allow run time variations to be absorbed with a low risk of overwriting.

[2] As an attempt of innovation with respect to the traditional "product driven" approach aimed at deploying solutions for specific mission problems.

features that vary over time. In this approach, problem solving consists of controlling components by means of external inputs in order to achieve a desired behavior. Hence different types of problems (e.g., planning, scheduling, execution or more specific tasks) can be modeled by identifying a set of inputs and relations among them that, together with the model of the components and a given initial set of possible temporal evolutions, will lead to a set of final behaviors which satisfy the requested properties; for instance, feasible sequences of states or feasible resource consumption[3].

Explanation in APSI

Being APSI the core of planning and scheduling applications we deploy for ESA, the explanation approach for TECO and TIAGO (presented later in this Section)is strongly grounded on modeling primitives and services provided by the framework.

APSI provides two general classes of modeling primitives: state variables and resources. Planning statements are specified as temporal assertions on timelines subject to temporal and transition constraints. Cause-effect relationships among planning statements and/or resource allocations are specified by means of temporal and logical synchronizations. Applications built using the framework can exploit the domain independent framework explanation primitives (for the solving process and for the solution) to build up a user oriented, application specific explanation significant for the final user.

A relevant aspect to support the generation of explanation is the fact that the APSI solving approach is a so-called flaw-based approach. This consists in an iterative, constructive paradigm which generates the final solution step-by-step driven by the identification and resolution (using the "right" solver) of flaws in the current plan. The causal relations between flaws and planning decisions are here exploited to drive the explanation process. In fact the flaws in a plan can be seen as the difference between this plan and an actual solution of the problem.

In general in APSI four elements are playing a role in generating the final explanation to the users:

1. The tree of flaws and decisions that are behind the generation of a solution. Here we are talking about trees and not simple chains as also failures in the solving process should be considered to provide a complete explanation. It is also worth remarking as the decisions can be resulting from the application of one of the solvers as well as from the propagation of the planning model.
2. The explanation of each decisions provided by the specific solver. In fact, all solving decisions are labeled with information about the solver who has taken the decision and the motivation of the decision.
3. A communication protocol. This is used to exchange the relevant information with the users of the system. This protocol has been introduced to address differences in backgrounds among the different partners. In fact, while the users are experts in their specific domain, they are not required to be experts in AI planning and scheduling.
4. An Explanation Generator module. The goal of this module is to interpret the information provided together with the solving decisions and of generating information for the system users by applying the given protocol.

It is worth remarking as the first two points are generic aspects already part of APSI. On the contrary the last two are specific of the final application and can have a different implementation from case to case.

In the remainder of the section we describe the generic explanation elements in the framework. In particular the different explanation generated by the model propagation functions and the solving process.

State Variables. State variables represent components that can take sequences of symbolic states subject to various (possibly temporal) transition constraints. This primitive allows the definition of *timed automata*.

The values taken by a timeline instantiated for a state variable component in the APSI framework (and temporal relations among them) can support explanation considering the timeline as a valid sequence of symbols accepted by the timed automata. A value for instance can be justified with the need of instantiating valid transitions, or with the need of enforcing temporal constraints on the durations of the states. If the timeline is not complete (it contains unspecified intervals of time or *gaps*), temporal constraints on unspecified intervals can be justified by the need of valid sequences of states to fill the gaps. If the timeline does not represent a valid sequence for the automata, information can be provided on violations: invalid transitions between specified values, invalid durations of the values or of the unspecified intervals. This low level explanation is used by solvers to build higher level explanations.

Resources. Resources are used to model any physical or virtual entity of limited availability, such that its timeline (or profile) represents its availability over time whereas a decision on the resource models a quantitative use/production/consumption of the resource over a time interval or in a time instant.

The timeline of a resource supports the explanation process by providing information about intervals of time that violate the resource capacity constraint. Given that, the explanation process can justify temporal relations on resource allocations, like deadline or release times, precedence or duration constraints generated by the solving process to remove peaks of consumption.

Synchronizations. The physical and technical constraints that influence the interaction between subsystems (modeled either as state variables or resources) are represented by means of temporal and logical synchronizations among the values taken by the automata and/or resource allocations on the timelines. Languages for timeline-based planning have

[3] A detailed description of the planning approach, state of the art and basic concepts is out of the scope of this paper. More information can be found, for example, in (Muscettola 1994; Frank and Jonsson 2003; Fratini and Cesta 2012).

constructs, called *synchronization* in APSI, to represent the interaction among the different timelines that model the domain. Conceptually these constructs define valid schema of values allowed on timelines and link the values of the timelines with resource allocations. Despite the syntactic differences, they allow the definition of Allen's relations (Allen 1983) like quantitative temporal relations among time points and time intervals as well as constraints on the parameters of the related values.

The synchronization construct on which the timeline based planning is grounded provides a significant support to the explanation process. The synchronization, in fact, gives teleological explanation for the values in the timeline. A value (or a resource allocation) can be traced back, trough a chain of synchronizations, to the goals that justify its allocation in the plan. These goals can originate in the planning problem or can be generated during the planning process to complete a timeline or to produce a resource. On the other way, following the synchronizations applied to justify a value or a resource allocation, it is possible to build a list of supports for that statement. In other words, the synchronization gives explanation about 'why' a given value is there, and to 'what' this value is a support for.

Problem Solving. An application in the APSI framework, being a generic planner and/or scheduler or a domain specific deployed application is designed as a collection of *solvers*. Based on the constraint-based paradigm, the search space of an APSI solver is made of planning and scheduling statements on timelines and temporal and data relations among them. An APSI solver provides explanation based on the actions taken to update the plan/schedule, grounded on the information collected on state variable and resource timelines.

Available solvers and applications include state-of-the-art binary and multi-capacity schedulers as well as integrated planners and schedulers (Fratini *et al.* 2015). Solvers are chosen and activated on a flaw detection base. When a flaw is detected on a timeline the planner activates the corresponding solver to fix the problem. A flaw is any type of violation in a plan. It can be a logical flaw, when unsupported actions are added to the plan, or a resource violation flaw, when a resource is over or under used, or any other impairment of temporal allocation on the values over a timeline. The flaw-based solving approach provides a natural way for building explanation by connecting issues detected in the partial plan with the explanation provided by the solver chosen for fixing it. The explanation of the plan process is then given as a sequence of steps taken to update the plan, with each step motivated by the issue detected and explained with the actions taken to fix it. Some of the solvers currently available in APSI are:

Domain Theory Unfolder The DTU detects unsupported statements in the plan and justify them by applying synchronizations. The DTU can apply the synchronization by expansion (adding new statements) or by unification (constraining existing statements). The DTU explains the process by listing the effects and the supports of a statement, i.e. expansion and unification applied to justify it.

Partial Order Scheduler The POS is supporting the scheduling process resulting from planning to guarantee temporal flexibility.[4] The explanation in this case refers to the consumer-producer relations created by the solver. Moreover in case of no solution the solver provides as explanation a set of activities which are associated to the unsolvable conflict.

Resource Activity Generator It makes sure that linear resources can be adequately managed by avoiding any overconsumption. The generator add production or consumption statements to the plan, grounding the choice on the domain theory (which states of a state variable can produce/consume resource, how much can be produced/consumed and what kind of opportunity are needed for the production/consumption). The explanation includes the resource violation that fostered the process and which production/consumption status has been chosen on the basis of the temporal position and entity of the violation and available opportunities.

Resource Profile Bounder It uses a MaxFlow algorithm to bounds position and duration of activities to assure that all the resource constraints and/or requirements are consistent. The explanation provided by a profile bounder links duration and position constraints generated with the resource violation that fired the solving process, for instance stating that a consumption activity has been shortened to avoid a resource overconsumpion in a specific intareval of a given resource timeline.

Timeline Completer It fills gaps in state variable timelines. The explanation provides information on which sequences of states have been generated considering the transition and temporal constraints induced by the automata.

As pointed out, the solving process is driven by the different flaws identified and solved - therefore by considering the flaws collected it is possible to have a first analysis of the planner behavior (and explain it).

However this is not sufficient. As shown in the TECO example below, one important factor to consider is also the order in which the different flaws are collected and moreover when they first appear – were they part of the problem or introduced during the solving process (by one of the solvers)?

Explanation in TECO

In the TECO operations, a proper explanation is crucial to have effective iterations between the different entities involved – the goal is make the users to better understand why a task has not been allocated and how the requests can be updated.

There might be different explanations (and combinations of these) for which a task is not allocated: wrong input files, ill-defined task requests, tasks conflicting with spacecraft status, tasks conflicting with other task requests, tasks not allocated because linked to conflicting tasks, etc. Once the

[4]The POS (Policella *et al.* 2007) is a set of activities partially ordered such that any possible complete order that is consistent with the initial partial order is a resource and time feasible schedule.

explanation is identified by the system, before distributing it, it is necessary to put it in a form that can be understood by the receivers.

A first aspect worthy of attention is that with this approach the explanation provided to the user is not generated directly by one of the different solvers.[5] This is required as 1) the single solvers has a limited view of the general solving process and 2) it is necessary to decouple the solvers from the final explanation generation process (and the associated protocol).

Figure 1 shows two simple examples where a task cannot be allocated. In both cases the explanation cannot be provided directly by a single solver. In both examples both the payload (TDP) and the ground activity timeline are represented.

Figure 1(a) contains two tasks: t_a which is associated to the payload TDP_a and triggers the mode of the payload from the mode X to W and back to X (see payload timeline). t_a requires a fixed start time (and duration). Task request t_b is instead flexible in time (in the figure this is represented by the double arrow on the ground timeline). The task is associated to the TDP_b payload and changes the mode of the payload from F to A. Also one of the domain constraints requires that mode W of payload TDP_a shall be synchronized with the mode A of TDP_b. Finally, the ground is not always available to execute the tasks (see not "not available" activity in the ground timeline).

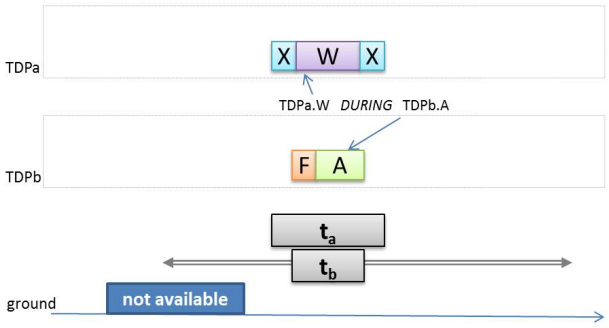
The solution of this problem contains only t_a. In fact to satisfy the domain constraint the planner adds the solving decision $et(t_b) \leq st(t_a)$ and pass it to the scheduler. The scheduler will then fail as t_b cannot be scheduled before the start of t_a due to the not availability of the ground. In generating the explanation for the missing allocation of t_a it will then be necessary to consider not only 1) the conflict identified by the scheduler but also 2) the decision taken by the planner, and 3) the domain theory (which contains the constraint $TDP_a.W$ during $TDP_b.A$).

In the second example, Fig. 1(b), only one task request, t_a, is present. Additionally it is required to switch off all the payloads (during the time interval represented on the ground timeline). In this case, while the task could be allocated in a time window, the planner will find a conflict between the final status of the task (X) and the value requested by the platform activity "TDPs OFF." In this case the explanation process will need to consider both 1) the planner conflict and 2) the domain theory.

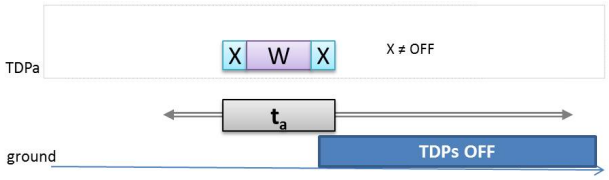
The objective of the previous examples is to show that for providing the correct explanation to the users it is necessary to consider all the decisions coming from the different solvers. Even if they represent very basic cases, the examples are also useful to give an idea of the effort needed to retrieve proper information especially when considering more complex and real cases, with more activities, resources, constraints, etc.

The approach implemented to generate explanations consists in "tracing back" all the different solving decisions that

[5]In TECO a planner and a scheduler are actually used(Policella *et al.* 2013).



(a) Planning decision and scheduling conflict



(b) Planning conflict

Figure 1: Example of explanations

are associated to a task request. This is done by exploiting the different annotations added during the solving process. Once all this information is collected, the explanation generator module identifies the specific case based both on the solvers which generate the annotations (for instance the scheduler) and on the content of the annotations (e.g., conflict with spacecraft activity).

Explanation in TIAGO

A key operational aspect in Cluster-II pass scheduling problem is the possibility for providing explanation on the choice of the passes, duration and temporal allocation. Users inspects visually the solution and need to understand why a ground station has been chosen, for instance in place of a different one that might looks better or more suitable. The explanation provided by the planner trough the APSI framework primitives described above is then used to synthesize an higher level explanation semantically significant for the final user.

The decision of allocating a pass in this domain is driven by the flaw in the on-board memory allocation. The memory is continuously being consumed by science operation. The position and the entity of the flaw give the justification for allocating a pass in the plan. The explanation provided by the Resource Activity Generator (that decided to allocate the pass) and the Resource Profile Bounder (that bounded the pass duration) includes information for the time window where the pass has been allocated and its duration. The earliest and latest start time for the possible pass allocation depends on the resource profile. The earliest start time depends on the minimal pass duration and on the amount of data in the memory: since a pass dumps at a rate higher than all the science data rate production, a pass cannot be allocated if

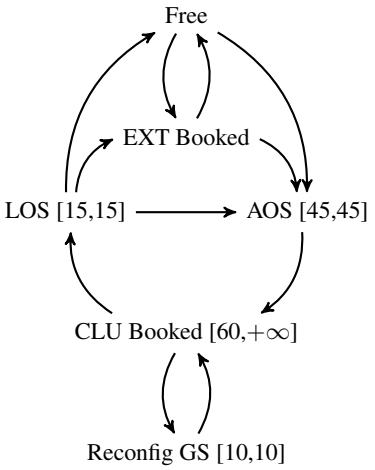


Figure 2: Ground station bookings state varible.

there are not enough data to download, which gives a lower temporal bound for the allocation. Conversely the latest start time for allocating a pass depends on the position of the flaw: the pass must start not too late to avoid the memory overwriting. The duration bounds depend on the ratio between the production rate in the interval and available dump rate.

