Question-Based Explainability in Abstract Argumentation

Philippe Besnard, Sylvie Doutre, Théo Duchatelle, Marie-Christine Lagasquie-Schiex

Question-Based Explainability in Abstract Argumentation

Philippe Besnard
Sylvie Doutre
Théo Duchatelle
Marie-Christine Lagasquie-Schiex

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3
118 route de Narbonne, 31062 Toulouse, France
{theo.duchatelle, besnard, doutre, lagasq}@irit.fr

Tech. Report
IRIT/RR- -2022- -01- -FR

April 2022

Abstract

This paper explores the definition of questions and the computation of explanations in general, and in the specific context of abstract argumentation. We aim at 1) defining a methodological way to generate questions asking for explanations in a certain context, and 2) defining explanations based on the questions they answer and on the context in which they are asked. Applied to abstract argumentation, the explanations that we define are designed to be visual, in the sense that they take the form of subgraphs of the argumentation graph which is a part of the context of the questions they apply to. Moreover, these explanations rely on the modular aspects of abstract argumentation semantics and can consequently be either aggregated or decomposed. Finally, we also investigate the adequacy of the explanations with several desirable properties that they should possess, with a particular focus on Grice's maxims of conversation.

Contents

1 Introduction

When it comes to explanations of decisions made using an Artificial Intelligence system, Abstract Argumentation (introduced in [1]) is increasingly studied as a formal tool to provide them. In Abstract Argumentation, the main object of study is an Argumentation Framework, which is a graph in which nodes are abstract arguments (in the sense that their internal structure is left unspecified) and the binary relation is a conflict relation. Given an Argumentation Framework, the objective is to find the arguments that can collectively be deemed receivable according to some criteria. Such sets of arguments are called *extensions* and the criteria that are considered desirable give rise to *semantics* for selecting extensions. The recent survey [2] indicates that Argumentation can be used to generate explanations in various domains (machine learning notably) and that explanations for the argumentative process itself are also necessary.

In this respect, the main questions which have been addressed so far concern the global acceptability status (credulous or skeptical) of an argument or of a set of arguments. In addition, the approach that is most often used consists in identifying some set(s) of arguments which act as explanation(s) ([3, 4, 5, 6, 7, 8]). One may however argue that, since the argumentative process of Abstract Argumentation already provides ways for selecting arguments, explaining this process by more selection of arguments (although different ones) may not be of much help. Furthermore, beyond the question of the global acceptability of an argument or a set of arguments, many other questions on the outcomes of argumentation or on the process of argumentation itself can be asked.

More generally, for any process that uses some context to compute some result, questions may arise. In this respect, we define a general formal tool to generate a wide range of questions. Using it, our answers (i.e. explanations) are tailored to the way a question is structured and generated. In the context of Abstract Argumentation, our answers take the form of relevant subgraphs, as in [9, 10, 11]. As such, our approach is a visual one, which has been shown to be helpful for humans to comply with reasoning principles [12], and which not only highlights arguments, but also subsets of attacks.

The paper is organised as follows: Section 2 recalls background notions relative to abstract argumentation, formal grammars and graphs theory. In Section 3 we define a general tool for the generation of questions and give

a specific instance for the domain of Abstract Argumentation. In Section 4, we define our explanations for Abstract Argumentation as answers to specific questions generated using our general tool. In Section 5, we provide a discussion on the quality of our explanations. Section 6 relates our approach to other existing ones. Section 7 concludes and presents some future works.

2 Preliminary Notions

In this section, we give the background notions that will be of use in this paper. They include some definitions about Argumentation frameworks, formal grammars and the description of some important operations over graphs.

2.1 Abstract Argumentation

We begin by recalling basic notions on Abstract Argumentation. The object handled in this formalism is called an Argumentation Framework.

Definition 1 (Argumentation Framework ([1]))**.** A *Dung's Argumentation Framework* is an ordered pair (A, R) such that $R \subseteq A \times A$.

Each element $a \in A$ is called an *argument* and aRb means that a *attacks* b. For $S \subseteq A$, we say that S attacks $a \in A$ iff bRa for some $b \in S$. Any argumentation framework can be represented as a directed graph.

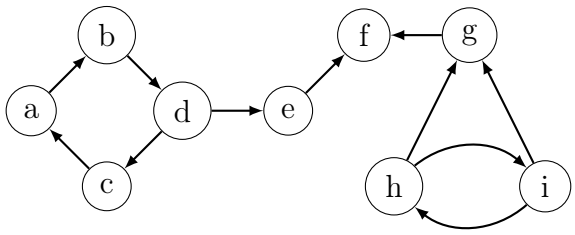


Figure 1: Example of an Argumentation Framework (AF) from [13]

The main asset of Dung's approach is the definition of semantics using some basic properties in order to define sets of acceptable arguments, as follows.

Definition 2. Given (A, R), an argument $a \in A$ is *acceptable* wrt $S \subseteq A$ iff for all $b \in A$, if bRa then cRb for some $c \in S$.

Definition 3. The *characteristic function* of $\mathcal{A} = (A, R)$ is $F_{\mathcal{A}} : 2^A \to 2^A$ such that $F_{\mathcal{A}}(S) = \{a \in A \mid a \text{ is acceptable wrt } S\}$ for any $S \subseteq A$.

The semantics originally defined in [1] are as follows.

Definition 4. Given $\mathcal{A} = (A, R)$, a subset S of A is said to be:

- *conflict-free* iff there are no a and b in S such that a attacks b,
- *admissible* iff S is conflict-free and for any $a \in S$, a is acceptable wrt S,
- *complete* iff S is admissible and for any $a \in A$, if a is acceptable wrt S then $a \in S$,
- *preferred* iff S is maximal (in the sense of set-inclusion) admissible,[1]
- *grounded* iff S is the least fixpoint for $F_{\mathcal{A}}$,
- *stable* iff S is conflict-free and S attacks any $a \in A \setminus S$.

Some properties have been proven in [1] establishing a link between the different semantics. For instance:

Proposition 1. *Given $\mathcal{A} = (A, R)$:*

- *There exists at least one preferred extension.*
- *Every preferred extension is complete, but not vice-versa.*
- *Every stable extension is preferred, but not vice-versa.*
- *The grounded extension is the least (with respect to set-inclusion) complete extension.*

Table 1 illustrates these semantics for the AF given in Figure 1.

[1] We write $\subseteq$-*maximal*.

	Admissible	Complete	Preferred	Grounded	Stable
$\varnothing$	✓	✓		✓	
$\{h\}$	✓	✓			
$\{i\}$	✓	✓			
$\{a, d\}$	✓	✓			
$\{b, c\}$	✓				
$\{a, d, h\}$	✓				
$\{a, d, i\}$	✓				
$\{b, c, h\}$	✓				
$\{b, c, i\}$	✓				
$\{b, c, e\}$	✓	✓			
$\{a, d, h, f\}$	✓	✓	✓		✓
$\{a, d, i, f\}$	✓	✓	✓		✓
$\{b, c, e, h\}$	✓	✓	✓		✓
$\{b, c, e, i\}$	✓	✓	✓		✓

Table 1: Acceptable sets of the AF of Figure 1 under the different semantics

2.2 Formal Grammars

Formal grammars provide a mechanism to generate languages, that is to say, sets of aggregation of symbols. The most known instance of formal grammars are Context-free grammars, that are used for example to describe programming languages. In this paper, we will use formal grammars as a way to generate the questions that we wish to provide answers to. This will allow us to link a certain answer to a certain pattern of questions. Note that, in the following definitions, we will use the writing conventions of the Backus-Naur Form (BNF, [14]) when describing formal grammars.

Definition 5 (Formal grammar ([15]))**.** A formal grammar is a 4-tuple (N, T, P, S) such that:

- N is a *finite* set of symbols that we call *nonterminal symbols*
- T is a *finite* set of symbols that we call *terminal symbols*
- P is a set of *production rules* of the form α `::=` β where α is a chain of terminal and nonterminal symbols that contains at least one nonterminal symbol and β is a chain of terminal and nonterminal symbols

- $S \in N$ is a particular nonterminal symbol that we call *start symbol*

In Context-free grammars, a nonterminal symbol always produces the same possibilities, no matter the words (a word being a set of symbols) that may be present around it. In a Context-sensitive grammar however, a nonterminal symbol can produce *different* possibilities *depending on* the words that surround it. Considering the same nonterminal symbol, there is then one production rule for each possible context. Context-sensitive grammars are thus more complex but also more general than Context-free grammars. Here, we have chosen to use and to present Context-sensitive grammars.

Definition 6 (Context-sensitive grammar [15])**.** A formal grammar (N, T, P, S) is *context-sensitive* if all its rules are of the form

$$uXv \texttt{ ::= } uxv$$

where $X \in N$ and u, v, x are arbitrary words. We call u the *left context* of X and v the *right context* of X.

Example. Consider the formal grammar (N, T, P, S) such that:

- $N = \{\texttt{<S>}, \texttt{<B>}, \texttt{<C>}, \texttt{<W>}, \texttt{<Z>}\}$
- $T = \{\texttt{a}, \texttt{b}, \texttt{c}\}$
- $S = \texttt{<S>}$

and P contains the following production rules:

1. `<S> ::= "a"<S><B><C>`
2. `<S> ::= "a"<B><C>`
3. `<C><B> ::= <C><Z>`
4. `<C><Z> ::= <W><Z>`
5. `<W><Z> ::= <W><C>`
5. `<W><C> ::= <B><C>`
6. `"a"<B> ::= "ab"`
7. `"b"<B> ::= "bb"`
8. `"b"<C> ::= "bc"`
9. `"c"<C> ::= "cc"`

This grammar generates the language $\{a^n b^n c^n \mid n \geq 1\}$. Note that this language cannot be generated by a Context-free grammar.

Looking at production rule 7, we see that <B> corresponds to the X in Definition 6. Consequently, its left context is "a" and its right context is the empty word. It turns into the word "b" which corresponds to the x in Definition 6. As an instance, we detail the production of the word $a^2b^2c^2$. The application of rules 1 and 2 gives "aa"<B><C><B><C>. Following this, rules 3, 4, 5 and 6 give "aa"<B><B><C><C>.[2] Then, by applying rules 7 and 8, we obtain "aabb"<C><C>. Finally, using rules 9 and 10 we have "aabbcc".[3]

2.3 Operations on Graphs

In this section we essentially provide ways to compute graphs from other graphs. We will use these operations to compute our explanations in later sections.

Definition 7 (Subgraph)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We say that G_1 is a *subgraph* of G_2 iff $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

So a subgraph G_1 of G_2 is included in G_2. The next definitions describe some particular subgraphs that are studied a lot in the literature.

Definition 8 (Induced subgraph)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We say that G_1 is an *induced subgraph of* G_2 (or the *induced subgraph of* G_2 *by* V_1) if G_1 is a subgraph of G_2 and for all $a, b \in V_1$, $(a, b) \in E_1$ iff $(a, b) \in E_2$.

We denote the induced subgraph of G by S as $G[S]_V$.

So, considering a graph G_2 and its induced subgraph G_1 by a set of *vertices* S, some vertices of G_2 can be missing in G_1 but all the edges concerning the kept vertices must be present in G_1.

Example. Figure 2 represents the induced subgraph of Figure 1 by $\{a, b, c, d, e, f\}$. Figure 3 represents the induced subgraph of the same graph by $\{e, f, g, h, i\}$.

Definition 9 (Partial subgraph)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We say that G_1 is a *partial subgraph of* G_2 (or the *partial subgraph of* G_2 *by* E_1) if G_1 is a subgraph of G_2 and $V_1 = V_2$.

We denote the partial subgraph of G by S as $G[S]_E$.[4]

[2] These four rules are used in order to transform <C><B> into <B><C>.

[3] Note that, in the rules 1 and 2, the numbers of "a", <B> and <C> are strictly identical; moreover the other rules cannot modify these numbers.

[4] The name *covering subgraph* is also used in the literature.

So, considering a graph G_2 and its partial subgraph G_1 by a set of *edges* S, all the vertices of G_2 must be present in G_1 but some edges can be missing in G_1.

Example. Figure 4 represents the partial subgraph of Figure 1 by $\{(a,b), (b,d), (d,c), (c,a), (d,e), (e,f)\}$. Figure 5 represents the partial subgraph of the same graph by $\{(g,f), (e,f), (h,g), (i,g), (h,i), (i,h)\}$.

Induced and partial subgraphs are examples of ways to compute a graph from another single graph. We now turn to how to combine several graphs in order to obtain another one. These are basically extensions of set-union and set-intersection to graphs.

Definition 10 (Graph union)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We define the *union* of G_1 and G_2 by $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

The union of two graphs, much like the union of two sets, represents the aggregation of the information contained in the two graphs.

Example. Let G_1, G_2, G_3, G_4 be the graphs of Figures 2, 3, 4 and 5 respectively. Then $G_1 \cup G_2$ and $G_3 \cup G_4$ are both represented by Figure 1.

Definition 11 (Graph intersection)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We define the *intersection* of G_1 and G_2 by $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$.

The intersection of two graphs, much like the intersection of two sets, contains the information that is present in both graphs at the same time.

Example. Let G_1, G_2, G_3, G_4 be the graphs of Figures 2, 3, 4 and 5 respectively. Then Figure 6 represents $G_1 \cap G_2$ while Figure 7 represents $G_3 \cap G_4$.

Finally, we recall the graph-theoretic notions of *successor function* and *predecessor function.*

Definition 12 (Successor and predecessor functions)**.** Let $G = (V, E)$ be a graph. The successor function of G is the function $E^+ : V \mapsto 2^V$ such that $E^+(v) = \{u \mid (v,u) \in E\}$ and the predecessor function of G is the function $E^- : V \mapsto 2^V$ such that $E^-(v) = \{u \mid (u,v) \in E\}$.
We also define extensions of the successor and predecessor functions to sets of vertices, *which we continue to denote by* E^+ *and* E^-, by $E^+(S) = \bigcup_{v \in S} E^+(v)$ and $E^-(S) = \bigcup_{v \in S} E^-(v)$, with $S \subseteq V$.

