Argument Schemes for Explainable Planning

Quratul-ain MAHESAR\textsuperscript{a} and Simon PARSONS\textsuperscript{a,b}

\textsuperscript{a}Department of Informatics, King’s College London, UK
\textsuperscript{b}School of Computing Science, University of Lincoln, Lincoln, UK

Abstract. Artificial Intelligence (AI) is being increasingly used to develop systems that produce intelligent solutions. However, there is a major concern that whether the systems built will be trusted by humans. In order to establish trust in AI systems, there is a need for the user to understand the reasoning behind their solutions and therefore, the system should be able to explain and justify its output. In this paper, we use argumentation to provide explanations in the domain of AI planning. We present argument schemes to create arguments that explain a plan and its components; and a set of critical questions that allow interaction between the arguments and enable the user to obtain further information regarding the key elements of the plan. Finally, we present some properties of the plan arguments.

Keywords. Argument schemes, explanation, planning

1. Introduction

Artificial intelligence (AI) researchers are increasingly concerned that whether the systems they build will be trusted by humans. Automated planning is one of the subfields of AI that focuses on developing intelligent techniques to determine efficient plans, i.e., a sequence of actions that should be performed in order to achieve a set of goals. Explainable AI Planning (XAIP) is a field that involves explaining AI planning systems to a user. The main goal of plan explanation is to help humans understand the plans produced by the planners. Approaches to this problem include explaining planner decision making processes as well as forming explanations from the models. Previous work on model-based explanations includes an iterative approach \cite{ref13} as well as using explanations for more intuitive communication with the user \cite{ref9}.

Argumentation is connected to the idea of establishing trusted AI by explaining the results and processes of computation, and has been used in many applications in multi-agent planning \cite{ref16} and practical reasoning \cite{ref1}. This work is an attempt to generate explanation arguments in the domain of AI planning, to answer questions such as Why A?, where A is an action in the plan, or How G?, where G is a goal. Questions like these are inherently based upon definitions held in the domain related to a particular problem and solution. Furthermore, questions regarding particular state information may arise, such as Why A here?. Thus, extracting relevant information about actions, states and goals from the planning model is required to provide explanations to the user. Further-
more, some users might be interested in a summarized explanation of the whole plan and consequently inquire further information regarding the elements of the plan.

In this work, we make a first attempt to formalise a set of argument schemes [17] that are aimed at creating arguments that explain and justify the plan and its key elements (i.e., action, state and goal). Furthermore, we present critical questions that allow the user to seek further information regarding the plan, and allow interaction between different arguments. Thus, the explanation arguments will enable a planning system to answer any such questions at a different granularity level. To make our argumentation-based explanations for the planning study concrete, we take a version of the classic blocks world, as a case study example.

2. Related Work

Our research is inspired by the works in practical reasoning and argumentation for multi-agent planning. However, our argument scheme based approach, generates explanations for a plan created by an AI planner, which we assume to be a single entity. One of the most well known scheme-based approach in practical reasoning is presented in [1], which is accompanied by a set of critical questions that allow agents to evaluate the outcomes on the basis of the social values highlighted by the arguments. Furthermore, in [15], a model for arguments is presented that contributes in deliberative dialogues based on argumentation schemes for arguing about norms and actions in a multi-agent system. [11] has proposed a similar scheme-based approach for normative practical reasoning where arguments are constructed for a sequence of actions.

[12] propose a framework that integrates both the reasoning and dialectical aspects of argumentation to perform normative practical reasoning, enabling an agent to act in a normative environment under conflicting goals and norms and generate explanation for agent behaviour. [2] have explored the use of situation calculus as a language to present arguments about a common plan in a multi-agent system. [14] present an argumentation-based approach to deliberation, the process by which two or more agents reach a consensus on a course of action.

The works that are closest to our research for generating plan explanations using argumentation are given in [3] and [7]. In [3], a dialectical proof based on the grounded semantics [4] is created to justify the actions executed in a plan. More recently, in [7], an Assumption-based argumentation framework (ABA) [6] is used to model the planning problem and generate explanation using the related admissible semantic [8]. Our work differs from both, as we present argument schemes to generate the arguments that directly provide an explanation. Moreover, we use the concept of critical questions to provide dialectical interaction with the user and arguments.