Moreover, in the Cluster-II domain some of the constraints on ground stations booking are modeled in a state variables definition. Before the actual begin of tracking, a ground station configuration activity of 45 min has to be considered. During this time, the ground station must be available and should not be booked by any other activity. After the end of tracking, a ground station post-pass activity of 15 min has to be considered. Similar to the prepass activity, no other activity should be booked during this time. When two consecutive passes of two different Cluster-II spacecrafts are scheduled on the same ground station, the time accounting for the post and prepass activities can be reduced to exactly 10 min (in order to change between the different carrier frequencies of the spacecraft). If this 10 min interval cannot be satisfied, a configuration time of at least 1 h (15 min post-pass and 45 min pre-pass activities) has to be considered. Moreover, for cost efficiency reasons, scheduled passes should have a duration of at least 1 h.

To represent the configuration time, intermediate states are introduced on the booking timeline of the ground station and transition rules are specified, hence the state variable in Figure 2 models the ground station booking logic. In this state variable, a cluster booking has a minimal duration of 60 minutes and it is possible to pass between two bookings either trough a reconfiguration status of 10 minutes or passing by an LOS (Loss Of Signal) status of 15 minutes and a successive AOS (Acquisition Of Signal) of 45 minutes. Non-cluster bookings are modeled as states of external booking and for them it is always required an AOS phase before that a Cluster pass can be booked.

In addition to that there are constraints among the values taken by the different state variables and the resource allocations: a ground station must be visible and should not be booked by any other satellite in order to be able to book a Cluster ground station pass. The pass is also synchronized with an activity that dumps the memory, according with the link budget available and it has to be scheduled against science modes (a pass start or end should stay 6 to 15 minutes away from a science mode switch because of the unaffordable computational effort on board to start/finish a pass and contemporary switch a science mode).

Hence the position and duration of a pass (in within the bounds decided by the resource allocator described above) is also constrained by other planning decisions: (a) the pass must be synchronized with ground station's visibility windows; (b) must be properly scheduled with respect to other events on the same ground station, to acquire and release the station; (c) must be synchronized with the science modes, to avoid start or end points to be too close to science modes switch; (d) must be properly synchronized with available downklink rates for the selected station and (e) must guarantee a proper de-overlapping with other passes scheduled for the same satellite on other ground stations. (a), (c) and (d) are enforced by the Domain Theory Unfolder, (b) is enforced bu the Timeline Completer and (e) is enforced by the schedulers. Combining the explanations provided by these solvers, to the user is provided a trace of pass allocation in the plan that includes: (a) a window of opportunity for allocating the pass, explained in terms of memory filling; (b) a choice of a ground station, explained in terms of station availabilities and other concurrent use of the same station; (c) a position and duration of a pass, explained in terms of station usage logic, other passes allocation for the same satellite and memory consumption.

Discussion

Our planning/solving approach is based on a constructive/iterative schema driven by flaw resolution. A flaw entails a resolution step which provides explanation on the updates performed on the plan to fix the flaw. This approach integrates different types of information that concur in explaining the plan process and result: temporal information, cause-effect relationships and resource allocations.

Compared to explanations that can be provided on action based planning (STRIPS and PDDL), our approach has the advantage of exploiting explicit temporal information and scheduling constraints to justify plan statements not only causally but also temporally (and in space domain often the most important issue is not to explain why something appears in the plan but mostly when things happens and in which order). On the other side, the timeline-based approach to planning does not make explicit the preconditions for an action. As a consequence, it is more difficult, in case of failures not induced by resource or temporal conflicts, to identify a closed set of violated logical properties to justify the issue.

Besides that, in our experience there are various aspects to be considered in order to make the explanation an added value in the process.

First of all the planner is part of a tool which can be seen as a partner of the end-user in solving the problems ('man in the loop' principle). Therefore, in order to have an efficient

collaboration, it is fundamental not only to explain the final solution but also the intermediate states and, more in general, the decisions taken during the solving process (Smith 2012). In this respect, the flaw based approach provides a step-by-step explanation not only on which actions are taken but also on why and when they are taken.

A second important aspect is the explanation of failures. In real applications it is not only important to be able to justify a plan, but we need often also to explain why a plan could not be found. This helps in tailoring system configurations and in analyzing problems. Again the flaw based approach gives a great support on this: starting from the unresolvable flaw that led to the failure of the planning process, it is often possible to frame the issue and decide for instance if it is a logical problem or a resource allocation problem.

Last but not least, the hierarchical approach that bases the explanation on increasing level of abstraction allows a good trade-off between generality and significance of the information. Temporal e logical allocation of values on the plan are explained, bottom up, as effect of propagation of temporal statements generated as consequence of solving steps (plan expansion or scheduling for instance), which in turns gives an explanation of higher level statements in the plan that play a role in that allocation. This analysis resulted useful in explaining solutions and failures of the planning process.

Pursuing a top down approach, the explanation starts from the high level goals and following the trace of the solving steps applied, frames the effects of the process on the plan (what provides support for the plan, which effect the goal has on the resource profiles and so on). This analysis proved to be a valid support for re-planning and mixed-initiative planning, where is useful to understand which effects a goal has on the whole plan and how it affects the planning process.

Notwithstanding this initial achievements, providing explanation remains in general an hard issue to face. First of all the solving steps and the set of constraints that influence a plan statement not always give a significant or usable information.

Regarding failure explanation for instance, a flaw can be the result of large set of combined effects, to the point that is not identifiable where and how to react. Also the flaw resolution process can sometimes make hard the explanation of the result. Some flaws can be resolved too early for instance, influencing the following steps of the process to the point that the final plan is not explainable. Optimization also makes the explanation more difficult: some optimization algorithms are not explainable (as genetic ones for instance).

Probably the most important issue is related with the explanation generation vs presentation: to be useful the explanation must be presented semantically close to the user, while is currently syntactically generated close to the data model and/or solving process. The synthesis of a semantically significant explanation, as shown in the examples presented in this paper, is still a cognitive process that requires a deep understating of the solving process first and of the planning domain also.

References

James Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

APSI. APSI Software Distribution Web Site. https://essr.esa.int/project/apsi-advanced-planning-and-scheduling-initiative, 2017.

A. Cesta, G. Cortellessa, M. Denis, A. Donati, S. Fratini, A. Oddi, N. Policella, E. Rabenau, and J. Schulster. MEXAR2: AI Solves Mission Planner Problems. *IEEE Intelligent Systems*, 22(4), 2007.

Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, Angelo Oddi, and Giulio Bernardi. Deploying Interactive Mission Planning Tools - Experiences and Lessons Learned. *JACIII*, 15(8):1149–1158, 2011.

S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, and S. Frye. Timeline-Based Space Operations Scheduling with External Constraints. In *ICAPS-10. Proc. of the 20th International Conference on Automated Planning and Scheduling*, 2010.

DARPA. Explanable Artificial Intelligence (XAI), August 2016. Defense Advanced Research Projects Agency, Program n. DARPA-BAA-16-53.

M. Fox, D. Long, and D. Magazzeni. Explainable Planning. In *IJCAI Workshop on XAI*, 2017.

J. Frank and A. Jonsson. Constraint Based Attribute and Interval Planning. *Journal of Constraints*, 8(4):339–364, 2003.

S. Fratini and A. Cesta. The APSI Framework: A Platform for Timeline Synthesis. In *Proceedings of the 1st Workshops on Planning and Scheduling with Timelines at ICAPS-12 (PSTL-12), Atibaia, Brazil*, 2012.

S. Fratini, N. Policella, N. Faerber, A. De Maio, A. Donati, and B. Sousa. Resource Driven Timeline-Based Planning for Space Applications. In *Proceedings of the 9^{th} International Workshop on Planning and Scheduling for Space, IWPSS15*, 2015.

S. Fratini, N. Faerber, N. Policella, and B. Sousa. TIAGO Tool for Intelligent Allocation of Ground Operations on Cluster-II. In *SPARK-17. Scheduling and Planning Applications woRKshop at ICAPS*, Pittsburgh, USA, 2017.

A.K. Jonsson, P.H. Morris, N. Muscettola, K. Rajan, and B. Smith. Planning in Interplanetary Space: Theory and Practice. In *AIPS-00. Proc. of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling*, pages 177–186, 2000.

N. Muscettola. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., editor, *Intelligent Scheduling*. Morgan Kauffmann, 1994.

N. Policella, A. Cesta, A. Oddi, and S. F. Smith. From Precedence Constraint Posting to Partial Order Schedules. *AI Communications*, 20(3):163–180, 2007.

N. Policella, H. Oliveira, and E. Benzi. Planning Spacecraft Activities: An automated approach. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling, ICAPS 2013*, 2013.

D.E. Smith, J. Frank, and A.K. Jonsson. Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review*, 15(1):47–83, 2000.

D.E. Smith. Planning as an iterative process. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI 2012*, pages 2180 – 2185, 2012.

Improving Explanation and Effectiveness of Interactions among Autonomous Vehicles and Pedestrians

Sara Manzoni[1], Simone Fontana[2], Andrea Gorrini[1], Domenico Sorrenti[2],

Stefania Bandini[13]

[1] CSAI - Department of Computer Science, University of Milano-Bicocca, Milan (ITALY)
[2] IRA lab - Department of Computer Science, University of Milano-Bicocca, Milan (ITALY)
[3] RCAST - The University of Tokyo, Tokyo (JAPAN)

Abstract

In this paper we describe a study involving an autonomous vehicle based on planning and interaction situations with other road users (i.e. *pedestrians*). The paper presents the research framework and directions towards enriching the vehicle information about the moving objects on its way with formal, explicit and shared representation of available findings about pedestrian on-road behaviors. Among the advantages of the integration of empirical data achieved through behavioral studies about pedestrian dynamics, we claim that a research effort in this direction may significantly contribute to improve trust, transparency, and quality of communication between heterogeneous road users, mainly to avoid and solve obstruction situations.

In the paper we describe first steps of this multidiscilpinary research and we present a representation of concepts related to pedestrian crossing behaviours at unsignalized pedestrian crossings (based on ontology formalism and derived from the reference literature). After, we introduce the movement planning model of the autonomous vehicle and we overview experiments we are developing to collect useful insights to improve knowledge on perception and interpretation of actions in interaction situations (involving pedestrians crossing in the presence of an autonomous vehicle). Further developments of the vehicle behavior to improve its capability to interpret the scene at real-time, as well as effectiveness of the vehicle communication with other road users are then discussed.

Introduction

Nowadays, autonomous driving is gaining increasing importance, with some vehicles supposed to perform at the SAE-level 5, already circulating in some cities. In this context, proper and interaction between a vehicle and the pedestrians that could be present in its surroundings is fundamental.

Pedestrian-aware planning, that is, planning keeping into consideration the behavior of pedestrians, is a longly studied problem in the robotics world. Initially, most of the works concentrated on the scenario of a robot moving in a crowded indoor space. For example Bennewitz *et al*. (Bennewitz et al. 2005) developed an approach that uses the prediction of the movement of a person to produce a better plan for the robot (*i.e*., a plan that would not interfere with the motion of the person). Pacchierotti *et al*. proposed a planner for traversing corridors with people coming from the opposite site (Pacchierotti, Christensen, and Jensfelt 2005). While these works focus on avoiding physical interference with a person, they do not address the important problem of the *perceived safety* of a plan (Kulkarni et al.). That is, the perceived level of danger induced by the robot motion. To allow a perception of safety, the robot should not move very close to a person, as this is perceived as dangerous by the person. This problem has been studied by Nonaka *et al*. and its solution is a fundamental step for the introduction of robotics in everyday world (Nonaka et al. 2004). It has also been addressed by Chen *et al*. who presented a method for the avoidance of the pedestrians whose trajectory is tracked (Chen, Ngai, and Yung 2008). The need of a predictive model of the movement of pedestrians is, for instance, discussed in (Nishino et al. 2016).
With the advent of autonomous driving vehicles, this topic gained an even larger importance as cars move much faster and can be more dangerous than small indoor service robots. For this reason safety and effectiveness in the interaction between pedestrian and autonomus vehicles are essential. Having a model capable to describe how to interact with the heterogenous types of pedestrians in the many different conditions is thus an essential part of an autonomous driving system: it would be not effective to deal similarly with an elderly person and with a group of teenagers; analogously, the vehicle behavior should change when the road is dry and there is, e.g., risk of black ice.

