Modeling argumentation based semantics using Non-Monotonic Reasoning

JUAN CARLOS NIEVES, ULISES CORTÉS
Universitat Politècnica de Catalunya
Software Department
c/Jordi Girona 1-3, E08034, Barcelona, Spain
(e-mail: {jcnieves,ia}@lsi.upc.edu)

MAURICIO OSORIO
Universidad de las Américas - Puebla
CENTIA
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
(e-mail: josorio@mail.udlap.mx)

9 December 2005

Abstract

Argumentation theory is an alternative style of formalizing non-monotonic reasoning. It seems, argumentation theory is a suitable framework for practical and uncertain reasoning, where arguments support conclusions. Dung’s approach is an unifying framework which has played an influential role on argumentation research and Artificial Intelligence. Even though the success of the argumentation theory, it seems that argumentation theory is so far from being efficiently implemented like the logic programming approach. We present a case of use of the well-known enumerate and eliminate approach to solve the decision problem of the admissible sets and the preferred semantics. By considering this approach, we identify a relationship between the decision problem of the Dung’s preferred semantics and the decision problems of the value-based argumentation’s preferred semantics.

Moreover, we introduce an efficient, clear, and elegant methodology to implement Dung’s approach based on the high-level representation of Answer Set Programming (ASP). This methodology is based on the definition of polynomial time mappings from an argumentation framework to logic programs. By using this methodology, we define a direct relationship between the preferred semantics and minimal models of logic programs. On the other hand, our methodology help us to define efficient extensions of the grounded semantics based on extensions of Well-Founded Semantics (WFS). We point out that our extensions of the grounded semantics offer some advantages to solve an open problem in argumentation based semantics

KEYWORDS: Answer Set Programming, Argumentation Based Semantics, Logic Programming.

1 Introduction

Argumentation theory is one of the research areas of the Artificial Intelligence (AI) that has been actively explored in the last two decades. Argumentation theory is a
suitable framework for practical and uncertain reasoning, where arguments support conclusions. One of the critical points in argumentation theory is the selection of arguments that support a conclusion. Nowadays, there are several approaches for dealing with the construction of arguments by exploring incomplete and uncertain information and resolving conflicts between them. Surveys of this research field are (Prakken and Vreeswijk 2002; Chesñevar et al. 2000).

Although there have been proposed several approaches for argument theory, Dung’s approach, presented in (Dung 1995), is an unifying framework which has played an influential role on argumentation research and AI. In fact, Dung’s approach has been influencing subsequent proposals for argumentation systems, e.g., (Bench-Capon 2002; Vreeswijk 1997). Besides, Dung’s approach is mainly relevant in fields where conflict management plays a central role. For instance, Dung showed that his theory naturally captures the solutions of the theory of n-person game and the well-known stable marriage problem. We are using Dung’s approach to implement decision-making in the medical domain (Tolchinsky et al. 2005).

Dung’s framework is captured by four argumentation semantics: stable semantics, preferred semantics, grounded semantics, and complete semantics. The central notion of these semantics is the acceptability of arguments. An argument is called acceptable if and only if it belongs to a set of arguments which is called extension. The extensions are determined by the argumentation semantics. Dung’s work is particularly relevant in bridging argumentation theory and logic programming. In fact, Dung’s approach can be viewed as a special form of logic programming with negation as failure.

The stable argumentation semantics was characterized by Dung in a clear methodology using logic programming meta-interpreters. On the other hand, the preferred semantics is one of the most relevant semantics from the Dung’s approach. This semantics is regarded as the most satisfactory approach, able to overcome some limitations of the stable argumentation semantics (Prakken and Vreeswijk 2002). However, there are few efficient algorithms in order to infer the preferred semantics (Cayrol et al. 2003; Besnard and Doutre 2004).

The grounded semantics is polynomial time computable and is regarded as a skeptical semantics. Dung already characterized it in terms of the well-known logic programming semantics called Well-Founded Semantics (WFS).

Even though the success of the argumentation theory as one style of formalizing non-monotonic reasoning, it seems that argumentation theory is so far from being efficiently implemented like the logic programming approach.

Answer Set Programming is the realization of much theoretical work on Non-monotonic Reasoning and AI applications. It represents a new paradigm for logic programming that allows, using the concept of negation as failure, to handle problems with default knowledge and produce non-monotonic reasoning. The most popular software implementations to compute answer sets are DLV (DLV) and SMODELS (SMO). The efficiency of such programs allowed to increase the list of practical applications e.g., planning, logical agents and Artificial Intelligence.

The idea of implementing argumentation theory using logic programming is not new, in fact Dung pointed out that his approach can be viewed as a special form
of logic programming with negation as failure. Kakas et al. (Kakas and Toni 1999), developed a proof theory in terms of derivations of trees where each node in a tree contains an argument against its corresponding parent node. In this paper, we extend our previous results, presented in (Nieves et al. 2005), and we present a case of use of the well-known enumerate and eliminate approach to solve the decision problem of the admissible sets and the preferred semantics. By considering this approach, we identify a relationship between the decision problem of the Dung’s preferred semantics and the decision problems of the value-based argumentation’s preferred semantics.

Moreover, we introduce an efficient, clear, and elegant methodology to implement Dung’s approach based on the high-level representation of ASP. This methodology is based on the definition of polynomial time mappings from an argumentation framework to logic programs. By using this methodology, we define a direct relationship between the preferred semantics and minimal models of logic programs. This result has a high impact in the definition of new efficient algorithms to solve the decision problem of the preferred semantics e.g., SAT and/or UNSAT algorithms.

On the other hand, our methodology help us to define efficient extensions of the grounded semantics based on extensions of WFS. Moreover, we point out that our extensions of the grounded semantics offer some advantages to solve the open problem in argumentation based semantics that consists in finding an argumentation semantics which could treat cycles without being affected by the length of the cycles (see (Prakken and Vreeswijk 2002) for more details).

The rest of the paper is structured as follows: In §2, we define the syntax and semantics of our logic programs. Also, it is presented a description of the Dung’s argumentation theory and it is presented a short description of the Bench-Capon’s value-based argumentation theory. In §3, we present a case of use of the enumerate and eliminate approach and identify a relationship between Dung’s preferred semantics and value-based argumentation’s preferred semantics. In §4, we define a relationship between minimal models and the preferred semantics. In §5, we define a mapping from an argumentation framework to a normal program in order to characterize the stable semantics. In §6, we define a mapping in order to characterize the preferred semantics. This mapping is based on positive disjunctive logic programs. In §7, we define a couple of direct characterizations of the grounded semantics. Moreover, we define some extensions of the grounded semantics. These extensions are defined in terms of rewriting systems. In §8, we present an alternative mapping in order to make some variants of the argumentation semantics presented in §7. These variants show some interesting properties w.r.t. the handling of cycles. In §9, we present a simple mapping in order to characterize the complete semantics. In §10, we present our conclusions. In Appendix A, we present some fundamental definitions in Answer Sets about abductive logic programs. And finally in Appendix B, we present some definitions w.r.t. logic programs with Extended Ordered Disjunction.
2 Background

In this section, we define the syntax of the logic programs that we will use in this paper. Also, we will introduce the definition of answer sets semantics, and will present a short description of the Dung’s argumentation theory. And, finally, we will present a short introduction to the Bench-Capon’s value-based argumentation framework.

Sometimes along this paper, we will use basic well-known definitions in complexity theory such as co-NP-complete problem, polynomial-time reducible, etc. We suggest the reader to consult (Cormen et al. 2001) if he/she needs to read more on such definitions.

2.1 Syntax

The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: \(p_0, p_1, \ldots\)

(ii) connectives: \(\lor, \land, \leftarrow, \neg, \perp, \top\)

(iii) auxiliary symbols: (, ).

Where \(\lor, \land, \leftarrow\) are 2-place connectives, \(\neg\) is 1-place connective and \(\perp, \top\) are 0-place connectives. The proposition symbols and \(\perp\) stand for the indecomposable propositions, which we call atoms, or atomic propositions. A literal is an atom, \(a\), or the negation of an atom \(\neg a\). Given a set of atoms \(\{a_1, \ldots, a_n\}\), we write \(\neg\{a_1, \ldots, a_n\}\) to denote the set of literals \(\{\neg a_1, \ldots, \neg a_n\}\).

A general clause, \(C\), is denoted:
\[a_1 \lor \ldots \lor a_m \leftarrow l_1, \ldots, l_n, \]
where \(m \geq 0\), \(n \geq 0\), each \(a_i\) is an atom, and each \(l_i\) is a literal. When \(n = 0\) and \(m > 0\) the clause is an abbreviation of \(a_1 \lor \ldots \lor a_m \leftarrow \top^2\), where \(\top\) is \(\neg \bot\). When \(m = 0\) the clause is an abbreviation of \(\bot \leftarrow l_1, \ldots, l_n^3\). Clauses of this form are called constraints (the rest, non-constraint clauses). Sometimes, we denote a clause \(C\) by \(A \leftarrow B^+, \neg B^−\), where \(A\) contains all the head atoms, \(B^+\) contains all the positive body atoms and \(B^−\) contains all the negative body atoms. We also use body\((C)\) to denote \(B^+ \cup \neg B^−\). A general program, \(P\), is a finite set of clauses. When \(A\) is a singleton set, the clause can be regarded as a normal clause. A normal program is a finite set of normal clauses, formally we understand a normal program as a conjunction of its normal clauses.

Given a set \(S\), \(\tilde{S}\) denotes the complement of \(S\). A signature \(\mathcal{L}\) is a finite set of elements that we call atoms. By \(\mathcal{L}_P\) we understand it to mean the signature of \(P\), i.e. the set of atoms that occurs in \(P\). Given a signature \(\mathcal{L}\), we write \(Prog_\mathcal{L}\) to denote the set of all programs defined over \(\mathcal{L}\).

We point out that our negation \(\neg\) corresponds to the default negation \textit{not} used in Logic Programming.

\(^1 l_1, \ldots, l_n\) represents the formula \(l_1 \land \cdots \land l_n\).

\(^2\) or simply \(a_1 \lor \ldots \lor a_m\).

\(^3\) In fact \(\bot\) is used to define \(\neg A\) as \(A \rightarrow \bot\).
2.2 Semantics

First, to define the answer set semantics, let us define some relevant concepts. Let $P$ be a logic program. The set of all ground atoms expressible in $\mathcal{L}_P$ is called the Herbrand base of $\mathcal{L}_P$ and is denoted by $\mathcal{H}(P)$. An assumption is a grounded negative literal in $\neg \mathcal{H}(P)$. A (Herbrand) interpretation $I$ for $P$ is a subset of $\mathcal{H}(P) \cup \neg \mathcal{H}(P)$ such that $I^+ \cap \neg I^- = \emptyset$, where $I^+$ and $\neg I^-$ are respectively $I \cap \mathcal{H}(P)$ and $I \cap \neg \mathcal{H}(P)$. $	ilde{I}$ denotes $\mathcal{H}(P) \setminus (I^+ \cup I^-)$. An atom $a$ is defined in $I$ if $a \in I^+ \cup I^-$ and undefined if $a \in \tilde{I}$. An atom $a$ is true in $I$ if $a \in I$, false if $\neg a \in I$. An interpretation $I$ is a partial model for $P$ if $P \cup I$ is consistent. A model is a total partial model. Finally, $M$ is a minimal model of $P$ if it does not exists a model $M'$ of $P$ such that $M' \subset M$.

By using Answer Set Programming, it is possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. Currently, there are several answer set solvers that find the answer sets of a program, such as: DLV (DLV ) and SMODELS (SMO ).

The answer set semantics was first defined in terms of the so called Gelfond-Lifschitz reduction (Gelfond and Lifschitz 1988) and it is usually studied in the context of syntax dependent transformations on programs. The following definition of an answer set for general programs generalizes the definition presented in (Gelfond and Lifschitz 1988) and it was presented in (Gelfond and Lifschitz 1991).

Let $P$ be any general program. For any set $S \in \mathcal{L}_P$, let $P^S$ be the general program obtained from $P$ by deleting

(i) each rule that has a formula $\neg l$ in its body with $l \in S$, and then

(ii) all formulas of the form $\neg l$ in the bodies of the remaining rules.

Clearly $P^S$ does not contain $\neg$, so $S$ is an answer set of $P$ iff $S$ is a minimal model of $P^S$.

