# Preferred Extensions of Argumentation Frameworks: Computation and Query Answering

Sylvie Doutre and Jérôme Mengin

**Mars 2001**

**Rapport IRIT/2001-03-R**

# PREFERRED EXTENSIONS OF ARGUMENTATION FRAMEWORKS: COMPUTATION AND QUERY ANSWERING

SYLVIE DOUTRE AND JÉRÔME MENGIN

Abstract. The preferred semantics for argumentation frameworks seems to capture well the intuition behind the stable semantics while avoiding several of its drawbacks. Although the stable semantics has been thoroughly studied, and several algorithms have been proposed for solving problems related to it, it seems that the algorithmic side of the preferred semantics has received less attention. In this paper, we propose algorithms, based on the enumeration of some subsets of a given set of arguments, for the following tasks: 1) deciding if a given argument is in a preferred extension of a given argumentation framework; 2) deciding if the argument is in all the preferred extensions of the framework; 3) generating the preferred extensions of the framework.

Résumé. La sémantique préférée des systèmes d'argumentation semble bien capturer l'intuition que la sémantique stable essaye de formalier, tout en en évitant plusieurs défauts. Alors que la sémantique stable a fait l'objet de nombreuses études, et que plusieurs algorithmes ont été proposés pour résoudre des problèmes liés à cette sémantique, il semble que la sémantique préférée ait reçu moins d'attention. Ce rapport présente des algorithmes pour répondre, étant donné un système d'argumentation, aux questions suivantes: 1) Un argument donné est-il dans au moins une extension ? 2) Un argument donné est-il dans toutes les extensions ? 3) Quelles sont les extensions du système ? Nos algorithmes sont basés sur une technique d'énumération de parties d'un ensemble d'arguments.

## Table des matières

# 1. Introduction

The argumentation frameworks of [Dun95, BDKT97] abstract many formal systems that have been developed for the automation of common-sense reasoning or to give a meaning to logic programs. Argumentation frameworks provide a unifying tool for the study of several aspects of these systems, notably their semantics.

Underlying these systems is some notion of deduction, often nonmonotonic. One important consequence of nonmonotonicity is that the set of consequences that can be deduced from such a system is usually inconsistent in the sense of classical logic. The semantics is then given in terms of *extensions*, where each extension corresponds to a possible interpretation of the world. All extensions of a theory are maximally consistent subsets of the set of nonmonotonic consequences of the theory, but various semantics accept more or less of these subsets as extensions. Probably the most widespread semantics is the stable semantics, which is also the most restrictive one. Its definition is very intuitive, but it gives problematic results for many theories; in particular, there are theories which have no stable extension. [Dun95] has proposed another semantics, the *preferred semantics* for argumentation frameworks, which seems to capture the intuition behind the stable semantics while avoiding several of its drawbacks.

The stable semantics has been thoroughly studied, and several algorithms have been proposed for the computation of stable extensions of nonmonotonic theories. Algorithms by e.g. [Rei87, Lev91, DMP97, Nie95] rely on some underlying monotonic deduction system that provides information about the possible conflicts between arguments: they then compute extensions so as to avoid these conflicts within each single extension. It has been shown by e.g. [Ino92] that there is a strong connection between the computation of the possible conflicts and consequence finding. These conflicts can be captured in a directed graph, and [CL91, DT93, DM94] have proved that stable extensions of an argumentation framework correspond to particular subsets of this graph called *kernels* in graph theory [Ber73]. This connection between graphs and nonmonotonic theories is the basis for an algorithm in [DMP97] that computes stable extensions of a default theory.

It seems that the algorithmic side of the preferred semantics has received less attention. The worst-case analysis of [DNT99, DNT00] show that the preferred semantics is at least as hard to compute as the stable semantics, and often harder (depending on what are accepted as consequences of argumentation frameworks: the elements of the union of the extensions or of their intersection). One question related to the preferred semantics has been studied by a number of authors: given an argumentation framework and an argument, is there a preferred extension of the framework that contains the argument? Vreeswijk and Prakken [VP00] propose a proof theory to answer that question, but stop short of actually giving an algorithm that would implement their proof theory. [DKT96] propose a proof procedure to answer the same question but in the restricted case where the argumentation framework corresponds to a logic program. In [DM00], we proposed an algorithm that computes all the preferred extensions of an argumentation framework. This algorithm is based on a technique called set-enumeration in [Rym], which can be used to generate all the subsets of a given set. This technique has been applied to many problems studied in Artificial Intelligence, like propositional deduction or the computation of stable extensions of some nonmonotonic logics. In this paper, we propose algorithms, also based on the enumeration of the subsets of a given set of arguments, for the following tasks: 1) deciding if a given argument is in an extension of a given argumentation framework; 2) deciding if the argument is in all the extensions of the framework; 3) generating the extensions of the framework. In particular, the third algorithm improves on that presented in [DM00].

The paper is built as follows: the next section presents the preferred semantics of argumentation frameworks, and a characterization in terms of graph-theoretic concepts. Set-enumeration is described in section 3, where we also exhibit properties of preferred extensions that can be used to reduce the number of generated sets of arguments. The algorithms are described in section 4. Section 5 concludes the paper with a comparison of our algorithms with related works.

## 2. The Preferred Semantics for Argumentation Frameworks

In this section, we present Dung's argumentation framework [Dun95] and the preferred semantics. We give a characterization of the preferred semantics in terms of graphs. We also define the notions of credulous and sceptical consequences.

### 2.1. Argumentation Frameworks.

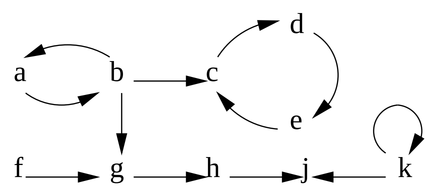
**Definition 2.1.** [Dun95] An *argumentation framework* is a pair $(A,R)$ where $A$ is a set of arguments and $R$ is a binary relation over arguments, i.e. $R \subseteq A \times A$. Given two arguments $a$ and $b$, $(a,b) \in R$ or $a\ R\ b$ means $a$ *attacks* $b$ ($a$ is said to be an *attacker* of $b$). Moreover, we say that a set $S \subseteq A$ of arguments attacks an argument $a$ if some argument $b$ in $S$ attacks $a$.

Thus an argumentation framework can be simply represented as a directed graph, where vertices are the arguments and edges correspond to the elements of $R$. Given $(A, R)$ an argumentation framework and $a$ an argument of $A$, we denote by:

- $R^+(a) = \{b \in A \mid (a,b) \in R\}$ the set of *successors* of $a$, that is, the set of arguments attacked by $a$;
- $R^-(a) = \{b \in A \mid (b,a) \in R\}$ the set of *predecessors* of $a$, that is, the set of arguments that attack $a$;
- $R^\pm(a) = R^+(a) \cup R^-(a)$ the union of $a$'s predecessors and successors.

Moreover, given a set $S \subseteq A$ of arguments and $\varepsilon \in \{+,-,\pm\}$, $R^\varepsilon(S) = \cup_{a \in S} R^\varepsilon(a)$.

**Example 2.2.** Let $AF_1 = (A,R)$ with $A = \{a,b,c,d,e,f,g,h,j,k\}$ and $R$ as indicated on the graph of the figure below. We use this example throughout the section to illustrate our definitions and propositions, and then as a running example for our algorithms.




To introduce the preferred semantics of an argumentation framework, we define the relation of defense, conflict-free sets and admissible sets of arguments:

**Definition 2.3.** Let $(A,R)$ be an argumentation framework.

- An argument $a \in A$ *is defended by* a set $S \subseteq A$ of arguments (or $S$ *defends* $a$) if and only if $\forall b \in A$, if $b\ R\ a$ then $S$ attacks $b$, i.e. $\exists c \in S$ such that $c\ R\ b$.
- A set $S \subseteq A$ is *conflict-free* if and only if there are no arguments $a$ and $b$ in $S$ such that $a$ attacks $b$.
- A set $S \subseteq A$ is *admissible* if and only if $S$ is conflict-free and $\forall x \in S$, $S$ defends $x$.

The equivalent formulation with graph-theoretic notations is:

- A set $S$ of arguments defends some argument $a$ if $R^-(a) \subseteq R^+(S)$.
- $S$ is conflict-free if $R^\pm(S) \cap S = \emptyset$ or equivalently if $R^+(S) \cap S = \emptyset$.
- $S$ is admissible if and only if $R^+(S) \cap S = \emptyset$ and $R^-(S) \subseteq R^+(S)$.

**Example 2.2 (cont.)** Let $S_1 = \{f,g\}$, $S_2 = \{f,j\}$ and $S_3 = \{f,h\}$. $R^+(S_1) = R^+(f) \cup R^+(g) = \{g,h\}$, $R^+(S_2) = \{g\}$ and $R^+(S_3) = \{g,j\}$.

$S_1$, $S_2$ and $S_3$ defend $h$ against the attack of $g$, since $g$ is a successor of each set. $S_2$ and $S_3$ are conflict-free, not $S_1$, since $R^+(S_1) \cap S_1 = \{g\}$, so $S_1$ can not be an admissible set. $S_2$ cannot defend its argument $j$ against the attacks of $h$ and $k$, since $R^-(S_2) = R^-(f) \cup R^-(j) = \{h,k\}$ and $R^-(S_2) \not\subseteq R^+(S_2)$, so $S_2$ cannot be an admissible set either. $S_3$ is able to defend every argument it contains since $R^-(S_3) = \{g\} \subseteq R^+(S_3)$, and because it is conflict-free, it is admissible.

2.2. **Preferred Semantics.** Dung [Dun95] defines the preferred semantics of an argumentation framework by the set of preferred extensions. Precisely, a preferred extension is a maximal (with respect to set inclusion) admissible set of arguments, that is:

**Definition 2.4.** Given an argumentation framework $(A, R)$, a set of arguments $S \subseteq A$ is a preferred extension if and only if:

1. $S$ is conflict-free;
2. $S$ defends every argument it contains;
3. $S$ is $\subseteq -$maximal such that 1 and 2.

The set of the preferred extensions of $(A, R)$ is denoted by $\mathrm{Pref}(A, R)$. Let us now give an original characterization of preferred extensions, based on the graph representation of argumentation frameworks:

**Proposition 2.5.** *Given some argumentation framework $(A, R)$, a subset $S$ of $A$ is a preferred extension if and only if the following conditions hold:*

1. $R^+(S) \cap S = \emptyset$;
2. $R^-(S) \subseteq R^+(S)$;
3. *for every non-empty* $X \subseteq A - S$, $X \cap R^+(S \cup X) \neq \emptyset$ *or* $R^-(X) \not\subseteq R^+(S \cup X)$.

*Proof.* By definition, $S \subseteq A$ is admissible if and only if properties 1. and 2. hold for $S$. Since preferred extensions are defined to be subsets of $A$ that are admissible and strictly included in no other admissible subset of $A$, $S$ is preferred if and only if properties 1. and 2. hold for $S$ and for every $X \subseteq A - S$ such that $X \neq \emptyset$, 1. or 2. does not hold for $S \cup X$. Let $S$ be a subset of $A$ such that properties 1. and 2. hold for $S$, $S$ is preferred if and only if for every non empty subset $X$ of $A - S$:

$$R^+(S \cup X) \cap (S \cup X) \neq \emptyset \text{ or } R^-(S \cup X) \not\subseteq R^+(S \cup X). \tag{2.1}$$

Since $S \cap R^+(S) = \emptyset$ and $R^-(S) \subseteq R^+(S)$, we have the following equivalences:

$$\begin{aligned} R^+(S \cup X) \cap (S \cup X) \neq \emptyset \quad &\Leftrightarrow \quad R^+(X) \cap X \neq \emptyset \text{ or } R^+(S) \cap X \neq \emptyset \text{ or } R^+(X) \cap S \neq \emptyset \\ &\Leftrightarrow \quad R^+(X) \cap X \neq \emptyset \text{ or } R^+(S) \cap X \neq \emptyset \text{ or } R^-(S) \cap X \neq \emptyset \\ &\Leftrightarrow \quad R^+(X) \cap X \neq \emptyset \text{ or } R^+(S) \cap X \neq \emptyset \\ &\Leftrightarrow \quad R^+(X \cup S) \cap X \neq \emptyset \end{aligned}$$

Moreover, since $R^-(S) \subseteq R^+(S)$, $R^-(S \cup X) \not\subseteq R^+(S \cup X)$ if and only if $R^-(X) \not\subseteq R^+(S \cup X)$. □

**Example 2.2 (cont.)** $AF_1$ possesses two preferred extensions: $\{b, d, f, h\}$ and $\{a, f, h\}$.

An argumentation framework $AF = (A, R)$ is said to be *well-founded* if and only if there exists no infinite sequence $a_0$, $a_1$, ..., $a_n$, ... such that for each $i$, $a_i \in A$ and $(a_{i+1}, a_i) \in R$ . In other words, $AF$ is well- founded if $AF$ is finite (i.e. set $R$ is finite) and contains no cycle. Dung exhibits interesting properties of the preferred semantics:

**Proposition 2.6.** [Dun95]
*Each admissible set is contained in a preferred extension.*
*Every argumentation framework possesses at least one preferred extension.*
*Every well-founded argumentation framework has exactly one preferred extension.*

Working with frameworks containing cycles is not straightforward, since they generally lead to multiple extensions. Moreover, these extensions generally contain contradictory arguments. In the next section, we recall the two types of conclusions that can be expected from argumentation frameworks with multiple preferred extensions. In the sequel, we distinguish between two kinds of cycles: odd ones (with an odd number of edges) and even ones (with an even number of edges). Some of the problems encountered with cycles will be illustrated with our example $AF_1$, which is composed of an even cycle ($aRbRa$) and of two odd ones ($cRdReRc$ and $kRk$: this is the smallest cycle we can find on a graph, that is $k$ is a self-attacking argument).