The variety of pedestrian behaviors observed at intersections has been the subject of interest of several researches (Evans and Norman 1998; Hamed 2001; Perumal and others 2014), aiming both at a better understanding and modeling of the behaviors of road users. Besides the understanding of human behaviors, pedestrians' movement models are studied also for several other objectives; for instance, for their integration into pedestrian simulation software platforms (Federici, Manenti, and Manzoni 2014) and for the improvement of the quality of crowd management support systems. Knowledge sources about pedestrian behaviors are thus quite fragmented and relates to several study areas.
Among recent results on pedestrian crossing behavior, environmental, demographical, social and psychological factors have shown to have a relevant impact on risky crossing behavior (Sisiopiku and Akin 2003), such as pedestrian age (Gorrini, Vizzari, and Bandini 2016) and movement as part of a group (NT. Dang et al. 2016). The crossing be-

Figure 1: The autonomous vehicle involved in LONGEVICITY experimental research. It implements a two-level planner and it is able to detect and avoid obstacle that may be present on its surrounding. Previous experimentations have been conducted mainly in its natural application scenario of a crowded unstructured space (like a public square). Within the planned experimentation the cart will be involved in interactions with participants that will be asked to cross the road in the presence of the autonomous vehicle.

havior of elderly pedestrians, for instance, is affected by the progressive decline of motor and cognitive abilities linked to ageing (Dommès et al. 2015), and by a scarce level of assertiveness in communicating to drivers their intention to cross. Moreover, a pedestrian crossing decision is influenced also by the behavior of others who are crossing at the same time, as the simultaneous motion with other pedestrians reduces indeed cautiousness due to diffusion of responsibility, and successful of crossing is probably overestimated (Harrell 1991).

These observations, and the opportunity to collaborate within the LONGEVICITY[1] project, motivated us at including a multidisciplinary study involving developers of IRA lab autonomous vehicle (see Figure 1) and competences on crowd studies from CSAI research center. A dedicated experimentation has been designed to better investigate the behavior of crossing pedestrians in the presence of an autonomous vehicle and to assess potential advantages on trust, interaction and transparency of the vehicle activity (including actions performed for communication aims, i.e. inter-actions (Chakraborti, Dudley, and Kambhampati 2018)).

Pedestrians and vehicles can interact in many different ways, but they always have to comply with the road code and have safety in high regard (World Health Organization 2015). Among the many different scenarios of interactions we focused our studies on a zebra crossings not regulated by a traffc light (i.e. Unsignalized Pedestrian Crossing, from now denoted as UPC). In this case, interaction between vehicles and pedestrians is defined as *obstruction*, (Ferber 1999). As in any obstruction situation, interaction at UPCs require a conflict resolution strategy to grant the success of individual coordinated actions. In a non-automated vehicular-traffic system, the resolution of conflictual situations is delegated to traffic rules, but it is often influenced also by culture-influenced customs; several heuristics are often observed on real roads.

We claim that this work could contribute not only to safer driving behavior of autonomous vehicles, but also to improve the trust into its decision-making, and the effectiveness of its communication with the other road users (Fox, Long, and Magazzeni). With the shared general goal of improving the quality and shareability of available information, we aim at reducing risk of inappropriate vehicle behavior, improving vehicle knowledge on other road users behavior and exploiting this knowledge towards improvement of trustability of its activity. To realistically consider application scenarios where autonomous vehicles and humans interact, potentially also with other partially robotic road entities, acceptance (as trusted co-user of a shared space) is an unavoidable aspect. High-quality interactions are also required to effectively engage and realise a semantically rich exchange of information with other agents, such as pedestrians or cyclists. Moreover, transparency of vehicles activity will be advantaged by a set of shared regulations they can be used.

An ontological representation of available knowledge about pedestrian crossing behavior is described in the next section. The motion planning model of the autonomous vehicle that will be involved in the experimentation, and the experimentation aims and main steps are then summarised.

Pedestrian Behaviors at Intersections

The ontology described in the following is an explicit knowledge representation of available knowledge about pedestrian behavior at UPC (i.e. zebra crossing without traffic light coordination of flows). In this case, interaction between vehicles and pedestrians is defined as *obstruction*, (Ferber 1999): any agent involved in such situation has individual skills to employ autonomously the available resources to accomplish their goals, and agents' goals are compatible because one agent crossing the road will not compromise the capabilities

[1]The project "LONGEVICITY - Social inclusion for the Elderly through Walkability" has the objective to study the cities of the future as characterized by the growing presence of long-lived and active citizens and by the need to design technologically advanced infrastructures, considering the increasing role that autonomous vehicles will have in this scenario. The LONGEVICITY project is funded by Fondazione Cariplo, within the call Scientific Research 2017 "Aging and social research: people, places and relations" along the period between April 2018 and December 2020 (Grant No. 2017-0938).

of the other agents to go straight on their way and vice-versa. In crossing situations where paths overlap, the agents need to coordinate their actions in order to share the use of the limited resource (i.e. the road portion that both want to occupy at the same time). A conflict resolution strategy to grant the success of individual actions shall be introduced. In a non-automated vehicular-traffic system, the resolution of conflictual situations is delegated to traffic regulation laws, but it is often influenced also by culture-influenced customs and several heuristics are often observed.

The exploitation of ontology-based (Mizoguchi and Ikeda 1998) explicit representation of road rules, of driving behaviors and heuristics has demonstrated to be enough expressive and practicable to enable vehicle context understanding (Feld and Muller 2011; Armand 2016; Mohammad 2015; Zhao et al. 2015). The main references for this work have been: an ontology (Armand 2016) that includes context concepts such as Mobile Entity (Pedestrian and Vehicle), Static Entity (Road Infrastructure and Road Intersection), and Context Parameters (*isClose*, *isFollowing*, and *isToReach*) and an ontology-based framework (Mohammad 2015; Zhao et al. 2015) for assessing the degree of risk in a road scene taking into account several factors (e.g. risk from other mobile objects like vehicles, pedestrians, cyclists, environmental risk related to weather and visibility conditions and, road environmental risk that is, related to road quality, road traffic signs and road type). Applying rules written in *Semantic Web Rule Language* (SWRL), the ontology provides an informed interpretation of road contexts. An ontology is an explicit specification of conceptualization (Gruber 1995). It describes concepts and relationships that are relevant to model a domain of interest. It specifies the vocabulary that is necessary to make assertions, and which may be inputs/outputs of knowledge-based agents. Domain ontologies, when are based on *Description Logics* (DL) (Baader et al. 2003), are described by two functional parts: *Terminological Box* (TBox) and *Assertional Box* (ABox). In the following subsections, pedestrian ontology developed in an explicitly, formal and exploitable to improve vehicle scene interpretation and decision-making.

Ontology TBox

TBox consists of the definition of all the concepts and relations that the ontology aims to describe, in our case detectable scene objects and contextual information about vehicle observable scene. It specifies: **concepts** (or classes) of the domain that the ontology aims to describe, organised into a taxonomy; **roles** that are properties that can be defined and assigned to concepts. Roles can be either **object properties** that define axioms in the form of binary relationships between two concepts, or **data properties** used to assign properties to single classes or instances of classes in the form: Concept1 - *Data Property* - Property Value. **Relations** between concepts are defined by hierarchical relations (concepts taxonomy), axioms (classes linked by object properties) and rules.

Figure 2 shows part of ontology TBox about taxonomic relationships among road entities that may be perceived in a driving situation (on a single lane road), while Figure 3

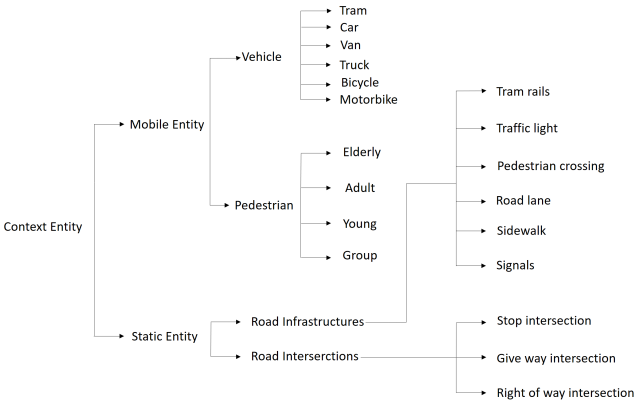


Figure 2: Graphical representation of TBox taxonomic relationships among road entities that may be perceived in a driving situation.

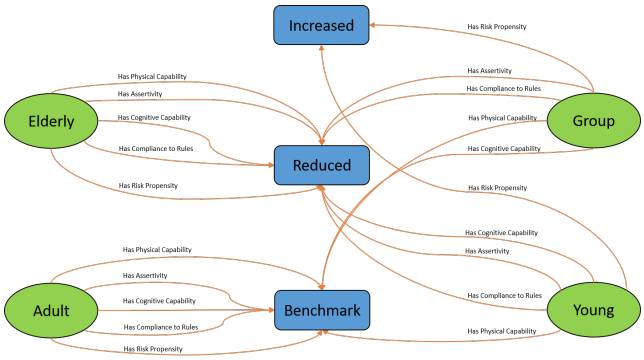


Figure 3: Graphical representation of TBox relationships between pedestrian profiles and relevant crossing features.

is a graphical representation of TBox relationships between pedestrian profiles and the following pedestrian profile properties: **cognitive capability** of subject's to estimate relevant motion coordination features (like distance from the subject and speed); **physical capability** to safely cross the road; **assertivity**, the ability to comunicate (more or less intentionally) the intention to cross; **compliance** with road rules; **propensity to risk**. Physical capability and propensity to risk are described as inversely proportional to age, while cognitive capability, assertivity and compliance decreases as the age diverges from the one of a reference pedestrian fully compliant to road traffic rules and with full capabilities and assertivity; for the present work, an adult person in the range of [25...50] years old.
The involvement of a child or of an elderly person in an interaction situation suggests its interpretation as potentially risky since, according to the reference ontology both pedestrian profiles are described as *less assertive and less respectful of traffic laws* and potentially less perceptive about communicative audio-visual signals. Rules like the ones exemplified in the Table manage this part of scene understading.

Each relationship refers to a pedestrian property and indicates whether the profile differs (with a reduction or increase) from a reference profile. Witin the T-Box ontology, these relationships are described in the form of axioms.

Ontology ABox

The ABox consists of instances of classes previously defined in the TBox, commonly called Individuals, that represent real life scenario, according to ontology interpretation. The ABox represents objects that are observed, and are understood thanks to prior knowledge available and explicitly described by TBox relationships (i.e. taxonomic relationships, axioms in the form of objects or data properties, or rules).

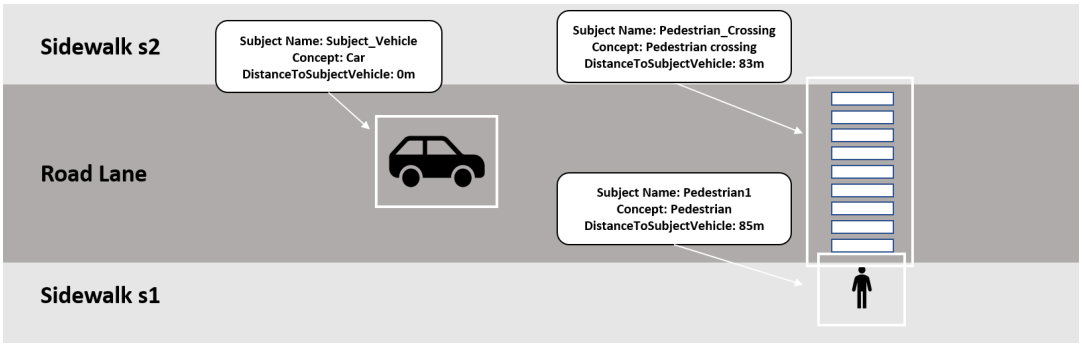
Figure 4 exemplifies an interaction situation at a unsignalized pedestrian crossing, with zebra crossing road infrastructure and right of way to pedestrians. Its description according to the above sketched TBox is shown. Information about detected road infrastructures, plausible profile of pedestrians and contextual information are available to be logged and exploited at, first real-time for improving decision making about actions to undertake in order to coordinate conflict resolution with pedestrian going to cross, as well. Moreover, this semantically rich contextual information is also available to improve explanation of vehicle decisions.

Vehicle Movement Planning

The autonomous vehicle involved in this experimental research (see Figure 1) is able to detect and autonomously avoid obstacles by computing its trajectories according to a two-level planning (i.e. global and local). Previous experimentations have been conducted (with a maximum speed of about 10 km/h) mainly in its natural application scenario of a crowded unstructured space (like a public square). Within the planned experimentation the cart will be involved in interactions with participants that will be asked to cross the road in the presence of the autonomous vehicle. A speed

Vehicle(?v0) $\wedge$ Adult(?p1) $\wedge$ Road_lane(?rl1) $\wedge$ IsCloseTo(?v0,?rl1) $\wedge$ IsToReach(?v0,?p1) $\rightarrow$ HasToAlert(?v0,?p1)	The adult pedestrian *p1* is close to road lane and want to cross, so the vehicle *v0* alert *p1* that can't cross because *v0* is upcoming
Vehicle(?v0) $\wedge$ Elderly(?p1) $\wedge$ Road_lane(?rl1) $\wedge$ IsCloseTo(?p1,?rl1) $\wedge$ IsToReach(?v0,?p1) $\rightarrow$ HasToDecelerate(?v0,?p1)	The elderly pedestrian *p1* is close to road lane and want to cross, so the vehicle *v0* doesn't alert because elderly have be a reduced cognitive capability.*v0* has to decelerate because *p1* has a higher risk profile than the standard.
Vehicle(?v0) $\wedge$ Young(?p1) $\wedge$ Pedestrian_crossing(?pc1) $\wedge$ Sidewalk(?s1) $\wedge$ Road_lane(?rl1) $\wedge$ IsOn(?v0,?rl1) $\wedge$ IsOn(?p1,?s1) $\wedge$ IsToReach(?v0,?pc1) $\wedge$ IsToReach(?p1,?pc1) $\rightarrow$ HasToDecelerate(?v0,?p1)	The young pedestrian *p1* is to reach pedestrian crossing, the vehicle *v0* is to reach the pedestrian crossing, *v0* has to decelerate because *p1* is a risk profile of pedestrian and so is more difficult to predict his action.

Table 1: Rules to describe relationship between observed pedestrians approaching unsignalized crossing, and its interpretation by taking into account pedestrian profiles.