2.3 Background: Argumentation

The fundamental Dung’s definition is the concept called argumentation framework which is defined as follows:

Definition 1
(Dung 1995) An argumentation framework is a pair $AF = (AR, attacks)$, where $AR$ is a set of arguments, and $attacks$ is a binary relation on $AR$, i.e. $attacks \subseteq AR \times AR$.

Following Dung’s reading, we say that $A$ attacks $B$ (or $B$ is attacked by $A$) if $attacks(A, B)$ holds. Similarly, we say that a set $S$ of arguments attacks $B$ (or $B$ is attacked by $S$) if $B$ is attacked by an argument in $S$.

Definition 2
(Dung 1995) A set $S$ of arguments is said to be conflict-free if there are no arguments $A, B$ in $S$ such that $A$ attacks $B$. 
Definition 3 (Dung 1995) (1) An argument \( A \in \text{AR} \) is acceptable with respect to a set \( S \) of arguments iff for each argument \( B \in \text{AR} \): If \( B \) attacks \( A \) then \( B \) is attacked by \( S \). (2) A conflict-free set of arguments \( S \) is admissible iff each argument in \( S \) is acceptable w.r.t. \( S \).

The (credulous) semantics of an argumentation framework is defined by the notion of preferred extensions.

Definition 4 (Dung 1995) A preferred extension of an argumentation framework \( \text{AF} \) is a maximal (w.r.t. inclusion) admissible set of \( \text{AF} \).

Another relevant semantics that Dung introduced is the stable semantics of an argumentation framework which is based in the notion of stable extension.

Definition 5 (Dung 1995) A conflict-free set of arguments \( S \) is called a stable extension iff \( S \) attacks each argument which does not belong to \( S \).

Dung proved an important relationship between preferred extensions and stable extensions.

Lemma 1 (Dung 1995) Every stable extension is a preferred extension, but no vice versa.

Definition 6 (Dung 1995) An argumentation framework \( \text{AF} \) is said to be coherent if each preferred extension of \( \text{AF} \) is stable.

Dung defined some important concepts w.r.t. the relationship between arguments when they are taking part of a sequence of attacks.

- An argument \( B \) indirectly attacks \( A \) if there exists a finite sequence \( A_0, \ldots, A_{2n+1} \) such that 1) \( A = A_0 \) and \( B = A_{2n+1} \), and 2) for each \( i, 0 \leq i \leq 2n \), \( A_{i+1} \) attacks \( A_i \).
- An argument \( B \) indirectly defends \( A \) if there exists a finite sequence \( A_0, \ldots, A_{2n} \) such that 1) \( A = A_0 \) and \( B = A_{2n} \), and 2) for each \( i, 0 \leq i \leq 2n \), \( A_{i+1} \) attacks \( A_i \).
- An argument \( B \) is said to be controversial w.r.t. \( A \) if \( B \) indirectly attacks \( A \) and indirectly defends \( A \).
- An argument is controversial if it is controversial w.r.t. some argument \( A \).

Definition 7 (Dung 1995)

1. An argumentation framework is uncontroversial if none of its arguments is controversial.
2. An argumentation framework is limited controversial if there exists no infinite sequence of arguments \( A_0, \ldots, A_n, \ldots \) such that \( A_{i+1} \) is controversial w.r.t. \( A_i \).

It is clear that every uncontroversial argument framework is limited controversial but not vice versa.
Theorem 1
(Dung 1995)

1. Every limited controversial argumentation framework is coherent.
2. Every uncontroversial argument framework is coherent and relatively grounded.

Remark 1
This theorem points out that the preferred semantics is equivalent to the stable semantics in any either limited controversial argumentation framework or uncontroversial argumentation framework.

Corollary 1
(Dung 1995) Every limited controversial argumentation framework possesses at least one stable extension.

Dung also defined a skeptical semantics which is called grounded semantics and it is defined in terms of a characteristic function.

Definition 8
(Dung 1995) The characteristic function, denoted by $F_{AF}$, of an argumentation framework $AF = \langle AR, \text{attacks} \rangle$ is defined as follows:
\[ F_{AF} : 2^{AR} \rightarrow 2^{AR} \]
\[ F_{AF}(S) = \{ A | A \text{ is acceptable w.r.t. } S \} \]

Definition 9
(Dung 1995) The grounded extension of an argumentation framework $AF$, denoted by $GE_{AF}$, is the least fixed point of $F_{AF}$.

Dung defined the concept of complete extension which provides the link between preferred extensions (credulous semantics), and grounded extension (skeptical semantics).

Definition 10
(Dung 1995) An admissible set $S$ of arguments is called complete extension iff each argument which is acceptable w.r.t. $S$, belongs to $S$.

Dung suggested a general method for generating metainterpreters in terms of logic programming for argumentation systems. This method is divided in two units: argument generation unit (AGU), and argument processing unit (APU). The AGU is basically the representation of the argumentation framework’s attacks and the APU consists of two clauses:

(C1) $\text{acc}(X) \leftarrow \neg \text{defeat}(X)$
(C2) $\text{defeat}(X) \leftarrow \text{attack}(Y, X), \text{acc}(Y)$

The meaning of C1 is that $X$ is acceptable if it is not defeated and the meaning of C2 is that an argument is defeated if it is attacked by an acceptable argument.
Then given an argumentation framework \( AF = (AR, \text{attacks}) \), \( P_{AF} \) denotes the logic program defined by 
\[
P_{AF} = APU + AGU \text{ where } APU = \{C1, C2\} \text{ and } AGU = \{\text{attacks}(A, B) \leftarrow (A, B) \in \text{attacks}\}
\]

For each extension \( E \) of \( AF \), \( m(E) \) is defined as follows:
\[
m(E) = AGU \cup \{\text{acc}(A) | A \in E\} \cup \{\text{defeat}(B) | B \text{ is attacked by some } A \in E\}
\]

**Theorem 2**
(Dung 1995) Let \( AF \) be an argumentation framework and \( E \) be an extension of \( AF \). Then

1. \( E \) is a stable extension of \( AF \) iff \( m(E) \) is an answer set of \( P_{AF} \)
2. \( E \) is a grounded extension of \( AF \) iff \( m(E) \cup \{\neg \text{defeat}(A) | A \in E\} \) is the well-founded model of \( P_{AF} \)

Now we present a short introduction of the Bench-Capon’s value-based argumentation framework. Bench-Capon extended, in (Bench-Capon 2002), the Dung’s argumentation framework to make defeated argument dependent on the relative importance of the values the arguments advance or protect. So, he re-introduced the definition of an argumentation framework.

**Definition 11**
(Bench-Capon 2002) A value-based argumentation framework (VAF) is a 5-tuple: 
\[
\text{VAF} = (AR, \text{Attacks}, V, \text{val}, \text{valpref})
\]
where \( AR, \text{and} \text{attacks} \) are as for a standard argumentation framework, \( V \) is a non-empty set of values, \( \text{val} \) is a function which maps from elements of \( V \), and \( \text{valpref} \) is a preference relation (transitive, irreflexive, asymmetric) on \( V \times V \). We say that an argument \( A \) relates to value \( v \) if accepting \( A \) promotes or defeats \( v \); the value in question is given be \( \text{val}(A) \). For every \( A \in AF, \text{val}(A) \in V \).

Therefore, Bench-Capon also extended the definition of a defeated argument.

**Definition 12**
(Bench-Capon 2002) Let \( \text{VAF} = (AR, \text{Attacks}, V, \text{val}, \text{valpref}) \) be a value-based argumentation framework. An argument \( A \in AR \) defeats an argument \( B \in AR \) iff both \( (A, B) \in \text{attacks} \) and not \( \text{valpref}(\text{val}(B), \text{val}(A)) \).

The definitions of conflict-free sets and admissible sets are quite similar to the standard definitions.

**Definition 13**
(Bench-Capon 2002) Let \( \text{VAF} = (AR, \text{Attacks}, V, \text{val}, \text{valpref}) \) be a value-based argumentation framework. \( S \subseteq AR \) is a conflict-free set of arguments if
\[
(\forall X)(\forall Y)((X \in S \land Y \in S) \rightarrow (\neg \text{attacks}(X, Y) \land \text{valpref}(\text{val}(X), \text{val}(Y))))
\]

\(^4\) Dung presented results w.r.t. other semantic, but we just cite the results w.r.t. stable extensions and grounded extensions
Definition 14
(Bench-Capon 2002) A conflict-free set of arguments S is admissible if (∀X)(X ∈ S → \text{acceptable}(X, S)).

The semantics of a valued-based argumentation framework is also defined in terms of extensions. So, the definition of the standard preferred extensions is generalized for value-based argumentation framework.

Definition 15
(Bench-Capon 2002) Let VAF = \langle AR, Attacks, V, val, valpref \rangle be a value-based argumentation framework. A set of arguments S ⊆ AR is a preferred extension of VAF if it is a maximal (w.r.t. set inclusion) admissible set of AR.

3 Declarative problem solving: Admissible sets and preferred extensions

In this section, we present a case of use of the \textit{enumerate} and \textit{eliminate} approach in order to find the admissible sets and the preferred extensions of an argumentation framework.

The \textit{enumerate} and \textit{eliminate} approach is a well known approach in ASP for declarative problem solving. This approach depends on how the possibilities are enumerated (see (Baral 2003) for details).

3.1 Admissible sets

Following the enumerate and eliminate approach we first need to enumerate the sets of arguments which could be admissible sets. In this encoding, we use the predicates \texttt{argument}(a_i), \texttt{argument}(a_j) and \texttt{attacked}(a_i, a_j) to represent that the argument a_j is attacked by the argument a_i (let us denote this encoding by Π(AF)).

\textbf{Declaration} : We have the domain specifications.
\begin{align*}
\text{argument}(a_1) & \leftarrow \ldots \ldots \text{argument}(a_m) \leftarrow . \\
\text{attacked}(a_i, a_j) & \leftarrow \ldots \ldots \text{attacked}(a_k, a_l) \leftarrow .
\end{align*}

\textbf{Enumeration} : The enumeration rules create the possible sets which could be admissible sets. We enumerate the possibility space which specifies that each argument a_j may or may not be admissible. The rules with their intuitive meaning are as follows:

- For each argument X, either X is admissible or not.
  \begin{align*}
  \text{admissible}(X) & \leftarrow \neg \text{not\_admissible}(X), \text{argument}(X) . \\
  \text{not\_admissible}(X) & \leftarrow \neg \text{admissible}(X), \text{argument}(X) .
  \end{align*}

- An admissible argument Y cannot be attacked by an admissible argument X
  \begin{align*}
  \leftarrow \text{admissible}(X), \text{admissible}(Y), \text{attacked}(X, Y) .
  \end{align*}

- An admissible argument X cannot be a not\_acceptable argument.
  \begin{align*}
  \leftarrow \text{admissible}(X), \text{not\_acceptable}(X), \text{argument}(X) .
  \end{align*}
Elimination: We use the elimination constraints to force that each admissible argument cannot be attacked by an admissible argument, and an admissible argument is an acceptable argument.

- An argument $X$ is not acceptable if it is attacked by an argument $Y$ such that $Y$ is not attacked by an admissible argument.
  
  \[
  \text{not acceptable}(X) \leftarrow \text{attacked}(Y, X),
  \]
  
  \[
  \neg \text{attacked by pref}(Y), \text{argument}(X), \text{argument}(Y).
  \]

  \[
  \text{attacked by pref}(Y) \leftarrow \text{argument}(Y),
  \]
  
  \[
  \text{admissible}(X), \text{attacked}(X, Y).
  \]

NOTE: The sets of arguments $a_i$ of the predicate $\text{admissible}(a_i)$, for each answer set of $\Pi(AF)$, correspond to the conflict-free sets of $AR$ that are admissible.

This encoding is formalized with the following definition and lemma.

Definition 16
Let $AF = \langle AR, attacks \rangle$ be an argumentation framework and $\text{Adm} = \{\text{admissible}(X) | X \in AR\}$. Let $f_{adm}$ be a function from $AR$ onto $\text{Adm}$ such that $f_{adm}(X) = \text{admissible}(X)$.

