# Complete Extensions as Clark's Completion Semantics

Mauricio Osorio[1], Juan Carlos Nieves[2], Alejandro Santoyo[1]

[1] Universidad de las Américas - Puebla
Depto. de Actuaría, Física y Matemáticas
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@googlemail.com, jasrvro@hotmail.com
[2] Department of Computing Science
Umeå University
SE-901 87, Umeå, Sweden
jcnieves@cs.umu.se

**Abstract.** According to Dung, the sets of arguments which can be considered as admissible from an argumentation framework can be regarded as *logic models* of a given logic program. Clark's completions defines a basic logic programming semantics which has influenced modern non-monotonic semantics such as Answer Set Semantics. The Complete Semantics is a fundamental argumentation semantics which identifies a set of admissible sets which contains the grounded, stable, preferred and ideal semantics. In this paper we introduce a characterization of the complete semantics in terms of logic models using Clark's completions. Given that we use a unique mapping which characterizes the grounded, stable, preferred and ideal semantics, our characterization argues for a strong bridge between argumentation semantics and logic programming semantics with negation as failure. This paper also seeks to draw attention to the correspondence we found between the complete semantics of argumentation frameworks and models of Clark's completion, since this correspondence also allowed us to identify the possibility of computing argumentation frameworks based on integer programming.

## 1 Introduction

Argumentation theory has become an increasingly important and exciting research topic in Artificial Intelligence (AI), with research activities ranging from developing theoretical models, prototype implementations, and application studies [3, 18]. The main purpose of argumentation theory is to study the fundamental mechanism humans use in argumentation and to explore ways to implement this mechanism on computers.

Currently formal argumentation research has been strongly influenced by abstract argumentation theory of Dung [9]. This approach is mainly orientated to manage the interaction of arguments by introducing a single structure called *Argumentation Framework (AF)*. An argumentation framework basically is a tuple of sets: a set of arguments and a set of disagreements between arguments called attacks. Indeed an argumentation framework can be regarded as a directed graph in which the arguments are represented by nodes and the attack relations are represented by arrows. In Figure 1, one can see an example of an argumentation framework and its graph representation.

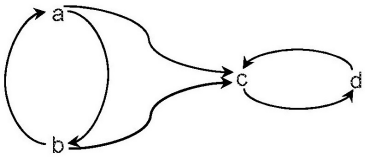


**Fig. 1.** Graph representation of $AF := \langle\{a, b, c, d\}, \{(a, b), (b, a), (a, c), (b, c), (c, d), (d, c)\}\rangle$.

In [9], four argumentation semantics were introduced: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of Dung's semantics is the *acceptability of the arguments*. Even though each of these argumentation semantics represents different patterns of selection of arguments, all of them are based on the basic concept of *admissible set*. Informally speaking, an admissible set presents a coherent and defendable point of view in a conflict between arguments. For instance, by considering the argumentation framework of Figure 1, one can find six admissible sets: $\{\}$, $\{a\}$, $\{b\}$, $\{d\}$ $\{a, d\}$, $\{b, d\}$. The Dung's semantics essentially identify subsets of these admissible sets, see Table 1.

Dung showed that argumentation can be viewed as logic programming with *negation as failure*. Specially, he showed that the grounded semantics can be characterized by the well-founded semantics [13], and the stable semantics by the stable model semantics [14]. In this setting, he showed that the sets of arguments which can be considered as admissible can be regarded as *logic models* of a given logic program. This result is of great importance because it introduces a general method for generating metainterpreters for argumentation systems and regards argumentation semantics from another point of view in order to identify non-monotonic reasoning features of them. Following this issue, the preferred semantics was characterized by the p-stable semantics [17] in [5]. Moreover, the preferred semantics was characterized by the stable model semantics in [16].

Against this background, in this paper we introduce new results which complete the understanding of Dung's semantics in terms of logic programming semantics with negation as failure. By considering an argumentation framework $AF$ and a uniform mapping of $AF$ into the logic program $\Pi^{acc}_{AF}$, we show that the Clarks's completion models of $\Pi^{acc}_{AF}$ characterize the complete extensions of $AF$. Additionally, this result makes the connection to use the method defined in [2] to compute complete extensions of a given $AF$, by computing the Clark's completion models of a logic program $\Pi^{acc}_{AF}$ using integer programming.

One of the main features of our approach is that we use a *unique theory* ( the logic program $\Pi^{acc}_{AF}$) for characterizing Dung's argumentation semantics. By using a unique theory, we identify an *isomorphism* between Dung's argumentation semantics and logic programming semantics. Therefore, by considering an argumentation framework $AF$ and the logic program $\Pi^{acc}_{AF}$, we get that:

- the well founded model of $\Pi^{acc}_{AF}$ characterizes the grounded extension of $AF$,
- the stable models of $\Pi^{acc}_{AF}$ characterize the stable extensions of $AF$,
- the p-stable models of $\Pi^{acc}_{AF}$ characterize the prefer extensions of $AF$,

– the Clark's completion models of $\Pi_{AF}^{acc}$ characterize the complete extensions of $AF$.

For instance, by considering the argumentation framework of Figure 1, the different Dung's semantics can be inferred by the $acc(x)$ atoms which appear in the models of $\Pi_{AF}^{acc}$, as is shown in Table 1.