Subject_Vehicle:

Object Property Assertions:

- **IsCloseTo** Pedestrian_crossing
- **IsCloseTo** Pedestrian1
- **IsOn** Road Lane

Data Property:

- **DistanceToSubjectVehicle** 0m
- **HasSpeedValue** 50 km/h

Pedestrian1:

Object Property Assertions:

- **IsCloseTo** Road Lane
- **IsCloseTo** Pedestrian_Crossing
- **IsCloseTo** Subject_Vehicle
- **IsOn** Sidewalk s1

Data Property:

- **DistanceToSubjectVehicle** 85m
- **HasSpeedValue** 3.7 km/h
- **IsAppraising** True

Pedestrian_Crossing:

Object Property Assertions:

- ∅

Data Property:

- **DistanceToSubjectVehicle** 83m

Action: Subject_Vehicle *has to stop* before Pedestrian_Crossing.

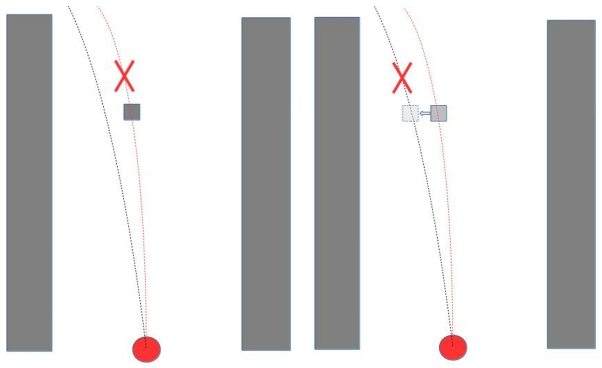
Figure 4: Example of a pedestrian crossing scenario (scale is not respected) and its interpretation according to ontology TBox rules, axioms (object and data properties) and concepts taxonomy.

profile between 20 km/h and 30 km/h will be designed properly. The best path to reach the goal is computed, first, taking into account only a previously determined static map (global plannig). This step is very similar to what a car navigation system does, although it does not have to necessarily use a topological map. The global planner calculates a path that the vehicle should follow. However, in practice, the vehicle will not be able to precisely stick to it. Dynamic objects, like pedestrians, cars or other movable objects are taken into account by local planner. Local planner takes as input a path calculated by the global planner and generates the best commands in order to follow it, taking into consideration any obstacle that the vehicle can perceive. Of course, it may happen that the vehicle has to deviate from the path, in order to avoid an unexpected obstacle.

During local planning, various feasible trajectories are generated, scored according to various criteria and the best one is chosen. Trajectories that should pass through occupied cells are obviously discarded (Figure 5(b)). A drawback of this type of local planning is that, while a trajectory represents an anticipation of future world state, its evaluation is taken into account only considering current environment state. If a trajectory intersects the current position of a crossing pedestrian, it is discarded, even though that person is moving and will not be there at the time the vehicle reaches that position. In this case, trajectory would be perfectly feasible. On the other hand, a local planner could choose a trajectory that passes through currently free space, but that would be occupied in the future because a person is moving (Figure 5(b)). This problem severely decreases the performance of a local planner, known for both indoor robots and autonomous vehicles. A performance study on our vehicle is object of current research.

Global planning is usually performed only once, unless the path reveals to be infeasible for some previously unexpected reason. On the other hand, local planning is performed cyclically, usually at a fixed rate, because it has to control the vehicle in order to avoid obstacles and keep moving to the goal. Finally, the commands computed by the local planner are used as set points for the control loops of the actuators, which run at a fixed and much higher frequency.

On this vehicle, the availability of semantically unambigous knowledge on pedestrian motion could improve local planner in both its decision-making and interaction with



(a) (b)

Figure 5: (a) An example of grid map with obstacles (in grey). The robot (in red) discards a trajectory that passes through an obstacle. (b) An example of the same scenario when the robot knows that the obstacle is a pedestrian and has a model of her/his motion, it can choose much better trajectories. The one on the left, indeed, is discarded not because it intersects an obstacle at the time of planning, but because it will intersect the path of the pedestrian.

other road users, for instance adapting its behavior (movement and online communicative actions) to different pedestrian profiles.

Experiments on Vehicle-Pedestrian Interaction at UPS

Within the LONGEVICITY project, the Complex Systems and Artificial Intelligence research center and Informatics and the Robotics for Automation laboratory of the University of Milano-Bicocca are sharing their competences with the aim of developing a cross-disciplinary approach to study pedestrian-autonomous vehicles interactions at UPC by means of the execution of an experimental campaign. In this methodological framework, a set of experiments will measure the interactions between a large sample of adult and elderly pedestrians and the autonomous vehicles developed by the IRA lab. The experiments will be executed with the authorisation of the Ethics Committee of the University of Milano-Bicocca, and they will be carried out in a private road of the campus of the university. Participants will be asked to cross the road in the presence of the autonomous vehicle; the crossing episodes will be video recorded from a zenithally point of view to facilitate video tacking analysis, focused on measuring the crossing behaviour of pedestrians by considering their trajectories, speed and crossing decision. In analogy with the results of the video-recorded observation performed in 2015 by the CSAI research center (Gorrini, Vizzari, and Bandini 2016), data analysis will be focused on measuring:

- the *appraising phase of pedestrians*: the pedestrian approaching the cross-walk decelerates to evaluate the distance and speed of oncoming autonomous vehicle;
- their *crossing decision*: the pedestrian decides to cross and speed up;
- the *safety gap accepted by pedestrians to safely cross*: the relation between the distance and speed of oncoming autonomous vehicles, when pedestrians decided to cross.

Achieved results will be compared with data from the literature, to highlight differences between the crossing behaviour of pedestrians in case of interaction with autonomous and human-driven vehicles. The experimental design includes several procedures to test the impact of different variables on the crossing behaviour of pedestrians (e.g., different braking patterns of the vehicles in terms of deceleration magnitude and distance from the pedestrian; presence of a driver supervising the autonomous vehicles; presence of the passenger only). Moreover, the experiments will aim at testing the effectiveness of a breaking light actuator on the front of the vehicle as a visual communication system to support the crossing decision of pedestrians. The final aim of the experiments is to collect useful insights to support the further development of the vehicles, through the exploitation and integration of the above described pedestrian ontology into the autonomous vehicle movement planning model, in order to improve its capability to interpret the scene at real-time.

Conclusions

The paper refers to an multidiscplinary research conducted at University of Milano-Bicocca towards the integration of available findings and knowledge about pedestrian dynamics at unsignalized pedestrian crossing to improve quality of scene interpretation, communication, and trustability of vehicle actions into the movement planner of the IRA lab autonomous vehicle.

The paper has presented the pedestrian ontology (in its T-Box and A-Box components) we developed to collect into a well-formed, machine readable and semantically rich framework part of available knowledge about behavioral studies findings about pedestrian dynamics. Vehicle-pedestrian interactions at UPCs is the reference case study here described. The adoption of an ontology as knowledge representation framework is due to advatages provided by any (formal, in the sense of semantically well-defined) tool to manage explicit knowledge. Evaluations about computational costs of pedestrian knowledge ontology implementation will be object of future investigation.

A possible direction towards the development of a learning behavior of the vehicle in interaction situations with pedestrians may consider several approaches as reasonable: Goal Reasoning (Wilson et al.), or Case-Based Reasoning (CBR (Kolodner 1993)), like for instance in (Floyd and Aha 2017) where stored information about previously adapted behaviors is used with the aim of taking into account human trustworthy of previous plan adaptations, is one of our preliminary preferred directions for further investigations.

Acknowledgements

The authors thank Alberto Castagna and Alessio Marella for their fruitful contribution to the first prototype of pedestrians ontology.

References

Armand, A. 2016. *Situation understanding and risk assessment framework for preventive driver assistance*. Ph.D. Dissertation, Université Paris-Saclay.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Bennewitz, M.; Burgard, W.; Cielniak, G.; and Thrun, S. 2005. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research* 24(1):31–48.

Chakraborti, T.; Dudley, A.; and Kambhampati, S. 2018. v2v communication for augmenting reality enabled smart huds to increase situational awareness of drivers. *VAM-HRI 2018*.

Chen, Z.; Ngai, D.; and Yung, N. 2008. Pedestrian behavior prediction based on motion patterns for vehicle-to-pedestrian collision avoidance. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, 316–321. IEEE.

Dommès, A.; Le Lay, T.; Vienne, F.; Dang, N.-T.; Beaudoin, A. P.; and Cuong, D. M. 2015. Towards an explanation of age-related difficulties in crossing a two-way street. *Accident Analysis & Prevention* 85:229–238.

Evans, D., and Norman, P. 1998. Understanding pedestrians' road crossing decisions: an application of the theory of planned behaviour. *Health Education Research* 13(4):481–489.

Federici, M. L.; Manenti, L.; and Manzoni, S. 2014. A checklist for the evaluation of pedestrian simulation software functionalities. *arXiv preprint arXiv:1404.7717*.

Feld, M., and Muller, C. 2011. The automotive ontology: managing knowledge inside the vehicle and sharing it between cars. 79–86.

Ferber, J. 1999. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Floyd, M. W., and Aha, D. W. 2017. Using explanations to provide transparency during trust-guided behavior adaptation 1. *AI Communications* 30(3-4):281–294.

Fox, M.; Long, D.; and Magazzeni, D. Explainable Planning. *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*.

Gorrini, A.; Vizzari, G.; and Bandini, S. 2016. Towards modelling pedestrian-vehicle interactions: Empirical study on urban unsignalized intersection. *arXiv preprint arXiv:1610.07892*.

Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.* 43(5-6):907–928.

Hamed, M. M. 2001. Analysis of pedestrians behavior at pedestrian crossings. *Safety science* 38(1):63–82.

Harrell, A. W. 1991. Factors influencing pedestrian cautiousness in crossing streets. *The Journal of Social Psychology* 131(3):367–372.

Kolodner, J. 1993. Case-based reasoning (morgan kaufmann series in representation and reasoning). *Morgan Kauffmann*.

Kulkarni, A.; Zha, Y.; Chakraborti, T.; Vadlamudi, S. G.; Zhang, Y.; and Kambhampati, S. Explicability as minimizing distance from expected behavior. *arXiv 1611.05497*.

Mizoguchi, R., and Ikeda, M. 1998. *Journal-Japanese Society for Artificial Intelligence* 13:9–10.

Mohammad, M. 2015. Ontology-based framework for risk assessment in road scenes using videos.

Nishino, N.; Tsugita, R.; Chugo, D.; Yokota, S.; Muramatsu, S.; and Hashimoto, H. 2016. Robot navigation according to the characteristics of pedestrian flow. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 5947–5952.

Nonaka, S.; Inoue, K.; Arai, T.; and Mae, Y. 2004. Evaluation of human sense of security for coexisting robots using virtual reality. 1st report: evaluation of pick and place motion of humanoid robots. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, 2770–2775. IEEE.

NT. Dang, V. C.; Thomson, J.; Gorrini, A.; Piwowarczyk, A.; Vienne, F.; Bandini, S.; and Pierre, G. S. 2016. Crossing the street in dyads: Social influence on pedestrian behaviour in an immersive virtual environment. In *Proceedings of the International Conference on Road Safety and Simulation International Conference - RSS2017, 17-19 October, 2017, Delft, Netherlands, in press, 2017*.

Pacchierotti, E.; Christensen, H. I.; and Jensfelt, P. 2005. Human-robot embodied interaction in hallway settings: a pilot user study. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, 164–171.

Perumal, V., et al. 2014. Study on pedestrian crossing behavior at signalized intersections. *Journal of traffic and transportation engineering (English edition)* 1(2):103–110.

Sisiopiku, V. P., and Akin, D. 2003. Pedestrian behaviors at and perceptions towards various pedestrian facilities: an examination based on observation and survey data. *Transportation Research Part F: Traffic Psychology and Behaviour* 6(4):249–274.

Wilson, M. A.; McMahon, J.; Wolek, A.; Aha, D. W.; and Houston, B. H. Goal reasoning for autonomous underwater vehicles: Responding to unexpected agents. *AI Communications* (Preprint):1–16.

World Health Organization. 2015. *Global status report on road safety 2015*. World Health Organization.

Zhao, L.; Ichise, R.; Mita, S.; and Sasaki, Y. 2015. An ontology-based intelligent speed adaptation system for autonomous cars. In Supnithi, T.; Yamaguchi, T.; Pan, J. Z.; Wuwongse, V.; and Buranarach, M., eds., *Semantic Technology*, 397–413. Cham: Springer International Publishing.

Visualizations for an Explainable Planning Agent

Tathagata Chakraborti[1] and **Kshitij P. Fadnis**[2] and **Kartik Talamadupula**[2] and **Mishal Dholakia**[2]
Biplav Srivastava[2] and **Jeffrey O. Kephart**[2] and **Rachel K. E. Bellamy**[2]

[1] Computer Science Department, Arizona State University, Tempe, AZ 85281 USA
tchakra2 @ asu.edu

[2]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA
{ kpfadnis, krtalamad, mdholak, biplavs, kephart, rachel } @ us.ibm.com

Abstract

In this paper, we report on the visualization capabilities of an Explainable AI Planning (XAIP) agent that can support human in the loop decision making. Imposing transparency and explainability requirements on such agents is especially important in order to establish trust and common ground with the end-to-end automated planning system. Visualizing the agent's internal decision making processes is a crucial step towards achieving this. This may include externalizing the "brain" of the agent – starting from its sensory inputs, to progressively higher order decisions made by it in order to drive its planning components. We also show how the planner can bootstrap on the latest techniques in explainable planning to cast plan visualization as a plan explanation problem, and thus provide concise model based visualization of its plans. We demonstrate these functionalities in the context of the automated planning components of a smart assistant in an instrumented meeting space.