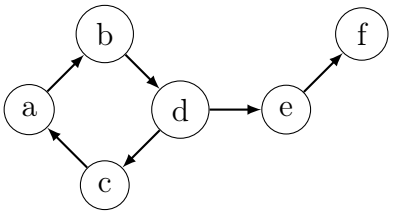


Figure 2: Induced subgraph of Figure 1 by $\{a, b, c, d, e, f\}$

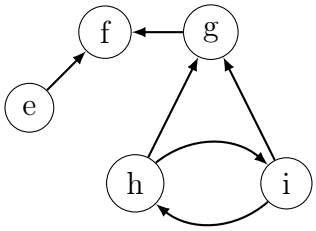


Figure 3: Induced subgraph of Figure 1 by $\{e, f, g, h, i\}$

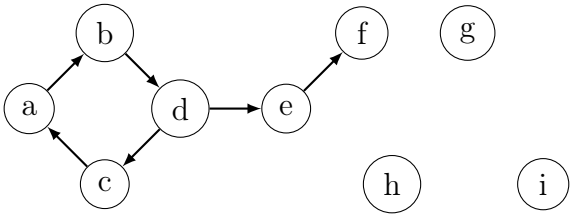


Figure 4: Partial subgraph of Figure 1 by $\{(a, b), (b, d), (d, c), (c, a), (d, e), (e, f)\}$

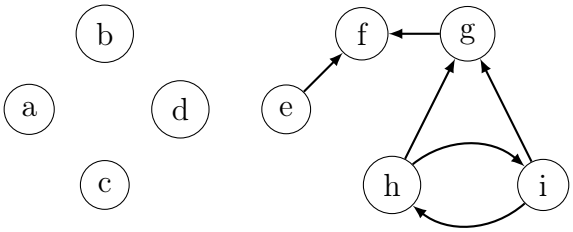


Figure 5: Partial subgraph of Figure 1 by $\{(g, f), (e, f), (h, g), (i, g), (h, i), (i, h)\}$

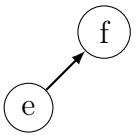


Figure 6: Graph intersection of Figure 2 and Figure 3

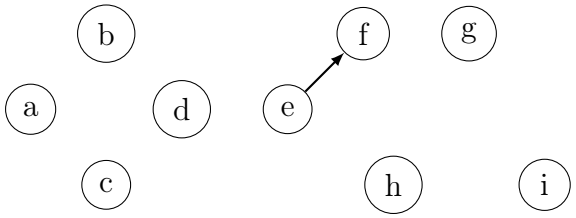


Figure 7: Graph intersection of Figure 4 and Figure 5

Note that, considering an Argumentation Framework, the successor function represents the arguments that are attacked by some argument(s) and the predecessor function represents the attackers of some argument(s). Since an Argumentation Framework is usually denoted by (A, R), the successor and predecessor functions are thus denoted R^+ and R^- in this context.

From the successor and predecessor functions, we define what we call their *n-step* extensions.

Definition 13 (N-step successor and predecessor functions)**.** Let $G = (V, E)$ be a graph and $n \geq 0$.

The n-step successor function of G is $E^{+n}(v) = \overbrace{E^+ \circ \cdots \circ E^+}^{\text{n times}}(v)$ and its n-step predecessor function is $E^{-n}(v) = \overbrace{E^- \circ \cdots \circ E^-}^{\text{n times}}(v)$. By convention, we have $E^{+0}(v) = E^{-0}(v) = v$.

Remark. We have $E^{+1}(v) = E^+(v)$ and $E^{-1}(v) = E^-(v)$

Thus, considering an Argumentation Framework, $R^{+2}(a)$ is the set of arguments defended by argument a, whereas $R^{-2}(a)$ is the set of defenders of a.

3 A grammar for generating questions

In this section, we present an instance of Context-sensitive grammar that generates an array of questions asking for explanations that we wish to be as close as possible to natural language. Of course with this constraint, this grammar could be simplified to become more concise and efficient. We describe this grammar as adaptable to the domain on which the questions are asked. We begin with the elements of the grammar that are independent from the domain and how to structure and use elements from the domain to instantiate such a grammar. We then give an instance relative to Abstract Argumentation that we will use throughout this paper.

3.1 A generic domain-independent grammar

We begin by defining the symbols used in this grammar. They represent the part of the grammar that is not relative to a specific domain.

Definition 14 (Questions grammar's symbols)**.**

- The set T of terminal symbols is the set of all letters, lowercase and capital, used in English, as well as the symbols "?" and "␣" (blank space)
- The set N of nonterminal symbols is {`<Q>`, `<RefState>`, `<ContrState>`, `<ElemInt>`, `<Prop>`, `<ContInfo>`}
- The start symbol is `<Q>`

The nonterminal symbols determine the grammatical structure of these questions. The structure we consider is a question made of one and possibly two constructions that we call *statements* and that can be put in contrast. The first statement of the question is called the *reference statement* and the second is called the *contrast statement*. They both have the same organisation, that is to say they are both made of the same objects, in the same order. These objects are *elements of interest*, *properties* and *contextual information*. What is represented by these objects is largely dependant on the domain we want to ask questions on. These abstract objects are thus used to adapt and instantiate the grammar to a specific domain. A schema of the structure of questions we wish to generate using this grammar can be seen on Figure 8.

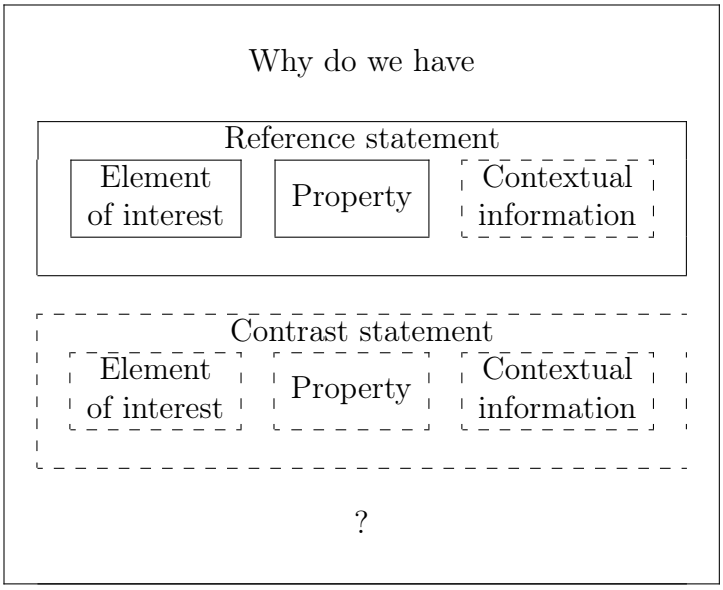


Figure 8: Structure of questions considered in this paper. Optional components are shown using dashed lines.

The reader may have noticed that, in order to complete our definition of a grammar, we still need a set of rules of production. The grammar's rules of production will in fact rely on the dependencies between elements of interest, properties and contextual information. Hence, we begin by formalising the abstract concepts we talked about previously.

Definition 15 (Constitutive elements)**.** We call *constitutive elements* a 5-tuple (E, P, C, D_P, D_C) where:

- E is a *finite* set of *elements of interest*
- P is a *finite* set of *properties* on elements of interest
- C is a *finite* set of *contextual information* for the properties
- $D_P : E \mapsto 2^P$ is a *dependence function* that associates an element of interest with the properties that can be assigned to it
- $D_C : P \mapsto 2^C$ is a *dependence function* that associates a property with the contextual information that is relevant for it

The idea is that elements of interest should represent precisely defined objects that we handle in the domain with which we wish to instantiate the grammar. Properties should then be understood as particular conditions that can be verified or not on these objects. To finish, Contextual information should represent additional parameters that are used to verify whether or not the properties hold.

We then define the reverse functions[5] of the dependence functions, that we will need in the definition of the productions rules.

Definition 16 (Reverse of dependence functions)**.** Let (E, P, C, D_P, D_C) be constitutive elements. We define the reverse of the dependence functions as:

- $D_P^- : P \mapsto 2^E$ is a function defined for all $p \in P$ by $D_P^-(p) = \{e \mid e \in E, p \in D_P(e)\}$
- $D_C^- : C \mapsto 2^P$ is a function defined for all $c \in C$ by $D_C^-(c) = \{p \mid p \in P, c \in D_C(p)\}$

Thus, given an element of interest e, $D_P(e)$ represents the properties that may hold on e; similarly given a property p, $D_P^-(p)$ represents the elements of interest on which p may hold. We now give the rules of production of our grammar.[6]

Definition 17 (Production rules for questions grammar)**.** Given constitutive elements (E, P, C, D_P, D_C), we define the following rules to produce questions requiring answers about these elements:

1.
```
<Q> ::= "Why do we have " <RefState> [<ContrState>] "?"
```

2.
```
<RefState> ::= <ElemInt> <Prop> [<ContInfo>]
```

3. Let $E = \{e_1, \ldots, e_n\}$. Then,

[5] We call them reverse functions because they are somewhat similar in spirit to inverse functions, but are not exactly inverse functions.

[6] Note that following our wish to generate questions that are as close as possible to natural language, the grammar generates questions in natural language. Of course, these questions could made more concise, for instance by replacing the beginning sequence `"Why do we have"` by `"Why"`.

> <ElemInt> ::= "e_1 " | ...| "e_n "

4. For all $e \in E$, let $D_P(e) = \{p_1, \ldots p_n\}$. Then, for all $e \in E$,

> "e " <Prop> ::= "e " ["not "] ("p_1 " | ...| "p_n ")

5. For all $p \in P$, let $D_C(p) = \{c_1, \ldots, c_n\}$. Then, for all $p \in P$,

> "p " <ContInfo> ::= "p " ("c_1 " | ...| "c_n ")

6.

> <ContrState> ::= "and not " [<ElemInt>] [<Prop>] [<ContInfo>]

7. For all $e \in E$, let $D_P(e) = \{p_1, \ldots, p_n\}$. Then, for all $e \in E$,

> "e " ["not "] ("p_1 " | ...| "p_n ") [<ContInfo>] "and not "
> <Prop> ::=
> "e " ["not "] ("p_1 " | ...| "p_n ") [<ContInfo>] "and not "
> ("p_1 " | ...| "p_n ")

8. For all $p \in P$, let $D_C(p) = \{c_1, \ldots, c_n\}$ and let $D_P^-(p) = \{e_1, \ldots, e_m\}$. Then, for all $p \in P$,

> ("e_1 " | ...| "e_m ") ["not "] "p " [<ContInfo>] "and not "
> <ContInfo> ::=
> ("e_1 " | ...| "e_m ") ["not "] "p " [<ContInfo>]
> "and not " ("c_1 " | ...| "c_n ")

9. For all $p \in P$, let $D_C(p) = \{c_1, \ldots, c_n\}$ and let $D_P^-(p) = \{e_1, \ldots, e_m\}$. Then, for all $p \in P$,

> ("e_1 " | ...| "e_m ") ["not "] "p " [<ContInfo>] "and not "
> ("e_1 " | ...| "e_m ") <ContInfo> ::=
> ("e_1 " | ...| "e_m ") ["not "] "p " [<ContInfo>]
> "and not " ("e_1 " | ...| "e_m ") ("c_1 " | ...| "c_n ")

Note. The members of the sets E, P and C must be strings of terminal symbols.

This grammar for questions is generic, but in the next section it will be instantiated in a specific context, and examples of sentences that it can generate will be built.

Using these rules, a question is enforced to possess a reference statement and may or may not possess a contrast statement (rule 1). A reference statement must be about an element of interest and a property of that element, but contextual information is optional (rule 2). A property following an element of interest must be chosen from the properties that can be applied on this element via the corresponding dependence function (rule 4). The same goes for contextual information with properties (rule 5). The three objects of a contrast statements are all optional but must follow the same order as in the reference statement (rule 6). If a property is present in a contrast statement devoid of element of interest, we consider that it refers to the reference statement's one. As such, it must be once again chosen from the properties that can be applied on this element via the corresponding dependence function (rule 7). The same goes for contextual information in a contrast statement devoid of property (rule 8 and 9).

Remark. With these rules of production, it is possible to generate structurally incorrect questions. For instance, using rule 6 and yielding the empty string after `"and not "`. Another example is a question like `"Why do we have e p and not e ?"` creating a self-contrast on e. This results from our wish to keep the rules as few and simple as possible. Please note that, on the other hand, since we cannot predict what question a user will ask, this allows for modelling users that may deliberately ask incorrect questions. Concerning rule 6, we are only interested in contrastive questions in which at least one object of the constrast statement is present.

3.2 An instance concerning Abstract Argumentation

We now turn to define the specific instance of our grammar we use for Abstract Argumentation. As previously said, to instantiate our grammar, one only needs to provide the constitutive elements, the production rules then follow mechanically. Hence, it is the subject of our next definition.

The constitutive elements we use are based on the object that we consider, that is to say, an Argumentation Framework. In order to have an idea of what kind of questions could be asked, we make the following hypothesis on the situation in which the need for explanation occurs:

A user asks for an explanation after she has been presented the result of a Formal Argumentation process (typically the selection of arguments via a semantics) by a program that we will refer to as the system. (H1)

With this in mind, we now define the constitutive elements that we use in this paper.