**Table 1.** Applying Dung's argumentation semantics to the argumentation framework of Figure 1

| The different extensions of $AF$ | Different kinds of Models of $\Pi_{AF}^{acc}$ |
|---|---|
| **Grounded extension** = $\{\}$ | $\mathbf{WFS}(\Pi_{AF}^{acc}) = \langle\{\}, \{\}\rangle$ |
| **Complete extensions** = $\{\{\}, \{a, d\}, \{b, d\}, \{d\}\}$ | **Clark's Completion models** of $\Pi_{AF}^{acc}$: $\{def(a), def(b), def(c), def(d)\}, \{\mathbf{acc(a)}, \mathbf{acc(d)}, def(b), def(c)\}, \{\mathbf{acc(b)}, \mathbf{acc(d)}, def(a), def(c)\}, \{\mathbf{acc(d)}, def(a), def(b), def(c)\}$ |
| **Stable extensions** = $\{\{a, d\}, \{b, d\}\}$ | **Stable models** of $\Pi_{AF}^{acc}$: $\{\mathbf{acc(a)}, \mathbf{acc(d)}, def(b), def(c)\}, \{\mathbf{acc(b)}, \mathbf{acc(d)}, def(a), def(c)\}$ |
| **Preferred extensions** = $\{\{a, d\}, \{b, d\}\}$ | **P-stable models** of $\Pi_{AF}^{acc}$: $\{\mathbf{acc(a)}, \mathbf{acc(d)}, def(b), def(c)\}, \{\mathbf{acc(b)}, \mathbf{acc(d)}, def(a), def(c)\}$ |

The characterization of argumentation semantics in terms of logic programming semantics does not only contribute to the inference of the well-accepted argumentation semantics but also this approach contributes by defining a method for studying the non-monotonic reasoning properties of the argumentation semantics.

It is worth to mentioning that in the literature, we can find different characterizations of argumentation semantics in terms of logic engines [20, 12]. For instance, in [20], there was presented another translation from Argumentation Frameworks to Logic Programs such that the Complete Extensions correspond to 3-valued Stable Models. This result is important but it goes in different direction from ours approach. First, from a theoretical point of view, we are interested in the concrete translation that was able to capture the other three semantics introduced by Dung as explained before. Second, from a practical point of view our translation as being based on the standard Clark's Completion only requires the well known classical logic where very efficient software exists. Recall that their translation is based on non classical logic (3-valued satble models).

In addition to the four argumentation semantics introduced in [9], in [10], the *ideal semantics* was introduced by Dung et al. The ideal semantics has been promoted as a solid argumentation semantics for performing skeptical reasoning. Indeed, the ideal semantics is regarded as an extension of the grounded semantics because the maximal ideal set is a super set of the grounded extension. Recently it was shown that the ideal sets of an argumentation framework can be characterized by logic models and $\Pi_{AF}^{acc}$. Moreover, it was shown that the maximal ideal set can be characterized by $WFS^+$ and $\Pi_{AF}^{acc}$ [15]. $WFS^+$ (also called the Well-Founded-By-Cases Semantics) [7, 19] has proved to be a sound logic programming semantics for performing skeptical reasoning. Indeed, Dix showed that $WFS^+$ is a well-behaved semantics [8] which means that this

semantics satisfies a basic principle of non-monotonic reasoning (see [8] for a formal definition of a well-behaved semantics).

The rest of the paper is divided as follows: In Section 2, a basic background about logic programming is introduced. After this, the basic definitions of Dung's argumentation semantics are presented. In Section 3, our study about the relationship between Clark's completion models and complete extensions is presented. In the third section, our mapping of an argumentation framework into logic programs is also presented. In the last section, we outline our conclusions and future work.

## 2 Background

In this section, we first define the syntax of a valid logic program, after that the Clark's completion semantics is presented. In the last part of this section, we present some basic concepts of argumentation theory.

### 2.1 Basic Notions

A signature $\mathcal{L}$ is a finite set of elements that we call atoms. A literal is an atom, $a$ (called *a positive literal*), or the negation of an atom $not\ a$ (called *a negative literal*). Given a set of atoms $\{a_1, \ldots, a_n\}$, we write $not\ \{a_1, \ldots, a_n\}$ to denote the set of literals $\{not\ a_1, \ldots, not\ a_n\}$. A normal clause is of the form:

$$a_0 \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n$$

in which $a_i$ is an atom, $0 \leq i \leq n$. When $n = 0$ the normal clause is an abbreviation of $a_0 \leftarrow \top$, where $\top$ and $\bot$ are the ever true and ever false propositions respectively. A normal logic program is a finite set of normal clauses.

Logic consequence in classic logic is denoted by $\vdash$. Given a set of proposition symbols $S$ and a theory (a set of well-formed formula) $\Gamma$, if $\Gamma \vdash S$ if and only if $\forall s \in S$ $\Gamma \vdash s$. Whenever we treat a logic program as a logic theory, each negative literal $not\ a$ is replaced by $\sim a$ such that $\sim$ is regarded as the classical negation in classic logic.