Introduction

Advancements in the fields of speech, language, and search have led to ubiquitous personalized assistants like the Amazon Echo, Google Home, Apple Siri, etc. Even though these assistants have mastered a narrow category of interaction in specific domains, they mostly operate in passive mode – i.e. they merely respond via a set of predefined scripts, most of which are written to specification. In order to evolve towards truly smart assistants, the need for (pro)active collaboration and decision support capabilities is paramount. Automated planning offer a promising alternative to this drudgery of repetitive and scripted interaction. The use of planners allows automated assistants to be imbued with the complementary capabilities of being nimble and proactive on the one hand, while still allowing specific knowledge to be coded in the form of domain models. Additionally, planning algorithms have long excelled (Myers 1996; Sengupta et al. 2017) in the presence of humans in the loop for complex collaborative decision making tasks.

eXplainable AI Planning (XAIP) While planners have always adapted to accept various kinds of inputs from humans, only recently has there been a concerted effort on the other side of the problem: making the outputs of the planning process more palatable to human decision makers. The paradigm of eXplainable AI Planning (XAIP) (Fox, Long, and Magazzeni 2017) has become a central theme around which much of this research has coalesced. In this paradigm, emphasis is laid on the qualities of *trust*, *interaction*, and *transparency* that an AI system is endowed with. The key contributions to explainability are the resolution of critical exploratory questions – why did the system do something a particular way, why did it not do some other thing, why was its decision optimal, and why the evolving world may force the system to replan.

Role of Visualization in XAIP One of the keys towards achieving an XAIP agent is visualization. The planning community has recently made a concerted effort to support the visualization of key components of the end-to-end planning process: from the modeling of domains (Bryce et al. 2017); to assisting with plan management (Izygon, Kortenkamp, and Molin 2008); and beyond (Sengupta et al. 2017; Benton et al. 2017). For an end-to-end planning system, this becomes even more challenging since the systems state is determined by information at different levels of abstraction which are being coalesced in the course of decision making. A recent workshop (Freedman and Frank 2017) outlines these challenges in a call to arms to the community on the topic of visualization and XAIP.

Contribution It is in this spirit that we present a set of visualization capabilities for an XAIP agent that assists with human in the loop decision making tasks: specifically in the case of this paper, assistance in an instrumented meeting space. We introduce the end-to-end planning agent, `Mr.Jones` (Chakraborti et al. 2017b), and the visualizations that we endow it with. We then provide fielded demonstrations of the visualizations, and describe the details that lie under the hood of these capabilities.

Introducing `Mr.Jones`

First, we introduce `Mr.Jones` (Chakraborti et al. 2017b), situated in the `CEL` – the Cognitive Environments Laboratory – at IBM's T.J. Watson Research Center. `Mr.Jones` is designed to embody the key properties of a proactive assistant while fulfilling the properties desired of an XAIP agent.

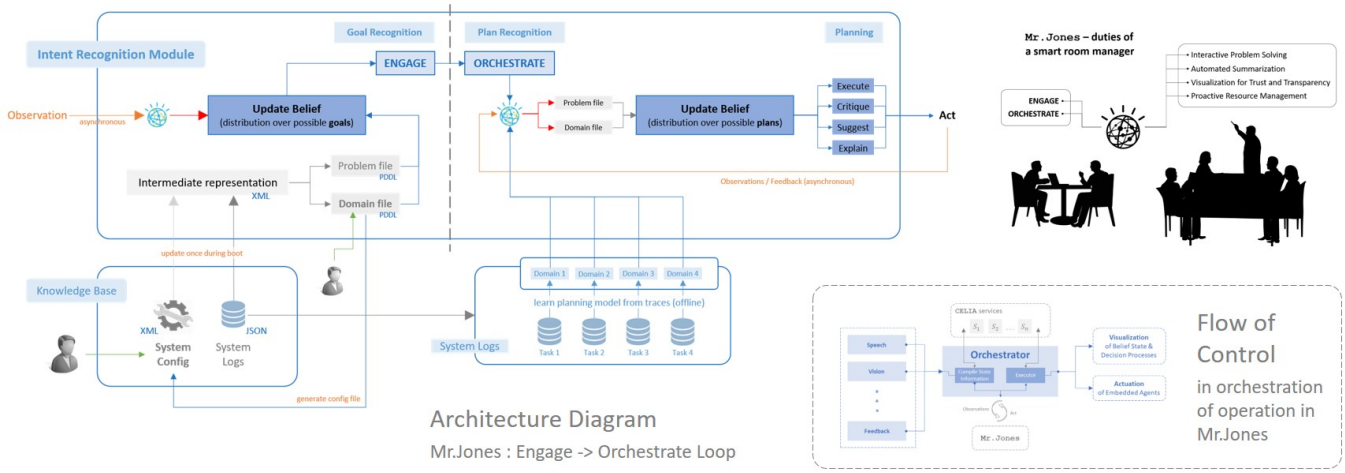


Figure 1: Architecture diagram illustrating the building blocks of `Mr.Jones` – the two main components *Engage* and *Orchestrate* situates the agent proactively in a decision support setting with human decision makers in the loop. The top right inset shows the different roles of `Mr.Jones` as a smart room orchestrator and meeting facilitator. The bottom right inset illustrates the flow of control in `Mr.Jones` – each service runs in parallel and asynchronously to maintain anytime response of all the individual components.

`Mr.Jones`: An end-to-end planning system

We divide the responsibilities of `Mr.Jones` into two processes – ***Engage***, where plan recognition techniques are used to identify the task in progress; and ***Orchestrate***, which involves active participation in the decision-making process via real-time plan generation, visualization, and monitoring.

ENGAGE This consists of `Mr.Jones` monitoring various inputs from the world in order to situate itself in the context of the group interaction. First, the assistant gathers various inputs like speech transcripts, live images, and the positions of people within a meeting space; these inputs are fed into a higher level symbolic reasoning component. Using this, the assistant can (1) *requisition* resources and services that may be required to support the most likely tasks based on its recognition; (2) *visualize* the decision process – this can depict both the agent's own internal recognition algorithm, and an external, task-dependent process; and (3) *summarize* the group decision-making process.

ORCHESTRATE This process is the decision support assistant's contribution to the group's collaboration. This can be done using standard planning techniques, and can fall under the aegis of one of four actions as shown in Figure 1. These actions, some of which are discussed in more detail in (Sengupta et al. 2017), are: (1) *execute*, where the assistant performs an action or a series of actions related to the task at hand; (2) *critique*, where the assistant offers recommendations on the actions currently in the collaborative decision sequence; (3) *suggest*, where the assistant suggests new decisions and actions that can be discussed collaboratively; and (4) *explain*, where the assistant explains its rationale for adding or suggesting a particular decision. The *Orchestrate* process thus provides the "support" part of the decision support assistant. The *Engage* and *Orchestrate* processes can be seen as somewhat parallel to the *interpretation* and *steering* processes defined in the crowdsourcing scenarios of (Talamadupula et al. 2013; Manikonda et al. 2017). The difference in these new scenarios is that the humans are the final decision makers, with the assistant merely supporting the decision making.

Architecture Design & Key Components The central component – the Orchestrator[1] – regulates the flow of information and control flow across the modules that manage the various functionalities of the `CEL`; this is shown in Figure 1. These modules are mostly asynchronous in nature and may be: (1) services[2] processing sensory information from various input devices across different modalities like audio (microphone arrays), video (PTZ cameras / Kinect), motion sensors (Myo / Vive) and so on; (2) services handling the different services of `CEL`; and (3) services that attach to the `Mr.Jones` module. The Orchestrator is responsible for keeping track of the current state of the system as well as coordinating actuation either in the belief/knowledge space, or in the actual physical space.

Knowledge Acquisition / Learning The knowledge contained in the system comes from two sources – (1) the developers and/or users of the service; and (2) the system's own memory; as illustrated in Figure 1. One significant barrier towards the adoption of higher level reasoning capabilities into such systems has been the lack of familiarity of developers and end users with the inner working of these technologies. With this in mind we provide an XML-based modeling interface – i.e. a *"system config"* – where users can easily configure new environments. This information in turn enables automatic generation of the files that are internally required by the reasoning engines. Thus system specific information is bootstrapped into the service specifications written by expert

[1] Not to be confused with the term *Orchestrate* from the previous section, used to describe the phase of active participation.

[2] Built on top of the Watson Conversation and Visual Recognition services on IBM Cloud and other IBM internal services.

developers, and this composite knowledge can be seamlessly transferred across task domains and physical configurations. The granularity of the information encoded in the models depends on the task at hand – for example, during the *Engage* phase, the system uses much higher level information (e.g. identities of agents in the room, their locations, speech intents, etc.) than during the *Orchestrate* phase, where more detailed knowledge is needed. This enables the system to reason at different levels of abstraction independently, thus significantly improving the scalability as well as robustness of the recognition engine.

Plan Recognition The system employs the probabilistic goal / plan recognition algorithm from (Ramirez and Geffner 2010) to compute its beliefs over possible tasks. The algorithm casts the plan recognition problem as a planning problem by compiling away observations to the form of actions in a new planning problem. The solution to this new problem enforces the execution of these observation-actions in the observed order. This explainsmini the reasoning process behind the belief distribution in terms of the possible plans that the agent envisioned (as seen in Figure 2).

Plan Generation The `FAST-DOWNWARD` planner (Helmert 2006) provides a suite of solutions to the forward planning problem. The planner is also required internally by the Recognition Module when using the compilation from (Ramirez and Geffner 2010), or in general to drive some of the orchestration processes. The planner reuses the compilation from the Recognition Module to compute plans that preserve the current (observed) context.

Visualizations in `Mr.Jones`

The `CEL` is a smart environment, equipped with various sensors and actuators to facilitate group decision making. Automated planning techniques, as explained above, are the core component of the decision support capabilities in this setting. However, the ability to plan is rendered insufficient if the agent cannot communicate that information effectively to the humans in the loop. Dialog as a means of interfacing with the human decision makers often becomes clumsy due to the difficulty of representing information in natural language, and/or the time taken to communicate. Instead, we aim to build visual mediums of communication between the planner and the humans for the following key purposes –

- *Trust & Transparency* - Externalizing the various pathways involved in the decision support process is essential to establish trust between the humans and the machine, as well as to increase situational awareness of the agents. It allows the humans to be cognizant of the internal state of the assistant, and to infer decision rationale, thereby reducing their cognitive burden.
- *Summarization of Minutes* - The summarization process is a representation of the beliefs of the agent with regard to what is going on in its space over the course of an activity. Since the agent already needs to keep track of this information in order to make its decisions, we can replay or sample from it to generate an automated visual summary of (the agent's belief of) the proceedings in the room.

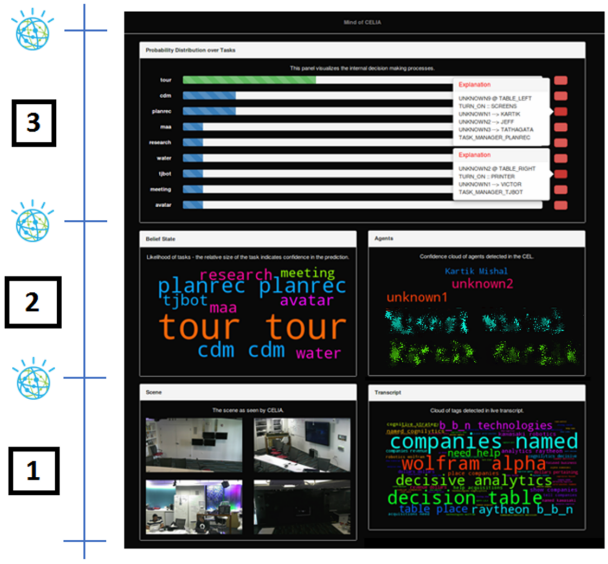


Figure 2: Snapshot of the mind of `Mr.Jones` externalizing different stages of its cognitive processes.

- *Decision Making Process* - Finally, and perhaps most importantly, the decision making process itself needs efficient interfacing with the humans – this can involve a range of things from showing alternative solutions to a task, to justifying the reasoning behind different suggestions. This is crucial in a mixed initiative planning setting (Horvitz 1999; 2007) to allow for human participation in the planning process, as well as for the planner's participation in the humans' decision making process.

Mind of `Mr.Jones`

First, we will describe the externalization of the "mind" of `Mr.Jones` – i.e. the various processes that feed the different capabilities of the agent. A snapshot of the interface is presented in Figure 2. The interface itself consists of five widgets. The largest widget on the top shows the various usecases that the `CEL` is currently set up to support. In the current `CEL` setup, there are nine such usecases. The widget represents the probability distribution that indicates the confidence of `Mr.Jones` in the respective task being the one currently being collaborated on, along with a button for the provenance of each such belief. The information used as provenance is generated directly from the plans used internally by the recognition module (Ramirez and Geffner 2010) and justifies why, given its model of the underlying planning problems, these tasks look likely in terms of plans that achieve those tasks. Model based algorithms are especially useful in providing explanations like this (Sohrabi, Baier, and McIlraith 2011; Fox, Long, and Magazzeni 2017). The system is adept at handling uncertainty in its inputs (it is interesting to note that in coming up with an explanatory plan it has announced likely assignments to unknown agents in its space). In Figure 2, `Mr.Jones` has placed the maximum confidence in the `tour` usecase.