3. Planning Model

In this section, we introduce a planning model which is based on an instance of the most widely used planning representation, PDDL (Planning Domain Definition Language), as given in [10]. We define the planning problem as follows.
Definition 3.1 (Planning Problem) A planning problem is a tuple $P = (O, Pr, △I, △G, A, Σ, G)$, where:
1. $O$ is a set of objects;
2. $Pr$ is a set of predicates;
3. $△I \subseteq Pr$ is the initial state;
4. $△G \subseteq Pr$ is the goal state, and $G$ is the set of goals;
5. $A$ is a finite, non-empty set of actions;
6. $Σ$ is the state transition system;

We define the predicates as follows.

Definition 3.2 (Predicates) $Pr$ is a set of domain predicates, i.e., properties of objects that we are interested in, that can be true or false. For a state $s \subseteq Pr$, $s^+$ are predicates considered true, and $s^- = Pr \setminus s^+$. A state $s$ satisfies predicate $pr$, denoted as $s \models pr$, if $pr \in s$, and satisfies predicate $¬pr$, denoted $s \models ¬pr$, if $pr \notin s$.

We define two types of actions, the standard sequential action, i.e., action, and the concurrent action.

Definition 3.3 (Action) An action $a = (pre, post)$ is composed of sets of predicates $pre, post$ that represent a’s pre and post conditions respectively. Given an action $a = (pre, post)$, we write $pre(a)$ and $post(a)$ for pre and post. Postconditions are divided into add($post(a)^+$) and delete($post(a)^-$) postcondition sets. An action $a$ can be executed in state $s$ iff the state satisfies its preconditions. The postconditions of an action are applied in the state $s$ at which the action ends, by adding the positive postconditions belonging to $post(a)^+$ and deleting the negative postconditions belonging to $post(a)^-$.

Definition 3.4 (Concurrent Action) A concurrent action $a_c$ is an action that can be concurrently executed with other concurrent actions. Two concurrent actions $a_i$ and $a_j$ (where $i \neq j$) are executable if their preconditions hold and their effects, i.e., postconditions are consistent. Furthermore, the effects of $a_i$ should not contradict the preconditions of $a_j$ and vice-versa.

We define the state transition system as follows.

Definition 3.5 (State Transition System) The state-transition system is denoted by $Σ = (S,A,γ)$, where:
- $S$ is the set of states.
- $A$ is a finite, non-empty set of actions.
- $γ : S \times A = S$ where:
  * $γ(S, a) = (S \setminus post(a)^-) \cup post(a)^+$, if $a$ is applicable in $S$;
  * $γ(S, a) = undefined$ otherwise;
  * $S$ is closed under $γ$.

We define the goal in a plan as follows.

Definition 3.6 (Goal) A goal achieves a certain state of affairs. Each $g \in G$ is a set of predicates $g = \{r_1, ..., r_n\}$, known as goal requirements (denoted as $r_i$), that should be satisfied in the state to satisfy the goal.
We define a plan as follows.

**Definition 3.7 (Plan)** A plan is a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$. The extended state transition function for a plan is defined as follows:

- $\gamma(S, \pi) = S$ if $|\pi| = 0$ (i.e., if $\pi$ is empty);
- $\gamma(S, \pi) = \gamma(\gamma(S, a_1), a_2, \ldots, a_n)$ if $|\pi| > 0$ and $a_1$ is applicable in $S$;
- $\gamma(S, \pi)$ is undefined otherwise.

A plan $\pi$ is a solution to a planning problem $P$ iff:

1. Only the predicates in $\Delta_I$ hold in the initial state: $S_1 = \Delta_I$;
2. the preconditions of action $a_i$ hold at state $S_i$, where $i = 1, 2, \ldots, n$;
3. $\gamma(S_i, \pi)$ satisfies the set of goals $G$.
4. the set of goals satisfied by plan $\pi$ is a non-empty $G\pi \neq \emptyset$ consistent subset of goals.

Each action in the plan can be performed in the state that results from the application of the previous action in the sequence. After performing the final action, the set of goals $G\pi$ will be true. We use the following Blocks World example to illustrate.

**Example 3.1** A classic blocks world consists of the following: (1) A flat surface such as a tabletop, (2) An adequate set of identical blocks which are identified by letters, (3) The blocks can be stacked one on one to form towers of unlimited height.

We have three predicates:

1. $\text{ON}(A, B)$ – block $A$ is on block $B$.
2. $\text{ONTABLE}(A)$ – block $A$ is on the table.
3. $\text{CLEAR}(A)$ – block $A$ has nothing on it.