A basic property between logic theories which will be considering in our study between logic programming semantics and argumentation semantics is the property of *conservative extension*. This property between logic theories is defined as follows:

**Conservative extension.**
*In mathematical logic, a logical theory $T_2$ is a conservative extension of a theory $T_1$ if the language of $T_2$ extends the language of $T_1$, every theorem of $T_1$ is a theorem of $T_2$, and any theorem of $T_2$ which is in the language of $T_1$ is already a theorem of $T_1$.*

The Clark's Completion of a given logic program is an old concept that has been intensively explored in logic programming literature in order to identify basic properties of logic programming semantics with negation as failure [1, 6]. It is defined as follows: Given a normal logic program $P$, its completion $Comp(P)$ is obtained in two steps:

1. each normal clause $a_0 \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n \in P$ is replaced with the formula:
$$a_0 \leftarrow a_1 \wedge \cdots \wedge a_j \wedge \sim a_{j+1} \wedge \cdots \wedge \sim a_n$$

2. for each symbol $a \in \mathcal{L}_P$, let $Support(a)$ denotes the set of all formulae with $a$ in the head. Suppose $Support(a)$ is the set:

$$\{a \leftarrow Body_1, \ldots, a \leftarrow Body_m\}$$

in which each $Body_i (1 \leq i \leq m)$ is of the form $a_1 \wedge \cdots \wedge a_j \wedge \sim a_{j+1} \wedge \cdots \wedge \sim a_n$. Replace $Support(a)$ with the single formula:

$$a \leftrightarrow Body_1 \vee \cdots \vee Body_m$$

If $Support(a) = \emptyset$ then replace it by $\sim a$.

*Example 1.* Let $P$ be the following normal program:

$$p \leftarrow a. \quad p \leftarrow b. \quad a \leftarrow not\ b. \quad b \leftarrow not\ a.$$

One can see that $Comp(P) = \{p \leftrightarrow a \vee b, a \leftrightarrow \sim b, b \leftrightarrow \sim a\}$. $Comp(P)$ has two models: $\{a, p\}$ and $\{b, p\}$.

An important concept that is closely related to the completion of a normal logic program is the so called *supported model*.

**Supported Models.**
*A model $M$ of a given normal logic program $P$ is called a* supported model *of $P$ if and only if for every $a \in M$ there is a rule*

$$a_0 \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n \in P$$

*such that $a = a_0$, $\{a_1, \ldots, a_j\} \subseteq M$ and $\{a_{j+1}, \ldots, a_n\} \cap M = \emptyset$.*

It is well known that given a normal logic program $P$, the models of $Comp(P)$ correspond to the supported models of $P$ [6]. The set of supported models of a logic program defines the so called *Clark's completion semantics* [1].

### 2.2 Argumentation theory

Now, we define some basic concepts of Dung's argumentation approach. The first one is an argumentation framework. An argumentation framework captures the relationships between arguments.

**Definition 1.** *[9] An argumentation framework is a pair $AF := \langle AR, attacks \rangle$, where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR*,* i.e. attacks $\subseteq AR \times AR$.

Any argumentation framework can be regarded as a directed graph. For instance, if $AF := \langle \{a, b, c, d\}, \{(a, b), (b, a), (a, c), (b, c), (c, d), (d, c)\} \rangle$, then $AF$ is represented as it is shown in Figure 1. 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 $S$.

Let us observe that an argumentation framework is a simple structure which captures the conflicts of a given set of arguments. In order to select *coherent points of views* from a set of conflicts of arguments, Dung introduced a set of *patterns of selection of arguments*. These patterns of selection of arguments were called *argumentation semantics*. Dung defined his argumentation semantics based on the basic concept of *admissible set*:

**Definition 2.** *[9] 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. An argument $a \in AR$ is acceptable with respect to a set $S$ of arguments if and only if for each argument $b \in AR$: If $b$ attacks $a$ then $b$ is attacked by $S$. A conflict-free set of arguments $S$ is admissible if and only if each argument in $S$ is acceptable w.r.t. $S$.*

By considering the concept of admissible set, in [9], Dung introduced four basic argumentation semantics: the grounded, stable, preferred and complete semantics. Even though all of them are based on admissible sets, each of them represent different pattern of selection of arguments. Three of them follow a credulous approach which means that given a set of conflicts of arguments, these semantics can suggest different sets of arguments which can be regarded as coherent points of views. These credulous argumentation semantics are defined as follows.

**Definition 3.** *[9] Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. An admissible set of argument $S \subseteq AR$ is:*

- stable *if and only if $S$ attacks each argument which does not belong to $S$.*
- preferred *if and only if $S$ is a maximal (w.r.t. inclusion) admissible set of $AF$ .*
- complete *if and only if each argument, which is acceptable with respect to $S$, belongs to $S$.*

In [9], Dung introduced a skeptical argumentation semantics which means that given a set of conflicts of arguments this semantics can suggest only one set of arguments which can be regarded as a coherent points of view. This semantics is called *grounded semantics* and is defined in terms of a *characteristic function*.