Below the largest widget is a set of four widgets, each

of which give users a peek into an internal component of Mr.Jones. The first widget, on the top left, presents a wordcloud representation of Mr.Jones's belief in each of the tasks; the size of the word representing that task corresponds to the probability associated with that task. The second widget, on the top right, shows the agents that are recognized as being in the environment currently – this information is used by the system to determine what kind of task is more likely. This information is obtained from four independent camera feeds that give Mr.Jones an omnispective view of the environment; this information is represented via snapshots (sampled at 10-20 Hz) in the third widget, on the bottom left. In the current example, Mr.Jones has recognized the agents named (anonymized) "XXX" and "YYY" in the scenario. Finally, the fourth widget, on the bottom right, represents a wordcloud based summarization of the audio transcript of the environment. This transcript provides a succinct representation of the things that have been said in the environment in the recent past via the audio channels. Note that this widget is merely a summarization of the full transcript, which is fed into the IBM Watson Conversation service to generate observations for the plan recognition module. The interface thus provides a real-time snapshot of the various sensory and cognitive organs associated with Mr.Jones- the eyes, ears, and mind of the CEL. The interface is organized at increasing levels of abstraction –

[1] *Raw Inputs* – These show the camera feeds and voice capture (speech to text outputs) as received by the system. These help in externalizing what information the system is working with at any point of time and can be used, for example, in debugging at the input level if the system makes a mistake or in determining whether it is receiving enough information to make the right decisions. It is especially useful for an agent like Mr.Jones, which is not embodied in a single robot or interface but is part of the environment as a whole. As a result of this, users may find it difficult to attribute specific events to the agent.

[2] *Lower level reasoning* – The next layer deals with the first stage of reasoning over these raw inputs – What are the topics being talked about? Who are the agents in the room? Where are they situated? This helps an user identify what knowledge is being extracted from the input layer and fed into the reasoning engines. It increases the situational awareness of agents by visually summarizing the contents of the scene at any point of time.

[3] *Higher level reasoning* – Finally, the top layer uses information extracted at the lower levels to reason about abstract tasks in the scene. It visualizes the outcome of the plan recognition process, along with the provenance of the information extracted from the lower levels (agents in the scene, their positions, speech intents, etc.). This layer puts into context the agent's current understanding of the processes in the scene.

Demonstration 1 We now demonstrate how the *Engage* process evolves as agents interact in the CEL. The demonstration begins with two humans discussing the CEL environment, followed by one agent describing a projection of the Mind of Mr.Jones on the screen. The other agent then discusses how a Mergers and Acquisitions (M&A) task (Kephart and Lenchner 2015) is carried out. *A video of this demonstration can be accessed at* https://www.youtube.com/watch?v=ZEHxCKodEGs. The video contains a window that demonstrates the evolution of the Mr.Jones interface through the duration of the interaction. This window illustrates how Mr.Jones's beliefs evolve dynamically in response to interactions in real-time.

Demonstration 2 After a particular interaction is complete Mr.Jones can automatically compile a summarization (or minutes) of the meeting by sampling from the visualization of its beliefs. *An anonymized video of a typical summary can be accessed at* https://youtu.be/AvNRgsvuVOo. This kind of *visual summary* provides a powerful alternative to established meeting summarization tools like text-based minutes. The visual summary can also be used to extract abstract insights about this one meeting, or a set of similar meetings together and allows for agents that may have missed the meeting to catch up on the proceedings. Whilst merely sampling the visualization at discrete time-intervals serves as a powerful tool towards automated summary generation, we anticipate the use of more sophisticated visualization (Dörk et al. 2010) and summarization (Shaw 2017; Kim, Chacha, and Shah 2015; Kim and Shah 2016) techniques in the future.

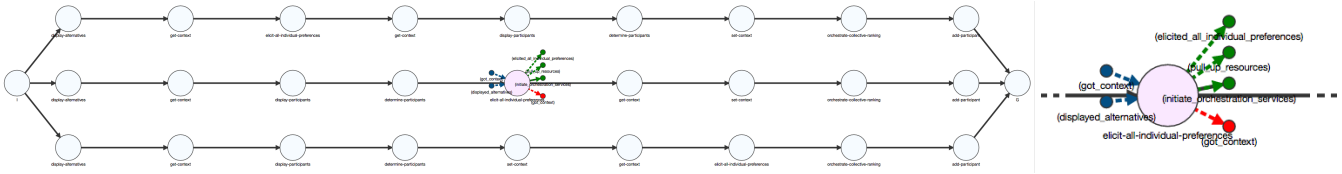
Model-Based Plan Visualization : Fresco

We start by describing the planning domain that is used in the rest of this section, followed by a description of Fresco's different capabilities in terms of *top-K plan visualization* and *model-based plan visualization*. We conclude by describing the implementation details on the back-end.

The Collective Decision Domain We use a variant of the Mergers and Acquisitions (M&A) task called *Collective Decision* (CD). The CD domain models the process of gathering input from a decision makers in a smart room, and the orchestration of comparing alternatives, eliciting preferences, and finally ranking of the possible options.

Top-K Visualization

Most of the automated planning technology and literature considers the problem of generating a single plan. Recently, however, the paradigm of Top-K planning (Riabov, Sohrabi, and Udrea 2014) has gained traction. Top-K plans are particularly useful in domains where producing and deliberating on multiple alternative plans that go from the same fixed initial state and the same fixed goal is important. Many decision support scenarios, including the one described above, are of this nature. Moreover, Top-K plans can also help in realizing unspecified user preferences, which may be very hard to model explicitly. By presenting the user(s) with multiple alternatives, an *implicit* preference elicitation can instead be performed. The Fresco interface supports visualization of the K top plans for a given problem instance and domain model, as shown in Figure 3a. In order to generate the Top-K plans, we use an experimental Top-K planner (Katz et al. 2018) that is built on top of Fast Downward (Helmert 2006).



(a) Top-K plan visualization showing alternative plans for a given problem. (b) Action Descriptions

Figure 3: Visualization of plans in `Fresco` showing top-K alternative solutions (K=3) for a given planing problem (left) and on-demand visualization of each action in the plan (zoomed-in; right) in terms of causal links consumed and produced by it.

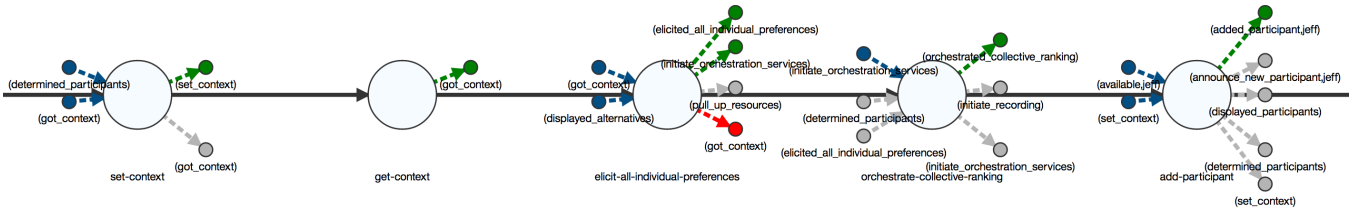


Figure 4: Visualization as a process of explanation – minimized view of conditions relevant to a plan. Blue, green and red nodes indicate preconditions, add and delete effects respectively. The conditions which are not necessary causes for this plan (i.e. the plan is still optimal in a domain without these conditions) are grayed out in the visualization (11 out of a total 30).

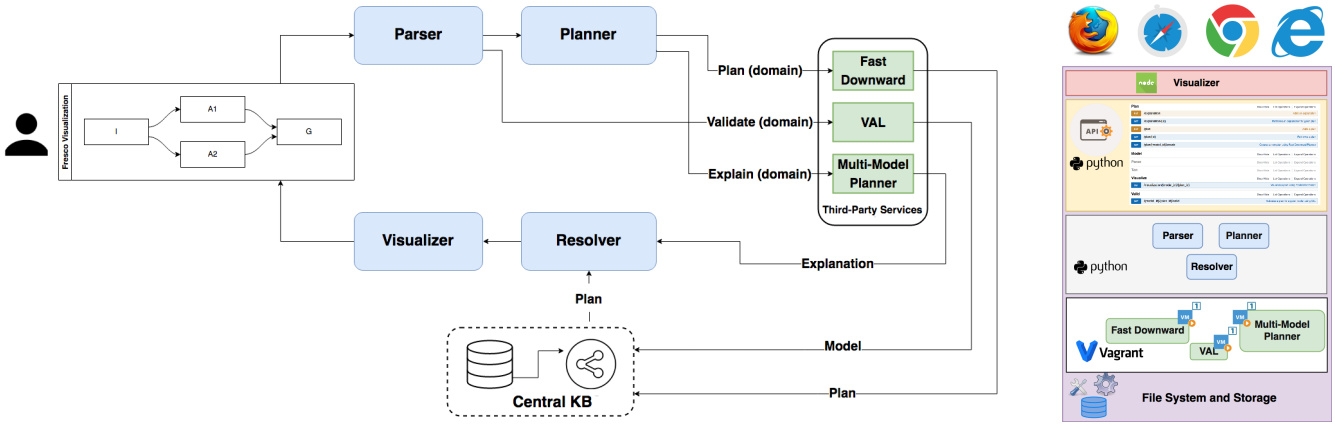
Model-based Plan Visualization

The requirements for visualization of plans can have different semantics depending on the task at hand – e.g. showing the search process that produced the plan, and the decisions taken (among possible alternative solutions) and trade-offs made (by the underlying heuristics) in that process; or revealing the underlying domain or knowledge base that engendered the plan. The former involves visualizing the *how* of plan synthesis, while the latter focuses on the *why*, and is model-based and algorithm independent. Visualizing the how is useful to the developer of the system during debugging, but serves little purpose for the end user who would rather be told the rationale behind the plan: why is this plan better than others, what individual actions contribute to the plan, what information is getting consumed at each step, and so on. Unfortunately, much of the visualization work in the planning community has been confined to depicting the search process alone (Thayer 2010; 2012; Magnaguagno et al. 2017). `Fresco`, on the other hand, aims to focus on the *why* of a plan's genesis, in the interests of establishing common ground with human decision-makers. At first glance, this might seems like an easy problem – we could just show what the preconditions and effects are for each action along with the causal links in the plan. However, even for moderately sized domains, this turns into a clumsy and cluttered approach very soon, given the large number of conditions to be displayed. In the following, we will describe how `Fresco` handles this problem of overload.

Visualization as a Process of Explanation We begin by noting that the process of visualization can in fact be seen as a *process of explanation*. In model-based visualization, as described above, the system is essentially trying to explain to the viewer the salient parts of its knowledge that contributed to this plan. In doing so, it is externalizing what each action is contributing to the plan, as well as outlining why this action is better that other possible alternatives.

Explanations in Multi-Model Planning Recent work has shown (Chakraborti et al. 2017a) how an agent can explain its plans to the user when there are differences in the models (of the same planning problem) of the planner and the user, which may render an optimal plan in the planner's model sub-optimal or even invalid–and hence unexplainable–in the user's mental model. An explanation in this setting constitutes a *model update* to the human such that the plan (that is optimal to the planner) in question also becomes optimal in the user's updated mental model. This is referred to as a *model reconciliation process* (MRP). The smallest explanation is called *minimally complete* (MCE).

Model-based Plan Visualization $\equiv$ Model Reconciliation with Empty Model As we mentioned previously, exposing the entire model to the user is likely to lead to cognitive overload and lack of situational awareness due to the amount of information that is not relevant to the plan in question. We want to minimize the clutter in the visualization and yet maintain all relevant information pertaining to the plan. *We do this by launching an instantiation of the model reconciliation process with the planner's model and an empty model as inputs.* An empty model is a copy of the given model where actions do not have any conditions and the initial state is empty (the goal is still preserved). Following from the above discussion, the output of this process is then the minimal set of conditions in the original model that ensure optimality of the given plan. In the visualization, the rest of the conditions from the domain are grayed out. (Chakraborti et al. 2017a) showed how this can lead to a significant pruning of conditions that do not contribute to the generation of a particular



(a) Architecture diagram of `Fresco`.

(b) Software stack

Figure 5: Illustration of the flow of control (left) in `Fresco` between the plan generator (FD), explanation generator (MMP), and plan validator (VAL) with the visualization modules. The MMP code base is in the process of being fully integrated into `Fresco`, and it is currently run as a stand-alone component. The software stack (right) shows the infrastructure supporting `Fresco` in the backend.

plan. An instance of this process on the CD domain is illustrated in Figure 4.

Note that the above may not be the only way to minimize information being displayed. There might be different kinds of information that the user cares about, depending on their preferences. This is also highlighted by the fact that an MCE is not unique for a given problem. These preferences can be *learned* in the course of interactions.

Architecture of `Fresco` The architecture of `Fresco`, shown in Figure 5a, includes several core modules such as the parser, planner, resolver, and visualizer. These modules are all connected in a feed-forward fashion. The parser module is responsible for converting domain models and problem instances into python objects, and for validating them using VAL (Howey, Long, and Fox 2004). Those objects are then passed on to the planner module, which relies on Fast-Downward (FD) and the Multi-Model Planner (MMP) (Chakraborti et al. 2017a) to generate a plan along with its explanation. The resolver module consumes the plan, the explanation, and the domain information to not only ground the plan, but also to remove any preconditions, add, or delete effects that are deemed irrelevant by the MMP module. Finally, the visualizer module takes the plan from the resolver module as an input, and builds graphics that can be rendered within any well-known web browser. Our focus in designing the architecture was on making it functionally modular and configurable, as shown in Figure 5b. While the first three modules described above are implemented using Python, the visualizer module is implemented using Javascript and the `D3` graphics library. Our application stack uses `REST` protocols to communicate between the visualizer module and the rest of the architecture. We also accounted for scalability and reliability concerns by containerizing the application with `Kubernetes`, in addition to building individual containers / virtual machines for third party services like VAL, Fast-Downward, and MMP.