Following are the two actions $a_1$ and $a_2$:

1. $a_1 : \text{UNSTACK}(A, B)$ – pick up clear block $A$ from block $B$;
   - $\text{pre}(a_1) : \text{CLEAR}(A) \land \text{ON}(A, B)$
   - $\text{post}(a_1)^+ : \text{ONTABLE}(A) \land \text{CLEAR}(B)$
   - $\text{post}(a_1)^- : \text{ON}(A, B)$
2. $a_2 : \text{STACK}(A, B)$ – place block $A$ onto clear block $B$;
   - $\text{pre}(a_2) : \text{ONTABLE}(A) \land \text{CLEAR}(A) \land \text{CLEAR}(B)$
   - $\text{post}(a_2)^+ : \text{ON}(A, B)$
   - $\text{post}(a_2)^- : \text{ONTABLE}(A) \land \text{CLEAR}(B)$

We have the following conditional statements:

- If block $A$ is on the table it is not on any other block.
- If block $A$ is on block $B$, block $B$ is not clear.

The initial and goal states of the blocks world problem are shown in Figure 1.

The initial state $\Delta_I$ is given by:

$\text{ONTABLE}(D) \land \text{ON}(C, D) \land \text{ON}(B, C) \land \text{ON}(A, B) \land \text{CLEAR}(A)$.

The goal state $\Delta_G$ is given by:

$\text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D)$.

The action sequence:

$\langle \text{UNSTACK}(A, B), \text{UNSTACK}(B, C), \text{UNSTACK}(C, D), \langle \text{STACK}(C, A), \text{STACK}(D, B) \rangle \rangle$ is a solution plan.
4. Argument Schemes for Explaining Plans

In scheme-based approaches [17] arguments are expressed in natural language and a set of critical questions is associated with each scheme, identifying how the scheme can be attacked. Below, we introduce a set of argument schemes for explaining a plan and its key elements, i.e., action, concurrent action, state and goal. The set of critical questions allow the user to ask for a summary explanation for the plan and consequently interrogate the elements of the plan. The explanation arguments constructed using the argument schemes allow the planner to answer any user questions.

**Definition 4.1** Given a planning problem P:
- \( \text{Hold}(\text{pre}(a), S) \) indicates that the precondition \( \text{pre}(a) \) of action \( a \) holds at state \( S \).
- \( \text{Execute}(a, S) \) indicates that action \( a \) is executed at state \( S \).
- \( \text{ExecuteC}(a_c, S) \) indicates that all the concurrent actions in the set \( a_c = \{a_0, a_1, \ldots, a_n\} \) are executed at state \( S \).
- \( \text{Achieve}(a, g) \) indicates that action \( a \) achieves goal \( g \).
- \( \text{Solution}(\pi, P) \) indicates that \( \pi \) is a solution to the planning problem \( P \).

**Definition 4.2** (Action Argument Scheme Arg\(_a\)) An action argument Arg\(_a\) explains how it is possible to execute an action \( a \):
- **Premise 1:** \( \text{Hold}(\text{pre}(a), S_1) \). In the current state \( S_1 \), the pre-condition \( \text{pre}(a) \) of action \( a \) holds.
- **Premise 2:** \( \gamma(S_1, a) = S_2 \). When we execute action \( a \) in the current state \( S_1 \), it results in the next state \( S_2 \).
- **Premise 3:** \( \text{Hold}(g, S_2) \). In the next state \( S_2 \), the goal \( g \) holds.
- **Premise 4:** \( \text{Achieve}(a, g) \): Action \( a \) achieves goal \( g \).
- **Conclusion:** \( \text{Execute}(a, S_1) \). Therefore, we should execute action \( a \) in the current state \( S_1 \).

**Example 4.1** We consider the blocks world of Example [3,7] The explanation argument for the first action UNSTACK\((A, B)\) is shown as follows. Where:
- \( \text{pre}(\text{UNSTACK}(A, B)) = \text{CLEAR}(A) \land \text{ON}(A, B) \).
- \( S_1 = \text{ONTABLE}(D) \land \text{ON}(C, D) \land \text{ON}(B, C) \land \text{ON}(A, B) \land \text{CLEAR}(A) \).
- \( S_2 = \text{ONTABLE}(D) \land \text{ON}(C, D) \land \text{ON}(B, C) \land \text{CLEAR}(A) \land \text{ONTABLE}(A) \land \text{CLEAR}(B) \).
- \( g = \text{ONTABLE}(A) \).

**Premise 1:**
\[ \text{Hold}(\text{CLEAR}(A) \land \text{ON}(A, B), \text{ONTABLE}(D) \land \text{ON}(C, D) \land \text{ON}(B, C) \land \text{ON}(A, B) \land \text{CLEAR}(A)) \]
In the current state $ONTABLE(D) \land ON(C,D) \land ON(B,C) \land ON(A,B) \land CLEAR(A)$, the pre-condition $CLEAR(A) \land ON(A,B)$ of action $UNSTACK(A,B)$ holds.