2.3. **Credulous and sceptical consequences.** We want to answer two important questions on preferred extensions: is a given argument contained in at least one preferred extension of an argumentation framework? Is it contained in every preferred extension? In other words, is an argument a credulous or a sceptical consequence of an argumentation framework under the preferred semantics?

**Definition 2.7.** Given an argumentation framework $(A, R)$ and an argument $a \in A$:

$a$ is a *credulous consequence* of $(A, R)$ under the preferred semantics, denoted by $(A, R) \mid\!\sim_{cr} a$, if and only if $a$ is contained in the union of the preferred extensions of $(A, R)$, or equivalently if $a$ is contained in at least one preferred extension of $(A, R)$.

$a$ is a *sceptical consequence* of $(A, R)$ under the preferred semantics, denoted by $(A, R) \mid\!\sim_{sc} a$, if and only if $a$ is contained in the intersection of all the preferred extensions of $(A, R)$, or equivalently if $a$ is contained in every preferred extension of $(A, R)$.

**Example 2.2 (cont.)** $b$, $a$, $d$, $f$ and $h$ are the credulous consequences of $AF_1$. The only sceptical consequences of $AF_1$ are $f$ and $h$.

## 3. Extension enumeration

3.1. **The basic technique.** Before we describe our algorithms in the next section, we present here the general technique on which they are based, and formal properties that will be used to speed-up the computations.

The enumeration of the subsets of a given set $A$ can be performed by exploring a binary tree, the nodes of which are labeled by a partition of $A$ into three sets $I$, $O$, and $U$. If $U = \emptyset$, the node is a leaf, corresponding to the subset $I$ of $A$. More generally, at any given node $n$, $I$ is a set of elements of $A$ that will be in every subset of $A$ found below $n$, and $O$ is a set of elements of $A$ that will be out of every subset of $A$ found below $n$, while $U = A - (I \cup O)$; thus $U$ is a set of elements that are *undecided* at that stage, they can end up in some of the sets below the current node and out of some other sets below that node. If $n$ is such that $U \neq \emptyset$, then $U$ has at most two children: one is labelled with the partition $(I \cup \{x\}, O, U - \{x\})$, the other is labelled with the partition $(I, O \cup \{x\}, U - \{x\})$ for some $x \in U$; if all the subsets of $A$ have to be enumerated, then $n$ has exactly two children.

Since we will be interested in the preferred extensions of some argumentation framework $(A, R)$, only some subsets of $A$ are of interest. Even in this case, it would be possible to enumerate all subsets of $A$, and test for each of them if it verifies a given property. However, it can happen that it is sufficient to generate only one child, or no child at all, for some nodes.

Before we investigate more closely the enumeration of preferred extensions, let us introduce notations that will lighten the presentation of our results. Since preferred extensions are conflict-free, and since the preferred extensions below a node labelled by $(I, O, U)$ contain $I$ but no element of $O$, it is legitimate to generate only nodes such that $R^{\pm}(I) \subseteq O$, so that $I$ remains conflict-free. We call *$R$-candidate* a triple of disjoint sets $(I, O, U)$ such that $R^{\pm}(I) \subseteq O$. Given such a $R$-candidate $C = (I, U, O)$, and an element $x$ of $U$, we denote by $C + x$ the triple $(I \cup \{x\}, O \cup R^{\pm}(x), U - (\{x\} \cup R^{\pm}(x))$, and by $C - x$ the triple $(I, O \cup \{x\}, U - \{x\})$. Given a binary relation $R$ and a $R$-candidate $C = (I, O, U)$, we denote by $\mathrm{Pref}^*(C, R)$ the set of preferred extensions that are below a node labelled by $(I, O, U)$:

$$\mathrm{Pref}^*((I, O, U), R) = \{S \in \mathrm{Pref}(I \cup O \cup U, R) \mid I \subseteq S \subseteq I \cup U\}$$

Our first result shows that we can have a complete enumeration while exploring only nodes that are $R$-candidates:

**Proposition 3.1.** *Let $R$ be a binary relation, let $C = (I, O, U)$ be a $R$-candidate, and let $x \in U$. If $x \notin R^+(x)$, then $C+x$ and $C-x$ are both $R$-candidates, and* $\mathrm{Pref}^*(C, R) = \mathrm{Pref}^*(C+x, R) \cup \mathrm{Pref}^*(C-x, R)$. *If $x \in R^+(x)$, then $C - x$ is a $R$-candidate, and* $\mathrm{Pref}^*(C, R) = \mathrm{Pref}^*(C - x, R)$.

*Proof.* Let us define

$$\mathrm{Max}^*(S, A, R) \equiv \forall X \subseteq A - S, X = \emptyset \vee X \cap R^+(S \cup X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(S \cup X).$$

Then $S \in \mathrm{Pref}(A, R)$ if and only if $S \cap R^+(S) = \emptyset$, $R^-(S) \subseteq R^+(S)$ and $\mathrm{Max}^*(S, A, R)$ holds.

In the proofs, we will use the following identities where $A$, $B$, $C$, and $D$ are sets:

$$\begin{aligned} A \cup (B - A) &= A \cup B \\ (A \cup B) \cap (C - B) &= (A \cap C) - B \\ ((A - B) \cup C) - (B \cup D) &= (A \cup C) - (B \cup D) \end{aligned}$$

Suppose first that $x \notin R^+(x)$. Let us prove that $C + x = (I', O', U') = (I \cup \{x\}, O \cup R^\pm(x), U - (\{x\} \cup R^\pm(x)))$ is a $R$-candidate. Note that since $x \notin R^+(x)$, $\{x\} \cap R^\pm(x) = \emptyset$. Let us prove that $I'$, $O'$ and $U'$ are disjoint sets:

- $I' \cap U' = (I \cup \{x\}) \cap (U - (\{x\} \cup R^\pm(x))) \subseteq (I \cup \{x\}) \cap (U - \{x\}) = (I \cap U) - \{x\} = \emptyset$ because $I$ and $U$ are disjoint sets;
- $I' \cap O' = (I \cup \{x\}) \cap (O \cup R^\pm(x)) = (I \cap O) \cup (I \cap R^\pm(x)) \cup (\{x\} \cap O) \cup (\{x\} \cap R^\pm(x)) = \emptyset$ (in particular, since $R^\pm(I) \subseteq O$, and since $x \in U$, $R^\pm(I) \cap \{x\} = \emptyset$, which is equivalent to $I \cap R^\pm(x) = \emptyset$);
- $U' \cap O' = (U - (\{x\} \cup R^\pm(x))) \cap (O \cup R^\pm(x)) = (U - \{x\}) \cap ((U - R^\pm(x)) \cap (O \cup R^\pm(x))) = (U - \{x\}) \cap ((U \cap 0) - R^\pm(x)) \subseteq U \cap O = \emptyset$

Clearly, $I' \cup O' \cup U' = I \cup O \cup U$ since $U$' is obtained by removing some elements from $U$ which are added to $I$ or $O$ in order to form $I'$ and $O'$. Finally, let us prove that $R^\pm(I') \subseteq O'$: $R^\pm(I') = R^\pm(I \cup \{x\}) = R^\pm(I) \cup R^\pm(x) \subseteq O \cup R^\pm(x) = O'$.

In order to prove that $\text{Pref}^*(C, R) \supseteq \text{Pref}^*(C + x, R)$, let $S \in \text{Pref}^*(C + x, R)$: then $R^+(S) \cap S = \emptyset$, $R^-(S) \subseteq R^+(S)$ and $\text{Max}^*(S, I \cup O \cup U, R) \equiv \text{Max}^*(S, I' \cup O' \cup U', R)$ holds. Moreover, $I \subseteq I' \subseteq S$, and $S \subseteq I' \cup U' = I \cup \{x\} \cup (U - (\{x\} \cup R^\pm(x))) \subseteq I \cup U$ because $x \in U$.

Let us now prove that $C - x = (I'', O'', U'')$ is a $R$-candidate. Clearly, $(I'', O'', U'')$ is a triple of disjoint sets and $I'' \cup O'' \cup I'' = I \cup O \cup U$ because $I$, $O$, $U$ are disjoint and $I'' = I$, and $O''$ is formed by adding to $O$ an element $x$ which is removed from $U$ to obtain $U''$. Moreover, $R^\pm(I'') = R^\pm(I) \subseteq O \subseteq O \cup \{x\} = O''$.

In order to prove that $\text{Pref}^*(C, R) \supseteq \text{Pref}^*(C - x), R)$, let $S \in \text{Pref}^*(C - x), R)$, then $R^+(S) \cap S = \emptyset$, $R^-(S) \subseteq R^+(S)$ and $\text{Max}^*(S, I \cup O \cup U, R) \equiv \text{Max}^*(S, I'' \cup O'' \cup U'', R)$ holds. Moreover, $I = I'' \subseteq S$, and $S \subseteq U'' \cup I'' = (U - \{x\}) \cup I \subseteq U \cup I$.

There remains to prove that $\text{Pref}^*(C, R) \subseteq \text{Pref}^*(C + x, R) \cup \text{Pref}^*(C - x, R)$. Let $S \in \text{Pref}^*(C, R)$: we will distinguish two cases, depending on wether $x \in S$ or not. Suppose first that $x \in S$, then since $I \subseteq S \subseteq I \cup U$, we have $I' \subseteq S$. Moreover, $R^+(S) \cap S = \emptyset$, thus $R^\pm(x) \cap S = \emptyset$, hence $S \subseteq I' \cup U'$: this proves that $S \in \text{Pref}^*(C + x, R)$. Suppose now that $x \notin S$, then we have $I'' = I \subseteq S$, and $I'' \cup U'' = I \cup U - \{x\}$, thus $S \subseteq I'' \cup U''$ because $x \notin S$: in this case, $S \in \text{Pref}^*(C - x, R)$.

Let us now suppose that $x \in R^+(x)$. The proof that $C - x = (I'', O'', U'')$ is a $R$-candidate is the same as before and we still have $I'' \cup O'' \cup I'' = I \cup O \cup U$ because $I$, and the proof that $\text{Pref}^*(C, R) \supseteq \text{Pref}^*(C - x, R)$ still holds too. In order to prove that $\text{Pref}^*(C, R) \subseteq \text{Pref}^*(C - x, R)$, let $S \in \text{Pref}^*(C, R)$: then $S \cap R^+(S) = \emptyset$, $R^-(S) \subseteq R^+(S)$ and $\text{Max}^*(S, I \cup O \cup U, R) \equiv \text{Max}^*(S, I'' \cup O'' \cup U'', R)$ holds. Morevoer $I'' = I \subseteq S$. Since $S \cap R^+(S) = \emptyset$, $x \notin S$, thus we have $I'' = I \subseteq S$, and $I'' \cup U'' = I \cup U - \{x\}$, thus $S \subseteq I'' \cup U''$ because $x \notin S$. Hence, $S \in \text{Pref}^*(C - x, R)$. □

Essentially, our algorithms will therefore select, at each stage, an undecided argument $x$ and generate two new $R$-candidates, by putting $x$ in $I$ or in $O$ (and adding predecessors and successors of $x$ to $O$ in the first case), thereby emptying $U$. Note that arguments that attack themselves are particular, since they can be in no preferred extension. In the sequel, we denote by $\text{Refl}(A, R)$ the set of these arguments:

$$\text{Refl}(A, R) = \{x \in A \mid x \in R^+(x)\}.$$

3.2. **Checking the leaves.** When $U$ is empty, we need to check if the leaf corresponds to a preferred extension. Let $(I, O, \emptyset)$ be the $R$-candidate that we find at the leaf, then $I$ is conflict-free because $R^\pm(I) \subseteq O$. We need to check that $I$ defends itself, so that $I$ is admissible, and some maximality condition: it must be the case that for every subset $X$ of $O$, $I \cup X$ is not admissible. We define the following property:

$$\text{Max}(I, O, U, R) \equiv \forall X \subseteq O, X = \emptyset \vee X \cap R^+(X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup U \cup X).$$

**Proposition 3.2.** *Let $R$ be a binary relation, and let $C = (I, O, U)$ be a $R$-candidate such that $U = \emptyset$. If $R^-(I) \subseteq R^+(I)$, and if* $\mathrm{Max}(I, O - R^\pm(I), \emptyset, R)$ *holds, then* $\mathrm{Pref}^*(C, R) = \{I\}$*, otherwise* $\mathrm{Pref}^*(C, R) = \emptyset$.

*Proof.* By definition, if $S \in \mathrm{Pref}^*((I, O, \emptyset), R)$ then $I \subseteq S \subseteq I$ since $U$ is empty. Thus the only possible element of $\mathrm{Pref}^*((I, O, U), R)$ is $I$. We know that $R^\pm(I) \subseteq O$ because $C$ is a $R$-candidate, and $I$ and $O$ are disjoint sets, thus $R^+(I) \cap I = \emptyset$. Therefore, $I \in \mathrm{Pref}^*((I, O, \emptyset), R)$ if and only if $R^-(I) \subseteq R^+(I)$ and $\mathrm{Max}^*(S, I \cup O, R)$ holds:

$$\begin{aligned}
&\mathrm{Max}^*(I, I \cup O, R)\\
&\quad\equiv \forall X \subseteq (I \cup O) - I, X = \emptyset \vee X \cap R^+(I \cup X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X)\\
&\quad\equiv \forall X \subseteq O, X = \emptyset \vee X \cap (R^+(I) \cup R^+(X)) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X))\\
&\quad\equiv \forall X \subseteq O - R^\pm(I), X = \emptyset \vee X \cap R^+(X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X))\\
&\quad\equiv \mathrm{Max}(I, O - R^\pm(I), \emptyset, R)
\end{aligned}$$

□

3.3. **Prunning properties.** In the remainder of the section, we study properties that can be applied to prune the binary search. First we have strong properties that can be used to stop the search at a given stage, because they guarantee the existence or the absence of preferred extensions that verify certain conditions. These properties will be used in the query answering algorithms: these algorithms do not need to exhibit extensions, but only test if there is some or no extension that contains a particular argument.