We define a straightforward generalization of $f_{adm}$ over a set $S \subseteq AR$ as follows:

$\forall S \subseteq \text{AR}, f_{adm}(S) = \{f_{adm}(s) | s \in S\}$. Furthermore, $f_{adm}$ is an invertible function, then the inverse function $f_{adm}^{-1}$ from $\text{Adm}$ onto $AR$ is defined as follows if $\text{admissible}(X) = f_{adm}(X)$ then $f_{adm}^{-1}(\text{admissible}(X)) = X$.

Lemma 2
Let $AF = \langle AR, attacks \rangle$ be an argumentation framework. Let $M$ be an answer set of $\Pi(AF)$ such that $M \cap \text{Adm} \neq \emptyset$. Then $f_{adm}^{-1}(M \cap \text{Adm})$ is an admissible conflict-free set of $AF$.

Proof
The proof is straightforward. □

Example 1
Let us consider the following argumentation framework: $AF = \langle AR, Attacks \rangle$, where $AR = \{v, nv, risv, cfs, pp, pt, ap\}$ and $\text{Attacks} = \{(v, nv), (nv, v), (risv, v), (cfs, nv), (pp, risv), (pt, risv), (ap, pp)\}$.

Then, the domain specification of the program $\Pi(AF)$ is defined according to $AF$. The answer sets of the program $\Pi(AF)$ are twelve, we show only some of them after intersecting them with the set $\text{Adm}$:

\[
\text{\{}\text{\}, \{\text{admissible}(ap)\}, \{\text{admissible}(cfs)\}, \{\text{admissible}(pt)\}, \text{\}, \{cfs\}, \{pt\}, \{cfs, pt\}, \{v, pt\}, \{cfs, ap\}, \{pt, ap\}, \{v, pt, ap\}, \{cfs, pt, v\}, \{cfs, pt, ap\} \text{\} and \{cfs, pt, ap\}.
\]
3.2 Preferred extensions

Following the case of use of the enumerate and eliminate approach we want to extend our encoding $\Pi(AF)$ in order to find the preferred extensions of an argumentation framework. As we already know the preferred extensions, by definition, are the maximal admissible sets of an argumentation framework, so we have to extend the elimination constraints of $\Pi(AF)$ in order to throw away the admissible sets which are not maximal. In fact, there are several approaches in order to look for maximal sets in ASP. For instance, we can use either ordered disjunctions clauses (Brewka et al. 2002), or maximal answer sets in combination with abductive logic programs.

We present an encoding using an extension of the ordered preferred disjunctions which was presented in (Zepeda et al. 2005). In order to compute this encoding, we use PSMODELS (Brewka et al. 2002). In Appendix B, we present a short background w.r.t. ordered preferred disjunctions.

Declaration : As in the formulation $\Pi(AF)$.

Enumeration : Also as in the formulation $\Pi(AF)$.

Elimination : This formulation has the elimination part of $\Pi(AF)$ plus the following extended preferred ordered disjunction clause.

- An argument is preferred to be admissible than the proposition symbol $all_{\text{pref}}$ to be true.

$$\neg \neg \text{admissible}(X) \times all_{\\text{pref}} \leftarrow \text{argument}(X).$$

Since the extended ordered disjunctions are not implemented in any answer sets solver, we use a mapping presented in (Zepeda et al. 2005) in order to compute the extended ordered disjunctions using standard ordered disjunctions (PSMODELS).

$$\text{aux}(X) \times all_{\text{pref}} \leftarrow \text{argument}(X).$$

$$\neg \neg \text{admissible}(I), \text{aux}(I), \text{argument}(I).$$

$$\text{aux}(I) \leftarrow \neg \text{tmp}(I), \text{argument}(I).$$

$$\text{tmp}(I) \leftarrow \neg \text{admissible}(I), \text{argument}(I).$$

$$\leftarrow \text{admissible}(I), \text{tmp}(I), \text{argument}(I).$$

NOTE: The sets of arguments $a_i$ of the predicate $\text{admissible}(a_i)$, for each answer set of about encoding, correspond to the preferred extensions of the argumentation framework $AF$.

In order to formalize an encoding using maximal answer sets with abductive logic programs. We are going to present some definitions and a lemma.

First of all, we need a preliminary definition about a bijective and restricted function defined on a subset of the signature of a program. This function assigns to each element, of the signature's subset, an element that does not occur in the signature of the original program. Moreover, this function will help to define an abductive logic program.
Definition 17
Let $L_P$ be the signature of a program $P$. Let $T_P$ be a signature of the same cardinality of $L_P$ such that $L_P \cap T_P = \emptyset$. Let $L_P^*$ be any fixed bijective function from $L_P$ onto $T_P$.

We shall denote the image of $a$ under $L_P^*$, namely, $L_P^*(a) = a^*$. We define the straightforward generalization of $L^*_P$ over $A \subseteq L_P$ as follows: $L_P^*(A) = \{ L_P^*(a) \mid a \in A \}$. Abusing of the notation, we let $A^*$ represents $L_P^*(A)$.

The abductive logic program corresponds to a particular translation of the program $\Pi(AF)$. Hence, we are going to present some definitions about maximal answer sets and the particular abductive logic program used to obtain the preferred extensions (Some other definitions are presented in Appendix A).

Definition 18
Let $\{ S_i : i \in I \}$ be a collection of subsets of $U$ such that $\bigcup_{i \in I} S_i = U$ and $A \subseteq U$. We say that $S_i$ is a maximal set w.r.t. $A$ among the collection $\{ S_i : i \in I \}$ iff there is no $S_j$ with $j \neq i$ such that $(S_i \cap A) \subset (S_j \cap A)$.

Definition 19
Let $P$ be a consistent program and $\{ M_i : i \in I \}$ be the collection of answer sets of $P$. Let $A \subseteq L_P$. We say that $M_i$ is a maximal answer set w.r.t. $A$ iff $M_i$ is an answer set of $P$ such that $M_i$ is a maximal set w.r.t. $A$ among the collection of answer sets of $P$.

The translation of a program w.r.t. a set of atoms consists in adding a set of constraints to the original program as follows:

Definition 20
Let $P$ be a program and $A \subseteq L_P$. We define the translation of program $P$ w.r.t. $A$ as $P \cup \text{Cons}_A$ where $\text{Cons}_A = \{ \leftarrow \neg a, \neg a^* \mid a \in A \}$.

Definition 21
Let $\text{Cons}_A$ be a set of constrains w.r.t. $A$ and $\Delta \subseteq A^*$. Then, we define the set of atoms that are not forbidden by $\text{Cons}_A$ w.r.t. $\Delta$, as follows:
$\text{Permitted}(\Delta) = \{ a \mid \leftarrow \neg a, \neg a^* \in \text{Cons}_A \text{ and } a^* \notin \Delta \}$.

Lemma 3
Let $P$ a program and $A \subseteq L_P$. Then, $M$ is a generalized answer set of the abductive program $(P \cup \text{Cons}_A, A^*)$ iff $M \setminus A^*$ is an answer set of $P$ and $\text{Permitted}(\Delta) \subseteq M$.

Proof
(Sketch) $M$ is a generalized answer set of the abductive program $(P \cup \text{Cons}_A, A^*)$ iff $M$ is an answer set of the program $P \cup \text{Cons}_A \cup A^*$ iff $M \setminus A^*$ is an answer set of the program $P \cup \text{Cons}_A$ and $\text{Permitted}(\Delta) \subseteq M$. \qed
Lemma 4
Let $P$ and $M$ be a program and an answer set of $P$ respectively. Let $A \subseteq \mathcal{L}_P$. Then $M$ is a maximal answer set of $P$ w.r.t. $A$ iff $M \cup A^*$ is a minimal generalized answer set of the abductive logic program $\langle (P \cup \text{Cons}_A), A^* \rangle$.

Proof
(Sketch) Using Lemma 3 and Definitions 58 and 56.

The inference of the preferred extensions is introduced as an instance of the Lemma 4. This instance is formalized with the following corollary.

Corollary 2
Let $AF = \langle AR, \text{attacks} \rangle$ be an argumentation framework and $A = f_{\text{adm}}(AR)$. Let $P = \Pi(AF)$ be the program obtained from $AF$ and $M$ be an answer set of $P$. Then $f_{\text{adm}}^{-1}(M \cap A)$ is a preferred extension of $AF$ iff $M \cup A^*$ is a minimal generalized answer set of the abductive logic program $\langle (P \cup \text{Cons}_A), A^* \rangle$.

This corollary takes advantage of the correspondence between the definition of a preferred extension as a maximal (w.r.t. inclusion) admissible set of $AF$ and the definition of a maximal answer set. Moreover, the Corollary 2 says that the preferred extension of an argumentation framework $AF$ is obtained by getting the minimal generalized answer sets of an abductive logic program.

3.3 Preferred extensions for value-based argumentation frameworks
In this subsection, we identify a relationship between the decision problem of the Dung’s preferred semantics and the decision problems of the value-based argumentation’s preferred semantics.

We start formalizing a relationship between the admissible sets of a value-based argumentation framework and Dung’s argumentation framework.

Lemma 5
Let $Q$ be the decision problem of the admissible sets of an argumentation framework, and let $Q'$ be the decision problem of the admissible sets of a value-based argumentation framework. Then $Q'$ is polynomial-time reducible to $Q$.

Proof
Let $VAF$ be a value-based argumentation framework and $AF$ an argumentation framework. And let $\Pi(AF)$ be the encoding defined in Section 3.1. By Lemma 2, we know that $\Pi(AF)$ solves $Q$.

We denote by $\Delta(VAF)$ the following encoding:

1. The predicate preferred relation defines a relation which is transitive, irreflexive and asymmetric.
   \[ \text{preferred} \left( v_i, v_j \right) \leftarrow . . . \text{val} \left( v_k, v_l \right) \]
   \[ \text{preferred} \left( v_i, v_j \right) \leftarrow \ldots \text{prefer}\left( v_i, v_j \right) \leftarrow . \]
2. An argument $X$ is preferred to argument $Y$, if $\text{val}(X, V_i), \text{val}(Y, V_j)$ and \text{preferred relation}(V_i, V_j) holds.

\[ \text{valpref}(X, Y) \leftarrow \text{val}(X, V_i), \text{val}(Y, V_j), \text{preferred relation}(V_i, V_j). \]

3. An argument $X$ defeats $Y$ if $X$ attacks $Y$ and $Y$ is not preferred that $X$.

\[ \text{attacked}(X, Y) \leftarrow \text{attacked}^*(X, Y), \neg \text{valpref}(X, Y). \]

Notice that for any value-based argumentation framework $VAF$, $\Delta(VAF)$ transforms $VAF$ to a standard argumentation framework. So $Q'$ is solved by $\Pi(\Delta(VAF))$. Moreover it is not difficult to see that $\Delta(VAF)$ is polynomial time computable.

Therefore, the following result is a straightforward result.

**Corollary 3**

Let $Q$ be the decision problem of the preferred extensions of an argumentation framework, and let $Q'$ be the decision problem of the preferred extensions of a value-based argumentation framework. Then $Q'$ is polynomial-time reducible to $Q$. This corollary suggests that any algorithm which determines the preferred extensions for a standard argumentation framework could be used to determine the preferred extensions of a value-based argumentation framework (by pre-processing the value-based argumentation framework).

In order to illustrate the Corollary 3, let us consider the following example.

**Example 2**

Let us consider the value-based argumentation framework $VAF := \langle AR, attacks, V, val, valpref \rangle$ of Fig. 1 from (Bench-Capon 2002), where $AR := \{a, b, c, d, e, f, g, h\}$, $attacks := \{(a, b), (b, c), (c, d), (d, a), (e, d), (e, f), (f, g), (g, h), (h, e)\}$, $V := \{\text{red, blue}\}$, $val(a) := \text{blue}$, $val(b) := \text{red}$, $val(c) := \text{red}$, $val(d) := \text{blue}$, $val(e) := \text{red}$, $val(f) := \text{red}$, $val(g) := \text{blue}$, $val(h) := \text{blue}$, and let us consider the case when $\text{red > blue}$, it means that $\text{valpref}(\text{red, blue})$.

By removing from attacks all the attacks such that $(X, Y) \in attacks$ and it is not hold $\text{valpref}(\text{val}(Y), \text{val}(X))$, we get the following standard argumentation framework: $AF := \langle AR', attacks' \rangle$ where $AR' := \{a, b, c, d, e, f, g, h\}$ and $attacks' := \{(b, c), (c, d), (d, a), (e, d), (e, f), (f, g), (g, h)\}$. It is easy to see that AF has one extended preferred $\{g, e, b, a\}$ which it also is a preferred extension of $VAF$.