**Premise 2:**

$\gamma(ONTABLE(D) \land ON(C,D) \land ON(B,C) \land ON(A,B) \land CLEAR(A), UNSTACK(A,B)) = ONTABLE(D) \land ON(C,D) \land ON(B,C) \land CLEAR(A) \land ONTABLE(A) \land CLEAR(B)$.

When we execute action $UNSTACK(A,B)$ in the current state $ONTABLE(D) \land ON(C,D) \land ON(B,C) \land ON(A,B) \land CLEAR(A)$, it results in the next state $ONTABLE(D) \land ON(C,D) \land ON(B,C) \land CLEAR(A) \land ONTABLE(A) \land CLEAR(B)$.

**Premise 3:**

$\text{Hold}(ONTABLE(A), ONTABLE(D) \land ON(C,D) \land ON(B,C) \land CLEAR(A) \land ONTABLE(A) \land CLEAR(B))$

In the next state $ONTABLE(D) \land ON(C,D) \land ON(B,C) \land CLEAR(A) \land ONTABLE(A) \land CLEAR(B)$, the goal $ONTABLE(A)$ holds.

**Premise 4:**

$\text{Achieve}(UNSTACK(A,B), ONTABLE(A))$

Action $UNSTACK(A,B)$ achieves goal $ONTABLE(A)$.

**Conclusion:**

$\text{Execute}(UNSTACK(A,B), ONTABLE(D) \land ON(C,D) \land ON(B,C) \land ON(A,B) \land CLEAR(A))$

Therefore, we should execute action $UNSTACK(A,B)$ in the current state $ONTABLE(D) \land ON(C,D) \land ON(B,C) \land ON(A,B) \land CLEAR(A)$.

**Definition 4.3** (Concurrent Action Argument Scheme $Arg_{ac}$) A concurrent action argument $Arg_{ac}$ explains how it is possible to execute all concurrent actions in the set $a_c = \{a_1, a_2, ..., a_n\}$.

- **Premise 1:** $\text{Hold}(\text{pre}(a_1), S_1) \land \text{Hold}(\text{pre}(a_2), S_1) \land ... \land \text{Hold}(\text{pre}(a_n), S_1)$. In the current state $S_1$, the preconditions of all the concurrent actions in the set $a_c$ hold.

- **Premise 2:** $\forall a_i, a_j \in a_c$ (where $i \neq j$) $\gamma(S_1, a_i) = S_2 \land \text{Hold}(\text{pre}(a_j), S_2)$. When we execute any concurrent action $a_i$ in the state $S_1$, it results in the state $S_2$, and the precondition $\text{pre}(a_j)$ of any other concurrent action $a_j$ holds in the state $S_2$.

- **Premise 3:** $\gamma(S_n, a_n) = S_G$. When we execute the last concurrent action $a_n$ in the state $S_n$, it results in the next state $S_G$.

- **Premise 4:** $\text{Hold}(G, S_G)$. In the next state $S_G$, the set of goals $G$ holds.

- **Premise 5:** $\text{Achieve}(a_c, G)$. The set of concurrent actions $a_c$ achieves the set of goals $G$.

- **Conclusion:** $\text{ExecuteC}(a_c, S_1)$. Therefore, we should execute all the concurrent actions in the set $a_c$ in the current state $S_1$.

**Example 4.2** The concurrent action argument $Arg_{ac}$ for the set of concurrent actions $a_c = \{STACK(C,A), STACK(D,B)\}$ in the Example 3.1 is shown as follows. Where:

- $\text{pre}(STACK(C,A)) = ONTABLE(C) \land CLEAR(C) \land CLEAR(A)$,
- $\text{pre}(STACK(D,B)) = ONTABLE(D) \land CLEAR(D) \land CLEAR(B)$,
• $S_1 = \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C)$.
• $S_2 = \text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$.
• $S_G = \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$.
• $G = \{\text{ON}(C, A), \text{ON}(D, B)\}$, i.e., the set of all the goals in the goal state $S_G$ that are not in the state $S_1$.

**Premise 1:**

$\text{Hold(ONTABLE}(C) \land \text{CLEAR}(C) \land \text{CLEAR}(A), \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C))$

\begin{align*}
&\land \text{Hold(ONTABLE}(D) \land \text{CLEAR}(D) \land \text{CLEAR}(B), \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C))
\end{align*}

In the current state $\text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C)$, the precondition $\text{ONTABLE}(C) \land \text{CLEAR}(C) \land \text{CLEAR}(A)$ of action $\text{STACK}(C, A)$ holds and the precondition $\text{ONTABLE}(D) \land \text{CLEAR}(D) \land \text{CLEAR}(B)$ of action $\text{STACK}(D, B)$ holds.