Our first condition guarantees that there can be no preferred extension below a node: since a preferred extension must defend itself, every element of $I$ at a given stage must be defensible; so every predecessor of $I$ must have predecessors that have not already been put out of the extensions currently explored; if this is not the case, then there is no hope of finding a preferred extension below the current node. Conversely, there is a condition that guarantees that there exists a preferred extension that contains a set $I$: if every element of $I$ is defended by $I$, that is, if every predecessor of $I$ is also a successor of $I$, then $I$ is admissible, therefore contained in at least one preferred extension (which may not however be found below the current node). Formally, the following holds:

**Proposition 3.3.** *Let $R$ be a binary relation, and let $C = (I, O, U)$ be a $R$-candidate. If $x \in R^-(I) - R^+(I)$ is such that $R^-(x) \subseteq O$, then* $\mathrm{Pref}^*(C, R) = \emptyset$. *Otherwise, if $R^-(I) - R^+(I) = \emptyset$, then* $\mathrm{Pref}(I \cup O \cup U, R) \neq \emptyset$ *and $I$ is admissible.*

*Proof.* Suppose first that $x \in R^-(I) - R^+(I)$ is such that $R^-(x) \subseteq O$, and suppose that $S$ is such that $I \subseteq S \subseteq I \cup U$, then $S \cap O = \emptyset$, therefore $R^-(x) \cap S = \emptyset$, i.e. $x \notin R^+(S)$, so $x \in R^-(S) - R^+(S)$, thus $R^-(S) \not\subseteq R^+(S)$. Thus $\mathrm{Pref}^*(C, R) = \emptyset$.

Suppose now that $R^-(I) - R^+(I) = \emptyset$, that is, $R^-(I) \subseteq R^+(I)$: then $I$ defends all its elements. Moreover, since $C$ is a $R$-candidate, $I$ is conflict-free. Therefore, $I$ is admissible, so $I$ is contained in at least one preferred extension of $(I \cup U \cup O, R)$. □

In particular, these conditions can be used to check if a given argument $a \in I$ is contained in at least one preferred extension. However, if we want to check if there is some extension that does not contain a given argument $a \in O$, then we need a stronger property:

**Proposition 3.4.** *Let $R$ be a binary relation, and let $(I, O, U)$ be a $R$-candidate such that $R^-(I) - R^+(I) = \emptyset$. If $a \in R^+(I)$, or if $a \in O - R^\pm(I)$ and* $\mathrm{Max}(I, O - R^\pm(I), U, R)$ *holds, then there exists* $S \in \mathrm{Pref}(I \cup U \cup O, R)$ *such that $a \notin S$.*

*Proof.* From the previous proposition we know that $I$ is contained in a preferred extension $S$ of $(I \cup O \cup U, R)$. If $a \in R^+(I) \subseteq R^+(S)$, since $S$ is conflict-free, $a \notin S$. Otherwise, if $a \in O - R^\pm(I)$ and $\mathrm{Max}(I, O - R^\pm(I), U, R)$ holds, then $a$ cannot be defended by $S$: let $X = \{a\}$, then either $X \cap R^+(X) \neq \emptyset$, that is $aRa$, in which case $a$ can be in no preferred extension, or $R^-(X) \not\subseteq R^+(I \cup U \cup X)$, thus, if $S \subseteq I \cup U$, $R^-(X) \not\subseteq R^+(S)$: since $S$ is preferred, this implies that $X \not\subseteq S$, that is, $a \notin S$. □

Let us now turn to properties that will enable us to prune one half of the subtree below some nodes. That is, we exhibit conditions that guarantee that a given $x \in U$ is such that every extension $S$ such that $I \subseteq S \subseteq I \cup U$ contains $x$, or such that none of these extensions contains $x$. We define the following sets:

$$\begin{aligned}
\mathrm{App}(I,U,R) &= \{x \in U \mid x \notin R^{\pm}(I) \cup R^{+}(U), R^{-}(x) \subseteq R^{+}(I \cup \{x\})\} \\
\mathrm{Undef}(I,U,R) &= \{x \in U \mid R^{-}(x) \not\subseteq R^{+}(I \cup U)\} \\
\mathrm{Heroes}(I,U,R) &= \{x \in U \mid \exists z \in I, \exists y \in R^{-}(z), \{x\} = R^{-}(y) \cap (U \cup I)\} \\
\mathrm{Traitors}(I,O,U,R) &= \{x \in U \mid \exists y \in R^{+}(x), \exists z \in O \cap R^{+}(y), \\
&\qquad z \notin R^{+}(I \cup U \cup \{z\}), R^{-}(z) - y \subseteq R^{+}(I)\}
\end{aligned}$$

$\mathrm{App}(I,U,R)$ is the set of undecided elements that cannot be in conflict with any extension $S$ below the current node (since $x \notin R^{\pm}(I) \cup R^{+}(U)$), and who are already defended by $I$, or who defend themselves ($R^{-}(x) \subseteq R^{+}(I \cup \{x\})$): these elements will be in every preferred extension below the current node. Similarily, the elements of $\mathrm{Undef}(I,U,R)$ will be in no extension $S$ below the current node, since they cannot be defended any more: not all their predecessors can be attacked ($R^{-}(x) \not\subseteq R^{+}(I \cup U) \supseteq R^{+}(S)$).

Elements of $\mathrm{App}(I,U,R)$ and of $\mathrm{Undef}(I,U,R)$ will be added to $I$ and $O$ respectively, and their fate will be sealed (at least below the current node). However, when no such rule can be applied, we may have to create two branches, in which case we will add an element $z$ to $I$ in one branch, to $O$ in the other, without any certainty that it truly belongs to where it is put; in fact, the presence of that element in one set or the other may only become fully justified when the fate of other elements is decided. In particular, if $z \in I$ only has one potential defender $x$ left, then $x$ must be added to $I$. $\mathrm{Heroes}(I,U,R)$ is the set of such last defenders of some elements of $I$. Conversely, if $z \in O$ has no attacker in $I$, then it must not be defensible by $I$; so if $z$ is already defended against all its attackers except one, whose attackers are not already in $O$, then these last potential defender of $z$ must not be in $I$. $\mathrm{Traitors}(I,O,U,R)$ is the set of those undecided elements who may have gone to $I$, but go to $O$ in order to support another element of $O$.

**Proposition 3.5.** *Let $R$ be a binary relation, let $C = (I,O,U)$ be a $R$-candidate, and let $x \in U$ such that $x \notin R^{+}(x)$. Then:*

1. *if* $x \in \mathrm{App}(I,U,R) \cup \mathrm{Heroes}(I,U,R)$, *then* $\mathrm{Pref}^*(C,R) = \mathrm{Pref}^*(C + x, R)$;
2. *if* $x \in \mathrm{Undef}(I,U,R) \cup \mathrm{Traitors}(I,O,U,R)$, *then* $\mathrm{Pref}^*(C,R) = \mathrm{Pref}^*(C - x, R)$.

*Proof.* Suppose first that $x \in \mathrm{App}(I,U,R) \cup \mathrm{Heroes}(I,U,R)$, let us prove that $\mathrm{Pref}^*(C,R) \subseteq \mathrm{Pref}^*(C + x, R)$. Let $(I',O',U') = C + x$, and let $S \in \mathrm{Pref}^*(C,R)$, then $R^{+}(S) \cap S = \emptyset$, $R^{-}(S) \subseteq R^{+}(S)$ and $\mathrm{Max}^*(I \cup O \cup U, R)$ holds: for every $X \subseteq (I' \cup O' \cup U') - S = (I \cup O \cup U) - S$, $X = \emptyset$ or $X \cap R^{+}(S \cup X) \neq \emptyset$ or $R^{-}(X) \not\subseteq R^{+}(S \cup X)$. If $x \in \mathrm{App}(I,U,R)$ then $x \notin R^{\pm}(I) \cup R^{+}(U)$ and $R^{-}(x) \subseteq R^{+}(I \cup \{x\})$. Let us show that $S \cap R^{\pm}(x) = \emptyset$: since $R^{-}(x) \subseteq O$, and since $S \subseteq I \cup U$, $x \notin R^{+}(S)$ because $(I \cup U) \cap O = \emptyset$. We now that $x \notin R^{-}(I \cup U)$, let us show that $x \notin R^{-}(U) \cap S$: suppose $y \in R^{+}(x) \cap U$, then, since $R^{-}(x) \cap S = \emptyset$, $R^{-}(y) \not\subseteq R^{+}(S)$, thus $y \notin S$. This proves that $x \notin R^{-}(S)$. Since $R^{\pm}(S) \subseteq R^{\pm}(I \cup U)$ (because $S \subseteq I \cup U$), and $x \notin R^{\pm}(I \cup U)$, $x \notin R^{\pm}(S)$, thus $S \cap R^{\pm}(x) = \emptyset$, therefore $S \subseteq (I \cup U) - R^{\pm}(x) \subseteq I \cup (U - R^{\pm}(x)) \subseteq I \cup \{x\} \cup (U - R^{\pm}(x)) = I' \cup U'$.

Moreover, consider $X = \{x\}$: since $x \notin R^{+}(I \cup U)$, $x \notin R^{+}(S \cup X)$, thus $X \cap R^{+}(S \cup X) = \emptyset$; and $R^{-}(X) = R^{-}(x) \subseteq R^{+}(I \cup \{x\}) = R^{+}(I \cup X) \subseteq R^{+}(S \cup X)$ because $I \subseteq S$, hence, since $S$ is a preferred extension, $X = \{x\} \not\subseteq (I \cup U \cup O) - S$, therefore $x \in S$, which proves that $I' \subseteq S$.

Let us now consider the case where $x \in \mathrm{Heroes}(R,I,U)$, that is, there exists $z \in I$ and $y \in R^{-}(z)$ such that $R^{-}(y) - O = \{x\}$: $S$ is preferred and contains $I$, thus $R^{-}(z) \subseteq R^{+}(S)$, thus $y \in R^{+}(S)$, and since $S \cap O = \emptyset$, it must be the case that $x \in S$. Since $I \subseteq S$, we conclude that $I' \subseteq S$. There remains to prove that $S \subseteq I' \cup U' = (I \cup U) - R^{\pm}(x)$. We know that $S \subseteq I \cup U$, so we have to prove that $S \cap R^{\pm}(x) = \emptyset$, which follows from the facts that $x \in S$ and $S \cap R^{+}(S) = \emptyset$.

Suppose now that $x \in \mathrm{Undef}(I,U,R) \cup \mathrm{Traitors}(I,O,U,R)$, and let us show that $\mathrm{Pref}^*(C,R) \subseteq \mathrm{Pref}^*(C - x, R)$. Let $(I'',O'',U'') = C - x$, and let $S \in \mathrm{Pref}^*(C,R)$, then $R^{+}(S) \cap S = \emptyset$, $R^{-}(S) \subseteq R^{+}(S)$ and for every $X \subseteq (I' \cup O' \cup U') - S = (I \cup O \cup U) - S$, $X \cap R^{+}(S \cup X) \neq \emptyset$. If $x \in \mathrm{Undef}(R,U,I)$, that is $R^-(x) \not\subseteq R^+(I \cup U)$ then, since $S \subseteq I \cup U$, $R^-(x) \not\subseteq R^+(S)$, thus $x \notin S$ because $R^-(S) \subseteq R^+(S)$, therefore $S \subseteq (U \cup I) - \{x\} = (U - \{x\}) \cup I = U' \cup I'$.

If $x \in \text{Traitors}(I, O, U, R)$: let $y \in R^+(x)$, $z \in R^+(y) \cap O$, such that $R^-(z) \subseteq O - \{z\}$ and $R^-(z) - y \subseteq R^+(I)$, and let $S \in \text{Pref}(R, I, O, U)$: then $\{z\} \subseteq A - S$, and $z \notin R^+(S \cup \{z\})$ because $R^-(z) \subseteq O - \{z\}$ and $O \cap S = \emptyset$, thus we must have that $R^-(z) \not\subseteq R^+(S \cup \{z\})$. But $R^-(z) - y \subseteq R^+(I) \subseteq R^+(S)$, thus it must be the case that $y \not\subseteq R^+(S)$, that is, $R^-(y) \cap S = \emptyset$. Thus we must have $x \notin S$. □

Prop. 3.5 gives conditions under which only one child of a given node can be generated: this interesting feature relies on the possibility to select an undecided element that has some particular property. If no such undecided element can be found, then we may have to generate two branches. The question that arises is if this can be further avoided, since one can expect the enumeration below a node that has two children to be on average twice as hard as below a node that only has one child. Recall that an argumentation framework that has no cycle only has one preferred extension. So it is natural to check if the set of undecided elements at a given node is cycle-free: we can hope that in this case there can only be one extension below that node. It is indeed the case:

**Proposition 3.6.** *Let $R$ be a finite binary relation, and let $C = (I, O, U)$ be a $R$-candidate such that* $\text{Undef}(I, U, R) = \emptyset$ *and $U$ contains no $R$-cycle. Then* $\text{Pref}^*(C, R) = \text{Pref}^*((I \cup U, O, \emptyset), R)$.