**Definition 4.** *[9] The characteristic function, denoted by $F_{AF}$, of an argumentation framework $AF = \langle AR, attacks \rangle$ is defined as follows:*
$F_{AF} : 2^{AR} \rightarrow 2^{AR}$
$F_{AF}(S) = \{A |$ *A is acceptable w.r.t. S* $\}$

**Definition 5.** *[9] The grounded extension of an argumentation framework AF, denoted by $GE_{AF}$, is the least fixed point of $F_{AF}$*

From the semantics introduced in [9], the grounded semantics was the only semantics which follows a skeptical approach.

Since the first argumentation semantics were introduced in [9], Dung's argumentation semantics have given place to different formal studies about the properties of them. One of these formal studies has been to regard them as formal non-monotonic reasoning inferences. In this setting, one can find that the argumentation semantics are close related to logic programming semantics with *negation as failure*. In the following sections, we will study different relations between argumentation semantics based on admissible sets and logic programming semantics with negation as failure.

## 3 Complete Semantics as Logic Programming Semantics

In this section, our main results are presented. In particular, we formalize a characterization of complete extensions in term of Clark's completion models. These results complete the relationships between the argumentation semantics introduced in [9] and some particular logic programming semantics with *negation as failure*. The formal proofs and an illustrative example of our results are presented in the appendix.

### 3.1 Mapping from argumentation frameworks to normal programs

The first step for studying the structure of an argumentation framework as a logic program is to manage an argumentation framework as a logic program. To this end, a mapping from an argumentation framework into a logic program will be presented. Let us observe that this mapping basically is *a declarative representation* of an argumentation framework by having in mind the the ideas of *conflictfreeness* and *reinstatement* which are the basic concepts behind the definition of admissible sets.

In this mapping, the predicate $def(x)$ is used, the intended meaning of $def(x)$ is "$x$ is a defeated argument" which means that $x$ cannot be part of an admissible set. A transformation function *w.r.t.* an argument is defined as follows.

**Definition 6.** *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework and $a \in AR$. We define the transformation function $\Pi(a)$ as follows:*

$$\Pi(a) = \bigcup_{b:(b,a)\in Attacks} \{def(a) \leftarrow not\ def(b)\} \cup \bigcup_{b:(b,a)\in Attacks} \{def(a) \leftarrow \bigwedge_{c:(c,b)\in Attacks} def(c)\}$$

The transformation function $\Pi$ with respect to an argumentation framework $AF$ is defined as follows:

**Definition 7.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define its associated normal program as follows:*

$$\Pi_{AF} := \bigcup_{a\in AR} \{\Pi(a)\}$$

As one can see in $\Pi_{AF}$, the language of $\Pi_{AF}$ only identifies the arguments which can be considered as defeated. By considering *total interpretations*, as the ones suggested by logic programming semantics as stable model semantics [14], we can assume that any argument which is not defeated in a model of $\Pi_{AF}$ will be acceptable. This means that given an argumentation framework $AF = \langle AR, Attacks\rangle$ if $M$ is a model of $\Pi_{AF}$, then any atom $def(x)$ which is false in $M$ will identify an argument $x$ which is acceptable. This assumption suggests a normal clause of the following form:

$$acc(x) \leftarrow not\ def(x).$$

where $acc(x)$ denotes that the argument $x$ can be considered as accepted. This clause essentially fixes as acceptable any argument which is not fixed as defeated in $\Pi_{AF}$. On

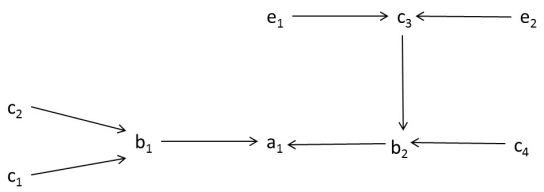


**Fig. 2.** Graph representation of $AF := \langle\{a1, b1, b2, c1, c2, c3, c4, e1, e2\}, \{(c1, b1), (c2, b1), (b1, a1), (b2, a1), (c3, b2), (c4, b2), (e1, c3), (e2, c3)\}\rangle$.

the other hand, this clause suggests an easy form for inferring the acceptable arguments of $AF$. Being aware of these observations, we extend $\Pi_{AF}$ as follows:

$$\Pi^{acc}_{AF} := \Pi_{AF} \cup \bigcup_{a \in AR} \{acc(a) \leftarrow \ not\ def(a)\}$$

It is important to observe that $\Pi^{acc}_{AF}$ is *a conservative extension* of $\Pi_{AF}$. Informally speaking, we can say that both $\Pi_{AF}$ and $\Pi^{acc}_{AF}$ represent the same mapping.

**Observation**
*For sake of simplicity of the proofs some part of the results will be concentrated on $\Pi_{AF}$; however, as we have seen, $\Pi_{AF}$ and $\Pi^{acc}_{AF}$ denote the same mapping of an argumentation framework into a normal logic program.*

In order to illustrate $\Pi_{AF}$ and $\Pi^{acc}_{AF}$, let us consider the following example.