Definition 18 (Constitutive elements for Abstract Argumentation)**.** Let (A, R) be any AF. Let x denote any argument ($x \in A$) and S denote any set of arguments ($S \subseteq A$). We consider the following constitutive elements:

- $E = E_x \cup E_S$ with $E_x = \{\texttt{x}, \texttt{x} \in A\}$ and $E_S = \{\texttt{S}, \texttt{S} \subseteq A\}$
- $P = \{\texttt{accepted}, \texttt{as a conflict-free extension}, \texttt{as an admissible extension}, \texttt{as a complete extension}, \texttt{as a stable extension}\}$
- $C = \{\texttt{in the extension S}, \texttt{for the conflict-free semantics}, \texttt{for the admissible semantics}, \texttt{for the complete semantics}, \texttt{for the stable semantics}, \texttt{in the AF (A,R)}\}$
- $D_P(\texttt{x}) = \{\texttt{accepted}\}$ for any $\texttt{x} \in E_x$
- $D_P(\texttt{S}) = P$ for any $\texttt{S} \in E_S$
- $D_C(\texttt{accepted}) = C$
- $D_C(\texttt{as a conflict-free extension}) = \{\texttt{in the AF (A,R)}\}$
- $D_C(\texttt{as an admissible extension}) = \{\texttt{in the AF (A,R)}\}$
- $D_C(\texttt{as a complete extension}) = \{\texttt{in the AF (A,R)}\}$
- $D_C(\texttt{as a stable extension}) = \{\texttt{in the AF (A,R)}\}$

Once again, before we detail our definition, please not that, as we already said in the introduction of Section 3, the grammar is aimed at generating questions that are as close as possible to natural language. Hence, some constitutive elements could be simplified. For example, we could have:

$P = \{\texttt{accepted}, \texttt{conflict-free}, \texttt{admissible}, \texttt{complete}, \texttt{stable}\}$

Elements of interest are supposed to capture what the user will focus on when asking questions. In the hypothetical situation in which we place ourselves in, it seems reasonable to assume that, if the need of explanation rises from being presented an extension, then the focus of the questions will be on arguments or groups of arguments.[7]

Once elements of interest have been introduced, selecting properties for them more or less comes down to identifying in which state they can be met. In our hypothetical situation, the user is presented an extension by the system. Hence, sets of arguments can have the property of being an extension for a certain semantics. Moreover, particular arguments can have the property of being part of such an extension, which can then be extended quite naturally to groups of arguments as well.

Finally, contextual information is what we need in order to verify the properties over the elements of interest. In order to verify if a given set of arguments is indeed an extension for a given semantics, we need the Argumentation Framework from which this set of arguments is taken from. Once again (A, R) is just a general notation scheme here, and it is perfectly allowed to ask questions on Argumentation Frameworks using different notations like (A', R'). The Argumentation Framework is also needed in order to verify that an argument or a group of arguments is part of an extension, along with this extension and the semantics under which it was computed. Please note that these three pieces of information represent everything that define the need for explanation in our hypothetical situation: some extension, computed under some semantics, in some Argumentation Framework. Hence, we will refer to these pieces of information collectively as *the question's context*.

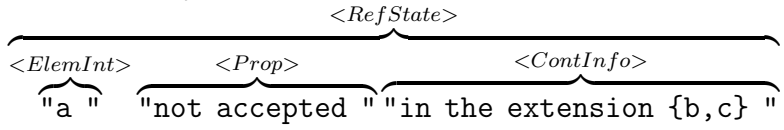
We now give some examples of questions that can be generated using our instantiation of our grammar.

Example (Derivation of a question on the acceptability of one argument).

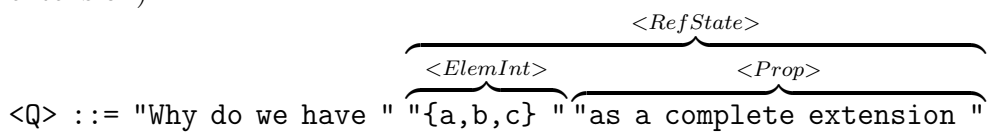
$$\texttt{<Q> ::= "Why do we have "} \overbrace{\underbrace{\texttt{"a "}}_{<ElemInt>}\ \underbrace{\texttt{"accepted "}}_{<Prop>}}^{<RefState>} \texttt{"?"}$$

[7] Even though we only used notations `x` and `S`, these are supposed to be general notation schemes and are not exclusive. Thus, it is perfectly allowed to ask questions about some precise arguments or sets of arguments using different notations such as a, b, c for arguments and T, U, S' for sets of arguments. We will use lowercase letters for arguments and capital letters for sets of arguments (except A and R that are used in the notation of an AF).

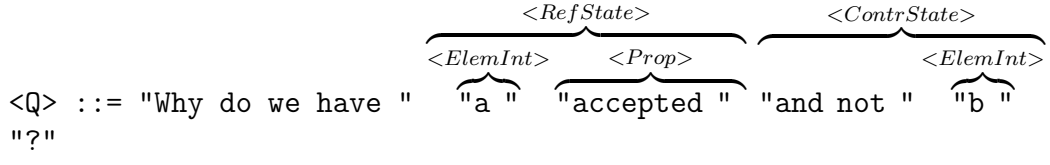
Example (Derivation of a question on the non-acceptability of one argument with context).



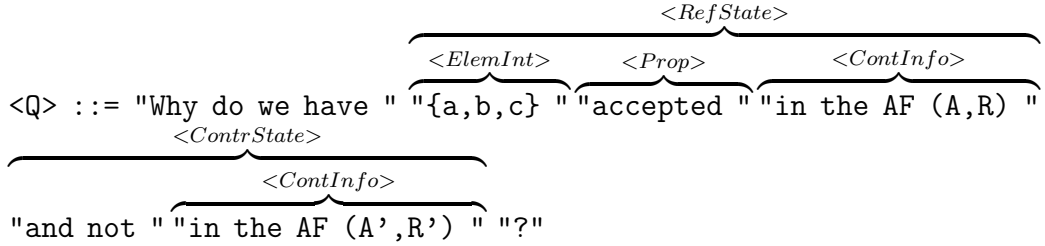
Example (Derivation of a question on several arguments being a complete extension).



Example (Derivation of a contrastive question on the acceptability of one argument compared to another).



Example (Derivation of a contrastive question on the non-acceptability of several arguments in different situations).



Using the grammar described in this section, we are now able to define some questions about the results of the argumentation process. In the next section, we turn to how to answer some questions that can be generated using our grammar.

4 Providing answers to questions about argumentation

We begin by stating some assumptions that we rely on in order to define our answers. We then continue by defining answers for questions about sets

of arguments being extensions for a certain semantics. Finally, we define answers for questions about arguments being accepted in some extension for a certain semantics.

4.1 Initial assumptions

In this section we make clear what are the assumptions we rely on for defining our explanations.

First of all, we recall the context of our work and so our first assumption (see in Sect.3.2):

> *A user asks for an explanation about the result of an abstract argumentation process* (H1)

Thus the user asks her question in reaction to a computer program, that we called *the system* presenting her the result of an argumentative process. We called *the question's context* the result that is presented to the user, the semantics under which it was computed and the Argumentation Framework that was used to compute it.[8] This behaviour is illustrated in Figure 9 in which we consider a user interested in a specific stable extension presented by the system.

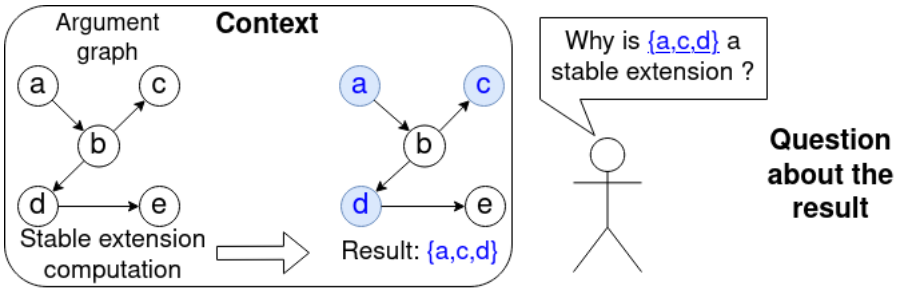


Figure 9: The first assumption: a user reacts to a given result produced by the system

Then we add some other assumptions. The first one is the following:

[8]Now, a transparent system should allow for the user to have access to this context. Still, an efficient system could focus on communicating only what is asked for, that is, in this case, the result of the argumentative process. The two are not mutually exclusive, so the system could be designed for only communicating the result it computed, but also always letting the user access the context that was part of the computation if required.

The user is able to understand Argumentation Frameworks. (H2)

Indeed, the purpose of explanations would then be to select the relevant parts of this context in order to facilitate the user's inspection process. The hypothesis we make is that, when presented with a graph interpreted as an Argumentation Framework, the user is able to understand that the nodes represent arguments and that the arcs represent conflicts between the arguments. The main reason we feel confident about this assumption is that if the user is not able to understand Argumentation Frameworks, it should not be complicated to describe how to "read" one.

The next hypothesis we make is directly related to the first one:

The user knows Abstract Argumentation semantics. (H3)

Thus, we assume that the user is already versed in Abstract Argumentation. As such, the explanations we define are not yet for ordinary users.[9]

Finally, as noted in [16], explanation is a social process, one that does not stop at the selection of an explanation. People receiving explanations expect them to obey to a certain number of rules, and evaluate their quality based on their adequecy with these rules. An example of such rules are Grice's maxims of conversation ([17]). Grice gave a set of simple rules that people tend to follow when engaging in cooperative conversation. Cooperative conversation is a discussion that happens between two or more agents that all make efforts in contributing to reaching a common goal which may be for instance exchanging information or achieving social bonding. Grice firstly gives one general principle to follow when engaging in cooperative conversation: *"Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged"*. Grice calls this the *Cooperative Principle*. Grice then gives four categories of maxims to follow in order to adhere to the Cooperative Principle that he calls *Quantity*, *Quality*, *Relation* and *Manner*. He gives the following maxims in these categories (directly cited from [17]) :

[9]This is however the objective we want to reach. Hence, we intend to drop this hypothesis in future work, so that we are able to generate explanations for any user and not only those that already know Abstract Argumentation.

1. *Quantity*: (a) Make your contribution as informative as required (for the current purpose of the exchange); (b) Do not make your contribution more informative than is required

2. *Quality*: (a) Do not say what you believe to be false; (b) Do not say that for which you lack adequate evidence

3. *Relation*: (a) Be relevant

4. *Manner*: (a) Avoid obscurity of expression; (b) Avoid ambiguity; (c) Be brief (avoid unnecessary prolixity); (d) Be orderly

Grice also gives what he calls *supermaxims*, namely "Try to make your contribution one that is true" (*Quality* category) and "Be perspicuous" (*Manner* category). And so the last assumption we make is:

> *Grice's maxims are correct and should thus be followed when engaging in cooperative conversation.* (H4)

In addition, we argue that seeking and providing explanation in a question-answer setting (such as the one we place ourselves in) certainly falls into the category of cooperative conversation. As such, we will make efforts to define explanations that adhere to these maxims, and use them as a way to evaluate our explanations.

4.2 Implicit context in questions

In this section, we deal with a kind of equivalence class of questions. By that we mean that some questions that are generated by our grammar have the same meaning and ask for the same explanation, even though their formulation is different. We illustrate the case we are interested in with the following example.

Example. Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Suppose that the system computed the result S under semantics σ using the Argumentation Framework (A, R). Consider then that the user asks the question `"Why do we have x accepted in the extension S for the` σ `semantics in the AF (A,R) ?"`. Consider now that, instead of that question, the user asks `"Why do we have x accepted`

?". We argue that, *in this particular context*, these questions are the same, in the sense that they ask for the same answer (or equivalently, for the same explanation).

Note. In the previous example, $\mathcal{A} = (A, R), S$ and σ are the context of the question. For the rest of the paper, in examples and definitions, we will always refer to the question's context using these notations.

Through this example, we raise the idea that some questions, albeit containing a different amount of explicit information, are in fact the same, *depending on the context in which they are asked.* In this situation, we interpret the less uttering question to be as informative as the more one. That is to say, we consider that both questions contain the same additional, *contextual*, information, which is only left implicit in the second question, whatever be the reason why this contextual information is left implicit.

Another example shows that such missing contextual information can also appear in contrastive questions but with a complete different impact.

Example. Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ under semantics σ. Consider then that the user asks the question "Why do we have x accepted in the extension S' for the σ' semantics in the AF (A', R') and not y ?". Consider now that, instead of that question, the user asks "Why do we have x accepted and not y ?". We argue that the contextual information in the contrast statement of each question is different.

To begin with, please note that the contextual information in the first question *may not correspond* to the question's context, hence the different notations with the primes. Through this second example, we see that the implicit information in the contrast statement of both questions in fact refers to the corresponding information of the reference statement. In the first question, we understand that the user puts the acceptance of x in contrast with the acceptance of y, *both* in the extension S' for semantics σ' in the AF (A', R'). Similarly, in the second question, we understand that the user puts the acceptance of x in contrast with the acceptance of y, *both* in the extension S for semantics σ in the AF (A, R) (this information being left implicit in the reference statement, it refers to the context of the question). As such, if an information is not stated in the contrast statement, be it element of interest, property or contextual information, we consider that this information is nonetheless present and in fact equal to the corresponding information in the reference statement.

To sum up our discussion in this question, when dealing with a question that is incomplete, in the sense that not all information is explicitly stated, we treat it as equivalent to a complete question using a method to retrieve the missing information. In the case of implicit information in the reference statement, we use the question's context to complete it. In the case of implicit information in the contrast statement, we refer to the reference statement in order to retrieve it.

4.3 Semantics Extensions

In this section, we focus on how to provide answers to a certain class of questions, namely the ones whose property in the reference statement is to be an extension of a given semantics. As an immediate consequence of this choice and of the constitutive elements we use to generate the questions, the elements of interest in the reference statement of such questions can only be sets of arguments, and the contextual information can only be argumentation frameworks. We also make an additional assumption: we do not consider contrastive questions in this section. Therefore, the scope of questions considered here is rather limited. It only consists of questions of the form `"Why do we have S [not] as an X extension ?"` or `"Why do we have S [not] as an X extension in the AF (A,R) ?"` with X ranging over conflict-free, admissible, complete and stable, (A, R) being potentially any Argumentation Framework and S being any subset of A.