Work in Progress

While we presented the novel notion of *explanation as visualization* in the context of AI planning systems in this paper via the implemention of the `Mr.Jones` assistant, there is much work yet to be done to embed this as a central research topic in the community. We conclude the paper with a brief outline of future work as it relates to the visualization capabilities of `Mr.Jones` and other systems like it.

Visualization for Model Acquisition Model acquisition is one of the biggest impediments towards the adoption of planning technologies. Our own work with `Mr.Jones` is not immune to this problem. Although we have enabled an XML-based modeling interface, the next iteration of making this easily consumable for non-experts involves two steps: first, we impose an (possibly graphical) interface on top of the XML structure to obtain information in a structured manner. We can thenl provide visualizations such as those described in (Bryce et al. 2017) in order to help with iterative acquisition and refinement of the planning model.

Tooling Integration Eventually, our vision – not restricted to any one planning tool or technology – is to integrate the capabilities of `Fresco` into a domain-independent planning tool such as `planning.domains` (Muise 2016), which will enable the use of these visualization components across various application domains. `planning.domains` realizes the long-awaited planner-as-a-service paradigm for end users, but is yet to incorporate any visualization techniques for the user. Model-based visualization from `Fresco`, complemented with search visualizations from emerging techniques like `WebPlanner` (Magnaguagno et al. 2017), can be a powerful addition to the service.

Mixed-Reality Finally, recent advances in mixed-reality technologies provide exciting opportunities for newer modes of interaction with AI agents (Williams et al. 2018). We explore such capabilities in the space of decision-support in (Sengupta, Chakraborti, and Kambhampati 2018).

Acknowledgements A significant part of this work was initiated and completed while Tathagata Chakraborti was an intern at IBM's T. J. Watson Research Center during the summer of 2017. The continuation of his work at ASU is supported by an IBM Ph.D. Fellowship.

References

Benton, J.; Smith, D.; Kaneshige, J.; and Keely, L. 2017. CHAP-E: A plan execution assistant for pilots. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning*, UISP 2017, 1–7.

Bryce, D.; Bonasso, P.; Adil, K.; Bell, S.; and Kortenkamp, D. 2017. In-situ domain modeling with fact routes. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning*, UISP 2017, 15–22.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017a. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*.

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

Dörk, M.; Gruen, D.; Williamson, C.; and Carpendale, S. 2010. A Visual Backchannel for Large-Scale Events. *IEEE Transactions on Visualization and Computer Graphics*.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *First IJCAI Workshop on Explainable AI (XAI)*.

Freedman, R. G., and Frank, J. D., eds. 2017. *Proceedings of the First Workshop on User Interfaces and Scheduling and Planning*. AAAI.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 159–166. ACM.

Horvitz, E. J. 2007. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine* 28(2):3.

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

Izygon, M.; Kortenkamp, D.; and Molin, A. 2008. A procedure integrated development environment for future spacecraft and habitats. In *Space Technology and Applications International Forum*.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Kephart, J. O., and Lenchner, J. 2015. A symbiotic cognitive computing perspective on autonomic computing. In *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, 109–114.

Kim, J., and Shah, J. A. 2016. Improving team's consistency of understanding in meetings. *IEEE Transactions on Human-Machine Systems* 46(5):625–637.

Kim, B.; Chacha, C. M.; and Shah, J. A. 2015. Inferring team task plans from human meetings: A generative modeling approach with logic-based prior. *Journal of Artificial Intelligence Research*.

Magnaguagno, M. C.; Pereira, R. F.; Móre, M. D.; and Meneguzzi, F. 2017. Web planner: A tool to develop classical planning domains and visualize heuristic state-space search. *ICAPS 2017 User Interfaces for Scheduling & Planning (UISP) Workshop*.

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

Muise, C. 2016. Planning.Domains. In *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*.

Myers, K. L. 1996. Advisable planning systems. *Advanced Planning Technology* 206–209.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Riabov, A.; Sohrabi, S.; and Udrea, O. 2014. New algorithms for the top-k planning problem. In *Proceedings of the Scheduling and Planning Applications woRKshop (SPARK) at the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 10–16.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2017. RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning. In *AAAI Fall Symposium on Human-Agent Groups*.

Sengupta, S.; Chakraborti, T.; and Kambhampati, S. 2018. MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making. *ICAPS UISP*.

Shaw, D. 2017. How Wimbledon is using IBM Watson AI to power highlights, analytics and enriched fan experiences. `https://goo.gl/r6z3uL`.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *AAAI*.

Talamadupula, K.; Kambhampati, S.; Hu, Y.; Nguyen, T.; and Zhuo, H. H. 2013. Herding the crowd: Automated planning for crowdsourced planning. In *HCOMP*.

Thayer, J. 2010. Search Visualizations. `https://www.youtube.com/user/TheSuboptimalGuy`.

Thayer, J. T. 2012. Heuristic search under time and quality bounds. *Ph. D. Dissertation, University of New Hampshire*.

Williams, T.; Szafir, D.; Chakraborti, T.; and Ben Amor, H. 2018. Virtual, augmented, and mixed reality for human-robot interaction. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 403–404. ACM.

Towards Explanation-Supportive Knowledge Engineering for Planning

Mauro Vallati and **Thomas L. McCluskey**
University of Huddersfield
Huddersfield, UK

Lukáš Chrpa
Czech Technical University &
Charles University in Prague, Prague, CZ

Abstract

In real-world planning applications, it is of pivotal importance that decisions and solutions can be explained, in order to maximise safety and increase the reliability of the planning component. There is a significant strand of research that focuses on exploiting planning models, and the planning process itself, in order to deal with a wide range of possible requests of explanation. However, there are explanations that can not be provided by considering only those elements, but that require additional knowledge –which is usually available during the knowledge engineering process of designing planning models.

In this paper, we introduce two classes of explanations that require a formal (or semi-formal) encoding of additional knowledge. We then describe how knowledge engineering tools can be extended in order to support the collection of such additional knowledge, and we briefly introduce the structure of an appropriate architecture.

Introduction

Automated planning has been successfully applied for decades, and is gaining importance, in several areas, including space exploration (Ai-Chang et al. 2004), machine tool calibration (Parkinson and Longstaff 2015), and urban traffic control (McCluskey and Vallati 2017) to mention a few. In applications, particularly in safety-critical ones, it is of paramount importance that generated solutions can be thoroughly explained and described. This is usually termed as *explainability*, and allows to increase the reliability of the planning component, and can support the validation of models and of automated reasoning.

There has been a growing strand of research focusing on explainability in automated planning. Works in the area cover the explanation and description of generated plans (Sohrabi, Baier, and McIlraith 2011; Caminada et al. 2014). A more recent line of work is focused on explaining also the planning process, by motivating and describing decisions taken by the planner during the exploration of the search space (Fox, Long, and Magazzeni 2017). This sort of analysis allows to provide answers to questions like *Why is this action performed? Why this other action has been ignored?*

Mentioned approaches are mainly focused on explaining plans and planning processes, and do not "question" the provided domain model that is used as the basis of the reasoning process. Therefore, any provided explanation is bounded to the model. In other words, a user would not be able to ask –or to get satisfying answers to– questions that involves a critical analysis of the model, but only questions about the search space that the considered model generates. On this matter, the work of Chakraborti et al. (2017) proposes to exploit explanations for dealing with the model reconciliation problem. Their approach takes explicitly into account models, and exploits explanation in order to minimise the divergence between the human mental model and the actual PDDL domain model. This can be done by considering only optimal plans. In a nutshell, if optimally generated plans are not self-explanatory for a human domain expert, than the domain model used by the planning framework does not correspond well with the mental model of the expert, and needs therefore to be refined. In this sense, explanations are not aimed at a final user of the planning framework, but at an expert that is "optimising" the system before deployment, or for maintenance purposes.

Knowledge planning models play a central role in any planning application, and their importance has been well-argued (McCluskey, Vaquero, and Vallati 2017). During the Knowledge Engineering (KE) process of encoding specifications into an appropriate domain model, a number of decisions are usually taken by experts. Some of those decisions are due to the knowledge of the encoder, while others are due to common knowledge about the application "context". Such knowledge is usually not stored anywhere, but can provide valuable information for explaining both plans and the planning process to stakeholders. In this paper, we argue about the importance of context awareness and annotation of decisions made by domain model encoders for improving the quality of the planning and plans explanation. We categorise the extended set of questions that can be answered by exploiting such knowledge. Finally, we propose the overall architecture that would allow to capture relevant knowledge, and analyse it in order to provide suitable explanations.

Knowledge Engineering for Planning

Knowledge Engineering in Planning and Scheduling (KEPS) was defined in the 2003 PLANET Roadmap (McCluskey et al. 2003), specifically for domain-independent planners, as the collection of processes involving (i) the acquisition, validation and verification, and maintenance of

planning domain models, (ii) the selection and optimisation of appropriate planning machinery, and (iii) the integration of (i) and (ii) to form planning and scheduling applications. KEPS can be seen as a special case of knowledge engineering, where the need for methodologies for acquiring, domain modelling and managing formally captured knowledge has long been accepted. It is also related to the area of capturing conceptual knowledge and developing domain models for Qualitative Reasoning in the general Modelling and Simulation area (Bredeweg et al. 2008). However, the peculiarities of automated planning applications distinguish KEPS from general knowledge-based and simulation systems. Firstly, KEPS is concerned with the acquisition and representation of knowledge about actions, processes, events, and the effect these have on a state. Secondly, this knowledge is to be used to create a system that synthesises plans rather than making a diagnosis or decision.

Studies on KEPS have led to the creation of several tools and techniques to support the design of domain knowledge structures, and the use of planners for real-world problems. Most of these tools have been presented in specialised workshops such as the *Knowledge Engineering for Planning & Scheduling* workshop and the *Verification and Validation of Planning Systems* workshop, as well as competitions such as the *International Competition on Knowledge Engineering for Planning & Scheduling* (Chrpa et al. 2017). The competitions have motivated the development of powerful KEPS systems and advances in domain modelling techniques, languages and analysis approaches. Well-known examples of existing tools include GIPO (Simpson, Kitchin, and McCluskey 2007), itSimple (Vaquero et al. 2012), KEWI (Wickler, Chrpa, and McCluskey 2014), PDDL-studio (Plch et al. 2012), and most recently the `planning.domains` framework.

Enhancing the Knowledge Engineering Process to Support Explainability

In order to describe how explainability can be supported also by the KE process, let us consider a variant of the well known Blocksworld domain as a running example. The main difference from the well-known classical planning domain, in our example there are two types of objects, categorised according to their shape: cubes and cones. Figure 1 shows a possible PDDL encoding of two operators from the domain model, that allow to move objects from an initial cube to another one. The name of the other operators are listed, but their details are not given due to the lack of space.

In a real-world planning application, the domain model is usually not shown to the final user, also because the typical user does not have a strong background in automated planning, and is usually not interested in such details. It is therefore reasonable to assume that the user will reason, and require explanations, on the basis of provided solution plans. For this reason, on the basis of the operators shown in Figure 1, let us consider the following problem, described in terms of initial and goal states, and a (very short) solution plan. For the sake of readability, predicates corresponding to objects stacked on the same tower are shown on a single line. In the

```
(:action cube-to-cube
:parameters (?b1  ?b2  ?b3 - cube)
:condition (and  (on ?b1 ?b2) (clear ?b1) (clear ?b3))
:effect (and (not (on ?b1 ?b2)) (on ?b1 ?b3)
  (not (clear ?b3)) (clear ?b2) ))

(:action cone-to-cube
:parameters (?b1  ?b2 - cube ?b3 - cone)
:condition (and (on ?b3 ?b1) (clear ?b2) (clear ?b3))
:effect (and (not (on ?b3 ?b1)) (on ?b3 ?b2)
  (not (clear ?b2)) (clear ?b1) ))

(:action cone-to-table ... )
(:action cube-to-table ... )
(:action cone-from-table ... )
(:action cube-from-table ... )
```

Figure 1: A possible encoding of a variant of BlocksWorld, considering cube and cone blocks.

solution plan, the first number indicates the time step.

```
initial state:
    (on-table cube1) (on cone1 cube1) (clear cone1)
    (on-table cube2) (on cube3 cube2) (clear cube3)
    (on-table cube4) (clear cube4)
    (on-table cube5) (clear cube5)

goal:
    (on cube2 cube1)
    (on cube5 cube3)

plan:
    [0] (cone-to-table cone1 cube1)
    [1] (cube-to-cube cube3 cube2 cube4)
    [2] (table-to-cube cube2 cube1)
    [2] (table-to-cube cube5 cube3)
```

Remarkably, the Explainable planning framework proposed by Fox, Long, and Magazzeni (2017) is capable of effectively addressing requests of explanation about the behaviour of the planner that generated the plan. However, there are some categories of questions that can not be dealt with by considering only the planning models and the planning process. Let us examine such categories by considering the following examples.

- *Why is cube3 moved on top of cube4, and not on cone1?*
  A possible and trivial answer to the first question can be obtained by analysing the domain model: there is no operator that allows to move a cube over a cone. Clearly, this sort of responses are not particularly informative for the user, as they do not provide additional information on top of what is already available for the user to directly investigate. Instead, a child that is used to play with building blocks could provide a very intuitive yet informative explanation: a cube stacked on a cone will fall, so you do not even try to stack them. Similarly, the child would be happy to answer questions like *Why am I not able to move a block that is not clear?* To explain the plans and, to some

extent, the knowledge of a planning model, the child is implicitly exploiting his awareness of the domain, that is commonly termed as context awareness. In other words, the knowledge needed for answering that class of questions, is given by the application context. Hereinafter we will refer to this class of explanations as *context-related*.