**Lemma 3.7.** *Let $R$ be a finite binary relation, let $(I, O, U)$ be a $R$-candidate such that $U$ contains no $R$-cycle and* $\text{Undef}(I, U, R) = \emptyset$, *then $R^+(U) \cap U = \emptyset$.*

*Proof.* Suppose that $x_0 \in U$ has a predecessor in $U$: there is $x_1 \in R^-(x_0) \cap U$, then, since $\text{Undef}(I, U, R) = \emptyset$, $x_1 \in R^+(I \cup U)$, and since $R^+(I) \cap U = \emptyset$, $x_1$ has a predecessor in $U$. We can iterate the argument, and construct a sequence $(x_i)_{i \geq 0}$ of elements of $U$ such that $x_{i+1} R x_i$ for every $i \geq 0$. Since $R$ is finite, $U$ must contain a cycle. Since $U$ contains no cycle, no element of $U$ has a predecessor in $U$, that is, $R^+(U) \cap U = \emptyset$. □

**Lemma 3.8.** *Let $R$ be a finite binary relation, let $(I, O, U)$ be a $R$-candidate such that $U$ contains no $R$-cycle and* $\text{Undef}(I, U, R) = \emptyset$, *and let* $S \in \text{Pref}^*((I, O, U), R)$, *then $U \subseteq S$.*

*Proof.* Let us prove that $S \cup U$ is admissible: since $S \subseteq I \cup U$, and since $O \cap (I \cup U) = \emptyset$, $S \cup U$ and $O$ are disjoint sets; moreover, we know that $R^+(S) \cap S = R^+(U) \cap U = \emptyset$, and that $R^\pm(I) \cap U = \emptyset$, thus $R^+(S \cup U) \cap (S \cup U) = \emptyset$. Let us now prove that $R^-(S \cup U) \subseteq R^+(S \cup U)$: we know that $R^-(S) \subseteq R^+(S)$, because $S$ is a preferred extension. Moreover, since $\text{Undef}(I, U, R) = \emptyset$, $R^-(U) \subseteq R^+(I \cup U) \subseteq R^+(S \cup U)$ because $I \subseteq S$. This proves that $S \cup U$ is admissible, so it is contained in a preferred extension $S'$ of $(I \cup O \cup U, R)$, which contains $S$ too. But since $S$ is itself preferred, we must have $S' = S$ (because of the maximality, with respect to set inclusion, of the preferred extensions), thus $U \subseteq S$. □

*Proof.* (of the proposition) Let us first prove that $\text{Pref}^*(C, R) = \text{Pref}^*((I \cup U, O, \emptyset), R)$: consider $S \in \text{Pref}^*(C, R)$, then $S \in \text{Pref}(A, R)$, where $A = I \cup U \cup O$, and $I \subseteq S \subseteq I \cup U$; but we also know, from the previous lemma, that $U \subseteq S$, therefore $I \cup U \subseteq S \subseteq (I \cup U) \cup \emptyset$: $S \in \text{Pref}^*((I \cup U, O, \emptyset), R)$.

For the converse, suppose now that $S \in \text{Pref}^*((I \cup U, O, \emptyset), R)$, then $S \in \text{Pref}(A, R)$ and $I \cup U \subseteq S \subseteq (I \cup U) \cup \emptyset$, which implies that $I \subseteq S \subseteq I \cup U$. Hence $S \in \text{Pref}^*((I, O, U), R)$. □

3.4. **Independance of the maximally connected components.** The following properties show that, to answer a query about an argument $a$, we can restrict the search space to the subgraph on the maximal connected component to which $a$ belongs. We prove that a preferred extension $I$ of an argumentation framework $(A, R)$ is the union of preferred extensions of the maximal connected components of $(A, R)$, and that each maximal connected component of $(A, R)$ possesses a preferred extension included in $I$. Thus, if $a$ belongs to at least one (resp. to every) preferred extension of the subgraph on its maximal connected component, then it belongs to at least one (resp. to every) preferred extension of the argumentation framework.

First, we give the definition of a maximal connected component and some well-known properties before proving the properties above.

**Definition 3.9.** Given an undirected graph $G = (V, E)$ with $V$ a set of vertices and $E$ a set of edges:

1. Two edges are *adjacent* if they have at least one vertex in common.
2. A *chain* between vertex $v$ and vertex $v'$ is a sequence of adjacent and distinct edges that are traversed in going from $v$ to $v'$.
3. An *isolated vertex* is a vertex such that there is no edge between it and any other vertex.
4. A *maximal connected component* is a set of vertices $V' \subseteq V$ such that $V' = \{v'\}$ with $v'$ an isolated vertex or such that:
   (a) every pair of $V'$ vertices is joined by at least one chain and
   (b) there is no edge between any vertex $v$' of $V$' and any vertex $v$ of $V - V'$.

The set of the maximal connected components of $G$ is denoted by $MC(V, E)$. It follows some properties:

**Proposition 3.10.** *Given a graph $G = (V, E)$,*

1. *The maximal connected components of $G$ are disjoints sets.*
2. *Each vertex of $V$ is contained in one and only one maximal connected component of $G$.*
3. *The union of the maximal connected components of $G$ is equal to $V$.*

*Proof.* (1) Assume that $\exists v$ such that $v \in M_1$ and $v \in M_2$. Then, by definition of a maximal connected component, $\exists v_1 \in M_1$ such that there is a chain between $v_1$ and $v$, just like $\exists v_2 \in M_2$ such that there is a chain between $v_2$ and $v$. So there is a chain between $v_1$ and $v_2$. But there is no edge (and then no chain) between a maximal connected component and any other argument out of the component. Contradiction with the assumption.

(2) Let $v$ a vertex of $V$.

If $v$ is isolated, then $M = \{v\}$ is a maximal connected component.

If $v$ is not isolated, let $M = \{v' \in V$ |there is a chain between $v$ and $v'\}$. Let $v_1, v_2 \in M$. There is a chain between $v_1$ and $v$, there is a chain between $v_2$ and $v$, so there is a chain between $v_1$ and $v_2$. Every pair of $M$'s vertices is joined by at least one chain. Let $v_1 \notin M$. If there were a chain between $v_1$ and some $v_2 \in M$, then there would be a chain between $v_1$ and $v$. This is impossible since $v_1 \notin M$. So $M$ is a maximal connected component.

Since maximal connected components are disjoint sets (Property 1), $v$ belongs to one and only one maximal connected component ($M$).

(3) Let $MC(V, E) = \{MCC_1, MCC_2, ..., MCC_n\}$. $\bigcup_{i=1}^{n} MCC_i \subseteq V$ since $\forall i,\ 1 \leq i \leq n,\ MCC_i \subseteq V$. Moreover, $V \subseteq \bigcup_{i=1}^{n} MCC_i$ since $\forall v \in V$, $\exists i,\ 1 \leq i \leq n$ such that $v \in MCC_i$ (Property 2). So $\bigcup_{i=1}^{n} MCC_i = V$. □

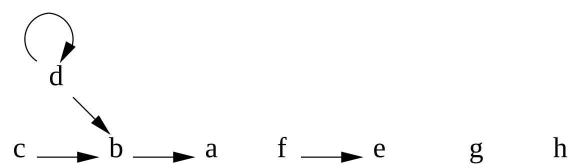
The maximal connected component to which a vertex $v$ of $G = (V, E)$ belongs is denoted by $MC_v(V, E)$.

Definitions 3.9 and properties 3.10 can be applied as well to directed graphs, considering the undirected graph corresponding.

**Definition 3.11.** Let $(A, R)$ a directed graph. The undirected graph corresponding to $(A, R)$ is $(A, E)$ with $E = \{\{x, y\} \mid (x, y) \in R\}$.

**Example 3.12.** Let $AF_2 = (A, R)$ be the argumentation framework with $A = \{a, b, c, d, e, f, g, h\}$ and $R$ as indicated on the graph below.

$MC(A, R) = \{\{a, b, c, d\}, \{e, f\}, \{g\}, \{h\}\}$, $MC_a(A, R) = \{a, b, c, d\}$.




Given an argumentation framework $AF = (A, R)$ and $A' \subseteq A$, Pref$(A', R)$ is the set of the preferred extensions of the subgraph on $A'$, that is Pref$(A', R) =$ Pref$(A', R')$ with $R' = \{(a_1, a_2) \in R \mid a_1 \in A',\ a_2 \in A'\}$.

**Example 3.12 (cont.)** Given $AF_2 = (A, R)$, $\mathrm{Pref}(A, R) = \{\{a, c, f, g, h\}\}$, $\mathrm{Pref}(MC_a(A, R), R) = \{\{a, c\}\}$.

**Definition 3.13.** Given $(A, R)$ an argumentation framework, we denote by $\mathrm{Adm}(A, R)$ the set of admissible sets of $(A, R)$, that is:
$\mathrm{Adm}(A, R) = \{S \subseteq A \mid R^+(S) \cap S = \emptyset \wedge R^-(S) \subseteq R^+(S)\}$

**Lemma 3.14.** *Given $(A, R)$ an argumentation framework and $MC(A, R) = \{MCC_1, MCC_2, ..., MCC_n\}$, if $\forall i$, $1 \leq i \leq n$, $M_i \in \mathrm{Adm}(MCC_i, R)$ then $\bigcup_{i=1}^n (M_i) \in \mathrm{Adm}(A, R)$.*

*Proof.* $\forall i, j$ such that $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$, there is no edge between $MCC_i$ and $MCC_j$ (Prop. 3.10), so there is no conflict between $MCC_i$ and $MCC_j$. Since $\forall i$, $1 \leq i \leq n$, $M_i \in \mathrm{Adm}(MCC_i, R)$, $\bigcup_{i=1}^n (M_i)$ is conflict-free.

Let $x \in \bigcup_{i=1}^n (M_i)$. Since $x$ belongs to one and only one maximal connected component (Prop. 3.10), there is one and only one $j$, $1 \leq j \leq n$ such that $x \in M_j$. $x$ does not have any attacker in $\bigcup_{i=1}^n (M_i) - M_j$ since we know that $\bigcup_{i=1}^n (M_i)$ is conflict-free. All the attackers of $x$ are in $MCC_j$, and since $M_j$ is an admissible set of $(MCC_j, R)$, $x$ is defended by $M_j$. So each element of $\bigcup_{i=1}^n (M_i)$ is defended by $\bigcup_{i=1}^n (M_i)$.

Since $\bigcup_{i=1}^n (M_i)$ is conflict-free and defends all its elements, $\bigcup_{i=1}^n (M_i) \in \mathrm{Adm}(\bigcup_{i=1}^n (MCC_i), R)$. Since $\bigcup_{i=1}^n (MCC_i) = A$ (Prop. 3.10), $\bigcup_{i=1}^n (M_i) \in \mathrm{Adm}(A, R)$. □

**Lemma 3.15.** *Given $(A, R)$ an argumentation framework and $MCC \in MC(A, R)$, if $I \in \mathrm{Adm}(A, R)$ then $I \cap MCC \in \mathrm{Adm}(MCC, R)$.*

*Proof.* Since $I$ is conflict-free, $I \cap MCC$ is conflict-free.

Let $x \in I \cap MCC$. $x$ is defended by $I$ since $I$ is an admissible set. All the defenders of $x$ are contained in $MCC$ since $x$ belongs to one and only one maximal connected component and no defender of $x$ can be in another maximal connected component (since then it would mean that there is an edge between two maximal connected components; impossible according to Prop. 3.10). So each element of $I \cap MCC$ is defended by $I \cap MCC$.

Since $I \cap MCC$ is conflict-free and defends all its elements, $I \cap MCC \in \mathrm{Adm}(MCC, R)$. □

**Proposition 3.16.** *Given $(A, R)$ an argumentation framework and $MC(A, R) = \{MCC_1, MCC_2, ..., MCC_n\}$, then if $\forall i$, $1 \leq i \leq n$, $M_i \in \mathrm{Pref}(MCC_i, R)$ then $\bigcup_{i=1}^n (M_i) \in \mathrm{Pref}(A, R)$.*

*Proof.* According to Lemma 3.14 and since every preferred extension is an admissible set, $\bigcup_{i=1}^n (M_i) \in \mathrm{Adm}(A, R)$. Let $Y \subseteq A - \bigcup_{i=1}^n M_i$ such that $Y \neq \emptyset$. Is $Y \cup (\bigcup_{i=1}^n M_i)$ an admissible set for $(A, R)$?

$Y \subseteq \bigcup_{i=1}^n (MCC_i - M_i)$ since $A - \bigcup_{i=1}^n (M_i) = \bigcup_{i=1}^n (MCC_i) - \bigcup_{i=1}^n (M_i) = \bigcup_{i=1}^n (MCC_i - M_i)$ according to Prop. 3.10 and since $M_i \subseteq MCC_i$ and $\forall j$, $1 \leq j \leq n$, $j \neq i$, $M_i \cap MCC_j = \emptyset$.

Let $X_i = (Y \cap MCC_i)$. $\bigcup_{i=1}^n X_i = Y$, so $Y \cup (\bigcup_{i=1}^n M_i) = \bigcup_{i=1}^n (X_i \cup M_i)$.

If $\bigcup_{i=1}^n (X_i \cup M_i)$ were admissible for $(A, R)$, then, according to Lemma 3.15, $\forall j$, $1 \leq j \leq n$, $\bigcup_{i=1}^n (X_i \cup M_i) \cap MCC_j$ would be admissible. But $\bigcup_{i=1}^n (X_i \cup M_i) \cap MCC_j = X_j \cup M_j$ and, since $Y \neq \emptyset$, there is $j$, $1 \leq j \leq n$, such that $X_j \neq \emptyset$, and then $X_j \cup M_j$ is not admissible since $M_j$ is a preferred extension of $(A, R)$.

So $Y \cup (\bigcup_{i=1}^n M_i)$ is not admissible for $(A, R)$. Then $\bigcup_{i=1}^n (M_i)$ is a preferred extension of $(A, R)$. □

**Proposition 3.17.** *Given $(A, R)$ an argumentation framework, $MC(A, R) = \{MCC_1, MCC_2, ..., MCC_n\}$ and $I \in \mathrm{Pref}(A, R)$, then $\forall i$, $1 \leq i \leq n$, $I \cap MCC_i \in \mathrm{Pref}(MCC_i, R)$.*

*Proof.* Let $I_i = I \cap MCC_i$ with $1 \leq i \leq n$. According to Lemma 3.15, and since every preferred extension is an admissible set, $I_i \in \mathrm{Adm}(MCC_i, R)$. Let $Y \subseteq MCC_i - I_i$ such that $Y \neq \emptyset$. Is $I_i \cup Y$ admissible for $(MCC_i, R)$?