With the way these questions are structured, we could be tempted to make assumptions on whether S is in fact (or not) an extension of the considered semantics prior to providing an answer. Indeed, a question like `"Why do we have S as an admissible extension ?"` tends to imply that S *is* in fact an admissible extension. We instead consider that the question merely implies that S is *perceived* as being an admissible extension by the user. We thus reject the assumption that such questions will not be asked on sets that are not extensions. In other words, we consider that the user could be wrong. As a consequence, the answers we define must provide elements supporting whether or not S is an extension of the considered semantics independently from how the user perceives it. That is to say, we should not restrain from providing elements showing that the user is wrong.

Example. Consider the Argumentation Framework depicted on Figure 1. Imagine that a user asks the question `"Why do we have { a,b,c } as a`

`complete extension ?"`. In this case, $\{a, b, c\}$ is not a complete extension, thus we would need to provide the elements that show what makes it impossible for $\{a, b, c\}$ to be a complete extension. Alternatively, imagine that a user asks the question `"Why do we have { a,d } not as a complete extension ?"`. In this case, $\{a, d\}$ is a complete extension, hence we should provide the elements that show the reasons for this set to be such an extension.

The next step is to discuss what these elements are. To define them, we adopt the modular view saying that the properties we consider (being an extension of a semantics) are a conjunction of different conditions. In other terms, to verify that a set of arguments is an extension of a given semantics, we must verify that this set respects a certain number of conditions. Alternatively, to verify that a set of arguments is not an extension of a given semantics, we must verify that this set does not respect one condition out of a certain number of them. This is undoubtedly a very basic viewpoint, but it is supported by the fact that semantics are precisely defined this way. Hence, to explain why a set of arguments is an extension of a certain semantics, we provide the relevant parts of the argumentation framework that allow to check that all the conditions defining being an extension for this semantics are satisfied. Consequently, an explanation here is made of two components:

- A part (subgraph) of the argumentation framework
- A checking procedure (that answers YES or NO)

When providing explanations, we will focus on having the checking procedures as simple and intuitive as possible. Thus, we will describe them informally in each case. In addition, if we wish to show an explanation for several conditions at once, we may show the aggregation of the reasons for every condition to hold.

The arguments of the extension which is given as input will be in blue in the explanation subgraph, and arguments or attacks that may cause a checking procedure to fail will appear in thick red.

4.3.1 Explanation for conflict-freeness

In light of the aspects discussed above, we go on by defining explanations for the considered semantics. We begin with the conflict-free semantics. Recall that a set of arguments is conflict-free if and only if there are no edges between its members. Hence, if we are to show why a set of arguments is conflict-free,

we must show a part of the graph that highlights the absence of edges within this set. We begin with an example in order to provide an intuition of how to define the explanation, and then move on to the formal definition.

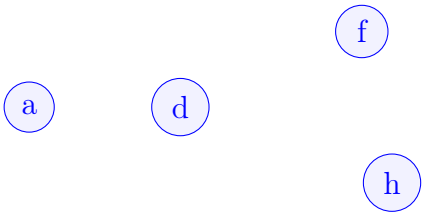


Figure 10: Explanation on why $\{a, d, h, f\}$ is conflict-free in Figure 1. All arcs between a, d, h and f are represented and since there are none, $\{a, d, h, f\}$ is conflict-free.

Figure 11: Explanation on why $\{a, d, e\}$ is not conflict-free in Figure 1. All arcs between a, d and e are represented, and we see that there is one between d and e.

Example. Consider the Argumentation Framework of Figure 1 and the questions `"Why do we have {a,d,h,f} as a conflict-free extension ?"` and `"Why do we have {a,d,e} as a conflict-free extension ?"`. Figures 10 and 11 show the answers for the first and second question respectively. Figure 10 in fact shows why $\{a, d, h, f\}$ is conflict-free while Figure 11 in fact shows why $\{a, d, e\}$ is not conflict-free. The idea is to make sure that all the arcs that are present within the given set are shown. Hence, if there is at least one, we can conclude the set is not conflict-free and if there is none, we can conclude it is.

Following the examples, we realise that we must compute a subgraph of the Argumentation Framework that shows any arc between the arguments of the set the question is about, if there are some. Preferably, this subgraph should be as small as possible to get rid of any irrelevant information.

Definition 19 (Explanation for conflict-freeness)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that:

- Production rule 1 does not yield the symbol `«ContrState»`
- Production rule 2 does not yield the symbol `«ContInfo»`

- Production rule 3 is used with the value `"S' "`
- Production rule 4 is used with the value `"as a conflict-free extension "`

The relevant subgraph to answer this question is[10]

$$\mathcal{A}[S']_V \tag{E1}$$

The checking procedure for (E1) is to verify that the set of edges in the resulting subgraph is $\varnothing$.

Note. We assume that the user could ask such a question on any possible set. Thus, the answer does not depend on S and σ (the context of the question) and instead depends on the information contained in the question. This note can be applied to all the explanations for semantics of this section.

Figure 12 illustrates the built answer giving the subgraph and the checking procedure (in this case, the used context is the context of the question, and $S = S'$).

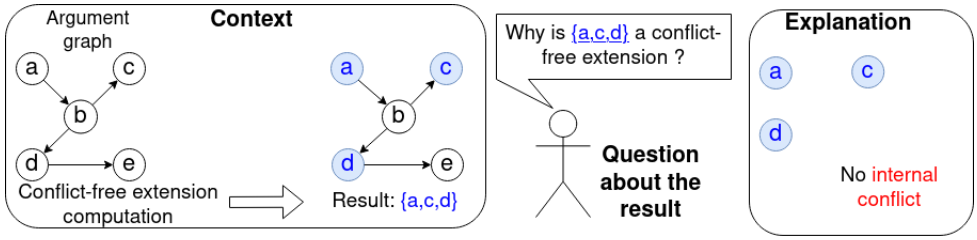


Figure 12: Explanation (E1) about the conflict-freeness

By defining the explanation as an induced subgraph, we make sure to have a reduced size as well as showing all the arcs that are concerned.

4.3.2 Explanation for admissibility

Next comes the admissible semantics. Recall that a set of arguments is admissible if and only if it is conflict-free and all its arguments are acceptable with respect to it. Since to be conflict-free is a condition to be admissible, part of the explanation for admissibility is the explanation for conflict-freeness.

[10]This is the induced graph by S'

The other part of the explanation involves the acceptability of all members of the set with respect to the set itself, which can be understood as defending the point of view represented by the set of arguments against its attackers. Therefore, if we are to show why a set of arguments only contains arguments that are acceptable with respect to it, we must show a part of the graph highlighting that all the attackers of this set are attacked in return. We begin by illustrating on some examples to give the intuition, and then formally define these explanations.

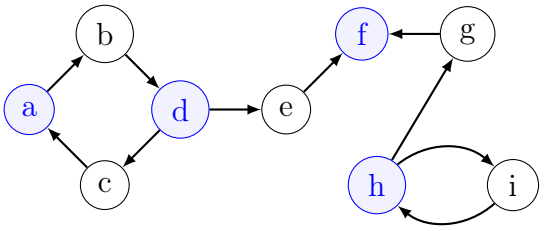


Figure 13: Explanation (E2) on why $\{a, d, h, f\}$ defends all its arguments in Figure 1. All the attackers of a, d, h and f are attacked in return.

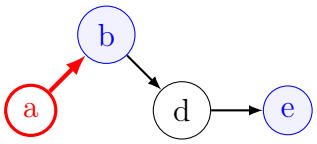


Figure 14: Explanation (E2) on why $\{b, e\}$ does not only contain arguments that are acceptable w.r.t. $\{b, e\}$ in Figure 1. b is attacked by a and a is not attacked by b or e in return.

Example. Consider the Argumentation Framework of Figure 1. Figures 13 and 14 show why $\{a, d, h, f\}$ defends all its arguments and why $\{b, e\}$ does not. In Figure 13, we can see that all the attackers of a, d, h and f are the endpoint of an arc whose origin is either a, d, h or f. In Figure 14, we see that a attacks b and neither b or e defends b against this attack.

Following the examples, we realise that we must compute a subgraph of the Argumentation Framework that includes both the set the question is about and its attackers. The only arcs that are relevant are those from the attackers to the set (to show that they are indeed attackers) and from the set to attackers (to show whether the set defends itself or not). So we must first compute an induced subgraph and then a specific partial subgraph of this induced subgraph.

Definition 20 (Explanation for defence)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$. The relevant subgraph to explain whether or not S contains only arguments that are acceptable w.r.t. S is

$$\left(\mathcal{A}[S \cup R^{-1}(S)]_V\right)[\{(a,b) \in R \mid a \in R^{-1}(S) \text{ and } b \in S, \text{ or } a \in S \text{ and } b \in R^{-1}(S)\}]_E \quad \text{(E2)}$$

The checking procedure for (E2) is to verify that every argument that does not belong to S in the resulting subgraph is the endpoint of an edge whose origin is in S.

Remark. Informally, (E2) is the subgraph of $\mathcal{A}$ in which we only keep the arguments of S, the attackers of S (so $R^{-1}(S)$) and the edges for which one extremity is an attacker of S and the other a member of S.

We can now define the explanation for admissibility. As we previously said, this explanation is in two parts: one for conflict-freeness and one for defence. Hence, the explanation for admissibility is defined as the set of these two components. These two subgraphs may be aggregated into a single one, using the union operator of Definition 10, hence giving an explanation for admissibility as a single subgraph. Such an explanation, even if more concise, may prevent a user to see at a glance the two parts which led to it. This is the reason why we choose to keep the explanation subgraphs separated in the definition of an explanation for admissibility. However, in examples later in this article, we may consider the aggregation of the subgraphs as the explanation.

Definition 21 (Explanation for admissibility)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 1 does not yield the symbol `«ContrState»`
- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 3 is used with the value `"S' "`
- Production rule 4 is used with the value `"as an admissible extension "`

The relevant set of subgraphs (with their checking procedures) to answer this question is given by

$$\text{(E1) and (E2) (applied on } S') \quad \text{(E3)}$$

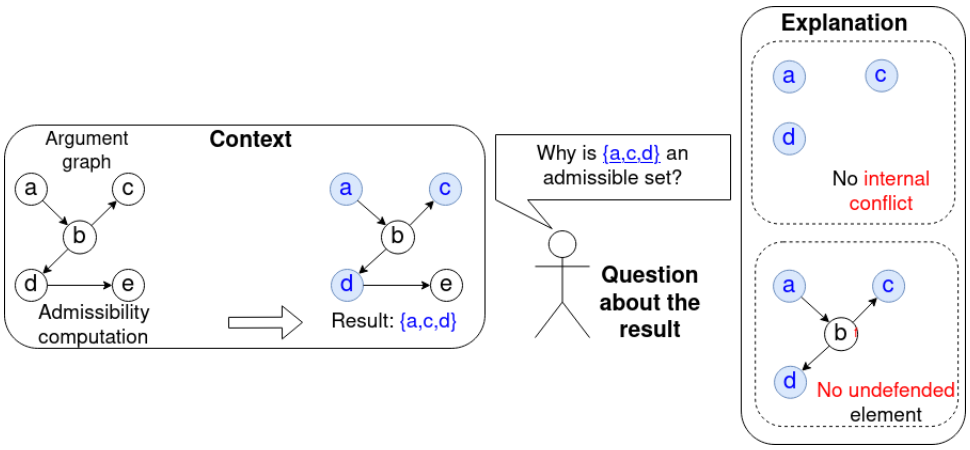


Figure 15: Explanation (E3) about admissibility

Figure 15 illustrates on a very simple example the built answer giving the subgraphs for (E1) and (E2) and the corresponding checking procedures (the used context is the context of the question, and $S = S'$.)

4.3.3 Explanation for completeness

We continue with the complete semantics. Recall that a set of arguments is complete if and only if it is admissible and all the arguments that are acceptable with respect to it are members of this set. Hence, part of the explanation for completeness is the explanation for admissibility. The other part of the explanation is about the membership of all acceptable arguments. This can be understood as adopting a point of view in which we take all the arguments we know we can defend (in a sense, we take as much as we are "forced" to). Thus, if we want to show why a set of arguments contains all the arguments it can defend, we must show a part of the graph highlighting that the arguments this set could defend are either defended (and so, part of it) or not defended (and so, not part of it). We begin by illustrating on some examples to give the intuition, and then formally define these explanations.

Example. Consider the Argumentation Framework of Figure 1. Figures 16 and 17 show why $\{a, d, h, f\}$ accepts all the arguments it defends and why $\{b, c\}$ does not. In Figure 16, we can see that all the arguments that can be reached in two steps via the attack relation from $\{a, d, h, f\}$ are in fact a, d, h and f themselves. If we consider the attackers of these arguments,

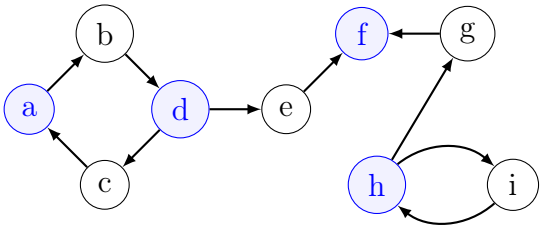


Figure 16: Explanation (E4) on why $\{a, d, h, f\}$ accepts all the arguments it defends in Figure 1. The arguments $\{a, d, h, f\}$ could defend are those that can be reached in two steps via the relation from either a, d, h or f. They are in fact a, d, h and f themselves, since they are all defended and all belong to the set.

