Argumentation for Evaluative Explanations of PDDL Plans

Anna Collins
Department of Informatics, King's College London
{anna.collins}@kcl.ac.uk

***Abstract*— For robotics applications, clear communication between an AI planning system and its human user is crucial. Depending on the level of expertise the user has, the availability and accuracy of live sensor data, and the level of detail in the planning model, good decision-making can be challenging. This paper presents an approach to evaluating a plan's causal structure and a user's decision to remove an action from a plan using domain-specific argumentation schemes. The approach is designed in a modular way and is intended to be used with any PDDL (Planning Domain Definition Language) planning system and easily customizable to its intended application.**

I. Introduction

Explainability is crucial for human-robot systems to be efficient in practical settings. In many scenarios (e.g. automated warehouse distribution centres) when a fault occurs in execution, quick decisions are crucial due to tight deadlines. If a robot fails to pick up an item from a shelf, should the human overseer leave his work to try and fix the robot? What impact will leaving the robot have on the remainder of the plan's execution?

Re-planning is a commonly cited tool to overcome changes in the real-world model, e.g. in situated robotics [1]. However, many instances exist where this is not possible. Plans might have longer approval periods before they can be used due to safety reasons, or perhaps the specific change cannot be captured by the system's model as it was written. Because of the relatively high level of uncertainty in most robotics domains, and the desire to keep planning models as simple as possible for complexity reasons, this is a common occurrence.

Providing a user insight into the causal structure of a plan is a crucial component of effective communication in the case of a fault in execution [2]. Extracting the plan's causal structure and implementing it in a logical way necessitates the use of argumentation; as described in [3], argumentation frameworks and causal models are two representations of the same idea. Recently, work using argumentation for explanations has been done for scheduling [4]. Other work involving causal explanations in planning had been done in task planning [5].

Additionally, a tool from argumentation theory called an argumentation scheme has been utilised in some areas to aid communication between AI systems and their users, e.g. in multi-agent problems [6]. They provide a template to capture common arguments used to communicate decisions. As of yet, no work has been done to provide a user decision support with the addition of the causal repercussions of each choice in planning. The ability to evaluate a user's choice as good or bad is an underappreciated tool in explainable AI [7].

In this work, we consider an ASPIC$^+$ [8] style framework with defeasible rules capturing the relationships between actions in a plan. We extend the work done in [9] to include an implementation of plan causal structure extraction with the addition of argumentation schemes to capture common user questions, in the style of [10].

We argue that if the common execution faults of a system can be represented in a layer separate from the PDDL (Planning Domain Definition Language) model, solving for an initial plan is easier, and the planner does not need to be specialised to deal with probabilistic modelling languages. The information about these faults can also be updated by non-technical users more easily due to argumentation's close connection with natural language.

Additionally, there is a need to tailor an explanation system to its users and the environment, and for it to remain flexible. Domain-independent explainable planning has been researched extensively and has wide-reaching applications [11][12]. However, more nuanced and flexible explanations can be tricky without the ability to capture human user expertise and change the types of explanations required. To that end, a customizable set of arguments would be beneficial.

This paper introduces the argumentation explanation system in the context of explaining the consequences of an execution fault, but it can be extended for more general use.

II. Motivating Example

Consider an automated warehouse with multiple robots and a human overseer. Each robot is tasked with specific delivery goals for which they are required to obtain packages from shelves and bring them to the various drop-off locations to be packed by human workers.

In such a setting, the time frames for completing the tasks are short, and the system may not be equipped with accurate sensors to update the current state of the system. Re-planning in the case of a robot fault, therefore, may not be a viable option. It is up to the human to determine what the best course of action would be. For this, the ability to query the system and quickly see the consequences of the fault would be helpful. Additionally, evaluating the cost of ignoring a fault could help determine the best course of action. Using a domain-specific argumentation scheme could allow the domain experts the ability to encode common issues into the system and use them for evaluations on the fly.

```
0 (move r-2 loc-1 loc-3)
1 (pick-up r-1 loc-2 pac-4 cap-2 cap-3)
2 (pick-up r-2 loc-3 pac-3 cap-1 cap-2)
3 (move r-1 loc-2 loc-5)
4 (drop r-1 loc-5 pac-4 cap-2 cap-3)
5 (move r-2 loc-3 loc-1)
6 (drop r-2 loc-1 pac-3 cap-1 cap-2)
7 (move r-1 loc-5 loc-2)
8 (pick-up r-2 loc-1 pac-1 cap-1 cap-2)
9 (pick-up r-2 loc-1 pac-2 cap-0 cap-1)
10 (move r-2 loc-1 loc-3)
11 (move r-2 loc-3 loc-2)
12 (drop r-2 loc-2 pac-1 cap-0 cap-1)
13 (drop r-2 loc-2 pac-2 cap-1 cap-2)
```