Let $I' = I \cup Y$. $I \subseteq I'$, but does $I' \subseteq I$ hold ? We know that $Y \not\subseteq I_i$. If $Y \subseteq I$, since $Y \subseteq MCC_i$, then $Y \subseteq (I \cap MCC_i) = I_i$. This is impossible since $Y \subseteq MCC_i - I_i$. So $I \cup Y \not\subseteq I$, and so $I' \neq I$: $I'$ is not a preferred extension of $(A, R)$.

According to Lemma 3.14, this result means that $\exists j$, $1 \leq j \leq n$, such that $I' \cap MCC_j \notin \mathrm{Adm}(MCC_j, R)$.

If $j \neq i$, then $Y \cap MCC_j = \emptyset$ and so $I' \cap MCC_j = I \cap MCC_j$. But, according to Lemma 3.15, since $I$ is an admissible set of $(A, R)$, $I \cap MCC_j$ is an admissible set of $(MCC_j, R)$. So $j = i$.

Since $I' \cap MCC_i = (I \cup Y) \cap MCC_i = (I \cap MCC_i) \cup (Y \cap MCC_i) = I_i \cup Y$, $I_i \cup Y$ is not an admissible set of $(MCC_i, R)$.

Then $I_i$ is a maximal admissible set of $(MCC_i, R)$ and $\forall k,\ 1 \leq k \leq n$, $I \cap MCC_k \in \text{Pref}(MCC_k, R)$. □

**Proposition 3.18.** *Given an argumentation framework $(A, R)$ and an argument $a \in A$, if $I \in \text{Pref}(A, R)$ and $a \in I$, then $\exists M \in \text{Pref}(MC_a(A, R), R)$ such that $a \in M$.*

*Proof.* According to Prop. 3.17, $\exists M \in \text{Pref}(MC_a(A, R), R)$ such that $I \supseteq M$. Since $a \in I$ and $MC_a(A, R)$ is the only maximal connected component $a$ belongs to (Prop. 3.10), then $a \in M$. □

In particular, this proposition means that, given $I$ with $a \in I$ and $I \in \text{Pref}(A, R)$, there is an admissible set $I'$ such that $\{a\} \subseteq I' \subseteq I$ and $I' \subseteq MC_a(A, R)$.

## 4. The Algorithms

The general extension enumeration algorithm is quite straightforward. We describe it by means of a recursive function PrefEnum, which, given a binary relation $R$ and a $R$-candidate $C = (I, O, U)$, returns true if $\text{Pref}^*(C, R) \neq \emptyset$, and returns false otherwise. At each step, the function tests if the answer can be found at the current node: StopCondTrue$(C, R)$ tests if the answer is true, StopCondFalse$(C, R)$ tests if it is false. If there is no easy answer, then the function generally calls itself on $(C + x, R)$ and on $(C - x, R)$, for a selected element $x$ of $U$. This selection of the branching $x$ is crucial, since it may be the case that for a good choice of $x$, only one branch needs to be explored (Prop. 3.5): for this reason, the selection function Select$(C, R)$ returns a triple $(x, b_I, b_O)$, where $b_I$ and $b_O$ are two booleans, indicating respectively if the branch in which $x$ is added to $I$, and the branch in which $x$ is added to $O$, have to be explored. When $U$ is empty or contains no more cycle (Prop. 3.6), the selection function returns $b_I$ and $b_O$ false and it only remains to test if $I \cup U$ is a preferred extension of $(I \cup U \cup O, R)$; this is the role of the condition FinalTest$(I \cup U, O, R)$.

**Function:** PrefEnum$(R, C)$
***Param.***: *a binary relation $R$, a $R$-candidate $C = (I, O, U)$*
***Result***: $\top$ *if an extension is found,* $\bot$ *otherwise*

1. if StopCondTrue$(C, R)$ then $\top$;
2. if StopCondFalse$(C, R)$ then $\bot$;
3. $(x, b_I, b_O) \leftarrow$ Select$(C, R)$;
4. if $\neg b_I \wedge \neg b_0$ then FinalTest$(I \cup U, O, R)$
5. else $(b_I \wedge \text{PrefEnum}(C + x, R)) \vee (b_O \wedge \text{PrefEnum}(C - x, R))$;

This skeleton of algorithm is used to answer the credulous and the sceptical query-answering problems (that is, checking if an argument is a credulous or a sceptical consequence of an argumentation framework), and also to compute preferred extensions.

Let us describe precisely for each kind of problem the call of PrefEnum, the conditions StopCondTrue, StopCondFalse and FinalTest and the function Select.

For this last function, we use a function Choose which, applied to a non-empty set, returns an unspecified element of the set; this set can contain arguments or sets of arguments as well. We also use a function Cycles which, given a set $X$ of arguments, returns the set of the sets of arguments which compose cycles in $X$.

**Example 2.2 (cont)** Given the argumentation framework $AF_1 = (A, R)$, Cycles$(A) = \{\{a, b\}, \{c, d, e\}, \{k\}\}$.

Given a set $X$ of arguments, we call:

Choose$(X)$ to choose an argument in $X$;

Choose(Cycles$(X)$) to choose a set of arguments which compose a cycle in $X$ (assuming that $X$ contains cycles);

Choose(Choose(Cycles$(X)$)) to choose an argument contained in a cycle of $X$ (assuming that $X$ contains cycles).

4.1. **Credulous Query Answering.** Given an argumentation framework $(A, R)$ and an argument $a$ of $A$, we want to check if $a$ is at least in one preferred extension of $(A, R)$. According to Prop. 4, it is sufficient to find an admissible set containing $a$ to be sure that $a$ is contained in a preferred extension.

Notice that a self-attacking argument does not belong to any preferred extension: we directly answer false to a credulous query on a such an argument.

For all the other credulous queries, we call the function PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A) + a$, and we try to "expand" $I$ for $I$ to be an admissible set. PrefEnum returns $\top$ if an admissible set is found, $\perp$ otherwise.

4.1.1. *Stop conditions.* We denote by $O_p$ the set $R^-(I) - R^+(I)$. According to Prop. 3.3, $I$ is an admissible set when $O_p = \emptyset$. On the contrary, the current set $I$ cannot lead to an admissible set when an argument in $I$ cannot be defended (Prop. 3.3). Then, given $C = (I, O, U)$ a $R$-candidate, the stop conditions are:
StopCondTrue$(C, R) \equiv O_p = \emptyset$
StopCondFalse$(C, R) \equiv \exists x \in O_p$ such that $R^-(x) \subseteq O$

4.1.2. Select *function.* For the StopCondTrue to be true, the set $O_p$ must be empty. $O_p$ contains predecessors of $I$ that are not yet successors of $I$, so to empty $O_p$, we try to make its undecided predecessors go in $I$.

Heroes$(I, U, R)$ contains undecided predecessors of $O_p$ that must go in $I$ (if they are not self-attacking): each of them is the only one able to attack a given argument of $O_p$. App$(I, U, R)$ contains undecided arguments which are conflict-free with $I \cup U$ and defended by $I$ when added to it. Among these arguments, we only look at the ones which are in $R^-(O_p)$, since the other ones would make $I$ (and $O_p$) grow without helping us to empty $O_p$. We choose to look at the arguments of Heroes$(I, U, R)$ before looking at the ones of App$(I, U, R)$.

Among the other undecided predecessors of $O_p$, we can find self-attacking arguments. These arguments must not be added to $I$ because then $I$ would not be conflict-free, so we have to put them in $O$. This treatment on Refl$(U \cap R^-(O_p), R)$ must be done before looking at the arguments of Heroes$(I, U, R)$. Actually, arguments of Heroes$(I, U, R)$ must be the last undecided predecessors of arguments of $O_p$ and then self-attacking arguments which would remain undecided could prevent some arguments from being in Heroes$(I, U, R)$. Self-attacking arguments do not prevent arguments from being in App$(I, U, R)$, but putting them in $O$ diminishes the number of undecided arguments and then App$(I, U, R)$ is easier to compute.

If none of the pruning properties is applicable, then we select an undecided predecessor of $O_p$. Actually, we know that $U \cap R^-(O_p)$ is not empty; if it were empty, then it would mean that $O_p = \emptyset$, and in this case StopCondTrue$((I, O, U), R)$ would be true, or that $R^-(O_p) \subseteq O$, and in this case StopCondFalse$((I, O, U), R)$ would be true.

The Select function is the following:

**Function:** Select$(C, R)$
***Param.:*** *a binary relation $R$, a $R$-candidate $C = (I, O, U)$*
***Result:*** *an argument and two booleans*

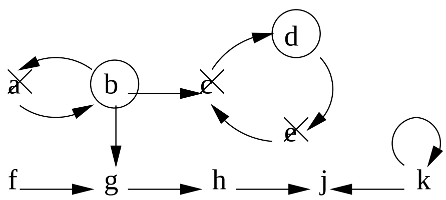
1. if Refl$(U \cap R^-(O_p)) \neq \emptyset$ then (Choose(Refl$(U \cap R^-(O_p))$), $\perp, \top$)
2. else if Heroes$(I, U, R) \neq \emptyset$ then (Choose(Heroes$(I, U, R)$), $\top, \perp$)
3. else if App$(I, U, R) \cap R^-(O_p) \neq \emptyset$ then (Choose(App$(I, U, R) \cap R^-(O_p)$), $\top, \perp$)
4. else (Choose$(U \cap R^-(O_p))$, $\top, \top$);

Notice that if we find an admissible set containing $a$, then this set is contained in $MC_a(A, R)$. Actually, when we call PrefEnum, $I$ contains $a$, so $I \subseteq MC_a(A, R)$. Assume that the algorithm is running and that the current set $I$ is contained in $MC_a(A, R)$. If $(x, \top, b_O) =$ Select$((I, O, U), R)$, then $x \in R^-(R^-(I) - R^+(I))$: $x$ is connected to $I$. Therefore, $(I \cup \{x\}) \subseteq MC_a(A, R)$. So, at the end of the run, if $I$ is an admissible set, then it is contained in $MC_a(A, R)$.

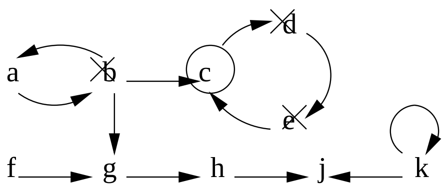
4.1.3. *Final Test.* The condition FinalTest is checked when the Select function returns the two booleans false; it is the case when the status of every argument is decided or when $U$ contains no more cycle. Here the Select function always returns an argument of $R^-(O_p)$ and at least one of the two booleans at $\top$. We do not have to decide the status of every argument to answer the credulous query. Therefore the condition $\text{FinalTest}(I,O,R)$ is not used in the credulous query-answering algorithm.

4.1.4. *Examples.* Let us run the algorithm on examples. We describe the call of PrefEnum and we show on the graph representation of the argumentation framework which arguments are in $I$ (circled), in $O$ (crossed) or in $U$ at the end of the run.

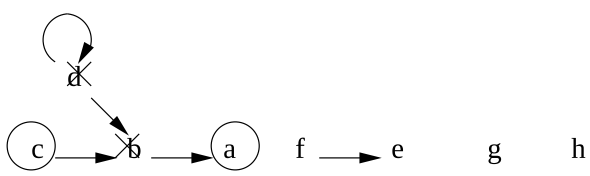
**Example 2.2 (cont.)** Given the argumentation framework $AF_1 = (A,R)$, is there a preferred extension containing $d$ ? We call PrefEnum on $R$ and on $(\emptyset,\emptyset,A)+d$. Since $d$ is in $I$, $c$ and $e$ go in $O$, because they are respectively predecessor and successor of $d$. Argument $c$ belongs to $O_p$ and it has only one undecided predecessor: $b$. Argument $b$ is in fact the only potential defensor of $d$: it belongs to $\text{Heroes}(I,U,R)$. So $b$ goes in $I$ and consequently its successor-predecessor $a$ goes in $O$. Now there is no more argument in $O$ without predecessor in $I$. $\text{StopCondTrue}((I,O,U),R)$ is true, we have found an admissible set : $\{d,b\}$, so $d$ belongs to a preferred extension.




**Example 2.2 (cont.)** Given $AF_1 = (A,R)$, is there a preferred extension containing $c$ ? We call PrefEnum on $R$ and on $(\emptyset,\emptyset,A)+c$. Since $c$ is in $I$, arguments $b$, $d$ and $e$ go in $O$. Argument $e$ is in $O_p$ because it is predecessor but not successor of $I$. In fact, its only predecessor is in $O$: we are in the case where $I$ can not defend all its elements, $\text{StopCondFalse}((I,O,U),R)$ is true, so there is no preferred extension containing $c$.

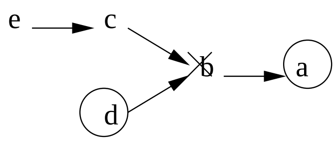



**Example 3.12 (cont.)** Given $AF_2 = (A,R)$, is there a preferred extension containing $a$ ? We call PrefEnum on $R$ and on $(\emptyset,\emptyset,A)+a$. Since $a$ is in $I$, $b$ is in $O_p$; $R^-(O_p) = \{c,d\}$. Argument $d$ is self-attacking, it belongs to $\text{Refl}(U \cap R^-(O_p))$, so it goes in $O$. Argument $c$ is now the only undecided argument which can defend $a$ against $b$: $c$ belongs to $\text{Heroes}(I,U,R)$ and goes in $I$. Then $O_p$ is empty, $\text{StopCondTrue}((I,O,U),R)$ is true, so $a$ is contained in a preferred extension.