*Example 2.* Let $AF := \langle AR, attacks\rangle$ be an argumentation framework such that $AR := \{a1, b1, b2, c1, c2, c3, c4, e1, e2\}$ and $attacks := \{(c1, b1), (c2, b1), (b1, a1), (b2, a1), (c3, b2), (c4, b2), (e1, c3), (e2, c3)\}$. The graph representation of $AF$ is presented in Figure 2. One can see that $\Pi_{AF}$ is:

$def(c3) \leftarrow \ not\ def(e1).$ $def(b1) \leftarrow \ not\ def(e2).$ $def(c3) \leftarrow \top.$
$def(b1) \leftarrow \ not\ def(c1).$ $def(b1) \leftarrow \ not\ def(c2).$ $def(b1) \leftarrow \top.$
$def(b2) \leftarrow \ not\ def(c3).$ $def(b2) \leftarrow \ notdef(c4).$ $def(b2) \leftarrow \top.$
$def(a1) \leftarrow \ not\ def(b1).$ $def(a1) \leftarrow def(c1), def(c2).$
$def(a1) \leftarrow \ not\ def(b2).$ $def(a1) \leftarrow def(c3), def(c4).$

On the other hand, $\Pi^{acc}_{AF}$ is $\Pi_{AF}$ union with the following set of clauses:

$acc(a1) \leftarrow \ not\ def(a1).$ $acc(b1) \leftarrow \ not\ def(b1).$
$acc(b2) \leftarrow \ not\ def(b2).$ $acc(c1) \leftarrow \ not\ def(c1).$
$acc(c2) \leftarrow \ not\ def(c2).$ $acc(c3) \leftarrow \ not\ def(c3).$
$acc(c4) \leftarrow \ not\ def(c4).$ $acc(e1) \leftarrow \ not\ def(e1).$
$acc(e2) \leftarrow \ not\ def(e2).$

Before moving on, let us observe that by considering either $\Pi_{AF}$ or $\Pi^{acc}_{AF}$, the grounded, stable and preferred semantics can be characterized by different logic programming semantics [5]. In order to introduce these characterizations, we introduce

the following notation: Given a set of arguments $E$: $tr(E) = \{acc(a)|a \in E\} \cup \{def(b)|b$ is an argument and $b \notin E\}$, $tr^{acc}(E) = \{acc(a)|a \in E\}$, $tr^{def}(E) = \{def(b)|a \in E\}$.

**Theorem 1.** *[5] Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework, $E \subseteq AR$ and $E^+ = \{b|a \in E$ and $(a, b) \in Attacks\}$. Then:*

- *$E$ is the grounded extension of $AF$ iff $\langle tr^{acc}(E) \cup tr^{def}(E^+), tr^{acc}(E^+) \cup tr^{def}(E)\rangle$ is the well-founded model of $\Pi^{acc}_{AF}$.*
- *$E$ is a stable extension of $AF$ iff $tr(E)$ is a stable model of $\Pi^{acc}_{AF}$.*
- *$E$ is a preferred extension of $AF$ iff $tr(E)$ is a p-stable model of $\Pi^{acc}_{AF}$.*

Let us observe that this theorem extended Theorem 62 from [9]. Theorem 62 from [9] introduced the first relationships between argumentation semantics and logic programming semantics.

### 3.2 Complete Extensions as Clark's Completion Models

In this subsection we first will show that the complete semantics can be understood in terms of the completion semantics, since the models of Clark's completion correspond to supported models, which lead us to our result.

Our characterization will take as starting point an characterization of complete extensions in terms of *logic models* introduced in [4].

**Proposition 1.** *[4] Let $\langle AR, attacks\rangle$ be an argument framework. A set $S \subseteq AR$ is a complete extension iff $S$ is a model of the formula*

$$\bigwedge_{a \in AR} ((a \rightarrow \bigwedge_{b:(b,a)\in attacks} \neg b) \wedge (a \leftrightarrow \bigwedge_{b:(b,a)\in attacks} (\bigvee_{c:(c,b)\in attacks} c))).$$

In order to introduce our results, let us introduce the following notation.

**Definition 8.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR$, then $m(A) = \{acc(x) \mid x \in A\} \cup \{def(x) \mid x \in AR \setminus A\}$.*

The appendix provides more insight about the main points of this section by means of an illustrative example, which lead us to the definition of three lemmas that support the proof of the following theorem:

**Theorem 2.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR$, $A$ is a complete extension of $AF$, iff $m(A)$ is a model of $comp(\Pi^{acc}_{AF})$.*

*Proof. See appendix in order to get a complete picture of the proof.*

### 3.3 Final Connection

Now that we have characterized complete extensions in terms of Clark's completion models, we can extend Theorem 1 as follows:

**Theorem 3.** *Let $AF = \langle AR, attacks \rangle$ be an argumentation framework and $E \subseteq AR$. Then:*

- *$E$ is the grounded extension of $AF$ iff $\langle tr^{acc}(E) \cup tr^{def}(E^+), tr^{acc}(E^+) \cup tr^{def}(E) \rangle$ is the well-founded model of $\Pi_{AF}^{acc}$.*
- *$E$ is a stable extension of $AF$ iff $tr(E)$ is a stable model of $\Pi_{AF}^{acc}$.*
- *$E$ is a preferred extension of $AF$ iff $tr(E)$ is a p-stable model of $\Pi_{AF}^{acc}$.*
- *$E$ is a complete extension of $AF$ iff $tr(E)$ is a Clark's completion model of $\Pi_{AF}^{acc}$.*

*Proof. The proof of Theorem 3 is direct by Theorem 1 and Theorem 2.*

Note that Theorem 3 uses the same program used for the characterization of the different semantics introduced in [9] and some particular logic programming semantics with *negation as failure*. Hence, $\Pi_{AF}^{acc}$ basically identifies an *isomorphism* between Dung's argumentation semantics and logic programming semantics.

