ARTÍCULO

Inferring Preferred Extensions by Pstable Semantics

[1]**José Luis Carballido** [2]**Juan Carlos Nieves**

[3]**Mauricio Osorio***

[1]Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias Físico Matemáticas, Puebla, México
carballido@fcfm.uap.buap.mx

[2]Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu

[3]Universidad de las Américas - Puebla, CENTIA,
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@gmail.com

Abstract

When Dung introduced his argumentation approach, he proved that it 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 model semantics respectively. However, Dung did not give any characterization of the preferred semantics in terms of logic programming semantics. In order to extend Dung's results, we will show that there exists a codification of an argumentation framework in terms of a logic program able to infer the *grounded*, *stable* and *preferred semantics* by considering the *well-founded*, *stable-model* and *pstable semantics* respectively. This result unifies, in logic programming with negation as failure, the three principal argumentation semantics of Dung's approach. Also the characterization of the preferred semantics by the pstable semantics suggests a new perception of the preferred semantics in terms of *logic foundations*.

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, 23, 22, 4]. 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.

Although several approaches have been proposed for capturing representative patterns of inference in argumentation theory, Dung's approach, presented in [7], is a unifying framework which has played an influential role on argumentation research and AI. In fact the model suggested by Dung has given rise to an extensive body research with particular concentration on the following, [3]:

*The authors are named by alphabetic order.

- Extension based semantics of argumentation.
- Algorithmic and complexity issues in argumentation.
- Dialogue processes for deciding acceptability.

Dung's approach is regarded as an abstract model where the main concern is to find the set of arguments which are considered as acceptable *i.e.*, to find sets of arguments which represent coherent points of view. The strategy for analyzing the attack relationships, and then inferring the sets of acceptable arguments, is based on extension based semantics.

The kernel of Dung's framework is supported by four extension based semantics (we will refer to them also as abstract argumentation semantics): *grounded semantics* (GS), *stable semantics* (SS), *preferred semantics* (PS), and *complete semantics* (CS). Although each abstract argumentation semantics represents a different pattern of inference in argumentation theory, all of them have as common point the concept of *admissible set*[1]. In fact the grounded, stable, preferred and complete semantics are nothing else than a set of admissible sets *i.e.*, a set of sets of arguments. An argumentation semantics can be regarded as mappings from an argumentation framework into a set of sets of arguments.

Regarding arguments as abstract concepts, as it is done in Dung's approach [7], is a powerful tool which affords a formalism that focuses on the relationship between individual arguments as a means of defining diverse ideas of acceptance. Some authors have pointed out that the grounded, stable and preferred semantics are of particular interest since they capture representative patterns of inference in argumentation theory [8, 2, 3].

When 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*. This result has at least two main implications:

1. It defines a general method for generating metainterpreters for argumentation systems and
2. it defines a general method for studying abstract argumentation semantics' properties in terms of logic programming semantics' properties.

Possibly, the two main challenges of studying Dung's approach in terms of logic programming are:

1. To find suitable logic programming codifications able to map an argumentation framework AF into a logic program P such that these codifications are polynomial time computable, and
2. To find suitable logic programming semantics able to capture the different patterns of inference of the argumentation semantics.

It is worth to comment that any logic programming semantics S can be regarded as a mapping from a logic program P into a set H of sets of literals, such that for each set of literals L in H, $P \cup L$ is consistent (in the strict sense of classical logic). We call each of the above sets of literals (such as L) a partial model of the program P.

In [7], Dung defined a logic programming codification (P_{AF}) in order to map an argumentation framework into a logic program. In fact by considering P_{AF}, Dung showed that the well-founded semantics (WFS) [9] is a proper logic programming semantics to capture the grounded semantics and the stable model semantics [10] is a proper logic programming semantics to capture the stable semantics. However, to the best of our knowledge, there does not exist a logic programming semantics able to capture the preferred semantics by using P_{AF}. Recently, in [15], it was shown that the preferred semantics can be captured by the minimal models of a propositional formula ($\alpha(AF)$). However, the minimal models of a propositional formula do not suggest at all a logic programming semantics with negation as failure.

In order to unify, in logic programming with negation as failure, the three principal argumentation semantics of Dung's approach, we identify the following problem:

Problem 1 *Do there exist three logic programming semantics $S1$, $S2$, $S3$, a logic program-*

[1] An admissible set represents a coherent point of view in a conflict of arguments.

ming codification f and a polynomial time computable function h, such that for every argumentation framework AF the following three conditions hold:

1. *$h(S1(f(AF))) = GS(AF)$*
2. *$h(S2(f(AF))) = SS(AF)$*
3. *$h(S3(f(AF))) = PS(AF)$?*

By Theorem 17 of [7], we can say that the well-founded semantics can be an instance of $S1$ and the stable model semantics can be an instance of $S2$. However, to the best of our knowledge, an instance of $S3$ has not been identified, neither has $f(AF)$.