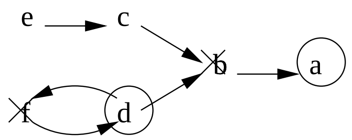



The three coming examples illustrate the case where an argument has two potential defenders and where one of its defenders belongs to $\text{App}(I,U,R) \cap R^-(O_p)$. In each example, we want to know if the argument $a$ belongs to a preferred extension of the given argumentation framework $(A,R)$. In each case, we call $\text{PrefEnum}(R,(\emptyset,\emptyset,A)+a)$.

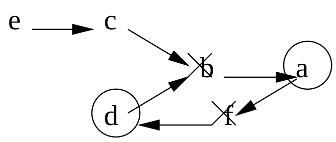
**Example 4.1.** Let $AF_3 = (A,R)$ with $A = \{a,b,c,d,e\}$ and $R = \{(e,c),(c,b),(d,b),(b,a)\}$. Since $a$ is in $I$, $b$ is in $O_p$; $R^-(O_p) = \{c,d\}$. $d$ has no predecessor and its only successor is $b$: $d$ belongs to$\text{App}(I,U,R) \cap R^-(O_p)$, so it goes in $I$. Then $O_p$ is empty, $\text{StopCondTrue}((I,O,U),R)$ is true, $a$ is contained in a preferred extension of $AF_3$.




**Example 4.2.** Let $AF_4 = (A, R)$ with $A = \{a, b, c, d, e, f\}$ and $R = \{(e, c), (c, b), (d, b), (b, a), (d, f), (f, d)\}$. Since $a$ is in $I$, $b$ is in $O_p$; $R^-(O_p) = \{c, d\}$. $d$ defends itself against its only predecessor, $f$, and its only successor is $b$: $d$ belongs to$\mathrm{App}(I, U, R) \cap R^-(O_p)$ so it goes in $I$, and $f$ goes in $O$. Then $O_p$ is empty, $\mathrm{StopCondTrue}((I, O, U), R)$ is true, $a$ is contained in a preferred extension of $AF_4$.




**Example 4.3.** Let $AF_5 = (A, R)$ with $A = \{a, b, c, d, e, f\}$ and $R = \{(e, c), (c, b), (d, b), (b, a), (a, f), (f, d)\}$. Since $a$ is in $I$, $b$ and $f$ are in $O$. $b$ is in $O_p$, $R^-(O_p) = \{c, d\}$. $d$ is defended by $I$ against its only predecessor $f$, and its only successor is $b$: $d$ belongs to $\mathrm{App}(I, U, R) \cap R^-(O_p)$ so it goes in $I$ and $f$ goes in $O$. Then $O_p$ is empty, $\mathrm{StopCondTrue}((I, O, U), R)$ is true, $a$ is contained in a preferred extension of $AF_5$.




4.2. **Sceptical Query Answering.** Given an argumentation framework $(A, R)$ and an argument $a$ of $A$, we want to check if $a$ is in every preferred extension of $(A, R)$. To this end, we look at the complementary of the sceptical query answering problem: we look for a preferred extension of $(A, R)$ not containing $a$.

Notice that a self-attacking argument does not belong to any preferred extension, so we directly answer false to a sceptical query on such an argument.

For all the other sceptical queries, we call the function PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A) - a$, and we attempt to build a preferred extension. If $\mathrm{PrefEnum}(R, (\emptyset, \emptyset, A)) - a)$ returns $\perp$, it means that $a$ is in every preferred extension, otherwise it is not.

In the sequel, we keep the notation $O_p$ to denote the set $R^-(I) - R^+(I)$ and we denote by $O_u$ the set $O - R^\pm(I)$.

4.2.1. *Stop conditions.* We are sure that we can build a preferred extension not containing $a$ when we have found an admissible set from which $a$ is rejected with a good reason. According to Prop. 3.4, this is the case:

1. when $a$ is attacked by the admissible set, or
2. when $a$ cannot be defended by any preferred extension containing $I$: this is the case when $a \in O_u$ and no part of $O_u$ can be defended by $I$ (the condition $\mathrm{Max}(I, O_u, U, R)$ is verified).

We are sure that we cannot build a preferred extension:

1. when the current set $I$ cannot defend all its arguments (Prop. 3.3), or
2. when some arguments have been rejected from $I$ whereas they should have been in $I$ in order for $I$ to be a maximal admissible set (Prop. 3.2[1]).

Given $C = (I, O, U)$ a $R$-candidate:
$\mathrm{StopCondTrue}(C, R) \equiv O_p = \emptyset \wedge (a \in R^+(I) \vee (a \in O_u \wedge \mathrm{Max}(I, O_u, U, R)))$

[1]Actually, even if $U \neq \emptyset$, if $\mathrm{Max}(I, O_u, \emptyset, R)$ is false (that is there is a conflict-free set $X \subseteq O_u$ defended by $I \cup X$) then $I$ will never be a maximal admissible set.

$\text{StopCondFalse}(C,R) \equiv (\exists x \in O_p,\ R^-(x) \subseteq O) \vee \neg\text{Max}(I, O_u, \emptyset, R)$

4.2.2. Select *function.* At the beginning of the run, $a$ is in $O_u$. For the first condition of StopCondTrue to be true ($a$ is attacked by an admissible set), we have to try to find a predecessor of $a$ in $\text{App}(I,U,R)$ in order for $a$ to be in $R^+(I)$ (Prop. 3.5); this predecessor cannot be in $\text{Heroes}(I,U,R)$ since this set contains undecided arguments which are predecessors of $O_p$. If we find $b \in \text{App}(I,U,R) \cap R^-(a)$, then we have to check if there is an admissible set containing $I$. To answer this credulous query, we use the function $\text{Select}((I,O,U),R)$ described to answer credulous queries (we call it SelectCred in the sequel).

If $R^-(a) \subseteq O$, then the first condition of StopCondTrue will never be true. Thus we try to make true its second condition ($a$ is rejected from any preferred extension containing $I$ because it is not defensible). To this end, we look at the arguments of $R^-(O_u)$ and we try to make them go in $O_u$ in order for $O_u$ to be a set of non defensible arguments. To make the arguments of $R^-(O_u)$ go in $O_u$, we check if they are contained in $\text{Refl}(U,R)$, $\text{Traitors}(I,U,R)$ or $\text{Undef}(I,U,R)$ (Prop. 3.5). Since it is easier to check if an argument is self-attacking than to check if it belongs to $\text{Traitors}(I,U,R)$ or $\text{Undef}(I,U,R)$, we will try to find at first a predecessor of $O_u$ in $\text{Refl}(U,R)$. Note that $\text{Traitors}(I,U,R)$ contains undecided predecessors of $O_u$, whereas $\text{Undef}(I,U,R)$ might contain other undecided arguments.

If none of these pruning properties is applicable, then we try to make the predecessors of $O_u$ go in $I$ thanks to $\text{App}(I,U,R)$.

According to Prop. 3.17, we only have to look for a preferred extension not containing $a$ on the subgraph of the maximal connected component of $a$. Then, according to Prop. 3.6, if $U \cap MC_a(A,R)$ is cycle-free, it only remains to test if $I \cup U$ is a preferred extension.

Otherwise, if $U$ is not cycle-free and $R^-(O_u) \cap U \neq \emptyset$ but if no undecided argument of $O_u$ can be put neither in $O$ nor in $I$, then we choose at random a predecessor of $O_u$, possibly cutting a cycle (Prop. 3.6). If $R^-(O_u) \cap U = \emptyset$, we choose an undecided argument that cuts a cycle in the maximal connected component of $a$.

Since the strategy of selection of an argument depends on $a$, we give $a$ as an additional parameter of the Select function. This function is the following:

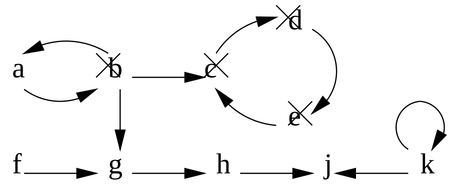
**Function:** $\text{Select}_a(C,R)$
***Param.***: *a binary relation $R$, a $R$-candidate $C = (I,O,U)$*
***Result***: *an argument and two booleans*

1. if $a \in R^+(I)$ then $\text{SelectCred}(C,R)$
2. else if $\text{App}(I,U,R) \cap R^-(a) \neq \emptyset$ then $(\text{Choose}(\text{App}(I,U,R) \cap R^-(a)), \top, \bot)$
3. else if $\text{Refl}(U \cap R^-(O_u), R) \neq \emptyset$ then $(\text{Choose}(\text{Refl}(U \cap R^-(O_u), R)), \bot, \top)$
4. else if $\text{Traitors}(I,U,R) \neq \emptyset$ then $(\text{Choose}(\text{Traitors}(I,U,R)), \bot, \top)$
5. else if $\text{Undef}(I,U,R) \cap R^-(O_u) \neq \emptyset$ then $(\text{Choose}(\text{Undef}(I,U,R) \cap R^-(O_u)), \bot, \top)$
6. else if $\text{App}(I,U,R) \cap R^-(O_u - \{a\}) \neq \emptyset$ then $(\text{Choose}(\text{App}(I,U,R) \cap R^-(O_u - \{a\})), \top, \bot)$
7. else if $\text{Cycles}(U \cap MC_a(A,R)) = \emptyset$ then $(\text{Choose}(U \cap MC_a(A,R)), \bot, \bot)$
8. else if $\exists e \in \text{Cycles}(U) \mid e \cap R^-(O_u) \neq \emptyset$ then $(\text{Choose}(e \cap R^-(O_u)), \top, \top)$
9. else if $U \cap R^-(O_u) \neq \emptyset$ then $(\text{Choose}(U \cap R^-(O_u)), \top, \top)$
10. else $(\text{Choose}(\text{Choose}(\text{Cycles}(U \cap MC_a(A,R)))), \top, \top)$;

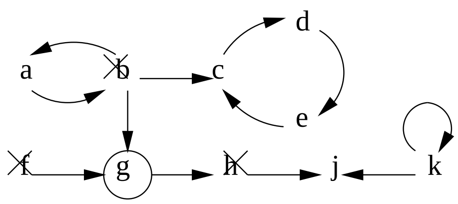
4.2.3. *Final test.* We verify the condition $\text{FinalTest}(I \cup U, O, R)$ when $U$ contains no more cycle. We must ensure that $I \cup U$ is a maximal admissible set, that is, according to Prop. 3.2:
$\text{FinalTest}(I \cup U, O, R) \equiv \text{Max}(I \cup U, O_u, \emptyset, R)$

4.2.4. *Examples.*

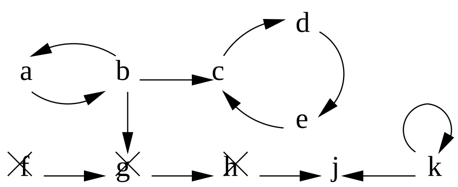
**Example 2.2 (cont.)** Given the argumentation framework $AF_1$, is argument $d$ contained in every preferred extension ? $MC_d(A,R) = A$, so we call PrefEnum on $R$ and on $(\emptyset, \emptyset, A) - d$. Argument $d$ is in $O_u$. Its only predecessor, $c$, can not be defended against its attacker $e$, because its only defensor was $d$: $c$ belongs to $\text{Undef}(I,U,R) \cap R^-(O_u)$, so it goes in $O$, precisely in $O_u$. Argument $e$ is a predecessor of $c$, and its only defensor against $d$ was $c$: $e$ belongs to $\text{Undef}(I,U,R) \cap R^-(O_u)$ and goes in $O_u$. Now argument $d$ has still a potential defensor against $c$, the undecided argument $b$. This argument must go in $O$, as we want to build an extension not containing $d$: $b$ belongs to $\text{Traitors}(I,U,R)$. We could think that $a$ also belongs to Traitors$(I, U, R)$, because it is a potential defensor of $c$ against $b$: it is not the case. $c$ can not be defended against $e$, so even if it could be defended against $b$, there would be no ambiguity on $c$'s membership of $I$ or $O$: $c$ is rejected from every extension. Now $O_p$ is empty and no argument of the set $O_u$ is defendable. StopCondTrue$((I, O, U), R)$ is true: there exists at least one preferred extension not containing $d$. The graph representation at the end of the run is the following:




**Example 2.2 (cont.)** Given $AF_1$, is argument $h$ contained in every preferred extension ? $MC_h(A, R) = A$, so we call PrefEnum on $R$ and on $(\emptyset, \emptyset, A) - h$. Argument $h$ has only one predecessor, $g$. We can not apply any pruning property to make $g$ go in $I$ or $O$. So we build two branches, one where $g$ goes in $I$ (branch 1), the other one where it goes in $O$ (branch 2). In branch 1, $f$ and $b$ go in $O$ because they are predecessor of $g$. But now $f$ belongs to $O_p$, and it does not have any predecessor: $g$ will never be defended against $f$. It means there is no extension in this branch. The graph representation at the end of the run on branch 1 is the following:

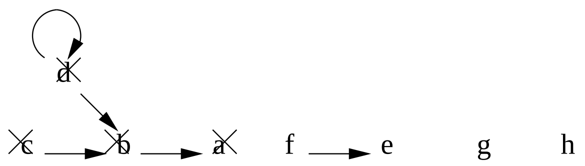



In branch 2, $f$ and $b$ belong to Traitors$(I, U, R)$ because they are potential defensors of $h$ against $g$. If we select $f$, then $f$ goes in $O$. But $\{f\} \subseteq O_u$ is conflict-free and $R^-(\{f\}) \subseteq R^+(I)$ since $f$ has no predecessor: StopCondFalse$((I, O, U), R)$ is true. This branch can not lead to a preferred extension because argument $f$ should be in $I$ for $I$ to be a maximal admissible set at the end of the run. The graph representation at the end of the run on branch 2 is the following:




Consequently, since PrefEnum returns false in each branch, $h$ is in every preferred extension of $AF_1$.