**4 Dung’s argumentation framework and UNSAT**

In this section, the use of minimal model is introduced in order to determine preferred extensions. The use of minimal model makes possible to introduce UNSAT in order to determine preferred extensions. The benefit of this result is that the best special-purpose algorithms and systems for UNSAT can be used to compute preferred extensions.

First of all, we introduce some definitions which are used in the rest of the paper. Most of our mappings, presented in this paper, use the predicate $d(X)$, where the intended meaning of $d(X)$ is “$X$ is defeated”.
Definition 22
Let $AF = (AR, Attacks)$ be an argumentation framework. Given a set of arguments $E \subseteq AR$, $s(E)$ is defined as follows:

$$s(E) := \{d(a)|a \in AR \setminus E\}$$

Essentially, we understand that if $E$ is a set of acceptable arguments then $s(E)$ will be the set of defeated arguments. In other words, $s(E)$ expresses the complement of the set $E$ w.r.t. $AR$.

Another important set of arguments w.r.t. an argument $A$ is the set of arguments which attacks $A$.

Definition 23
Let $AF = (AR, Attacks)$ be an argumentation framework. The direct defeaters of $A \in AR$ is the set $D(A) := \{B|(B, A) \in Attacks\}$.

4.1 Preferred extensions and UNSAT

We start by presenting a mapping between an argumentation framework to a propositional formula. This formula provides an efficient method for computing preferred extensions by using model checking throughout Unsatisfiability (UNSAT). UNSAT is the complement of Satisfiability (SAT), a problem for which very efficient systems have been developed in AI during the last decade.

Definition 24
Let $AF = (AR, attacks)$ be an argumentation framework, then $\alpha(AF)$ is defined as follows:

$$\alpha(AF) := \bigwedge_{A \in AR} (\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B)) \land (\bigwedge_{B \in D(A)} d(A) \leftarrow \bigwedge_{C \in D(B)} d(C))$$

Notice that $\alpha(AF)$ is essentially a propositional formula (just considering the atoms like $d(a)$ as $d_a$).

\[ \text{Fig. 1. A single chair of three arguments} \]

Example 3
Let $AF = (AR, attacks)$ be an argumentation framework, where $AR := \{a, b, c\}$ and attacks := $\{(a, b), (b, c)\}$ (see Fig. 1). Notice that $D(a) = \{\}$, $D(b) = \{a\}$ and $D(c) = \{b\}$, so if we consider the formula w.r.t. argument $a$, we obtain (in order to be syntactically clear we use uppercase letters as variables and lowercase letters as constants):

$$\left(\bigwedge_{B \in \{\}} d(a) \leftarrow \neg d(B)\right) \land \left(\bigwedge_{B \in \{\}} d(a) \leftarrow \bigwedge_{C \in D(B)} d(C)\right) \equiv \top \land \top \equiv \top$$
It is important to remember that the conjunction of an empty set is the truth value. Now if one considers the formula \( w.r.t. \) argument \( b \), we get

\[
\bigwedge_{B \in \{a\}} d(b) \leftarrow \neg d(B) \wedge \bigwedge_{B \in \{a\}} d(b) \leftarrow \bigwedge_{C \in \text{D}(B)} d(C) \equiv (d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top)
\]

And the formula \( w.r.t. \) argument \( c \) is

\[
\bigwedge_{B \in \{b\}} d(c) \leftarrow \neg d(B) \wedge \bigwedge_{B \in \{b\}} d(c) \leftarrow \bigwedge_{C \in \text{D}(B)} d(C) \equiv (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))
\]

Then \( \alpha(AF) \) is:

\[
(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))
\]

Besnard and Doutre (Besnard and Doutre 2004) already proved the following proposition:

**Proposition 1**

(Besnard and Doutre 2004) Let \( AF = \langle AR, attacks \rangle \) be an argumentation framework. A set \( S \subseteq AR \) is a preferred extension if \( S \) is a maximal model of the formula

\[
\bigwedge_{A \in AR} ((A \rightarrow \bigwedge_{B \in \text{D}(A)} \neg B) \wedge (A \rightarrow \bigwedge_{B \in \text{D}(A)} \bigvee_{C \in \text{D}(B)} C))
\]

Notice that \( \alpha(AF) \) is related to defeated arguments and the formula of Proposition 1 is related to acceptable arguments. It is not difficult to see that \( \alpha(AF) \) is the dual formula of the formula of proposition 1. For instance, let us consider the argumentation framework \( AF \) of Example 3. The formula related to \( AF \), according to Proposition 1, is:

\[
(\neg a \leftarrow b) \wedge (\bot \leftarrow b) \wedge (\neg b \leftarrow c) \wedge (a \leftarrow c)
\]

If we replace each atom \( x \) by the expression \( \neg d(x) \), we get:

\[
(\neg \neg d(a) \leftarrow \neg d(b)) \wedge (\bot \leftarrow \neg d(b)) \wedge (\neg \neg d(b) \leftarrow \neg d(c)) \wedge (\neg d(a) \leftarrow \neg d(c))
\]

Now, if we apply transposition to each implication

\[
(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))
\]

This last formula corresponds to \( \alpha(AF) \). The following lemma is a straightforward result of Proposition 1.

**Lemma 6**

Let \( AF = \langle AR, attacks \rangle \) be an argumentation framework and \( S \subseteq AR \). \( S \) is a preferred extension of \( AF \) iff \( s(S) \) is a minimal model of \( \alpha(AF) \).

In order to illustrate Lemma 6, let us consider again \( \alpha(AF) \) of Example 3. This formula has three models: \( \{d(b)\} \), \( \{d(b), d(c)\} \) and \( \{d(a), d(b), d(c)\} \). So the minimal
model is \{d(b)\}, this implies that \{a, c\} is a preferred extension of AF. In fact, each model of \(\alpha(AF)\) implies an admissible set, this means that \{a, c\}, \{a\} and \{\} are the admissible sets of AF.

There are several approaches to infer minimal models from a propositional formula. For instance in (Bell et al. 1994) were proposed several algorithms to compute minimal models using integer linear programming.

Another option is to use UNSAT’s algorithms to compute minimal models. This means that we can use UNSAT’s algorithms for inferring preferred extensions. This idea is formalized with the following lemma. Let S be a set of well formed formulae then we define \(n(S) := \bigwedge_{c \in S} c\).

\[\text{Lemma 7}\]

Let \(AF = \langle AR, attacks \rangle\) be an argumentation framework and \(S \subseteq AR\). \(S\) is a preferred extension of AF iff \(s(S)\) is a model of \(\alpha(AF)\) and \(\alpha(AF) \land n(\neg s(S)) \land \neg n(s(S))\) is unsatisfiable.

\[\text{Proof}\]

It is direct by Lemma 6.

Let us consider again the argumentation framework \(AF\) of Example 3 in order to illustrate Lemma 7. Let \(S = \{a\}\), so \(s(S) = \{d(b), d(c)\}\). We already know that \(\{d(b), d(c)\}\) is a model of \(\alpha(AF)\), so the formula to verify unsatisfiability is:

\[(d(b) \leftarrow \neg d(a)) \land (d(c) \leftarrow \top) \land (d(c) \leftarrow \neg d(b)) \land (d(c) \leftarrow d(a)) \land \neg d(a) \land (\neg d(b) \lor \neg d(c))\]

However, this formula is satisfiable by the model \(\{d(b)\}\), so \(\{a\}\) is not a preferred extension. Now, let \(S = \{a, c\}\), so \(s(S) = \{d(b)\}\). As we know \(\{d(b)\}\) is also a model of \(\alpha(AF)\), so the formula to verify unsatisfiability is:

\[(d(b) \leftarrow \neg d(a)) \land (d(b) \leftarrow \top) \land (d(c) \leftarrow \neg d(b)) \land (d(c) \leftarrow d(a)) \land \neg d(a) \land \neg d(c) \land \neg d(b)\]

It is easy to see that this formula is unsatisfiable, so \(\{a, c\}\) is a preferred extension. The relevance of Lemma 7 is that UNSAT is the prototypical and best-researched co-NP-complete problem.

The following lemma is a direct result of Lemma 7 and it characterizes the preferred extensions in terms of provability in classical logic.

\[\text{Lemma 8}\]

Let \(AF := \langle AR, attacks \rangle\) be an argumentation framework and \(S \subseteq AR\). \(S\) is a preferred extension of AF iff \(s(S)\) is a model of \(\alpha(AF)\) and \(\alpha(AF) \land n(\neg s(S)) \models n(s(S))\)

\[\text{Proof}\]

The proof is direct by Lemma 7.
In this section, we present our first mapping that is a single and clear mapping in order to infer the stable extensions of an argumentation framework. Given an argumentation framework $AF$, we determine its stable extensions by mapping $AF$ to a normal program $P$ and then computing $P$’s answer set models. The stable extensions give solution to a wide-ranging of argumentation frameworks that Dung called uncontroversial and limited controversial. In particular, the limited controversial argumentation framework always possesses at least one stable extension. By Using this mapping, we can infer the stable extensions using any answer set solver.

In (Nieves et al. 2005), we present an alternative procedural algorithms to infer the stable extensions for a coherent argumentation framework.

**Definition 25**
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. We define the transformation function $F_t$ of the argument $A$ as follows:

$$F_t(A) := A \leftarrow \bigwedge_{B \in D(A)} \neg B$$

Notice that if $D(A) = \emptyset$ then $F_t(A) := A$.

**Definition 26**
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. We define its associated normal program as follow:

$$P'_AF := \bigwedge_{A \in AR} F_t(A)$$

In order to illustrate the transformation, we consider the following example.

**Example 4**
Let us consider again the argumentation framework of Example 1: $AF = \langle AR, Attacks \rangle$, where $AR = \{v, nv, risv, cfs, pp, pt, ap\}$ and $Attacks = \{(v, nv), (nv, v), (risv, v), (cfs, nv), (pp, risv), (pt, risv), (ap, pp)\}$.

The resulting program $P'_AF$ after applying the transformation $F_t$ to each argument of $AF$ is:

- $v \leftarrow \neg nv, \neg risv$
- $nv \leftarrow \neg v, \neg cfs$
- $risv \leftarrow \neg pp, \neg pt$
- $pp \leftarrow \neg ap$
- $ap, pt, cfs$

The answer set of $P'_AF$ is $\{v, ap, pt, cfs\}$ which is the stable extension of the framework $AF$.

Formally, we can express this characterization with the following theorem.

**Theorem 3**
Let $AF$ be an argumentation framework and $E$ be a set of arguments. $E$ is a stable extension of $AF$ iff $E$ is an answer set of $P'_AF$. 
Proof
Due to Theorem 2, it suffices to prove that \( E \) is an answer set of \( P'_{AF} \) iff \( m(E) \) is an answer set of \( P_{AF} \), however it is easy to see that this condition holds.

\( \square \)

6 Preferred extensions and general programs

In Section 3.2, we presented a case of use of the enumerate and eliminate approach to infer the preferred extensions of an argumentation framework. In that approach, we inferred the preferred extensions based on an encoding of the definition of a preferred extension, i.e., using admissible sets.

Another option to infer the preferred semantics is using a straightforward mapping from an argumentation framework to a disjunctive logic program. This approach is an elegant and short form for inferring the preferred extensions of an argumentation framework using any disjunctive answer set solver.

We start this section by defining a simple mapping which helps to infer the preferred extensions in terms of defeated arguments.