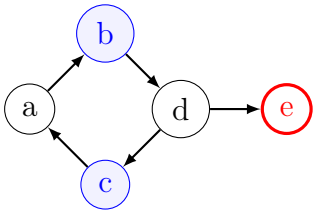


Figure 17: Explanation (E4) on why $\{b, c\}$ does not accept all the arguments it defends in Figure 1. e is defended by b but does not belong to the set.

we observe that they are all attacked by a, d, h or f. Since these arguments are all included in the set, we can conclude that $\{a, d, h, f\}$ accepts all the arguments it defends. In Figure 17, we see that e is defended by b, but does not belong to the set $\{b, c\}$. Hence, $\{b, c\}$ does not accept all the arguments it defends.

Following the examples, we realise that we must compute a subgraph of the Argumentation Framework induced by the set of arguments the question is about and the arguments this set could defend and the attackers of these arguments. Moreover, the only arcs that are relevant are those from the attackers to the arguments the set could defend (to show that they are indeed attackers) and from the set to these attackers (to show whether the set indeed defends these arguments or not). So the corresponding answer will be a specific partial subgraph of the induced subgraph.

Definition 22 (Explanation for reinstatement)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$. The relevant subgraph to explain whether or not S contains all arguments that are acceptable w.r.t. S is

$$\big(\mathcal{A}[S \cup R^2(S) \cup R^{-1}(R^2(S))]_V\big) \\ [\{(a,b) \in R \mid a \in R^{-1}(R^2(S)) \text{ and } b \in R^2(S)\} \\ \cup\{(a,b) \in R \mid a \in S \text{ and } b \in R^{-1}(R^2(S))\}]_E \tag{E4}$$

The checking procedure for (E4) is to verify that for every argument in $R^2(S)$ in the resulting subgraph, if it is in S then all its attackers are the endpoint of an edge whose origin is in S, otherwise (for the arguments that are not in S) at least one of its attackers is not the endpoint of an edge whose origin is in S.

Remark. Informally, (E4) is the subgraph of $\mathcal{A}$ in which we only keep the arguments of S, the arguments that S could defend (so $R^2(S)$) and the attackers of these arguments (so $R^{-1}(R^2(S))$). The edges of this subgraph are restricted to those from the attackers to the arguments S could defend and from S to the attackers.

Much like with the explanation for admissibility, we now have what we need to define the explanation for completeness. Once again, this explanation is the set of its different components.

Definition 23 (Explanation for completeness)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

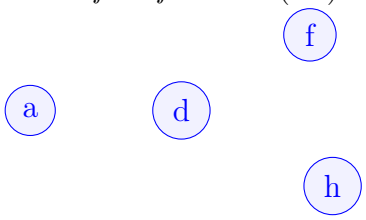
- Production rule 1 does not yield the symbol `«ContrState»`
- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 3 is used with the value `"S' "`
- Production rule 4 is used with the value `"as a complete extension "`

The relevant set of subgraphs (with their checking procedures) to answer this question is given by

$$\text{(E1) and (E2) and (E4)} \quad \text{(applied on } S') \tag{E5}$$

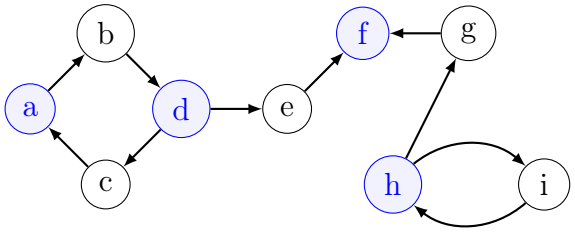
Example. Consider the Argumentation Framework of Figure 1 and the question `"Why do we have {a,d,h,f} as a complete extension ?"`. Figure 18 shows the answer for this question.

Conflict-freeness (E1)*:*



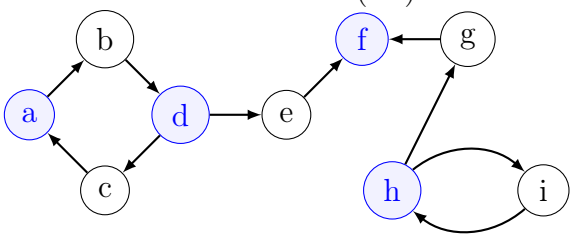
Result of the checking procedure: no internal conflict in $\{a, d, f, h\}$

Defence (E2)*:*



Result of the checking procedure: no undefended argument in $\{a, d, f, h\}$

Reinstatement (E4)*:*



Result of the checking procedure: any argument defended by $\{a, d, f, h\}$ belongs to $\{a, d, f, h\}$

Figure 18: Explanation (E5) on why $\{a, d, h, f\}$ is complete in Figure 1. It is the sequence of Figures 10, 13 and 16. Note that, in this example, the subgraph for the defense and the subgraph for the reinstatement are identical but not the checking procedures.

4.3.4 Explanation for stability

The last semantics we turn to is the stable semantics. Recall that a set of arguments is stable if and only if it is conflict-free and attacks all the arguments that do not belong to it. Therefore, part of the explanation for stability is the explanation for conflict-freeness. The other part involves the attack from S to its complement. This can be understood as adopting a dominant point of view in which we want to contradict everything that might be said against it. Consequently, if we want to show why a set of arguments is stable, we must show a part of the graph highlighting that this set of arguments attacks all the other arguments. We begin by illustrating on some examples the intuition, and then formally define these explanations.

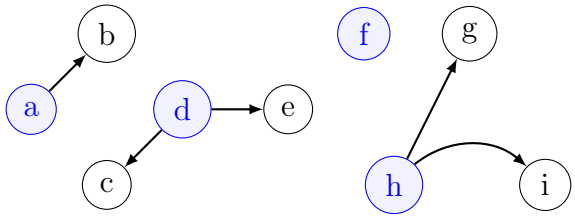


Figure 19: Explanation (E6) on why $\{a, d, h, f\}$ attacks all the other arguments in Figure 1. For all other arguments in the Argumentation Framework, there is an arc from a, d, h or f to that argument.

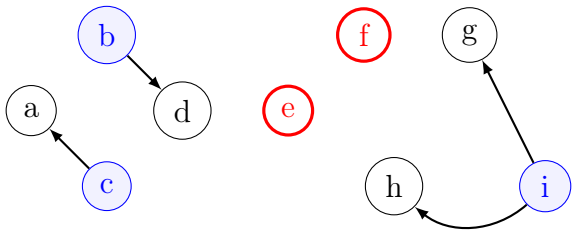


Figure 20: Explanation (E6) on why $\{b, c, i\}$ does not attack all the other arguments in Figure 1. e and f are not attacked by b, c or i.

Example. Consider the Argumentation Framework of Figure 1. Figure 19 shows why $\{a, d, h, f\}$ attacks all the other arguments in the Argumentation Framework and Figure 20 shows why $\{b, c, i\}$ does not. In Figure 19, we

can see that there is an arc from a, d, h or f[11] to every other argument in the Argumentation Framework. In Figure 20 however, we see that there are some arguments that are attacked by neither b, nor c, nor i (namely, e and f).

Following the examples, we realise that we must compute a subgraph of the Argumentation Framework that includes all the arguments. However, the only arcs that are relevant are those from the set the question is about to any argument that is not in that set. So we need to compute a specific partial subgraph.

Definition 24 (Explanation for complement attack)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$. The relevant subgraph to explain whether or not S attacks its complement is

$$\mathcal{A}[\{(a, b) \in R \mid a \in S \text{ and } b \notin S\}]_E \tag{E6}$$

The checking procedure for (E6) is to verify that all arguments not belonging to S are attacked by an argument of S.

Remark. Informally, (E6) is the partial subgraph of $\mathcal{A}$ in which we only keep the edges from S to arguments that are not in S.

In a similar fashion than with the previous semantics, we can now use the explanation for complement attack and the explanation for conflict-freeness, to define the explanation for stability. Again, it is defined as the set of its components.

Definition 25 (Explanation for stability)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 1 does not yield the symbol `«ContrState»`
- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 3 is used with the value `"S' "`

[11] In this example, f does not attack any argument, so its presence in the explanation is useless. Nevertheless, in a first step, it seems important to keep in each explanation all the elements of the set interesting the user, in order to give her the most complete view about the properties of this set. In future works, when we will study the notion of minimal explanations, this constraint will be probably relaxed.

- Production rule 4 is used with the value `"as a stable extension "`

The relevant set of subgraphs (with their checking procedures) to answer this question is given by

(E1) and (E6) (applied on S') (E7)

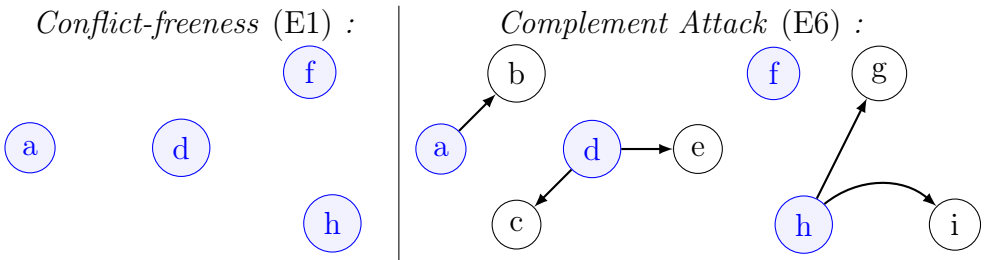


Figure 21: Explanation (E7) on why $\{a, d, h, f\}$ is stable in Figure 1.

Example. Consider the Argumentation Framework of Figure 1 and the question `"Why do we have {a,d,h,f} as a stable extension?"`. Figure 21 shows the answer for this question.

4.4 Acceptance

In this section, we turn to providing answers to another class of questions, namely the ones whose property in the reference statement is to be accepted in an extension of a given semantics. By "to be accepted in an extension", we mean "to be part of an extension". As such, this property can be understood as the standard relation of set-inclusion regarding a set respecting a certain number of conditions.

In addition, since we focus on questions in which the reference statement's property is to be accepted, we know that elements of interest can be either one argument or a set of arguments, and all possible contextual information can be explicitly mentioned. We also consider contrastive questions on acceptance, reducing a bit their scope by considering the additional assumption that:

If a contrast statement is present in the question, the contrast is only made on elements of interest, and not on the property or contextual information. (H5)

That is to say, we suppose that the property and contextual information in the contrast statement are the same as in the reference statement. This assumption is specific to this section and does not apply to the entire document. It only has the effect of focusing on a certain number of contrastive questions and not all that are possible in this specific context (that is, with the property in the reference statement being to be accepted in an extension of a given semantics).

Example. `"Why do we have a accepted ?"` and `"Why do we have {b,c} accepted and not d ?"` are examples of questions we are interested in, in this section. `"Why do we have a accepted in the extension {a,b,c} and not in the extension {d,e,f} ?"`, `"Why do we have {b,c} accepted and not as a stable extension ?"` and `"Why do we have a accepted in the AF (A,R) and not in the (A',R') ?"` are examples of questions that we do not consider in this setting.

In order to provide answers to this kind of questions, we rely on the structure of the question and its context. It is common to consider that a question of the form *"Why P ?"* is in fact a question of the form *"Why P and not Q ?"*, with Q left implicit. P and Q are often referred to as the *fact* and the *foil* respectively (see [16]). Thus, the key to this kind of question is to be able to identify the implicit foil.

Now, in our case, the minimal question is `"Why do we have e p ?"` with e an element of interest and p a property on e. The natural foil to this kind of question would then be that e does not enjoy property p. In other words, the question `"Why do we have e p ?"` is in fact the question `"Why do we have e p and not` $\overline{\texttt{p}}$ `?"` with $\overline{p}$ representing the absence of property p. Please note that here, the fact and foil are not exactly "p" and "$\overline{p}$" but rather "e being p" and "e being $\overline{p}$" (i.e. "e not being p"). So, the approach we use to answer questions relies on the following hypothesis:

To explain "e being p" is to show that "e not being p" is not possible. (H6)

Put it differently, to explain "e being p" is to show its *necessity*. This is already the approach we used in Section 4.3 on questions related to why a given set of arguments is (or not) an extension of some semantics. Independently from what the user perceives to be true, it holds that either "e being p" or "e being $\overline{p}$" is true, but not both. The choice we made was to show the element supporting the truth, and thus that its contrary is not possible. In the case of questions related to why an argument or set of arguments is (resp. is not) part of an extension of some semantics, we adopt the point of view of showing that the argument or set of arguments cannot not be part (resp. be part) of that extension. That is, we directly compute the foil suggested by the question and show it to the user so that she may come herself to the conclusion that the argument or set of arguments must (resp. must not) be part of that extension.

Example. Suppose we are in a situation in which the system works with the Argumentation Framework of Figure 1 and is required to provide an admissible extension. It presents the set $\{a, d, h\}$ to the user, which is indeed an admissible extension. Suppose then that the user asks `"Why do we have a accepted ?"`. In this case, the fact is "a being accepted" and the foil is "a not being accepted". To answer the user's question, we thus show that "a not being accepted" cannot hold. That is to say, we show the explanation for admissibility on $\{d, h\}$. As $\{d, h\}$ is, in fact, not an admissible extension of Figure 1, the answer will show why $\{d, h\}$ cannot be an admissible extension, and thus why a must be accepted (in this case, to defend d).

Example. Suppose we are in the same context as in the previous example. Suppose then that the user asks `"Why do we have i not accepted ?"`. To answer this question, we show the explanation for admissibility on $\{a, d, h, i\}$. This will show the user what is problematic with trying to accept i in addition of the arguments that were originally accepted. In this case, the user will see that accepting i gives rise to an internal conflict between h and i.

Remark. With this method, it is entirely possible to ask a question that suggests a valid extension that is different from the one originally presented by the system. For instance, consider the same context as in the previous example and suppose the user asks the question `"Why do we have h accepted ?"`. The system would then show the explanation for admissibility on $\{a, d\}$, which happens to be, in fact, an admissible extension of Figure 1. In this situation, our objective to show that it must be the case that h is accepted is a failure. We will elaborate more on this in following examples and in

Section 5.