In this paper, we will identify a complete answer to the question presented in Problem 1. We will show that there exists a logic program $f(AF) = \Psi_{AF}$ that represents an argumentation framework AF such that

- its well-founded model characterizes the grounded extension of AF;
- its stable models characterize the stable extensions of AF; and
- its pstable models characterize the preferred extensions of AF.

It is worth to comment that, to the best of our knowledge, this is the first time that by using *the same logic program*, the three main argumentation semantics of Dung's approach *i.e.*, the grounded, stable and preferred semantics, are captured.

Observe that the characterization of the preferred semantics by the pstable semantics suggests a new perception of the preferred semantics. The pstable semantics is a recently introduced logic programming semantics which is inspired in paraconsistent logics [21]. In fact, this semantics can be characterized by paraconsistent logics as the Cw and G'_3 logics [18]. The pstable semantics can be also characterized by modal logics as the S5 modal logic [20]. One important property of the pstable semantics is that by considering a simple transformation of a disjunctive logic program P_D into a normal logic program P_N, the pstable semantics of P_N corresponds to the stable model semantics of P_D over the common language [17].

The rest of the paper is divided as follows: In §2, some basic definitions of logic programming, pstable semantics and stable model semantics are presented; moreover a brief description of Dung's approach is presented. In §3, a mapping of an argumentation framework into a logic program is defined. In §4, we present our main results. In the last section, we present our conclusions.

2. Background

In this section, we define some basic concepts of logic programming, pstable semantics, stable model semantics and Dung's argumentation approach. We assume familiarity with basic concepts in classical logic and with semantics of logic programs *e.g.,* interpretation, model, *etc.* A good introductory treatment of these concepts can be found in [1, 14].

2.1. Logic programs: Syntax

A signature $\mathcal{L}$ is a finite set of elements that we call atoms. A literal is an atom, a (positive literal), or the negation of an atom $\neg a$ (negative literal). 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 disjunctive clause is a clause of the form:

$$a_1 \vee \cdots \vee a_m \leftarrow a_{m+1}, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_n$$

where a_i is an atom, $1 \leq i \leq n$. When $n = m$ and $m > 0$, the disjunctive clause is an abbreviation of the fact $a_1 \vee \cdots \vee a_m$. When $m = 0$ and $n > 0$ the clause is an abbreviation of

$$\bot \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_n$$

where $\bot$ is an atom that always evaluate to false. Clauses of this form are called constraints (the rest non-constraints). A disjunctive logic program is a finite set of disjunctive clauses. Sometimes, we denote a clause C by $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$, where $\mathcal{A}$ contains all the head atoms, $\mathcal{B}^+$ contains all the positive body literals and $\mathcal{B}^-$ contains all the negative body literals. We also use $body(C)$ to denote $\mathcal{B}^+, \neg\mathcal{B}^-$. When $\mathcal{B}^- = \emptyset$, the clause C is called a positive disjunctive clause. A set of positive disjunctive clauses is called positive disjunctive logic program. When $\mathcal{A}$ is a singleton set, the clause can be regarded as a normal clause. A normal program is a finite set of normal clauses. Also,

when $\mathcal{A}$ is a singleton set and $\mathcal{B}^- = \emptyset$, the clause can be regarded as a definite clause. A finite set of definite clauses is called a definite logic program.

We denote by $\mathcal{L}_P$ the signature of P, *i.e.*, the set of atoms that occur in P. Given a signature $\mathcal{L}$, we write $Prog_{\mathcal{L}}$ to denote the set of all the programs defined over $\mathcal{L}$.

2.2. Non Monotonic Reasoning: the pstable and stable semantics

In order to define the pstable semantics (introduced in [21]), we define some basic concepts. Logical inference in classical logic is denoted by $\vdash$. Given a set of proposition symbols S and a theory (a set of well-formed formulæ) Γ, $\Gamma \vdash S$ if and only if $\forall s \in S\ \Gamma \vdash s$. When we treat a logic program as a theory, each negative literal $\neg a$ is regarded as the standard negation operator in classical logic. Given a normal program P, if $M \subseteq \mathcal{L}_P$, we write $P \Vdash M$ when: $P \vdash M$ and M is a classical 2-valued model of P (*i.e.*, atoms in M are set to true, and atoms not in M to false; the set of atoms M is a classical model of P if the induced interpretation evaluates P to true). A model M of P is called minimal if there does not exist a model M' of P such that $M' \subset M$.