**Premise 2:**

$\gamma(\text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C), \text{STACK}(C, A)) = \text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B))$

When we execute the concurrent action $\text{STACK}(C, A)$ in the state $\text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C)$, it results in the next state $\text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$, and the precondition $\text{ONTABLE}(D) \land \text{CLEAR}(D) \land \text{CLEAR}(B)$ of the other concurrent action $\text{STACK}(D, B)$ holds in the next state $\text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$.

**Premise 3:**

$\gamma(\text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B), \text{STACK}(D, B)) = \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$.

When we execute the last concurrent action $\text{STACK}(D, B)$ in the state $\text{ONTABLE}(D) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$, it results in the next state $\text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$.

**Premise 4:**

$\text{Hold}(\{\text{ON}(C, A), \text{ON}(D, B)\}, \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B))$

In the next state $\text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ON}(C, A) \land \text{ON}(D, B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B)$, the set of goals $\{\text{ON}(C, A), \text{ON}(D, B)\}$ holds.

**Premise 5:**

$\text{Achieve}(\{\text{STACK}(C, A), \text{STACK}(D, B)\}, \{\text{ON}(C, A), \text{ON}(D, B)\})$. 
The set of concurrent actions \{\text{STACK}(C,A), \text{STACK}(D,B)\} achieves the set of goals \{\text{ON}(C,A), \text{ON}(D,B)\}.

Conclusion:
ExecuteC(\{\text{STACK}(C,A), \text{STACK}(D,B)\}, \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C)).

Therefore, we should execute all the concurrent actions in the set \{\text{STACK}(C,A), \text{STACK}(D,B)\} in the current state \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C).

Definition 4.4 (State Transition Argument Scheme Arg_s) A state transition argument Arg_s explains how the state S becomes true:

- Premise 1: \(\gamma(S_1, a) = (S_1 - \text{post}(a)^-) \cup \text{post}(a)^+ = S\). In the current state \(S_1\), we should execute the action \(a \in \pi\) by deleting the negative postconditions \(\text{post}(a)^-\) and adding the positive postconditions \(\text{post}(a)^+\) to \(S_1\), that results in the state \(S\).
- Conclusion: Therefore, the state \(S\) is true.

Example 4.3 The state transition argument Arg_s for the state \(S = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)\) in the Example 3.1 is shown as follows. Where:

- \(a = \text{UNSTACK}(A,B)\).
- \(\text{post}(a)^- = \text{ON}(A,B)\)
- \(\text{post}(a)^+ = \text{ONTABLE}(A) \land \text{CLEAR}(B)\)
- \(S_1 = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A)\).

Premise 1:
\(\gamma(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A), \text{UNSTACK}(A,B)) = (\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A) - \text{ON}(A,B)) \cup \text{ONTABLE}(A) \land \text{CLEAR}(B)\)

\(= \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)\).

In the current state \(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A)\), we should execute the action \text{UNSTACK}(A,B) by deleting the negative postconditions \(\text{ON}(A,B)\) and adding the positive postconditions \(\text{ONTABLE}(A) \land \text{CLEAR}(B)\) to the current state \(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A)\), that results in the state \(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)\).

Conclusion: Therefore, the state \(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)\) is true.

Definition 4.5 (Goal Argument Scheme Arg_g) A goal argument Arg_g explains how a feasible goal is achieved by an action in the plan:

- Premise 1: \(\gamma(S_1, a) = S_2\). In the current state \(S_1\), we should execute the action \(a \in \pi\), that results in the next state \(S_2\).
- Premise 2: \(\text{Hold}(g,S_2)\). In the next state \(S_2\), the goal \(g\) holds.
- Conclusion: Achieve(a,g): Therefore, the action \(a\) achieves the goal \(g\).

\(^1\)A goal is feasible if there is at least one plan that satisfies it.
Example 4.4 The goal argument $\text{Arg}_g$ for the goal $g = \text{ONTABLE}(A)$ in the Example 3.7 is shown as follows. Where:

- $a = \text{UNSTACK}(A,B)$.
- $S_1 = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A)$.
- $S_2 = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)$.

**Premise 1:**
\[
\gamma(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A), \text{UNSTACK}(A,B)) = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A).
\]
In the current state $\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A)$, we should execute the action $\text{UNSTACK}(A,B)$, that results in the next state $\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)$.