In the following, we will make a distinction between questions of the form `Why do we have X accepted?` (with `X` being either an argument `x` or a set of arguments `S`) and questions of the form `Why do we have X not accepted?`. The difference lies on using a negation on the property in the case of the latter questions. Hence, we will call the former *positive questions* and the latter *negative questions*.

4.4.1 Non-contrastive questions

In this section, we precisely define the answers to non-contrastive questions on acceptance. Taking into consideration the variation on elements of interest in the questions (an argument or a set or arguments), there are 2 possible positive non-contrastive questions and 2 possible negative non-contrastive questions.

In the case of a question on the acceptance of a set of argument, we consider that the set represents an aggregation of arguments. That is to say, the question is understood as a concatenation of questions on the acceptance of one argument, one for each argument of the set. As such, we will group both cases together by considering the case of a single argument equivalent to the case of a singleton set containing this argument.

The questions are answered using the principles illustrated above. We begin with the positive questions.

Definition 26 (Explanation for positive non-contrastive question on acceptance of a set of arguments)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 1 does not yield the symbol `«ContrState»`
- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 3 is used with the value `"S' "`
- Production rule 4 is used with the value `"accepted "`

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $S \setminus S'$.

Notice that in the subgraph which will be presented as an explanation, only the arguments of $S \setminus S'$ will be in blue (the explanation amounting to show why this set is or is not an extension under σ).

Example. Consider the Argumentation Framework of Figure 1 and the admissible extension $\{a, d, h\}$. Suppose the user asks the question `"Why do we have a accepted ?"`. Figure 22 shows the corresponding answer: $\{d, h\}$ without a is not admissible (it is conflict-free but does not respect the defence property) and thus that a is necessary in the extension presented by the system.[12]

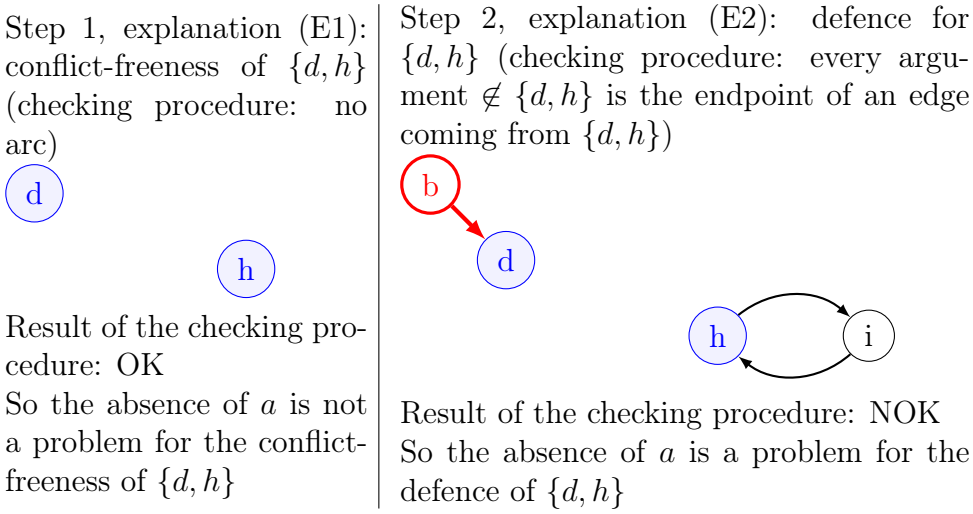


Figure 22: Explanation on why a is accepted in the admissible extension $\{a, d, h\}$: explanation (E3) on why $\{d, h\}$ is not admissible

We now turn to negative questions. The methodology is very similar to the positive questions, the difference being that instead of removing arguments from the extension, we add them.

Definition 27 (Explanation for negative non-contrastive question on acceptance of a set of arguments)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

[12]Note that, for some element the user is interested in, it could happen that the removal of this element does not produce any effect. This point will be discussed in Section 5.

- Production rule 1 does not yield the symbol «ContrState»
- Production rule 2 does not yield the symbol «ContInfo»
- Production rule 3 is used with the value "S' "
- Production rule 4 is used with the value "not accepted "

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $S \cup S'$.

Notice that in the subgraph which will be presented as an explanation, the arguments of $S \cup S'$ will be in blue (the explanation amounting to show why $S \cup S'$ is or is not an extension under σ).

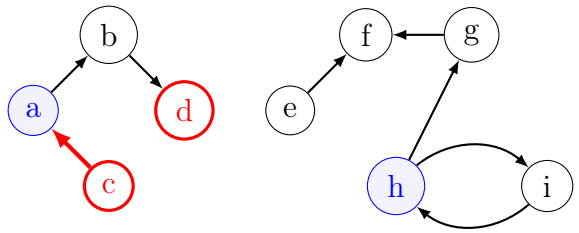


Figure 23: Explanation on why a is not accepted in the complete extension $\{h\}$: explanation (E5) on why $\{a, h\}$ is not complete. Considering $\{a, h\}$, the conflict-freeness is satisfied (no arc between a and h), but the defence property is not (a cannot be defended by h or by itself), nor is the reinstatement property (a defends d, which is not in the considered set). So the checking procedures corresponding to the step of defence and to the step of reinstatement fail.

Example. Consider the Argumentation Framework of Figure 1 and the complete extension $\{h\}$. Suppose the user asks the question "Why do we have a not accepted ?". Figure 23 shows the answer for this question.

4.4.2 Contrastive questions

In this section we turn to precisely define the answers to contrastive questions on acceptance. We begin by extending the distinction made between questions by the presence/absence of a negation on the property to contrastive questions, using the property of the contrastive statement as well

as the property of the reference statement. This yields 4 types of questions: positive-positive, positive-negative, negative-positive and negative-negative. Taking into consideration the variation on elements of interest in the questions, there are 2 possible questions for each type. Please note that since we use `"and not"` as the connection between the reference statement and the contrast statement, we must reverse the type of the property expressed in the contrast statement.

Example. The question `"Why do we have a not accepted and not b accepted ?"` is a negative-negative question, although the contrast statement's property is expressed as positive. Recalling the way we deal with implicit information, the question `"Why do we have a accepted and not b ?"` is then a positive-negative question.

To provide answers to these questions, we treat the contrast statement the same way as the reference statement. Note that a contrastive question is *not* a sequence of two questions; there is a specificity given by the contrast that influences the building of the answer. So the resulting graph can be obtained using a combination of the previous definitions given for the reference statement and for the contrast statement. We will only cover the case in which the element of interest is a set of arguments. The case in which it is one argument is dealt with in the same way as in the previous section.

Definition 28 (Explanation for positive-positive contrastive question on acceptance of a set of argument)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 6 is used and only yields the symbol `«ElemInt»` and `«Prop»`
- Production rule 3 is used with the value `"S' "` in the reference statement and `"S'' "` in the contrast statement
- Production rule 4 is used with the value `"accepted "` in the reference statement and `"not accepted "` in the contrast statement

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $(S \setminus S') \setminus S''$.[13]

In the subgraph which will be presented as an explanation, only the arguments of $(S \setminus S') \setminus S''$ will be in blue (the explanation amounting to show why such a set is or is not an extension under σ).

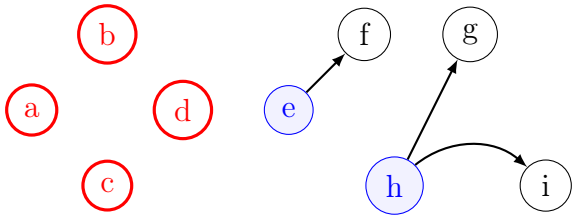


Figure 24: Explanation on why $\{b\}$ and $\{c\}$ are accepted in the stable extension $\{b, c, e, h\}$: explanation (E7) on why $\{e, h\}$ is not stable. Without b and c, e and h alone fail to attack every other argument in the Argumentation Framework. Hence, b and c are necessary in this stable extension. So it is the checking procedure corresponding to the step of complement attack that fails.

Example. Consider the Argumentation Framework of Figure 1 and the stable extension $\{b, c, e, h\}$. Suppose the user asks the question `"Why do we have {b} accepted and not {c} not accepted ?"`. Figure 24 shows the answer for this question.

Definition 29 (Explanation for positive-negative contrastive question on acceptance of a set of argument)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 6 is used and only yields the symbol `«ElemInt»`
- Production rule 3 is used with the value `"S' "` in the reference statement and `"S'' "` in the contrast statement

[13] So it is the combination of Def. 26 over the reference statement and Def. 26 over the contrast statement.

- Production rule 4 is used with the value `"accepted "`

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $(S \setminus S') \cup S''$.[14]

In the subgraph which will be presented as an explanation, the arguments of $(S \setminus S') \cup S''$ will be in blue (the explanation amounting to show why such a set is or is not an extension under σ).

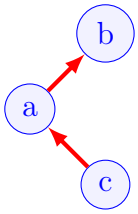


Figure 25: Explanation of why $\{d, h\}$ is accepted and not $\{b, c\}$ in the conflict-free extension $\{a, d, h\}$: explanation (E1) on why $\{a, b, c\}$ is not conflict-free. We see that removing d and h while adding b and c gives rise to internal conflicts in the extension. So it is the checking procedure corresponding to the step of conflict-freeness that fails.

Example. Consider the Argumentation Framework of Figure 1 and the conflict-free extension $\{a, d, h\}$. Suppose the user asks the question `"Why do we have {d,h} accepted and not {b,c} ?"`. Figure 25 shows the answer for this question.

Definition 30 (Explanation for negative-positive contrastive question on acceptance of a set of argument)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ. Consider a question generated using our grammar such that :

- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 6 is used and only yields the symbol `«ElemInt»`
- Production rule 3 is used with the value `"S' "` in the reference statement and `"S'' "` in the contrast statement

[14] So it is the combination of Def. 26 over the reference statement and Def. 27 over the contrast statement.

- Production rule 4 is used with the value `"not accepted "`

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $(S \cup S') \setminus S''$.[15]

In the subgraph which will be presented as an explanation, the arguments of $(S \cup S') \setminus S''$ will be in blue (the explanation amounting to show why such a set is or is not an extension under σ).

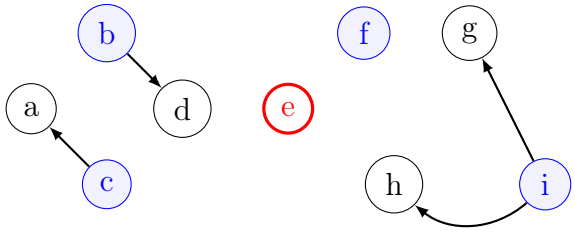


Figure 26: Explanation on why $\{b, c\}$ is not accepted and not $\{a, d\}$ in the stable extension $\{a, d, i, f\}$: explanation (E7) on why $\{i, f, b, c\}$ is not stable. If we add b and c to the extension while removing a and d we see that the extension fails to meet the conditions for being stable (e is not attacked by any argument of the extension). So it is the checking procedure corresponding to the step of complement attack that fails.

Example. Consider the Argumentation Framework of Figure 1 and the stable extension $\{a, d, i, f\}$. Suppose the user asks the question `"Why do we have` $\{b, c\}$ `not accepted and not` $\{a, d\}$`?"`.[16] Figure 26 shows the answer for this question.

Definition 31 (Explanation for negative-negative contrastive question on acceptance of a set of argument)**.** Let $\mathcal{A} = (A, R)$ be an Argumentation Framework and $S \subseteq A$ be an extension of $\mathcal{A}$ for semantics σ.

- Production rule 2 does not yield the symbol `«ContInfo»`
- Production rule 6 is used and only yields the symbol `«ElemInt»` and `«Prop»`

[15] So it is the combination of Def. 27 over the reference statement and Def. 26 over the contrast statement.

[16] Following our interpretation of implicit contexts (see Sect. 4.2), this question is in fact: `"Why do we have` $\{b, c\}$ `not accepted and not` $\{a, d\}$ `not accepted?"`

- Production rule 3 is used with the value `"S' "` in the reference statement and `"S'' "` in the contrast statement
- Production rule 4 is used with the value `"not accepted "` in the reference statement and `"accepted "` in the contrast statement

The relevant subgraphs and checking procedures to answer this question are given by the explanation for σ on $(S \cup S') \cup S''$.[17]

In the subgraph which will be presented as an explanation, the arguments of $(S \cup S') \cup S''$ will be in blue (the explanation amounting to show why such a set is or is not an extension under σ).

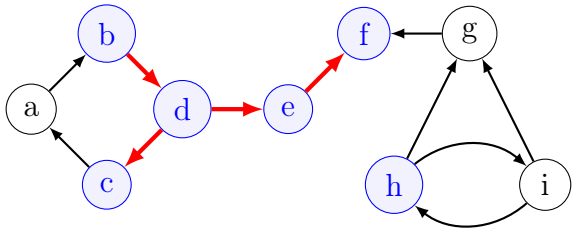


Figure 27: Explanation on why $\{h, f\}$ and $\{d\}$ are not accepted in the complete extension $\{b, c, e\}$: explanation (E5) on why $\{b, c, e, h, f, d\}$ is not complete. If h, f and d are added to the extension, internal conflicts appear. So it is the checking procedure corresponding to the step of conflict-freeness that fails.

Example. Consider the Argumentation Framework of Figure 1 and the complete extension $\{b, c, e\}$. Suppose that the user asks the question `"Why do we have {h,f} not accepted and not {d} accepted ?"`. Figure 27 shows the answer for this question.

4.5 Synthesis about answers

Before we discuss the quality of our explanations, we give in this section a synthesis about our approach in the argumentation context.

[17] So it is the combination of Def. 27 over the reference statement and Def. 27 over the contrast statement.

First of all, our assumptions (the two last ones concerning only a category of questions, those about the acceptance):[18]

(H1): *A user asks for an explanation after she has been presented the result of a Formal Argumentation process (typically the selection of arguments via a semantics) by a program that we will refer to as* the system.