**Definition 27**

Let \( AF := \langle AR, attacks \rangle \) be an argumentation framework and \( A \in AR \). We define the transformation function \( \Gamma(A) \) as follows:

\[
\Gamma(A) := \left( \bigwedge_{B \in D(A)} (d(A) \lor d(B)) \right) \land \left( \bigwedge_{B \in D(A)} (d(A) \leftarrow d(B)) \right)
\]

**Definition 28**

Let \( AF := \langle AR, attacks \rangle \) be an argumentation framework. We define its associated general program as follows:

\[
\Gamma_{AF} := \bigwedge_{A \in AR} \Gamma(A)
\]

**Remark 2**

Notice that \( \alpha(AF) \) (see Definition 24) is similar to \( \Gamma_{AF} \). However there are some important differences between both. In order to illustrate that differences, let us consider the argumentation framework \( AF := \langle AR, attacks \rangle \), where \( AR := \{a\} \) and \( attacks := \{(a, a)\} \). Then

\[
\Gamma_{AF} := (d(a) \lor d(a)) \land (d(a) \leftarrow d(a))
\]

and

\[
\alpha(AF) := (d(a) \leftarrow \neg d(a)) \land (d(a) \leftarrow d(a))
\]

It is clear that both formulae have a minimal model which is \( \{d(a)\} \), however \( \alpha(AF) \) has no answer sets. In fact both formulae are logically equivalent in classic logic but not in answer set semantics.

In the following theorem we formalize a characterization of the preferred extensions in terms of answer sets.
Theorem 4

Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. $S$ is a preferred extension of $AF$ iff $s(S)$ is an answer set of $\Gamma_{AF}$.

Proof

$S$ is a preferred extension of $AF$ iff $s(S)$ is minimal model of $\alpha(AF)$ (by Lemma 6) iff $s(S)$ is a minimal model of $\Gamma_{AF}$ (since $\Gamma_{AF}$ is logically equivalent to $\alpha(AF)$ in classical logic) iff $s(S)$ is an answer set of $\Gamma_{AF}$ (since $\Gamma_{AF}$ is a positive disjunctive program and for every positive disjunctive program $P$, $M$ is an answer set of $P$ iff $M$ is a minimal model of $P$) \qed

In order to illustrate the theorem, let us consider the following example.

Example 5

Let $AF := \langle AR, attacks \rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e\}$ and $attacks := \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}$ (see Fig. 2). So $\Gamma_{AF}$ is

\[
\begin{align*}
\textstyle d(a) \lor d(b). & \quad d(a) \leftarrow d(a). \\
\textstyle d(b) \lor d(a). & \quad d(b) \leftarrow d(b). \\
\textstyle d(c) \lor d(b). & \quad d(c) \leftarrow d(c). \\
\textstyle d(c) \leftarrow d(a). & \quad d(c) \leftarrow d(d). \\
\textstyle d(d) \lor d(c). & \quad d(d) \leftarrow d(b), d(c). \\
\textstyle d(e) \lor d(d). & \quad d(e) \leftarrow d(c).
\end{align*}
\]

$\Gamma_{AF}$ has two answer sets which are $\{d(a), d(c), d(e)\}$ and $\{d(b), d(c), d(e), d(d)\}$, so $\{b, d\}$ and $\{a\}$ are the preferred extensions of $AF$.

Fig. 2. An argumentation framework with two-length cycle and three-length cycle.

An alternative form for characterizing the preferred extensions without considering the predicate $d(X)$ is considering a new dual symbol for each argument of the argumentation framework. This idea is formalized with the following lemma. First, let us present some definitions.

Definition 29

Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define the function $\eta$ as $\eta : AR \rightarrow AR'$ such that $\eta(a) \notin AR$. $AR'$ has the same cardinality to $AR$ such that $AR \cap AR' = \emptyset$.

$\eta$ is a bijective function which assigns a new symbol to each argument of $AR$. Notice that the new symbol does not occurs in $AR$. We are going to denote the image of $A \in AR$ under $\eta$ as $A'$. 

Definition 30
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:

$$\Lambda(A) := (\bigwedge_{B \in D(A)} (A' \lor B')) \land (\bigwedge_{B \in D(A)} (A' \leftarrow \bigwedge_{C \in D(B)} C'))$$

Definition 31
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define its associated general program as follows:

$$\Lambda_{AF} := \bigwedge_{A \in AR} (\Lambda(A) \land (A \leftarrow \neg A'))$$

Lemma 9
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. $S$ is a preferred extension iff there is an answer set $M$ of $\Lambda_{AF}$ such that $S = M \cap AR$.

Proof
The proof is straightforward from Theorem 4. □

In order to illustrate the about lemma let us consider the following example.

Example 6
Let $AF := \langle AR, attacks \rangle$ be the argumentation framework of Example 5. So $\Lambda_{AF}$ is

$$a' \lor b', \quad a' \leftarrow a'.
\quad b' \lor a', \quad b' \leftarrow b'.
\quad c' \lor b', \quad c' \lor c'.
\quad c' \leftarrow a', \quad c' \leftarrow d'.
\quad d' \lor c', \quad d' \leftarrow b', c'.
\quad e' \lor d', \quad e' \leftarrow c'.
\quad a \leftarrow \neg a', \quad b \leftarrow \neg b'.
\quad c \leftarrow \neg c', \quad d \leftarrow \neg d'.
\quad e \leftarrow \neg e'.$$

$\Gamma_{AF}$ has two answer sets which are $\{a', c', e', b, d\}$ and $\{b', c', e', d', a\}$, so $\{b, d\}$ and $\{a\}$ are the preferred extensions of $AF$.

7 Grounded semantics

Dung already proved, in (Dung 1995), that the grounded semantics of an argumentation framework could be characterized by using the well-known logic semantics called Well-Founded Semantics (WFS) (Gelder et al. 1991). It is well-known that the grounded semantics is polynomial time computable. In this section, we present a direct characterization of the grounded semantics and present some possible extensions of the grounded semantics using some extensions of WFS. These extensions are still polynomial time computable.
7.1 Well-Founded Semantics

First of all, we present some definitions w.r.t. 3-valued logic semantics.

**Definition 32 (SEM)**

(Dix et al. 2001) For any logic program \( P \) we define \( \text{HEAD}(P) = \{ a | a \leftarrow B^+, \neg B^- \in P \} \) — the set of all head-atoms of \( P \). We also define

\[
\text{SEM}(P) = \langle P^{\text{true}}, P^{\text{false}} \rangle,
\]

where

\[
P^{\text{true}} := \{ p | p \leftarrow \in P \}, \quad P^{\text{false}} := \{ p | p \in \mathcal{L}_P \setminus \text{HEAD}(P) \}.
\]

In any 3-valued logic semantics, we have three possible values for any atom: \textit{true}, \textit{false}, and \textit{undefined}. On the other hand, we can also see any argument, in terms of argumentation semantics, as follows: \textit{accepted}, \textit{defeated}, and \textit{undefeated}.

**Definition 33 (3-valued extension)**

Given an argumentation framework \( \text{AF} := \langle \text{AR}, \text{attacks} \rangle \), and \( S, D \in \text{AR} \). A 3-valued extension is a tuple \( \langle S, D \rangle \), where \( S \cap D = \emptyset \) and \( S \) is an admissible set. We call an argument \( a \) acceptable if \( a \in S \), an argument \( b \) defeated if \( b \in D \), and an argument \( c \) undefeated if \( c \in \text{AR} \setminus \{ S \cup D \} \).

The extensions of \( WFS \), which we will consider, are defined in terms of rewriting systems, so we define some transformation rules for logic programs.

**Definition 34 (Basic Transformation Rules)**

(Dix et al. 2001) A transformation rule is a binary relation on \( \text{Prog}_L \). The following transformation rules are called basic. Let a program \( P \in \text{Prog}_L \) be given.

**RED**: This transformation can be applied to \( P \), if there is an atom \( a \) which does not occur in \( \text{HEAD}(P) \). **RED** transforms \( P \) to the program where all occurrences of \( \neg a \) are removed.

**RED**: This transformation can be applied to \( P \), if there is a rule \( a \leftarrow \in P \). **RED** transforms \( P \) to the program where all clauses that contain \( \neg a \) in their bodies are deleted.

**Success**: Suppose that \( P \) includes a fact \( a \) and a clause \( q \leftarrow \text{body} \) such that \( a \in \text{body} \). Then we replace the clause \( q \leftarrow \text{body} \) by \( q \leftarrow \text{body} \setminus \{a\} \).

**Failure**: Suppose that \( P \) contains a clause \( q \leftarrow \text{body} \) such that \( a \in \text{body} \) and \( a \notin \text{HEAD}(P) \). Then we erase the given clause.

**Loop**: We say that \( P_2 \) results from \( P_1 \) by \( \text{Loop}_A \) if, by definition, there is a set \( A \) of atoms such that

1. for each rule \( a \leftarrow \text{body} \in P_1 \), if \( a \in A \), then \( \text{body} \cap A \neq \emptyset \),
2. \( P_2 := \{ a \leftarrow \text{body} \in P_1 : \text{body} \cap A = \emptyset \} \),
3. \( P_1 \neq P_2 \).
Definition 35 (LLC')

Let \( \text{a} \) be an atom that occurs negatively in a program \( P \) and also appears in the head of some rule. Let \( P_1 \) be the program that results from \( P \) by removing \( \neg \text{a} \) from every clause of \( P \). Let \( \text{Success}^* \) denote the reflexive and transitive closure of the relation \( \text{Success} \). Suppose that \( P_1 \) relates to \( P_2 \) by \( \text{Success}^* \) and \( \text{a} \in P_2 \). In this case, we add \( \text{a} \leftarrow \) to \( P \).

Let \( CS_0 \) be the rewriting system such that contains the transformation rules: \( \text{RED}^+, \text{RED}^-, \text{Success}, \text{Failure}, \) and \( \text{Loop} \), and \( CS_1 := CS_0 \cup \{LLC'\} \).

We denote the uniquely determined normal form of a program \( P \) with respect to the system \( CS \) by \( \text{norm}_{CS}(P) \). Every system \( CS \) induces a semantics \( \text{SEM}_{CS} \) as follows:

\[
\text{SEM}_{CS}(P) := \text{SEM}(\text{norm}_{CS}(P))
\]

7.2 WFS\(^{LLC'} \) semantics

Now, we shall present some possible extensions of the grounded semantics that permit to infer more acceptable arguments in polynomial time. These extensions are based on extensions of the well-founded semantics. We start by considering the mapping presented in Definition 24, but now we present that mapping in terms of a logic program.

Definition 36

Let \( \langle AR, \text{Attacks} \rangle \) be an argumentation framework and \( A \in AR \). We define the transformation function \( \Psi(A) \) as follows:

\[
\Psi(A) := \left( \bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B) \right) \land \left( \bigwedge_{B \in D(A)} d(A) \leftarrow \bigwedge_{C \in D(B)} d(C) \right)
\]

Definition 37

Let \( \langle AR, \text{Attacks} \rangle \) be an argumentation framework. We define its associated normal program as follows:

\[
\Psi_{AF} := \bigwedge_{A \in AR} \Psi(A)
\]
First of all, we point out that the grounded semantics is also inferred by WFS and \( \Psi_{AF} \).

**Definition 38**

Let \( AF = \langle AR, \text{Attacks} \rangle \) be an argumentation framework. Given a set of arguments \( E \subseteq AR \), \( f(E) \) is defined as follows:

\[
f(E) := \{d(a)|a \in E\}
\]

**Lemma 11**

Let \( AF := \langle AR, \text{attacks} \rangle \) be an argumentation framework and \( S, D \subseteq AR \). \( S \) is the grounded extension of \( AF \) iff \( \langle f(D), f(S) \rangle \) is a well-founded model of \( \Psi_{AF} \).

**Proof**

(Sketch) It is well-known that given any normal program \( P \), \( WFS(P) := SEM(norm_{CS_0}(P)) \). So, let \( \langle f(D), f(S) \rangle \) be the well-founded model of \( \Psi_{AF} \). By \( \Psi_{AF} \)'s definition, if \( A \in AR \) such that \( D(A) = \emptyset \), then \( d(A) \notin \text{HEAD}(\Psi_{AF}) \) means \( d(A) \in f(S) \) and \( A \) is an acceptable argument because it is not attacked by any argument. Also, if \( B \in AR \) such that \( B \) is attacked by \( A \), then there is a rule \( r_1 \in \Psi_{AF} \) of the form \( r_1 : d(B) \leftarrow \neg d(A) \); therefore, \( r_1 \) is transformed by \( RED^+ \) to \( r_1 : d(B) \leftarrow \). this means \( d(B) \in f(D) \) and \( B \) is a defeated argument. Also if \( B \) is defended by \( A \), then there is rule \( r_2 \in \Psi_{AF} \) of the form \( r_2 : d(B) \leftarrow d(X_1), \ldots, d(A), \ldots, d(X_n) \), where \( X_i \in AR \) such that \( X_i \) defends \( B \); therefore, \( r_2 \) is deleted by \( \text{Failure} \). In general, one can see that the application of \( CS_0 \) over \( \Psi_{AF} \) will remove from \( \Psi_{AF} \) any rule \( r \in \Psi_{AF} \) such \( r \)'s head is an atom of the form \( d(A) \) and \( A \) is an acceptable argument; therefore, if \( d(B) \leftarrow \in norm_{CS_0}(\Psi_{AF}) \), then \( B \) is a defeated argument. So it is not difficult to see that if \( \langle f(D), f(S) \rangle \) is the well-founded model of \( \Psi_{AF} \) then \( S \) is the grounded extension of \( AF \).  