Fig. 1. Example plan with start times and durations removed for simplicity. Actions are enumerated to match vertices in causal graph found in Figure 2.

Consider the plan in Figure 1. There are three different actions relating to picking up and dropping off packages as well as moving between locations. The `r-1, r-2` represent two robots; `loc-1, loc-2, loc-3, loc-4, loc-5` represent five different locations; `pac-1, pac-2, pac-3, pac-4` represent four packages the robots need to deliver; lastly `cap-n` with $n = 1...3$ refers to the capacity changes when packages are picked up or dropped off.

The goals that this plan satisfy are delivering all four packages to their respective destinations.

The highlighted action in red (action 1) is used to demonstrate a fault during execution. At this point, the user would interact with the argumentation system to determine the consequences of this action failing to execute, and to access decision support.

III. Background on Argumentation

For capturing the causal structure of plans, we use ASPIC$^+$ arguments. This is not the only argumentation system applicable, and some others could be swapped for additional functionality.

For a full description of ASPIC$^+$ see [8]. The focus of this approach is on constructing nested arguments, and so we omit other aspects of ASPIC$^+$ from this general background. The following description is taken from [9].

We start with a language $\mathcal{L}$, closed under negation. A reasoner is then equipped with a set `Rules` of strict rules, denoted $\phi_1, \ldots, \phi_n \rightarrow \phi$, and defeasible rules, denoted $\phi_1, \ldots, \phi_n \Rightarrow \phi$, where $\phi_1, \ldots, \phi_n, \phi$ are all elements of $\mathcal{L}$. A knowledge base Δ is then a set of elements K from $\mathcal{L}$ and a set `Rules`. From Δ a set of arguments $\mathcal{A}(\Delta)$ is constructed, where an argument A is made up of some subset of K, along with a sequence of rules, that lead to a conclusion. Given this, $\texttt{Prem}(\cdot)$ returns all the premises, $\texttt{Conc}(\cdot)$ returns the conclusion and $\texttt{TopRule}(\cdot)$ returns the last rule in the argument. An argument A is then:

- ϕ if $\phi \in K$ with: $\texttt{Prem}(A) = \{\phi\}$; $\texttt{Conc}(A) = \phi$; $\texttt{Sub}(A) = \{A\}$; and $\texttt{TopRule}(A)$ = undefined.
- $A_1, \ldots, A_n \rightarrow \phi$ if A_i, $1 \leq i \leq n$, are arguments and there exists a strict rule of the form $\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow \phi$ in `Rules`. $\texttt{Prem}(A) = \texttt{Prem}(A_1) \cup \ldots \cup \texttt{Prem}(A_n)$; $\texttt{Conc}(A) = \phi$; and $\texttt{TopRule}(A) = \texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow \phi$.
- $A_1, \ldots, A_n \Rightarrow \phi$ if A_i, $1 \leq i \leq n$, are arguments and there exists a defeasible rule of the form $\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \Rightarrow \phi$ in `Rules`. $\texttt{Prem}(A) = \texttt{Prem}(A_1) \cup \ldots \cup \texttt{Prem}(A_n)$; $\texttt{Conc}(A) = \phi$; and $\texttt{TopRule}(A) = \texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \Rightarrow \phi$.