(H2): *The user is able to understand Argumentation Frameworks.*

(H3): *The user knows Abstract Argumentation semantics.*

(H4): *Grice's maxims are correct and should thus be followed when engaging in cooperative conversation.*

(H5): *If a contrast statement is present in the question, the contrast is only made on elements of interest, and not on the property or contextual information.*

(H6): *To explain "e being p" is to show that "e not being p" is not possible with e being an element of interest and p being a property for e.*

Then, let $\mathcal{A} = (A, R)$ be the Argumentation Framework that is the main component of the system, the questions we are interested in:

Questions about semantics extensions: let S be the set of arguments produced by the system for a given semantics σ (σ can be either the conflict-freeness, or the admissibility, or the completeness, or the stability), and let consider that the user asks about `S'` that denotes a set of arguments,

Q1 `"Why do we have S' [not] as an extension for semantics` σ`?"` (`S'` can be or not equals to S)

Note that Question **Q1** will be instantiated wrt σ. Moreover this question uses some subquestions related to the principles behind the semantics (so the defence, the reinstatement and the complement attack).

Questions about acceptance: let S be the set of arguments produced by the system for a given semantics σ and let consider that the user asks about `S', S"` that denote sets of arguments (eventually singletons),

[18]Note that the assumption (H6) could be also used in the questions about of semantics extensions. Nevertheless, due to the way our answers are built, it is useless.

Q2 (question +): `Why do we have S' accepted?`

Q3 (question −): `Why do we have S' not accepted?`

Q4 (question ++): `Why do we have S' accepted and not S" not accepted?`

Q5 (question +−): `Why do we have S' accepted and not S" accepted?`

Q6 (question −+): `Why do we have S' not accepted and not S" not accepted?`

Q7 (question −−): `Why do we have S' not accepted and not S" accepted?`

The answers corresponding to Question **Q1** are given in Tables 2 and 3.

The answers corresponding to questions **Q2** to **Q7** are given in Table 4.

It is worth noting that some contrastive questions are equivalent to non contrastive ones:

- given that $(S \setminus S') \setminus S'' = S \setminus (S' \cup S'')$, **Q4**: `Why do we have S' accepted and not S" not accepted?` is equivalent to **Q2**: `Why do we have S' ∪ S" accepted?`. So a positive-positive question can be expressed as a positive question.
- given that $(S \cup S') \cup S" = S \cup (S' \cup S'')$, **Q7**: `Why do we have S' not accepted and not S" accepted?` is equivalent to **Q3**: `Why do we have (S' ∪ S") not accepted?`. So a negative-negative question can be expressed as a negative question.

This can lead to a little simplification of our approach, considering only 2 kinds of contrastive questions, the positive-negative and the negative-positive questions (the other ones being transformed into non contrastive questions).

5 Quality of Explanations

In this section, we wish to provide several insights on the quality of the explanations we propose in this paper. To begin with, we clearly stated that we wished to adhere to Grice's maxims as much as possible. Thus, we use these maxims to evaluate our explanations.

Informal question about defence (Explanation (E2)): $\mathbf{Q_{def}}$ "Does S' contain only arguments that are acceptable w.r.t. S'?"	
Subgraph:	$\big(\mathcal{A}[S \cup R^{-1}(S')]_V\big)$ $[\{(a,b) \in R \mid a \in R^{-1}(S') \text{ and } b \in S',$ or $a \in S' \text{ and } b \in R^{-1}(S')\}]_E$
Checking Proc.:	every argument that does not belong to S' in the resulting subgraph is the endpoint of an edge whose origin is in S'
Informal question about reinstatement (Explanation (E4)): $\mathbf{Q_{reins}}$ "Does S' contain all arguments that are acceptable w.r.t. S'?"	
Subgraph:	$\big(\mathcal{A}[S \cup R^2(S') \cup R^{-1}(R^2(S'))]_V\big)$ $[\{(a,b) \in R \mid a \in R^{-1}(R^2(S')) \text{ and } b \in R^2(S')\}$ $\cup \{(a,b) \in R \mid a \in S' \text{ and } b \in R^{-1}(R^2(S'))\}]_E$
Checking Proc.:	for every argument in $R^2(S')$', if it is in S' then all its attackers are the endpoint of an edge whose origin is in S', otherwise (for the arguments that are not in S') at least one of its attackers is not the endpoint of an edge whose origin is in S'
Informal question about complement attack (Explanation (E6)): $\mathbf{Q_{compAtt}}$ "Does S' attack its complement?"	
Subgraph:	$\mathcal{A}[\{(a,b) \in R \mid a \in S' \text{ and } b \notin S'\}]_E$
Checking Proc.:	all arguments not belonging to S' are attacked by an argument of S'

Table 2: The answers given for some informal questions about the underlying principles used in semantics (the subgraph is the one that is built to answer the question and the checking procedure is always applied on this subgraph)

Q1 for conflict-freeness (Explanation (E1)): `"Why do we have S' [not] as a conflict-free extension?"`	
Subgraph:	$\mathcal{A}[S']_V$
Checking Proc.:	The set of edges is $\varnothing$
Q1 for admissibility (Explanation (E3)): `"Why do we have S' [not] as an admissible extension?"`	
Subgraph:	The ones for **Q1** about conflict-freeness (E1)
Checking Proc.:	
Subgraph:	The ones for the question about defence (E2)
Checking Proc.:	
Q1 for completeness (Explanation (E5)): `"Why do we have S' [not] as a complete extension?"`	
Subgraph:	The ones for **Q1** about conflict-freeness (E1)
Checking Proc.:	
Subgraph:	The ones for the question about defence (E2)
Checking Proc.:	
Subgraph:	The ones for the question about reinstatement (E4)
Checking Proc.:	
Q1 for stability (Explanation (E7)): `"Why do we have S' [not] as a stable extension?"`	
Subgraph:	The ones for **Q1** about conflict-freeness (E1)
Checking Proc.:	
Subgraph:	The ones for the question about complement attack (E6)
Checking Proc.:	

Table 3: The answers given for Question **Q1**, for each semantics. For some semantics, several subgraphs (and checking procedures) exist in the explanation, one for each underlying principle used in this semantics (S' is an element given by the user in her question)

Q2: `Why do we have S' accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $S \setminus S'$ `as an extension for` σ`?"`
Q3: `Why do we have S' not accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $S \cup S'$ `as an extension for` σ`?"`
Q4: `Why do we have S' accepted and not S" not accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $(S \setminus S') \setminus S''$ `as an extension for` σ`?"`
Q5: `Why do we have S' accepted and not S" accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $(S \setminus S') \cup S''$ `as an extension for` σ`?"`
Q6: `Why do we have S' not accepted and not S" not accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $(S \cup S') \setminus S''$ `as an extension for` σ`?"`
Q7: `Why do we have S' not accepted and not S" accepted?`	
Subgraph:	The ones for **Q1** applied to σ:
Check. Proc.:	`"Why do we have` $(S \cup S') \cup S''$ `as an extension for` σ`?"`

Table 4: The answers given for questions **Q2** to **Q7** wrt a given semantics σ (S is the result presented by the system to the user before she asks for an explanation and S', S'' are the elements given by the user in her question)

First, consider the maxims of **Quantity** that require to say what is necessary, but also to not say what is not necessary. We claim that our explanations contain both the necessary information and unnecessary information. The former results from our choice to answer to questions of the (minimal) form `"Why do we have e p ?"`, with e an element of interest and p a property on e, by showing that "e not being p" must not be the case, the latter results from our choice to look for all reasons that could allow to affirm it. To illustrate for the case of explanations on why a set of arguments is an extension under some semantics or not, we use Figures 10 and 11 on conflict-freeness. It is easy to see that all the necessary information is present in these explanations. On both cases, if one node (or arc) would have been discarded, we could have missed an information leading to a change of conclusion. On the other hand, one can see that in general, the explanations display more information than is necessary. In Figure 11, the arc from d to e is sufficient to conclude that the set is not conflict-free. Thus, the node a could be removed without changing the status of the conclusion. Similar observations can be made on the other explanations on why a set of arguments is an extension under some semantics or not. Concerning explanations on the acceptance of arguments in an extension, we already argued in the beginning of Section 4.4 that our explanations contain the necessary information. However, one can see, for instance on Figure 22, that these explanations may also contain unnecessary information. Indeed, in this example, the main reason for $\{d, h\}$ to not be admissible after the removal of a is that d is no longer defended by a. Hence, one could consider the information shown about h defending itself against i as superfluous.

The other categories of maxims are more straightforward. On the maxims of **Quality**, it is obvious that, since our explanations are computed using information which is available to both the system and the user, we have all the evidence needed to support them. Moreover, we believe that computing explanations using induced and partial subgraphs may never lead to *false* explanations, unless the original AF is twisted and modified prior to the explanations' computation, which is never done here. Concerning the maxim of **Relation**, one might consider the unnecessary information that is present in our explanations to be not relevant. On the other hand, we also make sure that all the relevant information is contained in our explanations. Finally, we believe that the category of **Manner** has more to do with a translation from our graphical explanations to a dialogue in natural language, which we are not interested in yet.

One of the strengths of our explanations is that they are built in a **modular** way, and thus rely on the specific features of the semantics at hand. We motivate our choice by showing why, in our opinion, explanations cannot rely only on a feature that is common amongst all the semantics such as defense. Indeed, *explanations* should make more understandable how results are obtained, how they are *computed*. Thus, one of the problem with *explaining* all semantics in the same way (that is, for instance, only showing defense in every case) might infer the incorrect bias that all semantics are in fact the same, since the explanations show that they are all *computed* in the same way. Moreover, it is worth noting that not all semantics are defined *based* on this notion of defence. For instance, stable extensions are stable because they attack all the arguments that are not part of it. Thus, when asking why a set of arguments is stable, or why some arguments are part of a stable extension, or why an argument is credulously/skeptically accepted under stable semantics, it seems misplaced, or even confusing, to invoke the fact that stable extensions defend all their arguments. The same goes for preferred or grounded semantics, since they add a constraint of maximality/minimality to the selection of arguments. One could even argue that it would not be a relevant answer for admissible extensions, because these extensions might defend arguments that are not part of them. Thus, justifying the presence of an argument in an admissible extension merely by the fact that it is defended by the other arguments might lead to the legitimate reply "Then why is this other argument not in the extension although it is defended as well?". As such, only invoking defence for justifying the selection of an argument only seems relevant in the case of complete semantics, where indeed, defence equates to acceptance.

It is worth noting that sometimes our explanations consist of the entire original AF (see for instance Figure 27). The scope of our explanations is fairly restrained in general (in the worst case, we keep arguments that are in a "distance" of 3 from the arguments[19] of the set for which we compute an explanation in the case of completeness). However, even with such a limited depth of search, if the set from which we begin the search contains arguments that are sparse and span all across the AF, one can easily see that the search will tend to cover the entire AF. Thus, in these situations, the explanations will indeed tend to be the entire AF. However, we have also seen through the examples of this paper that when considering less large sets, or sets that do

[19]So the elements that belong to $R^{-1}(R^2(S))$ for a given set S of arguments.

not span across all the AF, the explanations tend to be more local and to be restrained to precise areas of the AF.

Our explanations concern all the classical **semantics** defined by Dung, except the grounded and the preferred ones. This is because our definitions of explanations are graphical and rely on the *modular* aspect of semantics. In particular, the grounded and preferred semantics include an aspect of *minimality* and *maximality* respectively. Minimality (resp. maximality) may be shown by answering the question on why the set reduced (resp. augmented) of a non empty subset of arguments is not an extension under the considered semantics, and this, for any such subset. These subsets may be numerous, and the explanation that would hence be given may not be that intelligible. Such a solution is not that satisfactory in this sense. Another solution would consist to let the user ask why a given set she thought would be a minimal (resp. maximal) extension, is not. This would not be a direct explanation of why a set is minimal or maximal, but it may be helpful to a user who would have had in mind a different set. However, finding a direct graphical explanation of minimality and maximality (and then, of the grounded and preferred semantics) is a challenge for future work.

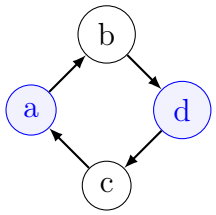


Figure 28: Explanation (E3) on why h is accepted in the admissible extension $\{a, d, h\}$. The removal of h does not lead to an internal conflict or a defenceless argument. Thus, h is not necessary in order to obtain the admissibility.

To finish with, we wish to focus on situations like the one presented on Figure 28. In this case, we supposed that the system delivered $\{a, d, h\}$ as an admissible extension and that the user asked the question `"Why do we have h accepted ?"`. Following our methodology expressed in our assumption (H6), the system thus attempts to show how it cannot be the case that h is not accepted by showing what would happen if h is indeed not accepted. In this case, this amounts to showing the explanation for admissibility on $\{a, d\}$ (Figure 28) and looking for a problematic situation like an internal conflict or a defenceless argument. However, such a situation does not occur since

$\{a, d\}$ is also an admissible extension in Figure 1. In consequence, our goal to show that h is necessary in the result presented by the system has failed. In order to solve this kind of problem, two possibilities could be taken into account:

1. We could relax our assumption (H6) and define an additional answer showing that the presence of h is not a problem (this additional answer completing the answer we already build based on the absence of h).

2. We could consider that the original question of the user becomes "If h being accepted is not necessary, then why is h accepted?". This question could naturally be interpreted as a suggestion from the user to the system to change its result. In other terms, the user's question could be interpreted as offering feedback to the system. So a possible improvement of our system could be the adaptation of its explanations to the preferences of the user by a sequence of such questions. The same can be said about the addition of arguments in the result instead of their removal.

Of course, this focused example on a specific question in a specific context can be extrapolated to a more general case, in which the computation of an answer results in an explanation for an eligible result.

6 Related works

In this section we present existing works on the computation of explanations for Abstract Argumentation, and we compare them to our approach.