**Premise 2:**
\[
\text{Hold}(\text{ONTABLE}(A), \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)).
\]
In the next state $\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A)$, the goal $\text{ONTABLE}(A)$ holds.

**Conclusion:**
\[
\text{Achieve}(\text{UNSTACK}(A,B), \text{ONTABLE}(A)).
\]
Therefore, the action $\text{UNSTACK}(A,B)$ achieves the goal $\text{ONTABLE}(A)$.

**Definition 4.6 (Plan Summary Argument Scheme $\text{Arg}_\pi$)** A plan summary argument $\text{Arg}_\pi$ explains that a proposed sequence of actions $\pi = (a_1,a_2,...,a_n)$ is a solution to the planning problem $P$ because it achieves a set of goals $G$:

- **Premise 1:** $\gamma(S_1,a_1) = S_2, \gamma(S_2,a_2) = S_3,...,\gamma(S_n,a_n) = S_{n+1}$. In the initial state $S_1 = \triangle_1$, we should execute the first action $a_1$ in the sequence of actions $\pi$ that results in the next state $S_2$ and execute the next action $a_2$ in the sequence in the state $S_2$ that results in the next state $S_3$ and carry on until the last action $a_n$ in the sequence is executed in the state $S_n$ that results in the goal state $S_{n+1} = \triangle_G$.
- **Premise 2:** $\text{Hold}(G, \triangle_G)$. In the goal state $\triangle_G$, all the goals in the set of goals $G$ hold.
- **Premise 3:** $\text{Achieve}(\pi, G)$. The sequence of actions $\pi$ achieves the set of all goals $G$.
- **Conclusion:** $\text{Solution}(\pi, P)$. Therefore, $\pi$ is a solution to the planning problem $P$.

Example 4.5 The plan summary argument $\text{Arg}_\pi$ for the solution plan given in the Example 3.7 is shown as follows.

**Premise 1:**
\[
\gamma(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A), \text{UNSTACK}(A,B)) = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A),
\]
\[
\gamma(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A), \text{UNSTACK}(B,C)) = \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B),
\]
\[
\gamma(\text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B), \text{UNSTACK}(C,D)) = \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C).\]
\[ \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C), \]
\[ \gamma(\text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{ONTABLE}(C), (\text{STACK}(C,A), \text{STACK}(D,B))) = \text{ON}(C,A) \land \text{ON}(D,B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D) \]

In the initial state \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{ON}(A,B) \land \text{CLEAR}(A), we should execute the action \text{UNSTACK}(A,B) that results in the next state \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A).

In the state \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{ON}(B,C) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A), we should execute the action \text{UNSTACK}(B,C) that results in the next state \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(C).

In the state \text{ONTABLE}(D) \land \text{ON}(C,D) \land \text{CLEAR}(A) \land \text{CLEAR}(B) \land \text{CLEAR}(C) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B), we should execute the action \text{UNSTACK}(C,D) that results in the next state \text{ONTABLE}(D) \land \text{CLEAR}(A) \land \text{CLE}

In the goal state \text{ON}(C,A) \land \text{ON}(D,B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D).

**Premise 2:**

\[ \text{Hold}(\{\text{ON}(C,A), \text{ON}(D,B), \text{ONTABLE}(A), \text{ONTABLE}(B), \text{CLEAR}(C), \text{CLEAR}(D)\}), \]
\[ \text{ON}(C,A) \land \text{ON}(D,B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D). \]

In the goal state \text{ON}(C,A) \land \text{ON}(D,B) \land \text{ONTABLE}(A) \land \text{ONTABLE}(B) \land \text{CLEAR}(C) \land \text{CLEAR}(D), all the goals in the set of goals \{\text{ON}(C,A), \text{ON}(D,B), \text{ONTABLE}(A), \text{ONTABLE}(B), \text{CLEAR}(C), \text{CLEAR}(D)\} \text{ hold.}

**Premise 3:**

\[ \text{Achieve}(\{\text{UNSTACK}(A,B), \text{UNSTACK}(B,C), \text{UNSTACK}(C,D), (\text{STACK}(C,A), \text{STACK}(D,B))\}), \]
\[ \{\text{ON}(C,A), \text{ON}(D,B), \text{ONTABLE}(A), \text{ONTABLE}(B), \text{CLEAR}(C), \text{CLEAR}(D)\}. \]

The sequence of actions \{(\text{UNSTACK}(A,B), \text{UNSTACK}(B,C), \text{UNSTACK}(C,D), (\text{STACK}(C,A), \text{STACK}(D,B)))\} achieves the set of all goals \{\text{ON}(C,A), \text{ON}(D,B), \text{ONTABLE}(A), \text{ONTABLE}(B), \text{CLEAR}(C), \text{CLEAR}(D)\}.