The pstable semantics was defined in terms of a single reduction which is defined as follows:

Definition 1 *[21] Let P be a normal program and M be a set of literals. We define*

$$RED(P, M) := \{a \leftarrow \mathcal{B}^+, \neg(\mathcal{B}^- \cap M) | \\ a \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P\}$$

Observe that M is model of P if and only if M is a model of $RED(P, M)$. For instance, let us consider the set of atoms $M_1 := \{a, b\}$ and the following normal program P_1:

$a \leftarrow \neg\, b, \neg\, c.$
$a \leftarrow b.$
$b \leftarrow a.$

We can see that $RED(P_1, M)$ is:

$a \leftarrow \neg b.$
$a \leftarrow b.$
$b \leftarrow a.$

As we can see, M_1 is a model of P and also of $RED(P, M)$. By considering the reduction RED, the *pstable* semantics for normal programs is defined as follows:

Definition 2 *[21] Let P be a normal program and M be a set of atoms. We say that M is a pstable model of P if $RED(P, M) \Vdash M$. We use pstable to denote the semantics operator of pstable models.*

Let us consider again M_1 and P_1 in order to illustrate the definition. We want to verify whether M_1 is a pstable model of P_1. First, we can see that M_1 is a model of P_1, *i.e.*, $\forall\ c \in P_1$, M_1 evaluates c to true. Now, we have to prove each atom of M_1 from $RED(P_1, M_1)$ by using classical inference, *i.e.*, $RED(P_1, M_1) \vdash M_1$. Let us consider the proof of the atom a, which belongs to M_1, from $RED(P_1, M_1)$.

1. $(\neg b \rightarrow a) \rightarrow ((b \rightarrow a) \rightarrow a)$
   Tautology
2. $\neg b \rightarrow a$
   Premise from $RED(P_1, M_1)$
3. $(b \rightarrow a) \rightarrow a$
   From 1 and 2 by Modus Ponens
4. $b \rightarrow a$
   Premise from $RED(P_1, M_1)$
5. a
   From 3 and 4 by Modus Ponens

Remember that the formula $\neg b \rightarrow a$ corresponds to the normal clause $a \leftarrow \neg b$ which belongs to the program $RED(P_1, M_1)$. The proof for the atom b, which also belongs to M_1, is similar. Then we can conclude that $RED(P_1, M_1) \Vdash M_1$. Hence, M_1 is a *pstable model* of P_1.

An important property of any pstable model, which was proved in [21], is that any pstable model is a minimal model.

Theorem 1 *Let P be a normal program, and M be a set of atoms. If M is a pstable model of P then M is a minimal model of P.*

Observe that the oppositive of this theorem does not hold *i.e.*, a minimal model might not be a pstable model. For instance, let P be the following program:

$$a \leftarrow \neg b$$

We can see that $\{b\}$ is a minimal model but not a pstable model of P.

The well known *stable model semantics* (see [10, 11]) is defined as follows.

Let P be any disjunctive program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the positive logic 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 formulæ of the form $\neg l$ in the bodies of the remaining rules.

Clearly P^S does not contain $\neg$. Then S is a stable model of P if S is a minimal model of P^S.

In order to illustrate this definition let us consider the following example:

Example 1 *For instance, let $S = \{b\}$ and P be the following logic program:*

$b \leftarrow \neg a.$ $\quad$ $b.$
$c \leftarrow \neg b.$ $\quad$ $c \leftarrow a.$

We can see that P^S is:

$b \leftarrow \top.$ $\quad$ $c \leftarrow a.$

Notice that P^S has three models: $\{b\}$, $\{b, c\}$ and $\{a, b, c\}$. Since the minimal model amongst these models is $\{b\}$, we can say that S is a stable model of P.

Observe that we have defined the pstable semantics for normal logic programs and the stable model semantics for disjunctive logic programs. One important property of the pstable semantics is that by considering a simple transformation of a disjunctive logic program P_D into a normal logic program P_N, the pstable semantics of P_N corresponds to the stable model semantics of P_D over a common language. This result was introduced in [17] and it is formalized as follows:

Theorem 2 *[17] Let P be a disjunctive logic program and M be a set of atoms of $\mathcal{L}_P$. Then there exists a normal program $Trans(P)$ that depends on P, such that M is a stable model of P if and only if there exists a pstable model N of $Trans(P)$ such that $M = N \cap \mathcal{L}_P$.*

That is to say: $Trans(P)$ is a mapping that transforms any disjunctive logic program into a normal logic program. The details of this mapping can be found in [17].

2.3. Argumentation theory

Now, we will define some basic concepts of Dung's argumentation approach. Mainly, we will define the grounded, stable and preferred semantics, and present Dung's result about how to regard his argumentation approach as logic programming with negation as failure.

The basic structure of Dung's argumentation approach is an argumentation framework. An argumentation framework captures the relationships between the arguments (all the definitions of this subsection were taken from the seminal paper [7]).