- *Why (some) actions can be executed in parallel?*, *Why the exploited hand is not indicated?*

  As in the previous case, a trivial explanation would be that actions are executed in parallel when they are independent, that is, they do not have conflicting effects and their effects do not interfere with pre-conditions of the other actions. In order to provide an informative answer to this question, it is necessary to reason in terms of decisions and assumptions made during the modelling process, on the basis of the available requirements specification. In the specific case, for instance, requirements may specify that more than one "hand" is available to move blocks. For safety reasons, one hand is not allowed to deal, at a given time step, with blocks involved in tasks performed by another hand. On top of that, an expert knowledge engineer for automated planning can come up with a model like the one presented in Figure 1, where the robotic hands are not explicitly modelled, in light of the fact that –regardless of the number of available hands– actions to be performed at the same time step in the plan can be easily serialised and distributed among available hands. This class of explanations is therefore related to requirements and modelling decisions and assumptions. We refer to this class as *assumption-related*.

Having identified two classes of explanations that can be addressed by extending the knowledge encoded in the models, it is now crucial to understand from where, and how, such additional knowledge can be extracted.

Knowledge required for providing context-related explanations can be obtained by linking, during the KE process of the domain model, objects and entities with both common sense and context-specific knowledge ontologies. Such ontologies can semantically enhance notions by deriving context-related constraints and relationships. They are commonly applied in areas such as robotics or smart homes (Kunze, Tenorth, and Beetz 2010; Alirezaie et al. 2017), where they support the interaction between robots (or environment) and humans, by supporting the interpretation of commands and instructions. Remarkably, there exists approaches to semantically enrich PDDL problem models, in order to improve the reasoning capabilities of domain-independent planning engines (Dornhege et al. 2012). A similar approach, that supports knowledge share among domains with similar contexts, could be taken also on the KE side of planning, as explained in the next section.

Assumption-related explanations require domain and model-specific knowledge. Some explanations can be directly derived by the requirement specifications, assuming they have been encoded using a formal or semi-formal language. However, many explanations –as those previously listed– require also knowledge about the engineering process, particularly in terms of decisions taken, impact of such decisions, and assumptions made. Decisions and assumptions have to be documented during the KE process, and should be linked to the relevant part of the model, or of the specification.

Utilising a Knowledge Engineering Environment for Generating Explanations

A knowledge engineering environment has the purpose of helping domain and AI experts collect together and formulate application knowledge to be used in a variety of ways. KEWI is such tool for encoding domain knowledge usable by domain experts for collecting knowledge that could be used in automated planning. It was developed in the context of a real industrial process application [PlanSig 2015]. Though KEWI was aimed at gathering knowledge about actions and change, it features a domain expert-friendly set of tools specifically for those not familiar with AI planning, within a *domain ontology*. KEWI's use of the 'concept' is fundamental to its language, and is related to the idea of an object class. A concept is defined by it's super-class and the roles that constrain how instances of the concept may be related to instances of other concepts. There are three different types of role constraints: "part-of" relations for aggregation, properties which contain attribute-values, and general relations. Properties of objects give an object's attribute a value. Both properties and relations must have some functional information using the :min :max keywords. This denotes the type of relation/property such many-many, one-many etc

The domain ontology describes a collection of concepts populated by objects which must be plugged into a concept hierarchy, ensuring that explanations can exploit more abstract knowledge about them. Thus more information is available to be involved in an explanation, both in terms of tightening up what makes a valid state, and in terms of relating concepts and objects. The first example below is a KEWI representation of the Dockworkers domain [NauHallab 04], encoded by Gerhard Wickler [Plansig paper 2015]. This shows examples of concept hierarchies (e.g. pallet and container are members of the more abstract class stackable). It asserts that in any state, a crane must be at exactly one location denoted via the "(:min 1) (:max 1))" terms. This gives us two essential pieces of information - a location must exist for those objects, and this location must be unique. As another example, the KEWI ontology asserts that a "pile" is attached to a location, and it holds a pallet and a stack of containers. This knowledge is generally not explicit in a planning domain model since it is unlikely to be important in the planning function.

Returning to the Blockworld example, a part of ontology of our variant of BlocksWorld as captured by KEWI is depicted in Figure 3. It captures relations between classes of objects with constraints determining how many instances of the relation can be true in a state.

Both these examples illustrate how better explanations are available in this context: explanation can refer to more accurate used of relations and objects, and to semantically related information such as aggregation and concept relations.

```
(:domain dwr
  (:class agent)
  (:class crane
    (:super-class agent)
    (:role at (:min 1)(:max 1)(:class location))
    (:role holds (:max 1)(:class container)))
  (:class robot
    (:super-class agent)
    (:role loaded-with (:max 1)(:class container))
    (:property has-colour (:min 1)(:max 1)
       (:type colour)))
  (:class location
    (:role occupied-by (:max 1)(:class robot)))
  (:class stackable)
  (:class container
    (:super-class stackable)
    (:role on (:max 1)(:class stackable))
    (:role piled-on (:max 1)(:class pallet))
    (:property paint (:min 1)(:max 1)
       (:type colour)))
  (:class pallet
    (:super-class stackable)
    (:role at (:min 1)(:max 1)(:class location))
    (:role top (:min 1)(:max 1)(:class stackable)))
  (:property colour
    (:values ( red green blue )))
  (:relation adjacent
    (:arguments ((?loc1 location)(?loc2 location)
    )))
```

Figure 2: A part of ontology of the Dock-worker's domain in KEWI

```
(:class object
(:role on (:max 1) (:class cube)))
(:class cube (:super-class object))
(:class cone (:super-class object))
(:class hand
(:role holding (:max 1) (:class object)))
(:class table
(:role on-table (:max k) (:class object)))
```

Figure 3: A part of ontology of our variant of BlocksWorld generated by KEWI

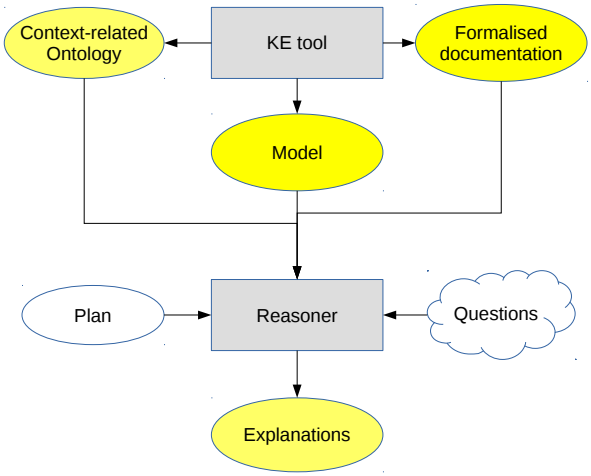


Figure 4: An overview of the components needed for supporting context-aware and assumption-aware explanations in planning.

Discussion

Three main approaches are traditionally exploited to formulate domain models.

1. Domain models are often formulated into a planner input language directly from a requirements specification using only (enriched) text editors (these are so-called *hand-crafted* domain models), such as the `planning.domains` framework.
2. Following a more principled process, requirements can be encoded firstly into a more application-oriented language such as UML in itSIMPLE, or in AIS-DDL in the KEWI interface, and then mapped into the target planner's input language. This can support a validation and verification processes, as well as improving maintenance and documentation of models.
3. Finally, there are approaches where automated acquisition tools can formulate partial or complete domain models. Tools, such as the LAMP system (Zhuo et al. 2010) and the LOCM system (Cresswell, McCluskey, and West 2013), can rely on different amount and type of available knowledge.

The first approach can hardly be modified to include the encoding of the additional knowledge that would be required in order to support context-related and assumption-related explanations. This is because the hand-crafting process is usually performed without producing significant documentation –sometimes even without any documentation at all– and is a try-and-fix process, where issues are tackled when they arise. This approach is supposed to be used for encoding mock-ups of domain models, in order to quickly testing some prototypes; however, evidence from recent ICKEPS indicate that this is usually not the case.

Tools designed for encoding models following the third approach, i.e. without humans in the loop, would also be hard to extend. The idea of such approaches is to generate models by considering available "raw" data, possibly collected from sensors, and to generate models that reflect the observed behaviour. The focus is not on human-readability, but on the encoding of models that can be immediately exploited for planning purposes. For this reason, it usually hard to link names and objects to the specific context. Moreover, assumptions are not made at the encoding level, but are implicitly given by the way in which tools have been designed and coded.

At this point, it should be clear that tools developed for encoding models following the depicted approach 2 are the best candidate to be extended for supporting explanations from the KE process. Tools like itSimple and KEWI come in the form of large frameworks that guide the encoding

of models. For this reason, they can be straightforwardly extended to support the level of documentation required for dealing with assumption-related explanations. Similarly to existing IDE for object-oriented programming, such KE tools can include plugins that facilitate documentation, and that underpin the semi-formalised encoding of comments, assumptions and motivations. This can be done by highlighting areas of the model that lack of "comments", and by posing *critical questions* (Atkinson and Bench-Capon 2007). Context-related explanations can be supported by appropriately designed plugins allowing to: (i) import an appropriate existing ontology, or design and fill a new ontology, (ii) link the ontology to the current KE process, and (iii) suggest, on the basis of the ontology, suitable object's types, predicates, etc. to be used. Aligning types and properties between the planning domain model and the ontology, allows to deal with context-related explanations.

Of course, extending the knowledge encoded during the KE process of a planning model is not enough to provide explanations. It is pivotal to provide a reasoner that is able to (i) analyse the questions, provided by the user or automatically generated, (ii) identify the most suitable source of knowledge, (iii) collect the available knowledge and perform automated reasoning, and (iv) provide one (or more) explanation addressing the question. Figure 4 shows an overview of an architecture capable of dealing with context- and assumption-related explanations. In yellow, we highlighted elements that can be a source of knowledge, or the product, of the explanation process.

Conclusion

It is pivotal that, in real-world planning applications, decisions and solutions can be explained, in order to maximise safety and increase reliability of the planning component.

In this paper we identified two categories of explanations that require additional knowledge, with regards to the knowledge provided in planning models and that can be extracted from the planning process, in order to be satisfyingly dealt with. Context-related explanations require information about the context that are not typically encoded in the planning models. Assumptions-related explanations are focused on the way in which models are encoded, and require information about decisions and assumptions made during the KE process.

We then described an architecture that would be able to deal with the introduced classes of explanations. Future work will focus on defining a suitable formalism for the additional knowledge, and in extending KE tools in order to effectively support the encoding of such knowledge during the KE process. Finally, we will investigate techniques and approaches –particularly from the argumentation community– that can be included in the automated reasoner shown in Figure 4.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.-J.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; et al. 2004. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems* 19(1):8–12.

Alirezaie, M.; Renoux, J.; Köckemann, U.; Kristoffersson, A.; Karlsson, L.; Blomqvist, E.; Tsiftes, N.; Voigt, T.; and Loutfi, A. 2017. An ontology-based context-aware system for smart homes: E-care@ home. *Sensors* 17(7):1586.

Atkinson, K., and Bench-Capon, T. 2007. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence* 171(10-15):855–874.

Bredeweg, B.; Salles, P.; Bouwer, A.; Liem, J.; Nuttle, T.; Cioaca, E.; Nakova, E.; Noble, R.; Caldas, A. L. R.; Uzunov, Y.; Varadinova, E.; and Zitek, A. 2008. Towards a structured approach to building qualitative reasoning models and simulations. *Ecological Informatics* 3(1):1 – 12.

Caminada, M. W. A.; Kutlák, R.; Oren, N.; and Vasconcelos, W. W. 2014. Scrutable plan enactment via argumentation and natural language generation. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS*, 1625–1626.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 156–163.

Chrpa, L.; McCluskey, T. L.; Vallati, M.; and Vaquero, T. 2017. The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. *AI Magazine* 38(1):104–106.

Cresswell, S.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using *LOCM*. *Knowledge Eng. Review* 28(2):195–213.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2012. Semantic attachments for domain-independent planning systems. In *Towards service robots for everyday environments*. 99–115.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*.

Kunze, L.; Tenorth, M.; and Beetz, M. 2010. Putting people's common sense into knowledge bases of household robots. In Dillmann, R.; Beyerer, J.; Hanebeck, U. D.; and Schultz, T., eds., *KI 2010: Advances in Artificial Intelligence*, 151–159.

McCluskey, T. L., and Vallati, M. 2017. Embedding automated planning within urban traffic management operations. In *Proceedings of the International Conference on Automated Planning and Scheduling ICAPS*.

McCluskey, T. L.; Aler, R.; Borrajo, D.; M.Garagnani; P.Haslum; Jarvis, P.; I.Refanidis; and Scholz, U. 2003. Knowledge Engineering for Planning Roadmap. *http://scom.hud.ac.uk/planet/home*.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering knowledge for automated planning: Towards a notion of quality. In *Proceedings of the Knowledge Capture Conference, K-CAP*, 14:1–14:8.

Parkinson, S., and Longstaff, A. P. 2015. Multi-objective optimisation of machine tool error mapping using automated planning. *Expert Syst. Appl.* 42(6):3005–3015.

Plch, T.; Chomut, M.; Brom, C.; and Barták, R. 2012. Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio. *ICAPS12 System Demonstration* 4.

Simpson, R.; Kitchin, D. E.; and McCluskey, T. 2007. Planning domain definition using gipo. *Knowledge Engineering Review* 22(2):117–134.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Vaquero, T. S.; Tonaco, R.; Costa, G.; Tonidandel, F.; Silva, J. R.; and Beck, J. C. 2012. itSIMPLE4.0: Enhancing the modeling experience of planning problems. In *System Demonstration – Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)*.

Wickler, G.; Chrpa, L.; and McCluskey, T. L. 2014. KEWI - A knowledge engineering tool for modelling AI planning tasks. In *KEOD 2014 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, 36–47.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence* 174(18):1540 – 1569.