Then, given $K = \{\alpha; \beta\}$ and `Rules` = $\{\alpha \rightarrow \gamma; \beta, \gamma \Rightarrow \delta\}$, we have the following arguments:

$$A_1 : \alpha$$
$$A_2 : \beta$$
$$A_3 : A_1 \rightarrow \gamma$$
$$A_4 : A_2, A_3 \Rightarrow \delta$$

When applied to planning, these arguments define a causal structure of a plan, as will be described in Section 4.

A. Argumentation Schemes

An argumentation scheme is a structure to capture common arguments. Following [10], this includes a set of critical questions, premises, and a conclusion. They are structured around a particular style of inference depending on the setting in which it is being used. For example, argumentation by practical reasoning. In this way, they are easily changed based upon the types of questions most asked and information relevant to a certain problem.

Formally, they are defined as a tuple, (I, CQ, P, C) consisting of an identifier, a set of critical questions, a set of premises and a conclusion C. With the motivating scenario in mind, a set of relevant critical questions may be: **(CQ1)** What goal(s) does the action in question impact? **(CQ2)** Does the system generally recommend an intervention for this type of fault? **(CQ3)** Can the user visually confirm the fault? **(CQ4)** Was a fault signal received?

Given our motivating example, an argumentation scheme with these critical questions may be instantiated as follows,

```
signal_received(fault, robot),
confirmed(user, fault),
impacts(robot, goal),
recommended(fault, intervention) ⇒int_req
```

where the `int_req` refers to whether or not a human intervention is prescribed.

$$\Delta = \left\{ \begin{array}{l} \texttt{signal\_received(arm, robot1)} \\ \texttt{confirmed(user, arm)} \\ \texttt{recommended(arm, human\_int)} \end{array} \right\}$$

B. Causal Chunks

In [9], the notion of causal 'chunks' was introduced with respect to planning. These chunks are defined as a subset of arguments centered around a theme. A theme can be an object in the original problem description, a goal, or any other component of the planning instance.

For the motivating example, the user's interest lies with the robot that experienced a fault. So, the themes of interest are 'robot 1' and the goals of the problem.

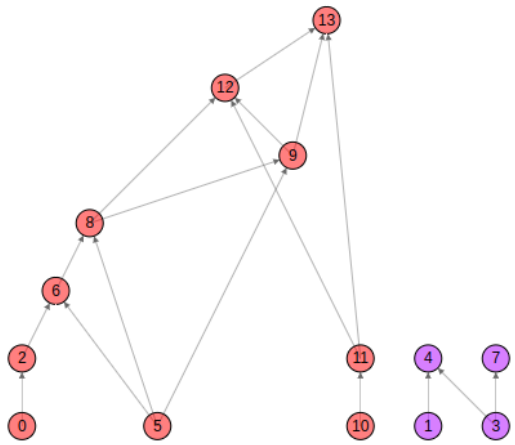

Fig. 2. Causal structure of the example plan given in Figure 1.

Other themes could be included, for example those pertaining to state variable information. An argumentation scheme surrounding the capacities of a robot could be created, necessitating a causal structure to be created which holds capacity information.

IV. Parsing Plans

Given a plan, problem, and domain, an argumentation framework is constructed using a PDDL parsing system. This system reads the files into a framework, extracting the relevant information about the causal structure of the plan and the goals of that specific problem. The causal structure of the plan is instantiated as a series of arguments with defeasible rules indicating precedence. Figure 2 shows the causal structure of the example plan given in Figure 1. Note that there are two distinct graphs corresponding to two causal 'chunks'. These reflect the two robots in the problem; the red subgraph represents `r-2` and the pink subgraph represents `r-1`. The vertices with no incoming edges correspond to actions whose preconditions were met in the initial state description.