Definition 3 *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., *$attacks \subseteq AR \times AR$.*

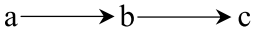

Figure 1. Graph representation of $AF := \langle \{a, b, c\}, \{(a, b), (b, c)\} \rangle$

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

Definition 4 *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.*

Dung defined his semantics based on the basic concept of *admissible set.*

Definition 5 *(1) 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. (2) A conflict-free set of arguments S is admissible if and only if each argument in S is acceptable w.r.t. S.*

For instance, the argumentation framework of Figure 1 has two admissible sets: $\{a\}$ and $\{a, c\}$.

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

Definition 6 *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF. The set of preferred extensions of AF will be referred as the preferred semantics of AF.*

The only preferred extension of the argumentation framework of Figure 1 is $\{a, c\}$. The grounded semantics is defined in terms of a *characteristic function.*

Definition 7 *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 8 *The grounded extension of an argumentation framework AF, denoted by GE_{AF}, is the least fixed point of F_{AF}.*

In order to illustrate the definition, let us consider the argumentation framework of Figure 1. Then

$$\begin{array}{ll} F^0_{AF}(\emptyset) := & \{a\}, \\ F^1_{AF}(F^0_{AF}(\emptyset)) := & \{a, c\}, \\ F^2_{AF}(F^1_{AF}(F^0_{AF}(\emptyset))) := & \{a, c\}, \end{array}$$

since $F^1_{AF}(F^0_{AF}(\emptyset)) = F^2_{AF}(F^1_{AF}(F^0_{AF}(\emptyset)))$, then $GE_{AF} = \{a, c\}$. Therefore the grounded extension of AF is $\{a, c\}$. We want to point out that one can insure that F_{AF} reaches a fixed-point by the fact that F_{AF} is monotonic and Tarsky's Lattice-Theoretical Fixpoint Theorem [25].

Another interesting semantics which was introduced in [7] is the *stable semantics.*

Definition 9 *A conflict-free set of arguments S is called a stable extension if and only if S attacks each argument which does not belong to S. The set of stable extensions of AF will be referred as the stable semantics of AF.*

Dung showed that this semantics coincides with the notion of stable solutions of n-person games [7].

There is an interesting relationship between the stable semantics and the preferred semantics which is that every stable extension is a preferred extension, but not vice versa. Let us consider the following example.

Example 2 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, such that $AR := \{a, b, c, d, e\}$ and $attacks := \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}$ (see Figure 2).*

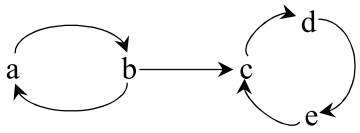

Figure 2. Graph representation of the argumentation framework $AF := \langle\{a, b, c, d, e\}, \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}\rangle$.

As we can see, AF has just one stable extension $S_1 = \{b, d\}$; however, AF has two preferred extensions: S_1 and $S_2 = \{a\}$.

Remark 1 *For the rest of the paper, we will say that an argument a is "*acceptable*" if a belongs to E such that E is a preferred, stable or the grounded extension. Hence an argument attacked by a will be called "*defeated*".*

In [7], Dung suggested a general method for generating metainterpreters in terms of logic programming for argumentation systems. This is the first approach which regards an argumentation framework as a logic program. 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) $acc(x) \leftarrow \neg d(x)$

(C2) $d(x) \leftarrow attacks(y, x), acc(y)$

The first one (C1) suggests that the argument x is acceptable if it is not defeated, and the second one (C2) suggests that an argument is defeated if it is attacked by an acceptable argument[2]. The predicate $d(x)$ denotes that "x is a defeated argument".

Definition 10 *Given an argumentation framework $AF = \langle AR, attacks\rangle$, P_{AF} denotes the logic program defined by $P_{AF} = APU + AGU$ where $APU = \{C1, C2\}$ and*

$$AGU = \{attacks(a, b) | (a, b) \in attacks\}$$

For each extension E of AF (namely a set of arguments), $m(E)$ is defined as follows:

$$m(E) = AGU \cup \{acc(a) | a \in E\} \cup \{d(b) | b \text{ is attacked by some } a \in E\}$$

The following theorem is the first result that suggests a clear relationship between two[3] argumentation semantics and two logic programming semantics.

Theorem 3 *[7] Let AF be an argumentation framework and E be an extension of AF. Then*

1. *E is a stable extension of AF if and only if $m(E)$ is a stable model of P_{AF}*
2. *E is a grounded extension of AF if and only if $m(E) \cup \{\neg d(A) | A \in E\}$ is the well-founded model of P_{AF}*

3. Mapping from argumentation frameworks to logic programs

As we commented in the previous section, Dung defined a mapping (P_{AF}) in order to regard argumentation frameworks as logic programs. In fact, by Theorem 3, one can see that P_{AF} is able to characterize the grounded and stable semantics; however, to the best of our knowledge, this mapping is unable to characterize the preferred semantics by considering a logic programming semantics with negation as failure.