Now, we introduce an extension of the grounded semantics by considering an extensions of WFS presented in (Dix et al. 2001) and was called \( WFS^{LLC'} \).

**Definition 39**

(Dix et al. 2001) \( CS_1 \) is a confluent rewriting system. It induces a 3-valued semantics that we call \( WFS^{LLC'} \).

By considering \( WFS^{LLC'} \), we define a new semantics for an argumentation framework as follows:

**Definition 40**

Let \( AF := \langle AR, \text{attacks} \rangle \) be an argumentation framework and \( S, D \subseteq AR \). \( \langle S, D \rangle \) is a \( WFS^{LLC'} \)-extension of \( AF \) iff \( \langle f(D), f(S) \rangle \) is a \( WFS^{LLC'} \)-model of \( \Psi_{AF} \).

**Example 7**

Let \( AF := \langle AR, \text{attacks} \rangle \) be an argumentation framework, where \( AR := \{a, b, c\} \) and \( \text{attacks} := \{(a,a), (a,b), (b,c), (c,b)\} \) (see Fig. 3). So \( \Psi_{AF} \) is:
Fig. 3. An argumentation framework with two-length cycle and a self-defeated argument.

\[
\begin{align*}
   d(a) &\leftarrow \neg d(a). & d(a) &\leftarrow d(a). \\
   d(b) &\leftarrow \neg d(a). & d(b) &\leftarrow \neg d(c). \\
   d(b) &\leftarrow d(a). & d(b) &\leftarrow d(e). \\
   d(c) &\leftarrow \neg d(b). & d(c) &\leftarrow d(c), d(a).
\end{align*}
\]

Now, we compute $WFS^{LLC'}(\Psi_{AF})$ by applying $CS_1$ to $\Psi_{AF}$. If we apply $LLC'$ w.r.t. the atom $d(a)$, then it is added $d(a) \leftarrow$ to $\Psi_{AF}$.

\[
\begin{align*}
   d(a) &\leftarrow \neg d(a). & d(a) &\leftarrow d(a). \\
   d(b) &\leftarrow \neg d(a). & d(b) &\leftarrow \neg d(c). \\
   d(b) &\leftarrow d(a). & d(b) &\leftarrow d(c). \\
   d(c) &\leftarrow \neg d(b). & d(c) &\leftarrow d(c), d(a). \\
   d(a) &\leftarrow. &
\end{align*}
\]

Applying $RED^-$, we get:

\[
\begin{align*}
   d(a) &\leftarrow d(a). & d(b) &\leftarrow d(c). \\
   d(b) &\leftarrow d(a). & d(b) &\leftarrow d(c). \\
   d(c) &\leftarrow d(c). & d(c) &\leftarrow d(c).
\end{align*}
\]

Applying $Success$, we get:

\[
\begin{align*}
   d(a) &\leftarrow. & d(b) &\leftarrow d(c). \\
   d(b) &\leftarrow. & d(b) &\leftarrow d(c). \\
   d(c) &\leftarrow \neg d(b). & d(c) &\leftarrow d(c).
\end{align*}
\]

Applying $RED^2$, we get:

\[
\begin{align*}
   d(a) &\leftarrow. & d(b) &\leftarrow d(c). \\
   d(b) &\leftarrow. & d(b) &\leftarrow d(c). \\
   d(c) &\leftarrow d(c). & d(c) &\leftarrow d(c).
\end{align*}
\]

Applying $Loop$, we get:

\[
\begin{align*}
   d(b) &\leftarrow. & d(a) &\leftarrow. \\
   d(a) &\leftarrow. & d(b) &\leftarrow d(c).
\end{align*}
\]

Finally, applying $RED^+$, we get:

\[
\begin{align*}
   d(b) &\leftarrow. & d(a) &\leftarrow.
\end{align*}
\]
So $WF^{LLC}_S(\Psi_{AF}) := \langle\{d(a), d(b)\}, \{d(c)\}\rangle$, this means that $\langle\{c\}, \{a, b\}\rangle$ is a $WF^{LLC}_S$-extension. Notice that $AF$ has an empty grounded extension and the only preferred extension of this example is $\{c\}$ which corresponds to the set of acceptable arguments of the $WF^{LLC}_S$-extension.

### 7.3 $WF^{WK}_S$ semantics

Now, let us consider another possible extension of the grounded semantics. To define this extension, first we define another simple transformation rule.

**Definition 41 (Weak-Cases)**

Let $P$ be a program and suppose the following condition holds: $C_1 \in P$, $C_2 \in P$, $C_1$ is of the form $a \leftarrow l$ and $C_2$ is of the form $a \leftarrow \neg l$. Then the Weak-Cases transformation replaces the clauses $C_1$ and $C_2$ in $P$ by the single clause $a$.

Let $CS_2$ be the rewriting system which contains the transformation rules $CS_0 \cup \{\text{Weak-Cases}\}$.

**Lemma 12**

The $CS_2$ is a confluent rewriting system. It induces a 3-valued semantics that we call $WF^{WK}_S$.

**Proof**

Since Weak-Cases is an instance of the transformation rule $T$-Weak-Cases, which is defined in (Dix et al. 2001), this lemma is straightforward from Theorem 13 of (Dix et al. 2001).

By considering $WF^{WK}_S$ is defined a new semantics for an argumentation framework as follows:

**Definition 42**

Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S, D \subseteq AR$. $(S, D)$ is a $WF^{WK}_S$-extension of $AF$ iff $(f(D), f(S))$ is a $WF^{WK}_S$-model of $\Psi_{AF}$.

![An argumentation framework with a floating argument.](image-url)
Example 8
Let \( AF := (AR, attacks) \) be an argumentation framework, where \( AR := \{a, b, c, d\} \) and \( attacks := \{(a, b), (b, a), (a, c), (b, c), (c, d)\} \) (see Fig. 4). Then \( \Psi_{AF} \) is:

\[
\begin{align*}
  &d(a) \leftarrow \neg d(b). \\
  &d(b) \leftarrow \neg d(a). \\
  &d(c) \leftarrow \neg d(b). \\
  &d(c) \leftarrow \neg d(a). \\
  &d(d) \leftarrow \neg d(c).
\end{align*}
\]

Now, we compute the WFS\(_{WK}\)-extension(\( \Psi_{AF} \)) by applying \( CS_2 \) to \( \Psi_{AF} \). We can see that it is possible to apply the transformation rule \( Weak-Cases \) w.r.t. atom \( d(c) \), so \( Weak-Cases \) transforms \( \Psi_{AF} \) to:

\[
\begin{align*}
  &d(a) \leftarrow \neg d(b). \\
  &d(b) \leftarrow \neg d(a). \\
  &d(c). \\
  &d(d) \leftarrow \neg d(c).
\end{align*}
\]

Applying \( RED^- \), we get:

\[
\begin{align*}
  &d(a) \leftarrow \neg d(b). \\
  &d(b) \leftarrow \neg d(a). \\
  &d(c). \\
  &d(d) \leftarrow \neg d(b), d(a).
\end{align*}
\]

Since the last program is the normal form of \( \Psi_{AF} \), then \( WFS_{WK}(\Psi_{AF}) := \langle \{d(c)\}, \{\}\rangle \), so \( \langle \{\}, \{c\}\rangle \) is the WFS\(_{WK}\)-extension of AF. Notice that the grounded extension of AF is an empty set, and there are two stable extensions which are also preferred extensions, these are: \( \{a, d\} \) and \( \{b, d\} \).

7.4 WFS\(_{WK+LCC'}\) semantics

Finally, we define a third extension of the grounded semantics considering the transformation rules of \( CS_0, LLC' \) and \( Weak-Cases \). So let \( CS_3 := CS_0 \cup \{LLC'\} \cup \{Weak-Cases\} \).

Lemma 13
The \( CS_3 \) is a confluent rewriting system. It induces a 3-valued semantics that we call \( WFS_{WK+LCC'} \).

Proof
It is straightforward from Theorem 15 of (Dix et al. 2001).

Definition 43
Let \( AF := (AR, attacks) \) be an argumentation framework and \( S, D \subseteq AR \). \( (S, D) \) is a WFS\(_{WK+LCC'}\) - extension of \( AF \) iff \( (f(D), f(S)) \) is a WFS\(_{WK}\) - model of \( \Psi_{AF} \).

None of both \( WFS_{LLC'} \) and \( WFS_{LLC'} \) extensions is the same to \( WFS_{WK+LCC'}\) - extension. In order to illustrate this difference let us consider the following example.
Example 9
Let $AF := \langle AR, \text{attacks} \rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e, f, m, n, p\}$ and $\text{attacks} := \{(a, b), (b, c), (c, a), (a, d), (d, e), (e, f), (m, e), (n, m), (n, p), (p, m), (p, n)\}$ (see Fig. 5). It is not difficult to see that $WF_{SLLC}^\prime$-extension $:= \langle \{\}, \{a, b, c, d, e\}\rangle$, $WF_{SWK}$-extension := $\langle \{\}, \{m\}\rangle$, and $WF_{SWK+LLC}^\prime$-extension $:= \langle \{\}, \{a, b, c, d, e, m\}\rangle$.

![Fig. 5. Example](image)

7.5 Formalizing extensions of the grounded semantics

Now, we formalize the extensions of the grounded semantics with the following theorem.

Theorem 5
Let $AF := \langle AR, \text{attacks} \rangle$ be an argumentation framework and $E$ be the grounded extension of $AF$. Then

a) 1. If $\langle S, D \rangle$ is the $WF_{SLLC}^\prime$-extension of $AF$ then $E \subseteq S$.
   2. If $\langle S, D \rangle$ is the $WF_{SWK}$-extension of $AF$ then $E \subseteq S$.
   3. If $\langle S, D \rangle$ is the $WF_{SWK+LLC}^\prime$-extension of $AF$ then $E \subseteq S$.

b) 1. The $WF_{SLLC}^\prime$-extension of $AF$ is polynomial time computable.
   2. The $WF_{SWK}$-extension of $AF$ is polynomial time computable.
   3. The $WF_{SWK+LLC}^\prime$-extension of $AF$ is polynomial time computable.

Proof
a) It is direct by Lemma 11.

b) It is not difficult to see that the mapping of Definition 37 is polynomial time computable; moreover, the rewriting systems $CS_1$, $CS_2$, and $CS_3$ are polynomial time computable (Dix et al. 2001).

This theorem is an important result because it makes a direct relationship between the grounded semantics and our new semantics. Also it points out that our new semantics are polynomial time computable.
8 Discussion: Odd and even length cycles

One of the semantics of the Dung’s approach which has played an influential role on argumentation research is the preferred semantics. However, it is well-known that the preferred semantics has some problems w.r.t. the treatment of cycles (Prakken and Vreeswijk 2002; Baroni et al. 2005). The authors in (Prakken and Vreeswijk 2002) underline:

“In fact, this seems one of the main unsolved problems in argumentation-based semantics.”

So it is open to find an argumentation semantics which could treat cycles without being affected by the length of the cycles. In this section, it is presented some simple variants of the argumentation semantics presented in Section 7 which present some advantages w.r.t. the treatment of cycles in an argumentation framework. This variant is based on the concept of acyclic argument.

Definition 44
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework and $A \in AR$. $A$ is an acyclic argument if there is not a sequence of attacks $A_0, \ldots, A_n$ such that 1) $A = A_0$ and $A = A_n$, and 2) for each $i$, $0 \leq i \leq n$, $A_{i+1}$ attacks $A_i$.

By considering the concept of acyclic argument, we present the following argumentation mapping in terms of normal programs.