## 4 Conclusions and Future Work

Since Dung introduced his abstract argumentation approach, he proved that his approach can be regarded as a special form of logic programming with *negation as failure*. In fact, he showed that the grounded and stable semantics can be characterized by the well-founded and stable models semantics respectively. This result is important because it defined a general method for generating metainterpreters for argumentation systems and regards argumentation semantics as non-monotonic reasoning inferences.

In this paper we give a complete characterization of Dung's argumentation semantics which were introduced in [9] in terms of logic programming semantics. By considering an argumentation framework $AF$ and a unique mapping of $AF$ into a logic program $\Pi_{AF}^{acc}$, we show that:

- the well founded model of $\Pi_{AF}^{acc}$ characterizes the grounded extension of $AF$ (Theorem 3),
- the stable models of $\Pi_{AF}^{acc}$ characterize the stable extensions of $AF$ (Theorem 3),
- the p-stable models of $\Pi_{AF}^{acc}$ characterize the prefer extension of $AF$ (Theorem 3) and
- the supported models of $\Pi_{AF}^{acc}$ characterize the complete extensions of $AF$ (Theorem 3).

The characterization of argumentation semantics in terms of logic programming semantics does not only contribute in the inference of the well-accepted argumentation semantics but also this approach contributes to study the non-monotonic reasoning properties of the argumentation semantics. Observe that this kind of results also help to understand the close relationship between two successful approaches of non-monotonic reasoning: argumentation theory and logic programming with negation as

failure. Hence, one interesting issue in argumentation research is to explore new characterizations of argumentation semantics in terms of logic programming semantics.

It is known that Dung's semantics have been shown to range from NP-complete to $\Pi_2^{(p)}$ complete [11]. Hence to identify efficient methods for computing them is a *hot topic*. In this concern, as an application of the results of this paper, we are exploring the use of mixed integer programming methods [2] for computing complete extensions of an argumentation framework. Indeed we are comparing mixed integer programming methods and answer set solvers for computing argumentation semantics. It is worth mentioning that in [2], a method for using integer programming to compute the Clark's completion models of a logic program was defined. Hence, this method takes advantage of the results of this paper for computing the complete semantics. The results of this study will be reported in a future paper.

## Acknowledgment

This research has been partially supported by VINNOVA (The Swedish Governmental Agency for Innovation Systems), the Swedish Brain Power and CONACyT (the Mexican Governmental Agency for science and technology) [CB-2008-01 No.101581].

## References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
2. C. Bell, A. Nerode, R. T. Ng, and V. S. Subrahmanian. Mixed integer programming methods for computing nonmonotonic deductive databases. *Journal of the ACM*, 41(6):1178–1215, 1994.
3. T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.
4. P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004),*, pages 59–64, June 2004.
5. J. L. Carballido, J. C. Nieves, and M. Osorio. Inferring Preferred Extensions by Pstable Semantics. *Iberoamerican Journal of Artificial Intelligence (Inteligencia Artificial) ISSN: 1137-3601, (doi: 10.4114/ia.v13i41.1029)*, 13(41):38–53, 2009.
6. K. L. Clark. *Logic and Databases*, chapter Negation as Failure, pages 293–322. Plenum Press, 1978.
7. J. Dix. A classification theory of semantics of normal logic programs: I. strong properties. *Fundam. Inform.*, 22(3):227–255, 1995.
8. J. Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundam. Inform.*, 22(3):257–288, 1995.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
10. P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.
11. P. E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artificial Intelligence*, 171(10-15):701–729, 2007.

12. U. Egly, S. A. Gaggl, and S. Woltran. Aspartix: Implementing argumentation frameworks using answer-set programmin. In M. G. de la Banda and E. Pontelli, editors, *International Conference of Logic Programming (ICLP)*, volume 5366 of *Lecture Notes of Computer Science*, pages 734–738. Springer, 2008.
13. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
14. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
15. J. C. Nieves and M. Osorio. Ideal extensions as logic programming models. Submitted paper to a journal, 2013.
16. J. C. Nieves, M. Osorio, and U. Cortés. Preferred Extensions as Stable Models. *Theory and Practice of Logic Programming*, 8(4):527–543, July 2008.
17. M. Osorio, J. A. Navarro, J. R. Arrazola, and V. Borja. Logics with Common Weak Completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.
18. I. Rahwan and G. R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.
19. J. S. Schlipf. Formalizing a logic for logic programming. *Ann. Math. Artif. Intell.*, 5(2-4):279–302, 1992.
20. Y. Wu, M. Caminada, and D. M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, 93(2-3):383–403, 2009.

# A Appendix

Let us illustrate the proof of our main results using the argumentation framework introduced in Figure 2 from Example 2 and by five steps; additionally, we will state three lemmas and their respective proofs which support the proof of Theorem 2.