In this section, we will present an alternative mapping in order to regard an argumentation framework as a logic programming. We will see that this mapping is really close to P_{AF}; however, we will show that this unique mapping is able to give answer to Problem 1 (presented in the introduction).

Like P_{AF}, our mapping will use the predicate $d(x)$, where the intended meaning of $d(x)$ is "x is a defeated argument", moreover, it will use the predicate $acc(x)$, where the intended meaning of $acc(x)$ is "x is an acceptable argument". First, we define a transformation function *w.r.t.* an argument.

Definition 11 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $a \in AR$. We define the transformation function $\Psi(a)$ as follows:*

$$\Psi(a) = \bigcup_{b:(b,a)\in attacks} \{d(a) \leftarrow \neg d(b)\} \cup \bigcup_{b:(b,a)\in attacks} \{d(a) \leftarrow \bigwedge_{c:(c,b)\in attacks} d(c)\}$$

In the program $\Psi(a)$, we can identify two parts for each argument $a \in AR$:

1. The first condition of $\Psi(a)$, $\bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow \neg d(b)\}$, suggests that the argument a is defeated when any one of its adversaries is not defeated.
2. The second condition of $\Psi(a)$, $\bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow \bigwedge_{c:(c,b)\in attacks} d(c)\}$, suggests that the argument a is defeated when all the arguments that defend[4] a from one of its attackers b are defeated.

The direct generalization of the transformation function Ψ to an argumentation framework is defined as follows:

Definition 12 *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework. We define its associated normal program as follows:*

$$\Psi_{AF} := \bigcup_{a\in AR} \{\Psi(a) \cup \{acc(a) \leftarrow \neg d(a)\}\}$$

[2]Dung uses the predicate *defeat* instead of the predicate d

[3]Dung presented results *w.r.t.* another semantics, but we just cite the results *w.r.t.* the stable extensions and the grounded extension.

[4]We say that c defends a if there exists b such that b attacks a and c attacks b.

Example 3 *Let $AF := \langle AR, attacks\rangle$ be the argumentation framework of Figure 1. We can see that Ψ_{AF} is:*

$$\begin{array}{ll} d(b) \leftarrow \neg d(a). & d(b) \leftarrow \top. \\ d(c) \leftarrow \neg d(b). & d(c) \leftarrow d(a). \\ acc(a) \leftarrow \neg d(a). & acc(b) \leftarrow \neg d(b). \\ acc(c) \leftarrow \neg d(c). & \end{array}$$

Observe that Ψ_{AF} has no normal clauses with the atom $d(a)$ in their head. This is essentially because Ψ_{AF} is capturing the arguments which could be defeated and the argument a will be always an acceptable argument because it has no adversary. Now, let us consider the argumentation framework AF of Figure 2. We can see that Ψ_{AF} is:

$$\begin{array}{ll} d(a) \leftarrow \neg d(b). & d(a) \leftarrow d(a). \\ d(b) \leftarrow \neg d(a). & d(b) \leftarrow d(b). \\ d(c) \leftarrow \neg d(b). & d(c) \leftarrow \neg d(e). \\ d(c) \leftarrow d(a). & d(c) \leftarrow d(d). \\ d(d) \leftarrow \neg d(c). & d(d) \leftarrow d(b), d(e). \\ d(e) \leftarrow \neg d(d). & d(e) \leftarrow d(c). \\ acc(a) \leftarrow \neg d(a). & acc(b) \leftarrow \neg d(b). \\ acc(c) \leftarrow \neg d(c). & acc(d) \leftarrow \neg d(d). \\ acc(e) \leftarrow \neg d(e). & \end{array}$$

In this case, Ψ_{AF} has normal clauses for all the arguments of AF. This is because all the arguments of AF at least has an adversary.

3.1. Preferred extensions as stable models

In this section, we will present an important relation between the preferred semantics and the stable model semantics that was introduced in [15]. This result will be useful to formalize the fact that the pstable models of Ψ_{AF} characterize the preferred extensions of AF.

We start by defining a mapping function which is a variation of the mapping of Definition 11.

Definition 13 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework and $a \in AR$. We define the transformation function $\Gamma(a)$ as follows:*

$$\Gamma(a) = \bigcup_{b:(b,a)\in attacks}\{d(a) \vee d(b)\} \cup \bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow \bigwedge_{c:(c,b)\in attacks} d(c)\}$$

Now we define the function Γ in terms of an argumentation framework.