Definition 45
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Phi(A)$ as follows:

- If $|D(A)| = 1$ and $A$ is an acyclic argument

$$\Phi(A) := (\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B))$$

otherwise

$$\Phi(A) := (\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B)) \land (\bigwedge_{B \in D(A)} d(A) \leftarrow \bigwedge_{C \in D(B)} d(C))$$

Definition 46
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. We define its associated normal program as follows:

$$\Phi_{AF} := \bigwedge_{A \in AR} \Phi(A)$$

Also by using the concept of acyclic argument, we present a mapping from an argumentation framework to a general program.
Definition 47
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Upsilon(A)$ as follows:

If $|D(A)| = 1$ and $A$ is an acyclic argument

$$\Upsilon(A) := (\bigwedge_{B \in D(A)} d(A) \lor d(B))$$

otherwise

$$\Upsilon(A) := (\bigwedge_{B \in D(A)} d(A) \lor d(B)) \land (\bigwedge_{B \in D(A)} d(A) \leftarrow \bigwedge_{C \in D(B)} d(C))$$

Definition 48
Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. We define its associated general program as follows:

$$\Upsilon_{AF} := \bigwedge_{A \in AR} \Upsilon(A)$$

Now, let us consider the mapping $\Phi$ and the WFS’s extensions in order to introduce some new argumentation semantics.

Definition 49
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S, D \subseteq AR$. Then

1. $(S, D)$ is a $WS^{LLC^\prime}$-extension of $AF$ iff $\langle f(D), f(S) \rangle$ is a $WS^{LLC^\prime}$-model of $\Phi_{AF}$.
2. $(S, D)$ is a $WS^{WK}$-extension of $AF$ iff $\langle f(D), f(S) \rangle$ is a $WS^{WK}$-model of $\Phi_{AF}$.
3. $(S, D)$ is a $WS^{WK+LLC^\prime}$-extension of $AF$ iff $\langle f(D), f(S) \rangle$ is a $WS^{WK+LLC^\prime}$-model of $\Phi_{AF}$.

By using the mapping $\Upsilon_{AF}$ and answer set semantics, we define another new semantics.

Definition 50
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. $S$ is an $\nu$-preferred extension of $AF$ iff $s(S)$ is an answer set of $\Upsilon_{AF}$.

Example 10
Let us consider again Example 8 (see Fig. 4), this example presents an argumentation framework which has an even-length cycle. Although the $WS^{WK}$-extension infers $c$ as a defeated argument, it does not infer $d$ as an accepted argument. We can hope to infer $d$ as an accepted argument if the only argument which attacks $d$ is a defeated argument. We could think that there is something wrong in the $WS^{WK}$-extension, however we have to notice that the extensions of the grounded semantics that we are presenting are so close to the mapping of the argumentation
framework to the logic program. So, any variation of the mapping is going to affect directly to the expected argumentation extension.

If we apply the mapping $\Phi$ to the argumentation framework of Example 8, the logic program $\Phi_{AF}$ is:

\[
\begin{align*}
\text{d(a)} & \leftarrow \neg \text{d(b)}. \\
\text{d(b)} & \leftarrow \neg \text{d(a)}. \\
\text{d(c)} & \leftarrow \neg \text{d(b)}. \\
\text{d(c)} & \leftarrow \neg \text{d(a)}. \\
\text{d(d)} & \leftarrow \neg \text{d(c)}. \\
\end{align*}
\]

Notice that the only acyclic argument with just one attack in Fig. 4 is $d$. The $\text{WF}_S^{WK}$-extension ($\text{WF}_S^{WK}$) of $\text{AF}$ is $\langle \{d\}, \{c\} \rangle$. This means that by using the $\text{WF}_S^{WK}$-extension, we can infer $d$ as an acceptable argument, $c$ as a defeated argument, and $\{a, b\}$ as undefeated arguments. Notice that, the preferred extensions of $\text{AF}$ are $\{a, d\}$ and $\{b, d\}$ which also are stable extensions and the ground extension is the empty set. We can see that the $\text{WF}_S^{WK}$-extension corresponds to the intersection of the preferred extensions. This is an important point since the $\text{WF}_S^{WK}$-extension is polynomial time computable. Moreover, it is important to point out that in the $\text{WF}_S^{WK}$-extension all the arguments belong to the even-length cycle are undefeated and the argument attacked by the even-length cycle is defeated.

By considering the $\text{WF}_S^{LLC}^\dagger$-extension and the $\text{WF}_S^{WK+LLC}^\dagger$-extension, we get $\langle \{\}, \{\} \rangle$ and $\langle \{d\}, \{c\} \rangle$ respectively.

Another simple variant considering general programs is presented with the mapping $\Upsilon$. For instance, by applying the mapping $\Upsilon$ to the argumentation framework of Example 8, the general program $\Upsilon_{AF}$ is:

\[
\begin{align*}
\text{d(a)} \lor \text{d(b)}. & \quad \text{d(a)} \leftarrow \text{d(a)}. \\
\text{d(b)} \lor \text{d(a)}. & \quad \text{d(b)} \leftarrow \text{d(b)}. \\
\text{d(c)} \lor \text{d(b)}. & \quad \text{d(c)} \leftarrow \text{d(b)}. \\
\text{d(c)} \lor \text{d(a)}. & \quad \text{d(c)} \leftarrow \text{d(a)}. \\
\text{d(d)} \lor \text{d(c)}. & \quad \text{d(d)} \leftarrow \text{d(c)}. \\
\end{align*}
\]

$\Upsilon_{AF}$ has two answer sets, which correspond to the minimal models of $\Upsilon_{AF}$: $\{\text{d(a)}, \text{d(c)}\}$ and $\{\text{d(b)}, \text{d(c)}\}$. This means that $\{a, d\}$ and $\{b, d\}$ are two $\upsilon$-preferred extensions. In fact, the $\upsilon$-preferred extensions correspond, in this case, to the preferred extensions of $\text{AF}$. We have to notice that $\Upsilon_{AF}$ is a variant of $\Gamma_{AF}$ (see Definition 28). Since the answer sets of $\Gamma_{AF}$ characterize the preferred extensions of $\text{AF}$, we can consider the $\upsilon$-preferred semantics as an extension of the preferred semantics. In the following example, we will see that both semantics are not the same.

Example 11
Let $\text{AF} = (\text{AR}, \text{attacks})$ be an argumentation framework, where $\text{AR} := \{a, b, c, d, e\}$ and $\text{attacks} := \{(a, c), (c, b), (b, a), (a, d), (b, d), (c, d), (d, c), (d, e)\}$ (see Fig. 6). $\text{AF}$ is a wide discussed argumentation framework (Prakken and Vreeswijk 2002; Baroni et al. 2005). The interesting point $w.r.t. \text{AF}$ is that intuitively we can expect to get $e$
as an accepted argument. However none of the Dung’s semantics could infer \( e \) as an accepted argument. There are not any stable extensions, the grounded extension is empty, the only complete extension is empty and also the only preferred extension is empty. Structurally, the only difference between the argumentation framework of Example 8 and AF is that one has an even length cycle and other one has an odd length cycle, however this ironic difference is so strong for Dung’s semantics approach.

Let us consider now the \( \Phi_{AF} \) program which is:

\[
\begin{align*}
  d(a) & \leftarrow \neg d(b). \\
  d(b) & \leftarrow \neg d(c). \\
  d(c) & \leftarrow \neg d(a). \\
  d(d) & \leftarrow \neg d(b). \\
  d(d) & \leftarrow \neg d(c). \\
  d(e) & \leftarrow \neg d(d).
\end{align*}
\]

The \( WFS^{WK-\phi}\)-extension(AF) is \( \langle \{ e \}, \{ d \} \rangle \), the \( WFS^{LLC'}-\phi\)-extension(AF) is \( \langle \{ e \}, \{ a, b, c, d \} \rangle \) and the \( WFS^{WK+LLC'}-\phi\)-extension(AF) is \( \langle \{ e \}, \{ a, b, c, d \} \rangle \). It is very interesting that all the semantics based on the mapping \( \Phi \) infer \( e \) as an accepted argument. Moreover, it is important to point out that in the \( WFS^{WK-\phi}\)-extension, all the arguments belonging to the odd-cycle are undefeated and the argument attacked by the odd-cycle is defeated. This means that \( WFS^{WK-\phi}\)-extension was not affected by the length of the cycle.

Now let us consider the general program \( \Upsilon_{AF} \).

\[
\begin{align*}
  d(a) \lor d(b). \\
  d(b) \lor d(c). \\
  d(c) \lor d(a). \\
  d(d) \lor d(a). \\
  d(d) \lor d(b). \\
  d(d) \lor d(c). \\
  d(e) \lor d(d).
\end{align*}
\]

\( \Upsilon_{AF} \) has an answer set: \( \{ d(a), d(b), d(c), d(d) \} \). This means that \( \{ e \} \) is an \( \upsilon \)-preferred extension. Notice that the AF’s \( \upsilon \)-preferred extension is different to the AF’s preferred extension.

![Fig. 6. An argumentation framework with a three-length cycle.](image)
Now, we present a direct relationship between the grounded semantics and the semantics based on the mapping $\Phi$. Moreover, we show that these semantics are polynomial time computable.

**Theorem 6**
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $E$ be the grounded extension of $AF$. Then

\textbf{a)}
1. If $⟨S, D⟩$ is the $WFS^{LLC′}_φ$-extension of $AF$ then $E \subseteq S$.
2. If $⟨S, D⟩$ is the $WFS^{WK}_φ$-extension of $AF$ then $E \subseteq S$.
3. If $⟨S, D⟩$ is the $WFS^{WK+LLC′}_φ$-extension of $AF$ then $E \subseteq S$.

\textbf{b)}
1. The $WFS^{LLC′}_φ$-extension of $AF$ is polynomial time computable.
2. The $WFS^{WK}_φ$-extension of $AF$ is polynomial time computable.
3. The $WFS^{WK+LLC′}_φ$-extension of $AF$ is polynomial time computable.

**Proof**
\textbf{a)} By definition, the $WFS^{LLC′}_φ$-extension is characterized by $CS_1$, $WFS^{WK}_φ$-extension is characterized by $CS_2$, and the $WFS^{WK+LLC′}_φ$-extension is characterized by $CS_3$. We know that the grounded extension is characterized by $CS_0$; moreover, $CS_0 \subseteq CS_1$, $CS_0 \subseteq CS_2$ and $CS_0 \subseteq CS_3$. Then the proof is direct by Lemma 11.

\textbf{b)} It is not difficult to see that the mapping $\Phi$ is polynomial. Also, the rewriting systems $CS_1$, $CS_2$, and $CS_3$ are polynomial time computable (Dix et al. 2001).

Finally, we present a relationship between the preferred semantics and the $ν$ preferred semantics.

**Theorem 7**
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework, $E$ be a preferred extension of $AF$ and $U$ be an $ν$-preferred extension then $E \subseteq U$.

**Proof**
This theorem follows by Theorem 4. $\square$

9 Complete semantics

In this section, we present a mapping in order to characterize the complete semantics. Our characterization, of the complete semantics in ASP, is based on the following proposition which was presented in (Besnard and Doutre 2004).

**Proposition 2**
(Besnard and Doutre 2004) Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. A set $S \subseteq AR$ is a complete extension iff $S$ is a model of the formula

$$\bigwedge_{A \in AR} ((A \rightarrow \bigwedge_{B \in D(A)} \neg B) \land (A \leftrightarrow \bigwedge_{B \in D(A)} \bigvee_{C \in D(B)} C))).$$

So, we present a single variation of the formula presented in Proposition 2 in order to compute the complete extensions in ASP.
Definition 51
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Upsilon(A)$ as follows:

$$\Upsilon(A) := (A \rightarrow \bigwedge_{B \in D(A)} \neg B) \land (A \leftrightarrow \bigwedge_{B \in D(A)} (\bigvee_{C \in D(B)} C)) \land (A \lor \neg A)$$

Definition 52
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define its associated general program as follows:

$$\Upsilon_{AF} := \bigwedge_{A \in AR} \Upsilon(A)$$

In order to prove our main theorem of this subsection, let us consider the following lemma:

Lemma 14
Let $P$ be any logic program. $M$ is a model (in classic logic) of $P$ iff $M$ is an answer set of $P \cup C$. Where $C := \{X \lor \neg X | X \in \mathcal{L}_P\}$.