**Conclusion:**

\[ \text{Solution}(\{\text{UNSTACK}(A,B), \text{UNSTACK}(B,C), \text{UNSTACK}(C,D), (\text{STACK}(C,A), \text{STACK}(D,B))\}, P). \]

Therefore, \{\text{UNSTACK}(A,B), \text{UNSTACK}(B,C), \text{UNSTACK}(C,D), (\text{STACK}(C,A), \text{STACK}(D,B))\} is a solution to the planning problem \(P\).

### 4.1. Argument Interactions and Properties of Plan Arguments

The five critical questions (CQs) given below describe the ways in which the arguments built using the argument schemes can interact with each other. These CQs are associated to (i.e., attack) one or more premises of the arguments constructed using the argument schemes and are in turn answered (i.e., attacked) by the other arguments, which are listed in the description.
CQ1: Is the plan $\pi$ possible? This CQ begins the dialogue with the user, and it is the first question that the user asks when presented with a solution plan $\pi$. The argument scheme $\text{Arg}_\pi$, answers the CQ by constructing the summary argument for the plan $\pi$.

CQ2: Is the action $a$ possible to execute? This CQ is associated with the following argument schemes: $\text{Arg}_{\pi}$, $\text{Arg}_{\delta}$, $\text{Arg}_{\alpha}$. The argument scheme $\text{Arg}_{\alpha}$, answers the CQ by constructing the explanation argument for the action $a$.

CQ3: Is the set of concurrent actions $a_c$ possible to execute? This CQ is associated with the argument schemes $\text{Arg}_{\delta}$. The argument scheme $\text{Arg}_{\delta}$, answers the CQ by constructing the explanation argument for the set of concurrent actions $a_c$.

CQ4: Is the state $S$ possible? This CQ is associated with the following argument schemes: $\text{Arg}_{\pi}$, $\text{Arg}_{\alpha}$, $\text{Arg}_{\gamma}$, $\text{Arg}_{\delta}$. The argument scheme $\text{Arg}_{\delta}$, answers the CQ by constructing the explanation argument for the state $S$.

CQ5: Is the goal $g$ possible to achieve? This CQ is associated with the argument scheme $\text{Arg}_{\pi}$. The argument scheme $\text{Arg}_{\pi}$, answers the CQ by constructing the explanation argument for the goal $g$.

We organise the arguments and their interactions, as presented earlier, by mapping into a Dung AF = ($\mathcal{A}$, $\mathcal{R}$) [5], where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$). $\text{Args} \subset \mathcal{A}$ and $\text{CQs} \subset \mathcal{A}$, where $\text{Args} = \{\text{Arg}_{\pi}, \text{Arg}_{\alpha}, \text{Arg}_{\gamma}, \text{Arg}_{\delta}, \text{Arg}_{\delta}\}$ and $\text{CQs} = \{CQ_1, CQ_2, CQ_3, CQ_4, CQ_5\}$. We use the grounded extension of the AF, denoted by $Gr$, to determine if a plan should be acceptable.

**Property 1:** For a plan $\pi$, the set of arguments $\text{Args}$ is complete, in that, if a $CQ \in \text{CQs}$ exists, then it will be answered (i.e., attacked) by an $\text{Arg} \in \text{Args}$.

**Proof.** Since, $(\text{Arg}_{\pi}, CQ_1) \in \mathcal{R}$, $(\text{Arg}_{\alpha}, CQ_2) \in \mathcal{R}$, $(\text{Arg}_{\gamma}, CQ_3) \in \mathcal{R}$, $(\text{Arg}_{\delta}, CQ_4) \in \mathcal{R}$, and $(\text{Arg}_{\delta}, CQ_5) \in \mathcal{R}$, therefore, a unique $\text{Arg} \in \text{Args}$ exists, that attacks a unique $CQ \in \text{CQs}$. Thus, $\text{Args}$ is complete.

**Property 2:** For a plan $\pi$, $\text{Arg}_{\pi} \in Gr$ iff $CQ \notin Gr$ when $(CQ, \text{Arg}_{\pi}) \in \mathcal{R}$, $CQ \in \text{CQs}$.