Definition 14 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework. We define its associated general program as follows:*

$$\Gamma_{AF} = \bigcup_{a\in AR} \Gamma(a)$$

In the following theorem, a characterization of the preferred semantics in terms of stable model semantics is formalized. Given an argumentation framework $AF = \langle AR, attacks\rangle$ and $E \subseteq AR$, $compl(E)$ is defined as $\{d(a)|a \in AR \setminus E\}$. Essentially, $compl(E)$ expresses the complement of E *w.r.t. AR.*

Theorem 4 *[15] Let $AF = \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $compl(S)$ is a stable model of Γ_{AF}.*

Example 4 *Let AF be the argumentation framework of Figure 2. We can see that Γ_{AF} is:*

$$\begin{array}{ll} d(a) \vee d(b). & d(a) \leftarrow d(a). \\ d(b) \vee d(a). & d(b) \leftarrow d(b). \\ d(c) \vee d(b). & d(c) \vee d(e). \\ d(c) \leftarrow d(a). & d(c) \leftarrow d(d). \\ d(d) \vee d(c). & d(d) \leftarrow d(b), d(e). \\ d(e) \vee d(d). & d(e) \leftarrow d(c). \end{array}$$

Γ_{AF} has two stable models which are $\{d(a), d(c), d(e)\}$ and $\{d(b), d(c), d(e), d(d))\}$, therefore $\{b, d\}$ and $\{a\}$ are the preferred extensions of AF.

4. Argumentation semantics as logic programming semantics with negation as failure

In order to present our main results, we need the following definition. For each extension E of AF (namely a set of arguments), $tr(E)$ is defined as

follows:

$$tr(E) = \{acc(a) | a \in E\} \cup \{d(b) | b \text{ is an argument and } b \notin E\}$$

Before proving that Ψ_{AF} is able to capture the stable, preferred and grounded semantics, we will present some interesting results *w.r.t. stable model equivalence*.

According to [12], P_1 is stable equivalent to P_2 if P_1 and P_2 have the same stable models. Also we can say that P_1 is *strongly equivalent* to P_2 if and only if for every logic program P, $P_1 \cup P$ and $P_2 \cup P$ have the same stable models. The intuitive idea of stable equivalence is that for any two programs which are stable equivalent we can replace one by the other in any large program without changing the stable models (declarative semantics). By having in mind this idea, we present the following lemmas:

Lemma 1 *Let P be a normal program and r be a normal rule of the form:*

$$y \leftarrow \neg y_1 \vee \cdots \vee \neg y_m$$

Suppose that y does not appear as the head of any rule of P. Then $P \cup \{r\}$ is stable-equivalent to $P \cup \{y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m\}$.

Proof: Let P and r be as defined in the conditions of the lemma. Let r' be $y \leftrightarrow \neg y_1 \vee \ldots \neg \vee y_m$. Let M be a set of atoms. Let $cl(M)$ denotes the complement of M *w.r.t.* the language of $P \cup \{r\}$. We will prove that M is a stable model of $P \cup \{r\}$ if and only if M is a stable model of $P \cup \{r'\}$. To this aim, we will freely use the characterization of stable models for arbitrary propositional formulæ in terms of *Gödel logic* G_3, see [19]. The proof is by cases:

(A) Suppose that there exists $y_i \in \{y_1, \ldots, y_m\}$ and $y_i \notin M$. Then $\{r\} \cup \neg cl(M) \cup \neg\neg M$ is equivalent in G_3 logic to $\{r'\} \cup \neg cl(M) \cup \neg\neg M$. Hence $P \cup \{r\} \cup \neg cl(M) \cup \neg\neg M$ is equivalent in G_3 logic to $P \cup \{r'\} \cup \neg cl(M) \cup \neg\neg M$. Thus, M is a stable model of $P \cup \{r\}$ if and only if M is a stable model of $P \cup \{r'\}$, as desired.

(B) Suppose on the other hand that $\{y_1, \ldots, y_m\} \subseteq M$. We have two subcases:

(B1) Suppose $y \in M$, then $\{r\} \cup \neg cl(M) \cup \neg\neg M$ is equivalent in G_3 logic to $\{\neg\neg y\} \cup \neg cl(M) \cup \neg\neg M$. Hence neither y nor $\neg y$ are provable in G_3 by $P \cup \{r\} \cup \neg cl(M) \cup \neg\neg M$ (here it is important the hypothesis that y does not appear as the head of any rule of P). So, M is not a stable model of $P \cup \{r\}$. It is easy to verify that $P \cup \{r'\}$ is inconsistent and so M is not a stable model of $P \cup \{r'\}$, as desired.