### Illustrative Example

**Step 1:** The argumentation framework from Figure 2 is instantiated with the propositonal formula from Proposition 1 considering just the atom $a_1$.

$$(a_1 \rightarrow (\sim b_1 \wedge \sim b_2) \wedge (a_1 \leftrightarrow (c_1 \vee c_2) \wedge (c_3 \vee c_4)))$$

Now for illustration purposes, it will be better to use the same notation as in previous examples, thus the previous formula becomes:

$$(acc(a_1) \rightarrow (\sim acc(b_1) \wedge \sim acc(b_2)) \wedge (acc(a_1) \leftrightarrow (acc(c_1) \vee acc(c_2)) \wedge (acc(c_3) \vee acc(c_4)))$$

**Step 2:** After decomposing the formula we will add a third one to get a conservative extension theory:

$\{acc(a_1) \rightarrow \sim acc(b_1) \wedge \sim acc(b_2),$
$acc(a_1) \leftrightarrow ((acc(c_1) \vee acc(c_2)) \wedge (acc(c_3) \vee acc(c_4)),$
$def(a_1) \leftrightarrow \sim acc(a_1)\}$

**Step 3:** which is logically equivalent to the following set of formulas

$$\{ \sim def(a_1) \rightarrow def(b_1) \wedge def(b_2),$$
$$\sim def(a_1) \leftrightarrow (\sim def(c_1) \vee \ \sim def(c_2)) \wedge (\sim def(c_3) \vee \ \sim def(c_4)),$$
$$acc(a_1) \leftrightarrow \sim def(a_1)\}$$

**Step 4:** Using De Morgan's laws and basic logical manipulations, a set of formulas that represent the argumentation framework's complete extensions are obtained (we will drop braces and commas for clarity purposes):

$$def(a_1) \leftarrow \sim def(b_1) \vee \ \sim def(b_2) \quad \text{(I)}$$
$$def(a_1) \leftarrow (def(c_1) \wedge def(c_2)) \vee (def(c_3) \wedge def(c_4)) \quad \text{(II)}$$
$$def(a_1) \rightarrow (def(c_1) \wedge def(c_2)) \vee (def(c_3) \wedge def(c_4)) \quad \text{(III)}$$
$$acc(a_1) \leftrightarrow \sim def(a_1) \quad \text{(IV)}$$

**Step 5:** From supported models definition, we know that the supported models of a program P correspond to the models of $Comp(P)$; therefore, we will calculate the $Comp_{a_1}(\Pi^{acc}_{AF})$ from the same example and we get:

$$def(a_1) \leftrightarrow \sim def(b_1) \vee \ \sim def(b_2) \vee (def(c_1) \wedge def(c_2)) \vee (def(c_3) \wedge def(c_4))$$

$$acc(a_1) \leftrightarrow \sim def(a)$$

and decomposing the above formulas we have:

$$def(a_1) \rightarrow \sim def(b_1) \vee \sim def(b_2) \vee (def(c_1) \wedge def(c_2)) \vee (def(c_3) \wedge def(c_4)) \quad (1)$$
$$def(a_1) \leftarrow \sim def(b_1) \vee \sim def(b_2) \vee (def(c_1) \wedge def(c_2)) \vee (def(c_3) \wedge def(c_4)) \quad (2)$$
$$acc(a_1) \leftrightarrow \sim def(a_1) \quad (3)$$

So far, we have a set of arguments (I,. . . ,IV) that represents a complete extension of a given theory if and only if this set is a model of the formula from Proposition 1 with respect to the argumentation framework from Example 2. On the other hand, we have computed $Comp_{a_1}(\Pi^{acc}_{AF})$ of the same argumentation framework and we got the formulas (1, 2, 3) which represent its supported models. Now we will see if they are equivalent:

We can see that 3 is equivalent to IV. 2 implies I, II. I, II imply 2. III implies 1, then we just need to prove that 1 implies III; to this end, we require to consider the completion of the arguments that attack $a_1$, namely $b_1$ and $b_2$, which are respectively:

$$def(b1) \leftrightarrow \sim def(c1) \vee \sim def(c2) \vee \top.$$
$$def(b2) \leftrightarrow \sim def(c3) \vee \sim def(c4) \vee \top.$$

thus, Formula 1 supported with the added rules can prove III even in the case where $\top$ was not present within the disjunctions. Now we are ready to formalize what this example has pointed out.

**Formal Proof**

**Definition 9.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, and $a \in AR$. Let $\Gamma_{aAF}$ be*

$$\{ \bigcup_{a\in AR} acc(a) \rightarrow \bigwedge_{b:(b,a)\in attacks} \sim acc(b)\} \bigcup$$
$$\{ \bigcup_{a\in AR} acc(a) \leftrightarrow \bigwedge_{b:(b,a)\in attacks} \bigvee_{c:(c,b)\in attacks} acc(c)\} \bigcup$$
$$\{ \bigcup_{a\in AR} def(a) \leftrightarrow \sim acc(a)\}$$

**Lemma 1.** *A set $A$ is a complete extension iff $m(A)$ is a model of $\Gamma_{aAF}$.*

*Proof. Applying the formula of Proposition 1 to an Argumentation Framework $AF$, considering just argument $a \in AR$, and by adding a third formula we got a conservative extension, which then was generalizated to get $\Gamma_{aAF}$; therefore, if $S \subseteq A$ is a model of the first formula then $A$, defined as in Definition 8, will also be a model of $\Gamma_{aAF}$, and also it will be a complete extension.*
*Note that this proof was illustrated in the first and second steps in our illustrative example.*