From this graph representation, the nested arguments are extracted. The argument structure for this plan is the following:

$$
\begin{aligned}
&A_1 : 0\\
&A_2 : 1\\
&A_3 : 3\\
&A_4 : 5\\
&A_5 : 10\\
&A_6 : A_1 \Rightarrow 2\\
&A_7 : A_5 \Rightarrow 11\\
&A_8 : A_3 \Rightarrow 7\\
&A_9 : A_1, A_6 \Rightarrow 6\\
&A_{10} : A_2, A_3 \Rightarrow 4\\
&A_{11} : A_1, A_6, A_9, A_4 \Rightarrow 8\\
&A_{12} : A_1, A_6, A_9, A_4, A_{11} \Rightarrow 9\\
&A_{13} : A_1, A_6, A_9, A_4, A_{11}, A_{12} \Rightarrow 12\\
&A_{14} : A_1, A_6, A_9, A_4, A_{11}, A_{12}, A_{13} \Rightarrow 13
\end{aligned}
$$

From this, we can model the improper execution of action 1 by setting this argument to false. Note the red colored arguments A_2 and A_{10}. These are the two arguments made false by action 1 not executing. Even though the action appears early in the plan, it doesn't affect many of the subsequent actions. This is because `r-1` only delivers one package, with the other three being delivered by `r-2`.

V. Implementing Argumentation Schemes

Given the argumentation scheme described in section 3.1, we can instantiate an argument based upon some knowledge base and plan from Figure 1.

Consider the knowledge base from Section 3.A. After the plan is parsed, the goals affected by the fault can be extracted. As seen from the nested arguments, only one goal-adding action is affected, action 4. This action involves the delivery of `pac-4`, which we label `g-4` for 'goal 4'. These facts are then imported into the knowledge base as shown below:

$$
\Delta = \left\{
\begin{array}{l}
\texttt{signal\_received(arm, robot1)}\\
\texttt{confirmed(user, arm)}\\
\texttt{impacts(r-1, goal4)}\\
\texttt{¬impacts(r-1, goal1)}\\
\texttt{¬impacts(r-1, goal2)}\\
\texttt{¬impacts(r-1, goal3)}\\
\texttt{recommended(arm, human\_int)}
\end{array}
\right\}
$$

The scheme can also be tailored to the number of goals; for instance, perhaps the user wishes to specify that an intervention is only warranted if either $g-1$ and $g-2$ or $g-3$ and $g-4$ are impacted. Using the running example, if action 1 fails, the following argument can be extracted:

```
signal_received(fault, r-1),
confirmed(user, fault),
(¬impacts(r-1, g-1) ∨ ¬impacts(r-1, g-2)),
(¬impacts(r-1, g-3) ∨ impacts(r-1, g-4)),
recommended(fault, intervention), ⇒
¬intervention
```

The premises which were found false are highlighted in red. By the implication in this scheme, it is concluded that no intervention should be taken. This is because `r-1` doesn't impact either `g-1` or `g-2`.

A. Presenting the Argument

In order to effectively communicate the argument to the user, the argument is parsed to natural language. A simple algorithm is used which parses the propositional logic formula to a preset list of natural language statements substituting the truth values of premises and conclusion. These equivalences are also easily customised at the point when the argumentation schemes are created, making it easy to tailor the language to the user's preferences. With the premises ordered based upon their truth-values, this argument corresponds to the following natural language argument:

"A signal is received indicating a fault with robot 1. The user confirms this fault to be accurate. The plan indicates robot 1 does impact goal 4. The system recommends an

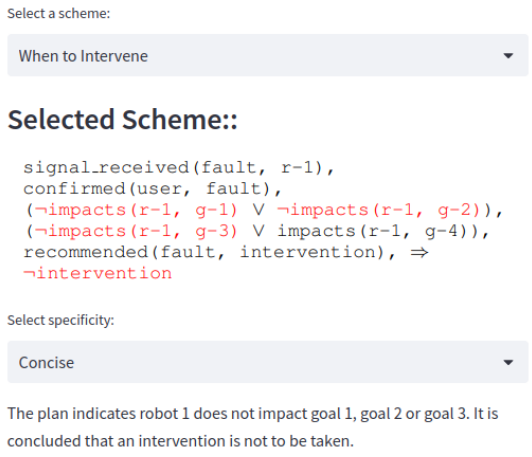

Fig. 3. Portion of the system's UI.

intervention for this fault. However, The plan indicates robot 1 does not impact goal 1, goal 2 or goal 3. It is concluded that an intervention is not to be taken."

If a more concise explanation is required, the true-valued premises can be removed, leaving the 'deciding' premise only:

"The plan indicates robot 1 does not impact goal 1, goal 2 or goal 3. It is concluded that an intervention is not to be taken."