**Proof.** Follows from Property 1. Since any CQ that attacks $\text{Arg}_{\pi}$ is in turn attacked by an $\text{Arg} \in \text{Args}$, therefore, $CQ \notin Gr$. Thus, $\text{Arg}_{\pi} \in Gr$.

**Property 3:** For a plan $\pi$, $\text{Arg}_{\pi} \in Gr$ iff $\forall g \in G \text{Arg}_{\delta} \in Gr$.

**Proof.** Since plan $\pi$ achieves all goals $g \in G$, and $\text{Arg}_{\delta} \in \text{CQ}$ attack all $CQ$s that attack the goal premises of $\text{Arg}_{\pi}$, therefore, $\forall g \in G \text{Arg}_{\delta} \in Gr$. Thus, $\text{Arg}_{\pi} \in Gr$.

**Property 4:** For a plan $\pi$, $\text{Arg}_{\pi} \in Gr$ iff the explanation is acceptable to the user.

**Proof.** Follows immediately from Properties 1, 2 and 3.

5. Conclusions and Future Work

In this paper, we have presented a novel argument scheme-based approach for creating explanation arguments in the domain of AI planning. The main contributions of our work are as follows: (i) We formalise a set of argument schemes that help in constructing arguments that explain a plan and its key elements; (ii) We present critical questions that allow the user to seek further information regarding the key elements of the plan, and the interaction between different arguments; (iii) We present properties of the plan arguments.
In the future, we aim to develop algorithms based on the argument schemes to automatically extract the arguments from the input planning model. Furthermore, we intend to build a scheme-based dialogue system for creating interactive dialectical explanations. Finally, our approach to generating explanation arguments is planner independent and therefore, can work on a wide range of input plans in classical planning, and in the future, we intend to extend this to richer planning formalisms such as partial order planning.

Acknowledgments: This work was supported by EPSRC grant EP/R033722/1, Trust in Human-Machine Partnership.

References

[1] K. Atkinson and T. J. M. Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. Artif. Intell., 171(10-15):855–874, 2007.
[2] A. Belesiotis, M. Rovatsos, and I. Rahwan. Agreeing on plans through iterated disputes. In 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010. Volume 1-3, pages 765–772. IFAAMAS, 2010.
[3] M. Caminada, R. Kutlak, N. Oren, and W. W. Vasconcelos. Scrutable plan enactment via argumentation and natural language generation. In AAMAS, 2014.
[4] M. Caminada and M. Podlaszewski. Grounded semantics as persuasion dialogue. In Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012, volume 245 of Frontiers in Artificial Intelligence and Applications, pages 478–485. IOS Press, 2012.
[5] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence, 77:321–357, 1995.
[6] P. M. Dung, R. A. Kowalski, and F. Toni. Assumption-Based Argumentation. Springer US, 2009.
[7] X. Fan. On generating explainable plans with assumption-based argumentation. In PRIMA 2018: Principles and Practice of Multi-Agent Systems, pages 344–361, 2018.
[8] X. Fan and F. Toni. On computing explanations in argumentation. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, pages 1496–1502. AAAI Press, 2015.
[9] M. Fox, D. Long, and D. Magazzeni. Explainable planning. 2017.
[10] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. An Introduction to the Planning Domain Definition Language, pages 1–169. Number 2 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2 edition, 2019.
[11] N. Oren. Argument schemes for normative practical reasoning. In Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers, volume 8306 of Lecture Notes in Computer Science, pages 63–78. Springer, 2013.
[12] Z. Shams, M. D. Vos, N. Oren, and J. A. Padget. Normative practical reasoning via argumentation and dialogue. In S. Kambhampati, editor, Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pages 1244–1250. IJCAI/AAAI Press, 2016.
[13] D. Smith. Planning as an iterative process. Proceedings of the National Conference on Artificial Intelligence, 3:2180–2185, 01 2012.
[14] Y. Tang and S. Parsons. Argumentation-based dialogues for deliberation. In 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands, pages 552–559. ACM, 2005.
[15] A. Tonio, T. J. Norman, and K. P. Sycara. Argumentation schemes for collaborative planning. In Agents in Practice - 14th International Conference, PRIMA 2011, Wollongong, Australia, November 16-18, 2011. Proceedings, volume 7047 of Lecture Notes in Computer Science, pages 323–335. Springer, 2011.
[16] A. Torreo, E. Onaindia, A. Komenda, and M. Tolba. Cooperative multi-agent planning. ACM Computing Surveys, 50(6):132, 2018.
[17] D. Walton. Argumentation Schemes for Presumptive Reasoning. Argumentation Schemes for Presumptive Reasoning. L. Erlbaum Associates, 1996.