**Lemma 2.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, and $a \in AR$. Then the following set of formulas*

$$\left\{ acc(a) \rightarrow \bigwedge_{b:(b,a)\in attacks} \sim acc(b) \right\} \bigcup$$
$$\left\{ acc(a) \leftrightarrow \bigwedge_{b:(b,a)\in attacks} \bigvee_{c:(c,b)\in attacks} acc(c) \right\} \bigcup$$
$$\{def(a) \leftrightarrow \sim acc(a)\}$$

*is logically equivalent to $\Gamma'_{aAF}$ defined as:*

$$\left\{ def(a) \leftarrow \bigvee_{b:(b,a)\in attacks} \sim def(b) \right\} \bigcup \quad \textit{(I)}$$
$$\left\{ def(a) \leftarrow \bigvee_{b:(b,a)\in attacks} \left( \bigwedge_{c:(c,b)\in attacks} def(c) \right) \right\} \bigcup \quad \textit{(II)}$$
$$\left\{ def(a) \rightarrow \bigvee_{b:(b,a)\in attacks} \left( \bigwedge_{c:(c,b)\in attacks} def(c) \right) \right\} \bigcup \quad \textit{(III)}$$
$$\{acc(a) \leftrightarrow \sim def(a)\} \quad \textit{(IV)}$$

*Proof. We used transposition to the first unnumbered formula to get $(I)$, and De Morgan's laws to the second unnumbered formula to get $(II)$ and $(III)$, and the third unnumbered formula is the same one than $(IV)$.*
*Note that this proof was illustrated in steps three to four in our illustrative example.*

**Lemma 3.** *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $\Gamma_{AF}$ as defined in Definition 9, then $\Gamma_{AF}$ is logically equivalent to $Comp(\Pi^{acc}_{AF})$.*

*Proof. First of all note that from Lemma 2 we know that $\Gamma_{AF}$ is equivalent to $\Gamma^{'}_{AF}$, then we will show that $\Gamma^{'}_{AF} \models \alpha$ where $\alpha \in Comp(\Pi^{acc}_{AF})$ for now is enough to note that for each $a \in AR, \Gamma^{'}_{aAF} \models Comp_a(\Pi^{acc}_{AF})$, where $\Gamma^{'}_{aAF}$ denotes its respective formulas w.r.t. the argument $a$ and $Comp_a(\Pi^{acc}_{AF_a})$ denotes its respective formulas w.r.t. the argument $a$.*

*Now, consider $\Gamma^{'}_{aAF}$ as defined in Lemma 2 and the following generalization of $Comp_a(\Pi^{acc}_{AF})$ as it was illustrated in Step 5 in Example 3:*

$$\left\{def(a) \rightarrow \bigvee_{b:(b,a)\in attacks} \sim def(b) \bigvee \left(\bigvee_{b:(b,a)\in attacks} \bigwedge_{c:(c,b)\in attacks} def(c)\right)\right\} \bigcup \quad (1)$$

$$\left\{def(a) \leftarrow \bigvee_{b:(b,a)\in attacks} \sim def(b) \bigvee \left(\bigvee_{b:(b,a)\in attacks} \bigwedge_{c:(c,b)\in attacks} def(c)\right)\right\} \bigcup \quad (2)$$

$$\{acc(a) \leftrightarrow \sim def(a)\} \quad (3)$$

*Note the following fact about these formulas: $3 \leftrightarrow IV$. $2 \models I, II$. $I, II \models 2$. $III \models 1$. Now we need to prove that $1 \models III$, and we can do it by considering $Comp_b(\Pi^{acc}_{AF})$ where $b : (b, a) \in attacks$, and doing so, we have two cases:*

1. *$d^{-}_{AF}(a) = 0$[3]. The argument $a$ is not attacked by any other one. In this case we have that $\Gamma_{bAF}$ is: $\{acc(a) \leftrightarrow \sim d(a),\ \sim d(a) \rightarrow \bot\}$, and on the other hand we have that $Comp_b(\Pi^{acc}_{AF})$ is: $\{acc(a) \leftrightarrow \sim d(a),\ \sim d(a) \leftarrow \top\}$, which are clearly equivalent.*

2. *$d^{-}_{AF}(a) \neq 0$. The argument $a$ do is attacked by other arguments. In this case we have that $Comp_a(\Pi^{acc}_{AF}) \cup Comp_b(\Pi^{acc}_{AF}) \models III$.*

*Note that the idea of this proof was illustrated in step five in our illustrative example.*

*Proof of Theorem 2*

*We know by Lemma 1 that $A$ is a complete extension of $\Gamma_{aAF}$, and we know by Lemma 3 that $\Gamma_{AF}$ is equivalent to $\Pi^{acc}_{AF}$, therefore $A$ is also a model of $\Pi^{acc}_{AF}$.*

[3] Borrowing some notation from graph theory. Bondy, Adrian, Murty, U.S.R., Graph Theory, Springer, 2008