The UI created for initial testing is shown in Figure 3.

VI. Alternate Schemes

Many different iterations of the 'When to Intervene' argumentation scheme can be made to fit different critical questions a user might have. The user may require more or fewer premises to support the recommendation. For example, one could break this scheme down into three different schemes: (1) 'When to Intervene', (2)'When to Intervene (with goals)', and (3) 'When to Intervene' (with goals and actions). These correspond to an increasing number of premises:

```
signal_received(fault, robot),
confirmed(user, fault),
recommended(fault, intervention) ⇒int_req

signal_received(fault, robot),
confirmed(user, fault),
impacts(robot, goal),
recommended(fault, intervention) ⇒int_req

signal_received(fault, robot),
confirmed(user, fault),
impacts(robot, goal),
enables(robot, action),
recommended(fault, intervention) ⇒int_req
```

VII. Initial Testing and Discussion

This argumentation explanation system has been implemented into a web app with Streamlit (see Figure 3). Along with the warehouse domain, initial testing using IPC planning domains was conducted to determine the performance of the plan causal graph extraction. The domains 'driverlog', 'tidybot', and 'zeno_travel' were selected, and the benchmark instances from the competition used for testing. There was no significant difference in performance between the smallest and largest instances, supporting our motivation to have a modular system which acts independently of planners. For these domains, argumentation schemes in the style of the 'When to Intervene' scheme were tested. The time to extracting a conclusion from these schemes, which did not include chained arguments, was also negligible, as expected. One perceived drawback of incorporating the causal structure of the entire plan in the user interface is its unwieldy size as the instances grow larger. This can be remedied by only showing the relevant subgraphs to the user's question.

VIII. Conclusions and Future Work

We claim the benefits of using an argumentation layer on top of a planning system for robotics are threefold: (1) it enables the underlying planning model to remain simple(r); (2) it is easier to obtain inputs from domain experts based on their observations of common faults; (3) it allows for easier parsing to natural language for wider use. In future work, careful testing will be required to confirm these benefits, including user studies in a human-robot setting. Also, more complicated argumentation schemes including chained arguments could be considered. Argumentation schemes based upon different forms of inference could be studied, along with the ability of mixing schemes for more robust explanations.

References

[1] M. Cashmore, A. Coles, B. Cserna, E. Karpas, D. Magazzeni, and W. Ruml, "Replanning for situated robots," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 665–673.

[2] T. Miller, "Contrastive explanation: A structural-model approach," *The Knowledge Engineering Review*, vol. 36, p. e14, 2021.

[3] A. Bochman, "Propositional argumentation and causal reasoning." in *IJCAI*, 2005.

[4] K. Cyras, D. Letsios, R. Misener, and F. Toni, "Argumentation for explainable scheduling," 2019, https://arxiv.org/abs/1811.05437.

[5] G. Canal, S. Krivic, P. Luff, and A. Coles, "Task plan verbalizations with causal justifications," in *ICAPS 2021 Workshop on Explainable AI Planning (XAIP)*, 2021.

[6] A. R. Panisson, P. McBurney, and R. H. Bordini, "A computational model of argumentation schemes for multi-agent systems," *Argument & Computation*, vol. 12, no. 3, pp. 357–395, 2021.

[7] T. Miller, "Explainable ai is dead, long live explainable ai! hypothesis-driven decision support," 2023, https://arxiv.org/abs/2302.12389.

[8] S. Modgil and H. Prakken, "A general account of argumentation with preferences," *Artificial Intelligence*, vol. 195, pp. 361–397, 2013.

[9] A. Collins, D. Magazzeni, and S. Parsons, "Towards an argumentation-based approach to explainable planning," in *ICAPS 2019 Workshop XAIP Program Chairs*, 2019.

[10] D. Walton, C. Reed, and F. Macagno, *Argumentation schemes*. Cambridge University Press, 2008.

[11] J. Hoffmann and D. Magazzeni, "Explainable ai planning (xaip): overview and the case of contrastive explanation," *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*, pp. 277–282, 2019.

[12] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable ai planning and decision making," *arXiv preprint arXiv:2002.11697*, 2020.