Following the line of [2], our work is an exploration of AF-based explanations. This survey distinguishes works that define explanations as: subgraphs, changes, extensions, dialogue-games.

We begin with the category of *subgraphs*, in which our work falls. Another example of work defining explanations as *subgraphs* is [9]. It was categorised in the second category in [2], a choice we do not agree with since the authors of [9] seek to explain the credulous non acceptance of some argument, not by changing its status, but by finding a strongly rejecting subframework. A strongly rejecting subframework is an induced subgraph of an Argumentation Framework that does not credulously accept an argument, and nor do its supergraphs (that are still induced subgraphs of the original AF). As such,

strongly rejecting subframeworks capture the core argumentative reasons for why an argument is not credulously accepted under a certain semantics. Our approach differs in that we define explanations for why a set of arguments is (not) an extension of a given semantics, or why some arguments are (not) part of an extension. In addition, there are some semantics not considered in our work (namely the grounded and preferred semantics), our graphs are not only induced subgraphs but also partial subgraphs, our second kind of explanations are contrastive, and each explanation results from the question it answers. [10] also studies subgraphs as explanations for the credulous non acceptance of some argument for a given semantics (except the grounded semantics). The difference here is that the authors consider both induced and partial subgraphs as explanations. However, they do it separately, and do not consider a combination of both like we do in our work.
A specific kind of graph that is used in explaining argumentative results is *defence trees*. Defence trees are trees where nodes are arguments and each successor of a node is an attacker of that node. As such, they can be used to prove whether an argument is defended or not. While not being subgraphs technically speaking, one can easily retrieve the subgraph represented by a defence tree using the original AF. Some works, like [11], use defence trees as explanations for argumentative results. The authors of [11] argue that a defence tree is a dialogical explanation for an argument since it can be used to show that it is defended. Other works, like [3], use them to compute their notion of explanations.

We now turn to the second category, which concerns *changes*. It consists in identifying what elements to remove from the AF in order to modify a given result. This is the method used in [18], in which the authors explain why an argument is not credulously accepted under admissibility. Their explanations consist of sets of arguments or attacks to remove from the AF in order to make the considered argument credulously accepted under admissibility in the resulting subgraph. Such sets were also studied in [19] (in which they were called "*diagnosis*") although the authors restrained themselves to the case where a given semantics does not yield any extension. Diagnoses are parts of the study of [10], which provides a way of computing them (as well as explanations as subgraphs) using logical formulas and provide complexity results. As noted in [10], diagnoses can be seen as a kind of dual of the computation of induced and partial subgraphs. Indeed, each diagnosis infers an induced or partial subgraph, and conversely, each induced or partial subgraph is computed using (the complement of) a diagnosis. As such, one

could wonder what are the properties of the complement of a set used to compute a certain induced or partial subgraph, or what can be said about the induced or partial subgraph computed from the complement of a given diagnosis.

The third category of approach consists of taking *sets of arguments* as explanations. This is probably the most widely used approach to this problem. In most of the works using this method, the point of view is to consider that explanation equates to justification. Hence the restriction to sets of arguments as explanations, since given a set of arguments, the original AF can be used to justify it by the mean of the attack relation. In [3], the authors define an explanation semantics, called related admissibility, which provides all the reasons why an argument belongs to an admissible set. The idea is to get rid of all the arguments that are not relevant for the acceptance of the considered argument, that is, those that are not connected to it via the attack relation. In [4, 5], the authors propose a basic framework to compute explanations as sets of arguments for the credulous/skeptical acceptance or non-acceptance of an argument. It is a framework for explanations since it can be parameterised in order to modify the way explanations are computed. In their work, the authors focus on some human biases used to select explanations such as simplicity (taken as minimality), sufficiency and necessity. In subsequent works ([13, 20]) the authors extend their framework to adapt it to Structured Argumentation (adding another parameter to control the form of the explanation) and to compute contrastive explanations (the intersection of why an argument (the fact) is accepted and why a set of arguments (the foils) are not). [6] proposes strong explanations for credulous acceptance of a set of arguments under a given semantics. A strong explanation is a set of arguments such that for every subgraph induced by a superset of the explanation, there exists an extension of the considered semantics that includes the set to explain. As such, the authors use subgraphs but only as means to compute their explanations rather than as explanations themselves. Some other works define their explanations from the observation that in the computation of an extension, some parts are non-deterministic choices, while others, deterministic, result from the first ones. For instance, in [7], the authors base their approach on the observation that each Strongly Connected Component (SCC) of an Argumentation Framework can be seen as making a choice for accepting conflict-free sets of arguments. From these choices results the rest of the accepted arguments. Thus, in a set of arguments, each argument can be explained by the set of arguments that were

chosen in a given SCC. Similarly, in [8], the authors observe that complete and admissible semantics are computed firstly by computing the grounded (resp. strongly admissible) extension, then making choices in even cycles, and finally computing the grounded (resp. strongly admissible) extension again. As such, they define the arguments chosen in the even cycles as the explanations for some complete or admissible extension.
Although the links between subgraph-based methods of explanation and extension-based methods are less direct than with diagnosis-based methods, there are still some that can be studied. Indeed, one could wonder what are the links between a subgraph computed by a subgraph-based method and the subgraph induced by the set computed by an extension-based method. Conversely, what can be said about the set that was used to compute an induced subgraph and the set computed by an extension-based method? Is the explanation of an extension-based method included in the explanation of a subgraph-based method? What about the converse? Those are questions that could help explore the links between the two methods. At a more fundamental level, one may note that the subgraph induced by the set computed by an extension-based method may contain more attacks than the subgraph that may be defined directly as an explanation. This may reveal that, in extension-based explanations, arguments are considered as the only relevant elements to explain, while in subgraph-based approaches, attacks and thus the structure of the AF are also considered relevant.

7 Conclusion and Future Work

To conclude, we have provided a general, methodological and procedural explanation system. This system relies on a formal grammar to generate the questions to which answers are required. The grammar is designed in such a way that it can be instantiated for any domain, using the notion of constitutive elements, provided that the context of the questions is known. Moreover, given an instance of this grammar, it is easy to adapt it to different questions by modifying the constitutive elements that are used. In this paper, we focused on a particular instance for the domain of Abstract Argumentation.

The instance we gave generates a wide range of questions, but we only considered some of them. More precisely, we dealt with questions related to why a given set of arguments is (or not) an extension under some semantics and questions related to why a given argument or set of arguments is (or not)

part of an extension of some semantics. In the former case, we stopped at non-contrastive questions, while we studied a particular case of contrast in the latter case, namely the contrast with another argument or set of arguments.

The answers we provide for these questions (i.e. explanations) are defined as subgraphs, tied to how a given question is generated using the grammar. That is, we use both the question's structure as well as its context to determine what kind of the subgraph to compute, and then use the content of the questions to compute the correct subgraph. Moreover, the context of the question allows for both dealing with the case of implicit information in the question and defining contrastive answers. The choice of subgraphs as explanations naturally yields visual explanations that are more easily understandable, and can always be used to potentially generate other forms of explanations.

The answers to questions related to why a given set of arguments is (or not) an extension under some semantics make use of the modularity of Abstract Argumentation semantics. That is, we take advantage of how semantics are composed in one another, and of the different principles that govern them in order to build our explanations. They subsequently take the form of several subgraphs illustrating each principle that composes the semantics at hand. These subgraphs, each of them associated with a specific checking procedure, can then be shown sequentially or aggregated together in one explanation.

The answers questions related to why a given argument or set of arguments is (or not) part of an extension of some semantics are always contrastive answers. More precisely, for these questions, we consider the question's context as the fact, and the question as providing the necessary elements to deduce the foil. We thus proceed to compute said foil as our answers using the content of the questions as well as our previous kind of answers. In the case of contrastive questions, we subsequently naturally interpret the contrast in the question as additional information to compute the foil.

We also discuss several aspects of our explanations, including but not limited to their accordance with Grice's maxims of conversation and some particular cases of answers.

Our present work can be extended in many ways. First, our general grammar could be refined in order to generate more questions or include some subtleties in the generation of questions in order to tailor even more precisely the answers we provide. On this subject, the answers we define for now are

tied to a specific sequence of production rules. We intend to shift from this way of doing to a more modular one, associating specific modifications of the answer to each production rule. Amongst the questions that we already considered there are some variants that we left aside for now. Those typically concern contrastive questions for questions related to the status of (non-)extension of a given set of arguments or particular cases of contrasts for questions related to some argument or set of arguments being part of a given extension. For instance, it could prove interesting to answer to questions like `"Why do we have S as a conflict-free extension and not as an admissible extension ?"` or `"Why do we have {a, b} accepted in the AF (A, R) and not in the AF (A', R') ?"`. For now we only considered the case of conflict-free, admissible, complete and stable semantics, we also intend to study answers to questions related to grounded and preferred semantics. Finally, a formal study of the properties of our explanations, as well as an empirical evaluation of their quality are planned.

References

[1] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (2) (1995) 321–357. doi:10.1016/0004-3702(94)00041-x.

[2] K. Cyras, A. Rago, E. Albini, P. Baroni, F. Toni, Argumentative XAI: A survey, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence IJCAI, IJCAI Organization, Online Event / Montreal, Canada, 2021, pp. 4392–4399. doi:10.24963/ijcai.2021/600.

[3] X. Fan, F. Toni, On computing explanations in argumentation, in: B. Bonet, S. Koenig (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press, Austin, Texas, USA, 2015, pp. 1496–1502.

[4] A. Borg, F. Bex, Necessary and sufficient explanations in abstract argumentation, Computing Research Repository (CoRR) abs/2011.02414 (2020). arXiv:2011.02414.

[5] A. Borg, F. Bex, Necessary and sufficient explanations for argumentation-based conclusions, in: J. Vejnarová, N. Wilson (Eds.), Proceedings of the 16th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU, Vol. 12897 of Lecture Notes in Computer Science, Springer, Prague, Czech Republic, 2021, pp. 45–58. doi:10.1007/978-3-030-86772-0_4.

[6] M. Ulbricht, J. P. Wallner, Strong explanations in abstract argumentation, in: Proceedings of The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI Press, Online event, 2021, pp. 6496–6504.

[7] B. Liao, L. van der Torre, Explanation semantics for abstract argumentation, in: H. Prakken, S. Bistarelli, F. Santini, C. Taticchi (Eds.), Computational Models of Argument - Proceedings of COMMA 2020, Vol. 326 of Frontiers in Artificial Intelligence and Applications, IOS Press, Perugia, Italy, 2020, pp. 271–282. doi:10.3233/FAIA200511.

[8] R. Baumann, M. Ulbricht, Choices and their consequences - explaining acceptable sets in abstract argumentation frameworks, in: M. Bienvenu, G. Lakemeyer, E. Erdem (Eds.), Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning KR, IJCAI Organization, Online event, 2021, pp. 110–119. doi:10.24963/kr.2021/11.

[9] Z. G. Saribatur, J. P. Wallner, S. Woltran, Explaining non-acceptability in abstract argumentation, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), Proceedings of the 24th European Conference on Artificial Intelligence ECAI, Vol. 325 of Frontiers in Artificial Intelligence and Applications, IOS Press, Santiago de Compostela, Spain, 2020, pp. 881–888. doi:10.3233/FAIA200179.

[10] A. Niskanen, M. Järvisalo, Smallest explanations and diagnoses of rejection in abstract argumentation, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning KR, IJCAI Organization, Rhodes, Greece, 2020, pp. 667–671. doi:10.24963/kr.2020/67.

[11] T. Racharak, S. Tojo, On explanation of propositional logic-based argumentation system, in: A. P. Rocha, L. Steels, H. J. van den Herik (Eds.), Proceedings of the 13th International Conference on Agents and Artificial Intelligence ICAART, Vol. 2, SCITEPRESS, Online Streaming, 2021, pp. 323–332. doi:10.5220/0010318103230332.

[12] S. Vesic, B. Yun, P. Teovanovic, Graphical representation enhances human compliance with principles for graded argumentation semantics, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2022), to appear.

[13] A. Borg, F. Bex, A basic framework for explanations in argumentation, IEEE Intelligent Systems 36 (2) (2021) 25–35. doi:10.1109/mis.2021.3053102.

[14] J. W. Backus, The syntax and semantics of the proposed international algebraic language of the zurich ACM-GAMM conference, in: Proceedings of the 1st International Conference on Information Processing, UNESCO, Paris, France, 1959, pp. 125–131.

[15] N. Chomsky, G. A. Miller, Introduction to the formal analysis of natural languages, in: R. D. Luce, R. R. Bush, E. Galanter (Eds.), Handbook of Mathematical Psychology, Vol. 2, John Wiley and Sons, Inc., New York and London, 1963, Ch. 11, pp. 269–321.

[16] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38. doi:10.1016/j.artint.2018.07.007.

[17] H. P. Grice, Logic and conversation, in: P. Cole, J. L. Morgan (Eds.), Speech Acts, Vol. 3 of Syntax and Semantics, Brill, Leiden, The Netherlands, 1975, pp. 41–58. doi:10.1163/9789004368811_003.

[18] X. Fan, F. Toni, On explanations for non-acceptable arguments, in: E. Black, S. Modgil, N. Oren (Eds.), Theory and Applications of Formal Argumentation, TAFA - Third International Workshop, Vol. 9524 of Lecture Notes in Computer Science, Springer, Buenos Aires, Argentina, 2015, pp. 112–127. doi:10.1007/978-3-319-28460-6_7.

[19] M. Ulbricht, R. Baumann, If nothing is accepted - repairing argumentation frameworks, Journal of Artificial Intelligence Reseach 66 (2019) 1099–1145. doi:10.1613/jair.1.11791.

[20] A. Borg, F. Bex, Contrastive explanations for argumentation-based conclusions, Computing Research Repository (CoRR) abs/2107.03265 (2021). arXiv:2107.03265.