**Example 3.12 (cont.)** Given $AF_2 = (A, R)$, is argument $a$ contained in every preferred extension ? $MC_a(A, R) = \{a, b, c, d\}$, so we call PrefEnum on $R$ and on $(\emptyset, \emptyset, \{a, b, c, d\}) - a$. Argument $b$ must go in $O$ since it is the only predecessor of $a \in O_u$: $b$ belongs to Traitors$(I, U, R)$. Argument $d$ is self-attacking, so it goes in $O$. $c$ remains the only undecided predecessor of $b$, so to justify the presence of $b$ in $O$, $c$ goes in $O$ ($c$ belongs to Traitors$(I, U, R)$). But argument $c$ is defended by $I$ (it has no predecessor) and it is not self-attacking: $c$ should be in $I$ for $I$ to be a maximal admissible set (we have $\neg$Max$(I, O_u, \emptyset, R)$). StopCondFalse$((I, O, U), R)$ is true, we cannot build a preferred extension not containing $a$, so $a$ belongs to every preferred extension of $MC_a(A, R)$, and then it belongs to every preferred extension of $(A, R)$.




4.3. **Computation of the extensions.** To compute the preferred extensions, we call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$.

4.3.1. *Stop conditions.* StopCondTrue$(C, R)$ is always false, because we can say we have found a preferred extension only when the status of every argument is decided; this test will be done by FinalTest$(I, O, R)$ at the end of the run. We know that the current set $I$ cannot lead to a preferred extension when an argument in $I$ is not defensible (Prop. 3.3). Then, given $C = (I, O, U)$ a $R$-candidate:
StopCondTrue$(C, R) \equiv \perp$;
StopCondFalse$(C, R) \equiv \exists x \in O_p$ such that $R^-(x) \subseteq O$.

4.3.2. Select *function.* There is no particular strategy to compute all the extensions. We will just apply at first the pruning conditions that make arguments go in $I$ or $O$ on a justified way (looking at the arguments of App$(I, U, R)$ and Undef$(I, U, R)$). According to Prop. 3.6, if these two conditions are not applicable, then we check if $U$ is cycle-free (and then PrefEnum will call FinalTest$(I, O, R)$). If $U$ is not cycle-free, we look at the arguments of Heroes$(I, U, R)$ and Traitors$(I, U, R)$ to justify the presence of arguments put in $I$ or $O$ at a branching point. Finally, if none of these pruning conditions are applicable, we choose an undecided argument that cuts a cycle.

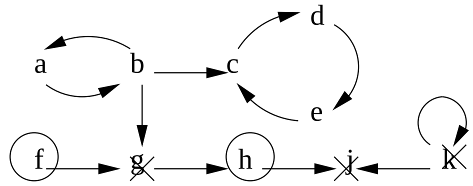
**Function:** Select$(C, R)$
***Param.***: *a binary relation $R$, a $R$-candidate $C = (I, O, U)$*
***Result***: *an argument and two booleans*

1. if Refl$(U, R) \neq \emptyset$ then (Choose(Refl$(U, R)$), $\perp, \top$)
2. else if App$(I, U, R) \neq \emptyset$ then (Choose(App$(I, U, R)$), $\top, \perp$)
3. else if Undef$(I, U, R) \neq \emptyset$ then (Choose(Undef$(I, U, R)$), $\perp, \top$)
4. else if Cycles$(U) = \emptyset$ then (Choose$(U)$, $\perp, \perp$)
5. else if Heroes$(I, U, R) \neq \emptyset$ then (Choose(Heroes$(I, U, R)$), $\top, \perp$)
6. else if Traitors$(I, U, R) \neq \emptyset$ then (Choose(Traitors$(I, U, R)$), $\perp, \top$)
7. else (Choose(Choose(Cycles$(U)$)), $\top, \top$);

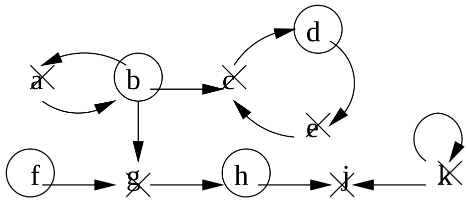
4.3.3. *Final test.* Since we want to enumerate all the extensions, FinalTest must always be $\perp$, otherwise the function PrefEnum could stop before having finished to enumerate of all the extensions. FinalTest must be written as a condition that, as a side effect, tests if $I$ is a preferred extension, and if true, outputs $I$. To this end, we introduce the function Print. According to Prop. 3.2:
FinalTest$(I \cup U, O, R) \equiv$ (**if** Max$(I \cup U, O_u, \emptyset, R)$ **then** Print$(I)$) $\wedge \perp$:

4.3.4. *Examples.*

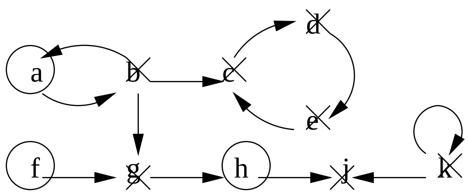
**Example 2.2 (cont.)** Let us compute all the preferred extensions of $AF_1$. We call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$. Argument $k$ is self-attacking, so it goes in $O$. Argument $f$ is not attacked: it belongs to App$(I, U, R)$, so it goes in $I$. As it is in $I$, its successor $g$ goes in $O$. Argument $h$ is attacked by $g$ but it is defended by $f$; moreover, it is conflict-free with $I$, so it belong to App$(I, U, R)$ and then goes in $I$. Consequently, $h$'s successor (argument $j$) goes in $O$. No more pruning property is applicable and $U$ is not cycle-free. The graph representation of $AF_1$ is the following:



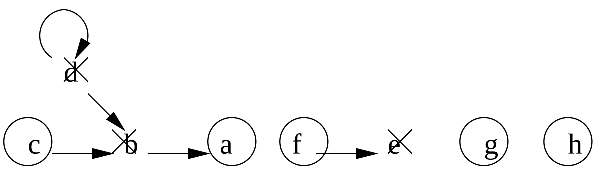
Then the Select function returns an argument that cuts a cycle, for example argument $d$. We build two branches, one where $d$ goes in $I$, the other one where $d$ goes in $O$. When $d$ goes in $I$, we have the same reasoning on arguments $a$, $b$, $c$ and $e$ as in example 4.1.4. The status of every argument being decided, it only remains to verify FinalTest$(I \cup U, O, R)$: it is true, so $\{b, d, f, h\}$ is a preferred extension. The graph representation of $AF_1$ at the end of the run on the first branch is:




When $d$ goes in $O$, we have the same reasoning on arguments $b$, $c$ and $e$ as on example 4.2.4. Argument $a$ is not in conflict with $I \cup U$ and defends itself against $b$: $a$ belongs to App$(I, U, R)$, so it goes in $I$. The status of every argument being decided, we do a FinalTest on $I = \{a, f, h\}$: it is a preferred extension of $AF_1$. The graph representation of $AF_1$ at the end of the run on the second branch is:




**Example 3.12 (cont.)** Let us compute the preferred extensions of $AF_2 = (A, R)$. We call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$.Argument $d$ is self-attacking, so it goes in $O$. Argument $c$ belongs to App$(I, U, R)$, so it goes in $I$ and its successor $b$ goes in $O$. Argument $a$ is conflict-free with $I \cup U$ and defended by $I$: $a$ belongs to App$(I, U, R)$, it goes in $I$. Argument $f$ belongs to App$(I, U, R)$, it goes in $I$ and its successor $e$ goes in $O$. $g$ and $h$ belong to App$(I, U, R)$, they go in $I$. Then the status of every argument is decided, FinalTest$(I, O, R)$ is true, $\{a, c, f, g, h\}$ is the preferred extension of $AF_2$.

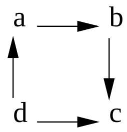



## 5. Related works

5.1. **Credulous query answering.** [VP00] have proposed a proof theory for the credulous consequence relation of the preferred semantics, in form of an argument game. To decide if an argument $x$ is in a preferred extension, the credulous argument game takes the form of a dialogue between two players, a proponent (PRO) and an opponent (CON). PRO and CON propose arguments in turn. The role of PRO is to try to build an admissible set 'around' $x$. CON helps PRO by finding attackers to arguments advanced by PRO. We can establish a correspondance between PRO and our set $I$, and between CON and $R^{-}(I)$. This game has precise rules, but we must keep in mind that every predecessor of PRO is potentially an argument that CON could propose, and every argument which could defend PRO against one of its attackers not yet attacked could be potentially proposed by PRO.

Our credulous query answering algorithm refines the rules of this game by giving precise strategies to choose attackers and defenders. As attackers, we choose predecessors of PRO that are not successors of PRO, because PRO must not defend against arguments that it attacks. As a defender, we choose in priority an argument that has not already been decided, that is an argument not already proposed by PRO and which is neither predecessor nor successor of PRO. If there is no such argument, then we simply choose some argument that defends PRO and that is not in conflict with PRO.

**Example 5.1.** Let $(A, R)$ be an argumentation framework with $A = \{a, b, c, d\}$ and $R = \{(a, b), (b, c), (c, d), (d, a)$.

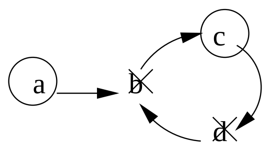



We want to check if $a$ is in a preferred extension. In the credulous argument game, PRO gives $a$, then CON proposes $d$, PRO responds with $c$, CON gives $b$, then PRO repeats $a$. CON can not respond because his only possible proposal would be an argument he already gave ($d$) and he can not repeat himself (the game would never end otherwise). The game stops, PRO has built an admissible set containing $a$. With our strategy, the game would have stopped when PRO proposed $c$: argument $b$ was already attacked by the set of arguments of PRO, so $c$ had not to be defended against it.

The sceptical argument game proposed by [VP00] only works in a particular case: when preferred extensions coincide with stable extensions. This is a very restrictive condition, because stable extensions do not always exist, in particular when the graph possesses cycles of odd length. Thus [VP00] are not able to answer any sceptical query on $AF_1$ (example 2.2).

5.2. **Cycle-cut sets.** The idea of looking for an element of a cycle to branch at a stage when no pruning property can be applied (Prop. 3.6) was first put forward by [DMP97]. However, [DMP97] proposes to compute a feedback vertex set first, that is, a set $F$ of vertices of a graph $(A, R)$ such that the graph restricted to $A - F$ contains no cycle. They then run an enumeration-like algorithm, choosing an undecided element of $F$ every time a branching point is reached. Notice that finding a minimal feedback-vertex set of a given graph $(A, R)$ is known to be an $NP$-complete problem, although heuristics can be used to find a small one. However, there are cases where elements of a feedback vertex set, even a minimal one, are of no use:

**Example 5.2.** Let $A = \{a, b, c, d\}$ and let $R = \{(a, b), (b, c), (c, d), (d, b)\}$. Since $a$ has no predecessor, it has to be in $I$; then, the successor of $a$, that is $b$, goes in $O$; then $d$ cannot be defended, so it goes in $O$, whereas $c$ has to be in $I$ since it is defended by $a$ and it is not adjacent to it. Although there is a cycle here ($bRcRdRb$), the tree explored by our algorithm has a single branch.




This is why it may be more efficient to look for a cycle only when necessary. Moreover, in order not to find several times the same cycle in different branches of the algorithm, we can maintain a set of all cycles that have been found so far during the execution of the algorithm.

5.3. **Computation of stable extensions.** We recall the definition of stable extensions and we show that our algorithm for the computation of preferred extensions can also be used to compute the stable extensions. We also compare our algorithm to other ones dedicated to the computation of stable extensions. Finally, we propose a new computation of stable extensions.

5.3.1. *The stable semantics for argumentation frameworks.* A subset $S$ of the set of vertices $A$ of some argumentation framework $(A, R)$ is a *stable extension* of $(A, R)$ if and only if it is conflict-free and dominant, that is if every element of $A$ which is not in $S$ has a predecessor in $S$. This can be expressed in terms of graph-theoretic concepts as follows:

**Proposition 5.3.** *A subset $S$ of the set of vertices $A$ of some argumentation framework $(A, R)$ is a stable extension of $(A, R)$ if and only if it verifies:*

1. *$R^+(S) \cap S = \emptyset$ ($S$ is conflict-free); and*
2. *$A - S \subseteq R^+(S)$ ($S$ attacks every argument which does not belong to $S$).*

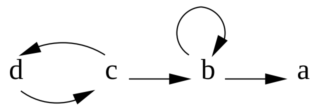
We recall some well-known results of [Dun95]:

**Proposition 5.4.** [Dun95]
*Every stable extension is also a preferred one (but the converse is not true).*
*The unique preferred extension of a well-founded argumentation framework is also a stable extension.*

**Example 2.2 (cont.)** $AF_1$ has no stable extension since argument $k$ is self-attacking (so rejected from any stable extension since stable extensions have to be conflict-free) and it has no other predecessor than itself, so it cannot be attacked by any conflict-free set.

Let us take another example of argumentation framework:

**Example 5.5.** Let $AF_7 = (A, R)$ with $A = \{a, b, c, d\}$ and $R$ as indicated on the graph below:




$AF_7$ possesses a stable extension: $\{a, c\}$ which is also preferred, and it is has a second preferred extension: $\{d\}$.

5.3.2. *Computation of stable extensions as a special case of preferred extensions.* The algorithm for the computation of preferred extensions can be used to compute the stable extensions of an argumentation framework, since these are preferred extensions as well. We adapt the condition StopCondFalse$(C, R)$ to the case of stable extensions: if at the current node, an argument of $O$ is not a successor of $I$ and has all its predecessors in $O$, then we know this node cannot lead to a stable extension. Then, given $C = (I, O, U)$ a $R$-candidate:
StopCondFalse$(C, R) \equiv \exists x \in (O - R^+(I))$ such that $R^-(x) \subseteq O$.
The test we have to do at a leaf to be sure that $I$ is a stable extension (every argument out of $I$ is attacked by $I$) is equivalent to $\neg$StopCondFalse$(C, R)$. The role of FinalTest is reduced to print $I$:
FinalTest$(I \cup U, O, R) \equiv$ Print$(I) \wedge \bot$
What about the pruning properties ? All of them can be used to compute the stable extensions, since every stable extension is also a preferred one. Just notice that Traitors will never be used, since it applies to arguments which are predecessors of argument which are in $O$, themselves predecessors of arguments of $O$: if such a situation occurs, the StopCondFalse is true and the algorithm would have stopped before looking for Traitors.