(B2) Suppose $y \notin M$, then $\{r\} \cup \neg cl(M) \cup \neg\neg M$ is equivalent in G_3 logic to $\{r'\} \cup \neg cl(M) \cup \neg\neg M$. Hence $P \cup \{r\} \cup \neg cl(M) \cup \neg\neg M$ is equivalent in G_3 logic to $P \cup \{r'\} \cup \neg cl(M) \cup \neg\neg M$. Thus, M is a stable model of $P \cup \{r\}$ if and only if M is a stable model of $P \cup \{r'\}$, as desired.

■

Lemma 2 *Let P be a normal program that includes the clause:*

$$x \leftarrow \neg y$$

Suppose in addition that all the rules in P with head equal to y are:

$$y \leftarrow \neg y_1. \quad \ldots \quad y \leftarrow \neg y_m.$$

Then P is stable-equivalent to $P \cup \{x \leftarrow y_1, \ldots, y_m\}$.

Proof: We will use freely one basic fact that is a direct consequence of the results in [12]. If two programs P_1 and P_2 are equivalent in *Gödel logic* G_3 then P_1 is strongly equivalent to P_2.

Suppose the hypothesis of the lema, namely that P is of the form:

$$y \leftarrow \neg y_1. \quad \ldots \quad y \leftarrow \neg y_m.$$
$$x \leftarrow \neg y.$$

and Q (that includes the rest of the rules).

Since $(\neg y_1 \rightarrow y) \wedge \cdots \wedge (\neg y_m \rightarrow y)$ is equivalent in G_3 to $\neg y_1 \vee \cdots \vee \neg y_m \rightarrow y$.

Then P is stable equivalent to P_1:

$$y \leftarrow \neg y_1 \vee \cdots \vee \neg y_m.$$
$$x \leftarrow \neg y.$$
$$Q.$$

By Lemma 1, P_1 is stable equivalent to P_2:

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m.$
$x \leftarrow \neg y.$
$Q.$

Since

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m$
$\neg y \rightarrow x$

is equivalent in G_3 to:

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m$
$\neg\neg y_1 \wedge \cdots \wedge \neg\neg y_m \rightarrow x$

Then P_2 is stable equivalent to P_3:

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m.$
$x \leftarrow \neg\neg y_1 \wedge \cdots \wedge \neg\neg y_m.$
$Q.$

Since $\neg\neg y_1 \wedge \cdots \wedge \neg\neg y_m \rightarrow x$ is equivalent in G_3 to:

$\neg\neg y_1 \wedge \cdots \wedge \neg\neg y_m \rightarrow x$
$y_1 \wedge \cdots \wedge y_m \rightarrow x$

We obtain that P_3 is stable equivalent to P_4:

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m.$
$x \leftarrow \neg\neg y_1, \ldots, \neg\neg y_m.$
$x \leftarrow y_1, \ldots, y_m.$
$Q.$

By similar reasoning as above we obtain the following chain of equivalences. P_4 is stable equivalent to P_5:

$y \leftrightarrow \neg y_1 \vee \cdots \vee \neg y_m.$
$x \leftarrow \neg y.$
$x \leftarrow y_1, \ldots, y_m.$
$Q.$

P_5 is stable equivalent to P_6:

$y \leftarrow \neg y_1 \vee \cdots \vee \neg y_m.$
$x \leftarrow \neg y.$
$x \leftarrow y_1, \ldots, y_m.$
$Q.$

P_6 is stable equivalent to P_7:

$y \leftarrow \neg y_1.$
$\ldots$
$y \leftarrow \neg y_m.$
$x \leftarrow \neg y.$
$x \leftarrow y_1, \ldots, y_m.$
$Q.$

Hence, by transitivity we obtain our desired result.

■

Let us note that the program Q in the previous proof is not arbitrary; it is the complement with respect to P of a set consisting of special rules in P. Therefore the lemma establishes the stable equivalence, but not strong stable equivalence, between two programs.

Another interesting result *w.r.t.* stable equivalence is formalized by the following lemma:

Lemma 3 *Let P be a normal program, let $r_1, r_2, ..., r_n$ be definite rules such that the two programs P and $P \cup r_i$ have the same stable models for any $i \in \{1, 2..., n\}$. Then P and $P \cup r_1 \cup r_2 \cup ... \cup r_n$ have the same stable models.*

Proof:

⇒ Let us assume that M is a stable model of P. By definition this means that M is a minimal classical model for P^M. Now by hypothesis M is a minimal classical model for each of the programs $(P \cup r_i)^M = P^M \cup r_i$, for each $i \in \{1, 2, ..., n\}$.
From this, it is clear that M is a classical model for $P^M \cup r_1 \cup r_2 \cup ... \cup r_n$. Any proper subset of M can not be a model of this last program since that would contradict our original assumption, hence M is minimal. Now the desired conclusion follows after observing that $P^M \cup r_1 \cup r_2 \cup ... \cup r_n = (P \cup r_1 \cup r_2 \cup ... \cup r_n)^M$.