Proof
The proof is straightforward because $C$ infers all the possible combinations of the signature of $P$. \square

The characterization of the complete semantics is formalized with the following theorem.

Theorem 8
Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. $S$ is a complete extension of $AF$ iff $S$ is an answer set of $\Upsilon_{AF}$.

Proof
The proof is direct by Lemma 14 and Proposition 2. \square

Example 12
Let us consider the following single argumentation framework $AF := \langle AR, attacks \rangle$, where $AR := \{a, b\}$ and $attacks := \{(a, b), (b, a)\}$. So $\Upsilon_{AF}$ is

$$\begin{align*}
a &\rightarrow \neg b, & a &\leftrightarrow a, \\
a &\lor \neg a, & b &\rightarrow \neg a, \\
b &\lor \neg b.
\end{align*}$$

$\Upsilon_{AF}$ has three answer sets: $\{a\}, \{b\}, \{\}$, These answer sets are the complete extensions of $AF$. 
Remark 3
Notice that $AF$ has an empty complete extension which is not a preferred extension.

Notice that $\Upsilon_{AF}$ is defined under free programs\textsuperscript{5} to allow negation as failure in the head. Even though there are not answer sets solvers for free programs, one can transform any free program to a general program which is valid program for disjunctive answer set solvers \textit{e.g.}, DLV. In (Osorio et al. 2001) were presented some transformations that preserve answer set semantics.

In order to illustrate how to infer the complete extensions of any argumentation framework using a disjunctive answer set solver, let us consider again the argumentation framework of Example 12. We replace any bi-conditional clause $a \leftrightarrow b$ by $a \leftarrow b$ and $b \leftarrow a$, and any negative literal $\neg a$ that appears, in the head of some clause, by a new positive literal $not\_a$ and add the constrain $\leftarrow not\_a, a$ and the clause $not\_a \leftarrow \neg a$ to $\Upsilon_{AF}$. So, by applying this transformations to $\Upsilon_{AF}$, we obtain:

\[
\begin{align*}
not\_b & \leftarrow a. & a & \leftarrow a. & a & \leftarrow a. \\
(\neg a) & \cup not\_a. & b & \leftarrow b. & b & \leftarrow b. \\
not\_a & \leftarrow b. & b & \leftarrow b. & b & \leftarrow b. \\
(\neg a) & \cup not\_a. & \leftarrow not\_a, b. & \leftarrow not\_b, b. & not\_b & \leftarrow \neg b. \\
not\_a & \leftarrow \neg a. & not\_a & \leftarrow \neg a. & not\_a & \leftarrow \neg a. \\
not\_b & \leftarrow \neg b. & not\_b & \leftarrow \neg b. & not\_b & \leftarrow \neg b.
\end{align*}
\]

This program has three answer sets: \{not\_a, b\}, \{not\_b, a\}, and \{not\_a, not\_b\}. If we remove the new literals \textit{w.r.t} $\Upsilon_{AF}$, we get \{b\}, \{a\}, and \{\} which are the complete extensions of $AF$. The transformations used in this example were presented in (Osorio et al. 2001).

10 Conclusions
Argumentation theory is an alternative style of formalizing non-monotonic reasoning. Argumentation theory is a suitable framework for practical and uncertain reasoning, where arguments support conclusions. One of the critical points in argumentation theory is the selection of arguments that support a conclusion.

Dung’s approach, presented in (Dung 1995), is an unifying framework which has played an influential role on argumentation research and AI. In fact, Dung’s approach has been influencing subsequent proposals for argumentation systems, \textit{e.g.}, (Bench-Capon 2002; Vreeswijk 1997). Besides, Dung’s approach is mainly relevant in fields where conflict management plays a central role.

Even though the success of the argumentation theory, it seems that argumentation theory is so far from being efficiently implemented like the logic programming approach. We introduce an \textit{efficient}, \textit{clear}, and \textit{elegant} methodology to implement

\textsuperscript{5} A free program is a set of free clauses where a free clause is build from a disjunction of literals in the head and a conjunction of literals in the body (see Osorio et al. 2004; Osorio et al. 2001 for more details).
Dung’s approach based on the high-level representation of ASP. This methodology is based on the definition of polynomial time mappings from an argumentation framework to logic programs. By using this methodology, we define a direct relationship between the preferred semantics and minimal models of logic programs. Our methodology also helps us to define efficient extensions of the grounded semantics based on extensions of WFS.

One of the semantics of the Dung’s approach which has played an influential role on argumentation research is the preferred semantics. However, it is well-known that the preferred semantics has some problems \textit{w.r.t.} the treatment of cycles (Prakken and Vreeswijk 2002; Baroni et al. 2005). Our extensions of the grounded semantics present some advantages \textit{w.r.t.} the treatment of cycles in an argumentation framework.

\section*{References}

DLV. http://www.dbai.tuwien.ac.at/proj/dlv/.


Appendix A : Abductive logic programs

We present some fundamental definitions in Answer Sets about abductive logic programs in order to make this paper self contained.

Definition 53 (Baral 2003) Let $P$ and $P'$ be a pair of programs such that $P \subseteq P'$. We say that $P'$ is a conservative extension of $P$ if the following condition holds: $M$ is an answer set for $P$ iff there is an answer set $M'$ for $P'$ such that $M = M' \cap L_P$.

Definition 54 Let $P$ and $P'$ be a pair of programs. We say that $P$ and $P'$ are equivalent, denoted as $P \equiv_{ASP} P'$, if they have the same answer sets.

The following definitions are slight similar to the definitions given in (Balduccini and Gelfond 2003).
J. C. Nieves, M. Osorio, and U. Cortés

Definition 55 (Abductive Logic Program)
An abductive logic program is a pair \( \langle P, A \rangle \) where \( P \) is an arbitrary program and \( A \) a set of atoms, called abducibles.

Definition 56 (Extended Generalized Answer Set and Generalized Answer Set)
Let \( \langle P, A \rangle \) be an abductive logic program and \( \Delta \subseteq A \). (1) \( \langle M, \Delta \rangle \) is an extended generalized answer set of \( \langle P, A \rangle \) iff \( M \) is an answer set of \( P \cup \Delta \). (2) Let \( \langle M, \Delta \rangle \) be an extended generalized answer set of \( \langle P, A \rangle \) then we say that \( M \) is a generalized answer set of \( \langle P, A \rangle \).

Definition 57 (Extended Generalized Answer Set Inclusion Order)
Let \( \langle M_1, \Delta_1 \rangle \) and \( \langle M_2, \Delta_2 \rangle \) be extended generalized answer sets of the abductive program \( \langle P, A \rangle \), we define \( \langle M_1, \Delta_1 \rangle < \langle M_2, \Delta_2 \rangle \) iff \( \Delta_1 \subset \Delta_2 \).

Definition 58 (Minimal Extended Generalized Answer Set)
\( \langle M, \Delta \rangle \) is a minimal extended generalized answer set of the abductive program \( \langle P, A \rangle \) iff \( \langle M, \Delta \rangle \) is an extended generalized answer set of \( \langle P, A \rangle \) and it is minimal w.r.t. extended generalized answer set inclusion order.

Minimal Generalized Answer Set: Let \( \langle M, \Delta \rangle \) be a minimal extended generalized answer set of the abductive program \( \langle P, A \rangle \) then we say that \( M \) is a minimal generalized answer set of \( \langle P, A \rangle \).

Appendix B: Logic Programs with Extended Ordered Disjunction

In (Brewka 2002) Brewka introduced the connective \( \times \), called ordered disjunction, to allow an easy and natural representation of preferences and desires. While the disjunctive clause \( a \vee b \) is satisfied equally by either \( a \) or \( b \), to satisfy the ordered disjunctive clause \( a \times b \), \( a \) will be preferred to \( b \), i.e. a model containing \( a \) will have a better satisfaction degree than a model that contains \( b \) but does not contain \( a \). For example, the natural language statement “I prefer coffee than tea” can be expressed as coffee \( \times \) tea. The definition presented here is that of (Osorio et al. 2004), where ordered disjunctions is extended to wider classes of logic programs \(^6\).

Definition 59 (Ordered Logic Programs)
An extended ordered disjunction rule is either a clause as defined in Section 2.1, or a formula of the form: \( f_1 \times \ldots \times f_n \leftarrow g \) where \( f_1, \ldots, f_n, g \) are (well formed) propositional formulas. An extended ordered disjunction program is a finite set of extended ordered disjunction rules.

The formulas \( f_1 \ldots f_n \) are usually called the choices of a rule and their intuitive reading is as follows: if the body is true and \( f_1 \) is possible, then \( f_1 \); if \( f_1 \) is not possible, then \( f_2 \); \ldots; if none of \( f_1, \ldots, f_{n-1} \) is possible then \( f_n \). The particular case

\(^6\) Moreover, while the extension introduced in (Osorio et al. 2004) is in the context of Answer Sets, the extension introduced in (Brewka et al. 2004) for the operator \( \times \) is in a different context.
where all $f_i$ are literals and $g$ is a conjunction of literals corresponds to the original LPODs as presented by Brewka in (Brewka 2002), and we shall call them standard ordered disjunction programs. If additionally $n = 0$ the clause is a constraint (equiv. $⊥ ← g$). If $n = 1$ it is an extended clause and if $g = ⊤$ the clause is a fact and can be written as $f_1 × \ldots × f_n$. An extended ordered disjunction program and a standard ordered disjunction program as defined by Brewka can be called just extended ordered program and standard ordered program respectively where no ambiguity arises.

Now, we present the semantics of programs with extended ordered disjunction.

Most of the definitions presented here are taken from (Brewka 2002; Brewka et al. 2002). The only relevant difference is the satisfaction degree. The reader may notice that the satisfaction degree as defined here is just a straightforward generalization of Brewka’s definition, according to our notation (see (Osorio et al. 2004)).

Definition 60
(Brewka 2002) Let $r := f_1 × \ldots × f_n ← g$ be an ordered rule. For $k ≤ n$ the $k$-th option of $r$ is defined as follows: $r_k := f_k ← g, not f_1, \ldots not f_{k−1}$. Let $P$ be an extended ordered program. $P'$ is a split program of $P$ if it is obtained by replacing each rule $r := f_1 × \ldots × f_n ← g$ in $P$ by one of its options $r^1, \ldots, r^n$. Let $M$ be a set of atoms. $M$ is an answer set of $P$ iff it is an answer set of a split program $P'$ of $P$. Let $M$ be an answer set of $P$ and let $r := f_1 × \ldots × f_n ← g$ be a rule of $P$. We define the satisfaction degree of $r$, denoted by $deg_M(r)$, as follows:

- if $M ∪ ¬(LP \setminus M) \models_1 g$, then $deg_M(r) = 1$.
- if $M ∪ ¬(LP \setminus M) \not\models_1 g$ then $deg_M(r) = \min \{i \mid M ∪ ¬(LP \setminus M) \vdash_1 f_i\}$.

Theorem 9
(Brewka 2002) Let $P$ be an extended ordered program. If $M$ is an answer set of $P$ then $M$ satisfies all the rules in $P$ to some degree.

Definition 61 (Preferred Answer Set)
(Brewka et al. 2002) Let $P$ be an extended ordered program and $L$ a set of literals. We define $S^i_L(P) = \{r ∈ P \mid deg_L(r) = i\}$. Let $M$ and $N$ be answer sets of an extended ordered program $P$. $M$ is inclusion preferred to $N$, denoted as $M >_i N$, iff there is an $i$ such that $S^i_N(P) ⊂ S^i_M(P)$ and for all $j < i$, $S^j_M(P) = S^j_N(P)$. $M$ is cardinality preferred to $N$, denoted as $M >_c N$, iff there is an $i$ such that $|S^i_M(P)| > |S^i_N(P)|$ and for all $j < i$, $|S^j_M(P)| = |S^j_N(P)|$. $S$ is a $k$-preferred answer set (where $k \in \{\text{inclusion, cardinality}\}$) of $P$ if $S$ is an answer set of $P$ and there is no $S'$ answer set of $P$, $S ≠ S'$, such that $S' >_k S$.

\footnote{Brewka’s LPODs use the strong negation connective. Here we will consider only one type of negation but this does not affect the results given in (Brewka 2002).}

\footnote{Note that since we are not considering strong negation, there is no possibility of having inconsistent answer sets.}