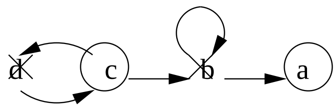
It seems easier to compute the stable extensions than the preferred ones: one of the stop conditions is simpler and we do not have to do any final check.

Let us take a couple of examples to illustrate the computation of stable extensions:

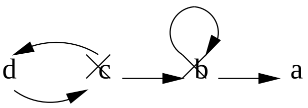
**Example 2.2 (cont.)** We know that $AF_1 = (A, R)$ has no stable extension. This is verified when we call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$. Argument $k$ is self-attacking so it goes in $O$. $k$ has no predecessor, it will never be a successor of $I$: StopCondFalse$((I, O, U), R)$ is true, the run ends.

Let us run the algorithm on $AF_7$:

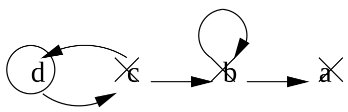
**Example 5.5 (cont.)** To compute the stable extension of $AF_7$, we call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$. Argument $b$ is self-attacking so it goes in $O$. We cannot apply any more pruning property. $AF_7$ has cycles, so we choose an argument cutting a cycle, for example $c$. We build two branches, one where $c$ goes in $I$ (branch 1), the other one where it goes in $O$ (branch 2). In branch 1, since $c$ is in $I$, $d$ goes in $O$. Argument $a$ is conflict-free with $I \cup U$ and is defended by $I$, so it goes in $I$ ($a \in$ App$(I, U, R)$). The status of every argument is decided, and every argument in $O$ is successor of $I$, so $\{a, c\}$ is a stable extension. The graph representation of $AF_7$ at the end of the run on branch 1 is:




In branch 2, $c$ is in $O$: argument $b$ now has its only predecessor in $O$. The StopCondFalse is true, the current set $I$ cannot lead to a stable extension. The graph representation at the end of the run on branch 2 is:




We have found the stable extension of $AF_7$. Had we wanted to compute the preferred extensions, the run on branch 1 would have been the same, but we would not have stopped on branch 2. Argument $d$ would not have been in conflict with $I \cup U$ and it would have defended itself against $c$ ($d \in \text{App}(I, U, R)$): so $d$ would have gone in $I$. Argument $a$ would have gone in $O$ since it would have belonged to $\text{Undef}(I, U, R)$. Then the status of every argument would have been decided, the final checks would have been true, so $\{d\}$ would have been a preferred extension.




5.3.3. *Comparison with related algorithms.* The algorithms of [DMP97], [Nie95] dedicated to the computation of stable extensions explore the equivalent of our $R$-candidates, but they use stronger properties of stable extensions:

**Proposition 5.6.** *Let $(A, R)$ be some argumentation framework, and let $(I, O, U)$ be a partition of $A$. Let $x \in U$, and let $S$ be a stable extension of $(A, R)$ such that $I \subseteq S \subseteq (I \cup U) - \{x\}$, then:*

1. if $x \notin R^+(I \cup U)$, then $x \in A - S \not\subseteq R^+(S)$;
2. if $x \in U$ is such that there exists $y \in O$, such that $R^-(y) \cap (I \cup U) = \{x\}$, then $y \in A - S \not\subseteq R^+(S)$.

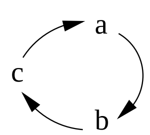
In other words, if there is $x \in U$ such that $x \notin R^+(I \cup U)$ or such that $x$ is the only predecessor in $I \cup U$ of some $y \in O$, then we do not have to compute the subsets of $A$ that contain $I$ but not $x$ since we know that they cannot be stable extensions: we have to add $x$ to $I$.

Undecided arguments that verify the condition of Property 1 of Proposition 5.6 above can be compared with the elements of $\text{App}(I, U, R)$: in the case of stable extensions, we can add an element $x$ to $I$ as soon as all its predecessors are in $O$, but in the case of preferred extension we need to ensure that the predecessors of $x$ have predecessors in $I$. In the case of stable extensions, this verification will be done later.

Undecided arguments that verify the condition of Property 2 of Prop. 5.6 can be compared with the elements of $\text{Heroes}(I, U, R)$. These last elements are the only ones able to attack a predecessor of $I$. The elements which verify Property 2 are the only ones able to justify the absence of an argument of a stable extension; we call them the "braves" in the sequel. All the elements of $\text{Heroes}(I, U, R)$ verify Property 2, but the converse is not true because the braves do not necessarily defend an argument of $I$. This property cannot be applied to the computation of preferred extensions, since an argument can be rejected without being a successor of $I$ (for example when it is not defensible).

The example below proves that the properties of Prop. 5.6 cannot be used for the computation of preferred extensions:

**Example 5.7.** Let $A = \{a, b, c\}$ and let $R = \{(a, b), (b, c), (c, a)\}$.



This is the classical odd cycle with three elements which has no stable extension, but one preferred extension: $\emptyset$. Suppose we choose to branch on $a$, then we put $a$ in $I$ in one branch and $a$ in $O$ in the other. Going down the branch in which $a \in O$, we can apply the properties above in two ways: (1.) the only predecessor $a$ of $b$ is in $O$, so we would have to put $b$ in $I$; (2.) $c$ is the only predecessor of $a$ which is already in $O$, so we would have to put $c$ in $I$. This branch cannot lead to the only preferred extension which is empty. Since on the other branch $I$ already contains an element, it cannot lead to the preferred extension either. This proves that the use of these properties is not complete with respect to preferred extensions.

Let us compute the stable extensions of $AF_7 = (A, R)$ (Example 5.5) using [DMP97]'s algorithm:

**Example 5.5 (cont.)** First of all, we compute a feedback vertex set (see section 5.2); let $F = \{c, b\}$ this set. None of the properties of Prop. 5.6 is applicable, so we choose at random an argument in $F$, for example $c$, and we build two branches, one where we put $c$ in $I$ (branch 1), the other one where we put $c$ in $O$ (branch 2).

In branch 1, $d$ and $b$ go in $O$ since they are successors of $c$. According to property 1, $a$ goes in $I$ since its only predecessor is in $O$. The status of every argument is now decided. The set $\{c, a\}$ is conflict-free and attacks every argument in $A - \{c, a\}$, so it is a stable extension.

In branch 2, $d$ goes in $I$ according to property 1. We cannot decide the status of the remaining undecided arguments thanks to properties 1 and 2. $F$ still contains an undecided argument: $b$. $b$ cuts the last remaining cycle in the set of undecided arguments. So we take $b$ and we build two branches: one where we put $b$ in $I$ (branch 21), the other one where we put it in $O$ (branch 22).

In branch 21, $a$ goes in $O$ since it is a successor of $I$. The status of every argument is decided, but $\{d, b\}$ is not conflict-free since $b$ is self-attacking, so it is not a stable extension.

In branch 22, $a$ goes in $I$ according to property 1. The status of every argument is decided, the set $\{d, a\}$ is conflict-free, but it does not attack $b$ which is in $A - \{d, a\}$, so $\{a, d\}$ is not a stable extension.

Then the only stable extension of $AF_7$ is $\{c, a\}$.

The search tree in this example is deeper than the search tree that we obtained with our algorithm. A reason to this is that [DMP97] does not have a particular look at self-attacking arguments; these arguments are simply components of a cycle of length one. In that sense, we have choosen $b$ as an argument to cut a cycle, and we have built two branches, one where we have put it in $I$: this is obviously contradictory with the conflict-free condition of a stable extension.

We have seen in section 5.2 that the computation of a feedback vertex set [DMP97] is sometimes of no use. This example shows in addition that a feedback vertex set should not contain self-attacking arguments. This suggests that our technique for cutting cycles could be more suitable in this case than [DMP97]'s one, but it remains to prove that it can really be used here.

We show in the next section that we can use the properties of Prop. 5.6 to speed up the computation of stable extensions, and that the use of these conditions combined with some of our pruning conditions improves the algorithm proposed by [DMP97].

5.3.4. *Direct computation of stable extensions.* We describe two new sets of arguments that verify properties 1 and 2 of Prop. 5.6:

AppBis$(I, U, R) = \{x \in U \mid x \notin R^+(I \cup U)\}$ for Property 1

Braves$(I, U, R) = \{x \in U \mid \exists y \notin (I \cup U)$ such that $R^-(y) \cap (I \cup U) = \{x\}\}$ for Property 2

Let us write a new Select function using these properties. App is replaced by AppBis, Heroes is replaced by Braves and we leave out Traitors (which is not applicable to the computation of stable extensions - see section 5.3.2). We also leave out Undef since undecided elements of this set go in $O$ thanks to Property 1: an argument $x$ is not defensible if all its defenders are in $O$; in such a situation, all the attackers of $x$ go in $I$ according to Property 1, and then $x$ goes in $O$.

We keep the pruning condition Refl to put in $O$ self-attacking arguments and we use our technique for cutting cycles.

**Function:** Select$(C, R)$

***Param.:*** *a binary relation $R$, a $R$-candidate $C = (I, O, U)$*

***Result***: *an argument and two booleans*

1. if $\text{Refl}(U,R) \neq \emptyset$ then $(\text{Choose}(\text{Refl}(U,R)), \bot, \top)$
2. else if $\text{AppBis}(I,U,R) \neq \emptyset$ then $(\text{Choose}(\text{AppBis}(I,U,R)), \top, \bot)$
3. else if $\text{Braves}(I,U,R) \neq \emptyset$ then $(\text{Choose}(\text{Braves}(I,U,R)), \top, \bot)$
4. else if $\text{Cycles}(U) = \emptyset$ then $(\text{Choose}(U), \bot, \bot)$
5. else $(\text{Choose}(\text{Choose}(\text{Cycles}(U))), \top, \top)$;

We can see on the example below that this new function produces less branches (and then less leaves) than our first algorithm (see section 5.3.2) and than [DMP97]*'s* one.

**Example 5.5 (cont.)** We call PrefEnum on $R$ and on the $R$-candidate $(\emptyset, \emptyset, A)$. Argument $b$ is self-attacking, so it goes in $O$. Argument $a$ goes in $I$ since its only predecessor is in $O$ ($a \in \text{AppBis}(I,U,R)$). Argument $c$ goes in $I$ since it is the only undecided predecessor of $b$ ($c \in \text{Braves}(I,U,R)$). Argument $d$ being attacked by $c$, it is out of any extension and goes in $O$. The status of every argument is now decided and every argument in $O$ is attacked by $I$: we have found the stable extension of $AF_7$, $\{a,c\}$.

Notice that, thanks to the condition $\text{Refl}(I,U,R)$ , $b$ has gone directly in $O$. Moreover, the search tree is reduced to a single branch; remind it had at least two branches with the previous algorithms (examples 5.3.2 and 5.3.3).

## References


[BDKT97] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[Ber73] C. Berge. *Graphs and Hypergraphs*, volume 6 of *North-Holland Mathematical Library*. North-Holland, 1973.

[CL91] G. Chaty and F. Levy. Default logic and kernel in digraph. Technical Report 91-9, LIPN, Université Paris Nord, Avenue J.B. Clément, 93430 Villetaneuse, 1991.

[DKT96] P.M. Dung, R.A. Kowalski, and F. Toni. Synthesis of Proof Procedures for Default Reasoning. In *Proceedings of the International Workshop on Logic Program Synthesis and Transformation (LOPSTR'96)*, 1996.

[DM94] Y. Dimopoulos and V. Magirou. A graph-theoretic approach to default logic. *Information and computation*, 112:239–256, 1994.

[DM00] S. Doutre and J. Mengin. An Algorithm that Computes the Preferred Extensions of Argumentation Frameworks. In *ECAI'2000, Third International Workshop on Computational Dialectics (CD'2000)*, pages 55–62, August 2000.

[DMP97] Y. Dimopoulos, V. Magirou, and C.H. Papadimitriou. On kernels, Defaults and Even Graphs. *Annals of Mathematics and AI*, 1997.

[DNT99] Y. Dimopoulos, B. Nebel, and F. Toni. Preferred Arguments are Harder to Compute than Stable Extensions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 36–41, 1999.

[DNT00] Y. Dimopoulos, B. Nebel, and F. Toni. Finding Admissible and Preferred Arguments Can be Very Hard. In *Proceedings of KR'200*, pages 53–61, 2000.

[DT93] Y. Dimopoulos and A. Torres. Graph-theoretic structures in logic programs and default theories. Technical Report 93-264, Max-Planck-Institut für Informatik, Saarbrücken, Germany, November 1993.

[Dun95] P.M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[Ino92] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56:301–353, 1992.

[Lev91] F. Levy. Computing Extensions of Default Theories. In Rudolph Kruse and Pierre Siegel, editors, *Symbolic and Quantitative Approaches to Uncertainty, Proceedings of the European Conference ECSQAU*, number 548 in Lecture Notes in Computer Science, pages 219–226, New York, 1991. Springer Verlag.

[Nie95] I. Niemelä. Towards Efficient Default Reasoning. In *Proceedings IJCAI'95*, pages 312–318, 1995.

[Rei87] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987.

[Rym] R. Rymon. Search through Systematic Set Enumeration. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 539–550. Morgan Kaufmann, 1992.

[VP00] G.A.W. Vreeswijk and H. Prakken. Credulous and Sceptical Argument Games for Preferred Semantics. In *Proceedings of the 7th European Workshop on Logics in Artificial Intelligence (JELIA'2000)*, 2000.