⇐ Now let us assume that M is a minimal classical model of $(P \cup r_1 \cup r_2 \cup ... \cup r_n)^M = P^M \cup r_1 \cup r_2 \cup ... \cup r_n$. Then it is clear that M models P^M. Let us assume for a moment that M is not a minimal model of P^M; then there would exist a set $N \subsetneq M$ such that N is minimal model for P^M.
Then by hypothesis N is a minimal model of $(P \cup r_i)^M = P^M \cup r_i$ for $i \in \{1, 2, ..., n\}$. Hence N is a model for $P^M \cup r_1 \cup r_2 \cup ... \cup r_n$, that is to say: N is a model of $(P \cup r_1 \cup r_2 \cup ... \cup r_n)^M$. This contradicts the minimality of M and the result follows.

■

By considering Theorem 3, Lemma 2 and Lemma 3, we will show that the stable modes of Ψ_{AF} characterize the stable extensions of AF.

Theorem 5 *Let AF be an argumentation framework and E be a set of arguments. Then E is a stable extension of AF if and only if $tr(E)$ is a stable model of Ψ_{AF}.*

Proof: Let P be the grounding of the program P_{AF} (see §2.3). Hence, P is of the form:

$P = \bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow acc(b)\} \cup \bigcup_{a\in AR}\{acc(a) \leftarrow \neg d(a)\}$

Now let P' be the program obtained from P by applying the well-know principle of partial evaluation (PPE) ([5]) to P:

$P' = \bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow \neg d(b)\} \cup \bigcup_{a\in AR}\{acc(a) \leftarrow \neg d(a)\}$

Since the stable model semantics is closed under PPE [5], then by Theorem 3, E is a stable extension of AF iff $tr(E)$ is a stable model of P_{AF} iff $tr(E)$ is a stable model of P'.

Now, let $\Psi_{AF} = P_1 \cup P_2$ such that P_1 contains all the definite clauses of Ψ_{AF} and $P_2 = \Psi_{AF} \setminus P_1$. This means that

$P_2 = \bigcup_{b:(b,a)\in attacks}\{d(a) \leftarrow \neg d(b)\} \cup \bigcup_{a\in AR}\{acc(a) \leftarrow \neg d(a)\}$

Observe that $P_2 = P'$, hence by Lemma 2 and Lemma 3, P' is stable equivalent to Ψ_{AF}. Therefore, E is a stable extension of AF iff $tr(E)$ is a stable model of P' iff $tr(E)$ is a stable model of Ψ_{AF}.

■

Example 5 *In order to illustrate this theorem, let us consider the argumentation framework of Figure 2. As we saw in Example 3, Ψ_{AF} is:*

$d(a) \leftarrow \neg d(b).$ $\quad$ $d(a) \leftarrow d(a).$
$d(b) \leftarrow \neg d(a).$ $\quad$ $d(b) \leftarrow d(b).$
$d(c) \leftarrow \neg d(b).$ $\quad$ $d(c) \leftarrow \neg d(e).$
$d(c) \leftarrow d(a).$ $\quad$ $d(c) \leftarrow d(d).$
$d(d) \leftarrow \neg d(c).$ $\quad$ $d(d) \leftarrow d(b), d(e).$
$d(e) \leftarrow \neg d(d).$ $\quad$ $d(e) \leftarrow d(c).$
$acc(a) \leftarrow \neg d(a).$ $\quad$ $acc(b) \leftarrow \neg d(b).$
$acc(c) \leftarrow \neg d(c).$ $\quad$ $acc(d) \leftarrow \neg d(d).$
$acc(e) \leftarrow \neg d(e).$

Observe that Ψ_{AF} can be split in two subprograms:

P_1: $d(a) \leftarrow d(a).$ $\quad$ $d(b) \leftarrow d(b).$
$d(c) \leftarrow d(a).$ $\quad$ $d(c) \leftarrow d(d).$
$d(d) \leftarrow d(b), d(e).$ $\quad$ $d(e) \leftarrow d(c).$

P_2: $d(a) \leftarrow \neg d(b).$ $\quad$ $d(b) \leftarrow \neg d(a).$
$d(c) \leftarrow \neg d(b).$ $\quad$ $d(c) \leftarrow \neg d(e).$
$d(d) \leftarrow \neg d(c).$ $\quad$ $d(e) \leftarrow \neg d(d).$
$acc(a) \leftarrow \neg d(a).$ $\quad$ $acc(b) \leftarrow \neg d(b).$
$acc(c) \leftarrow \neg d(c).$ $\quad$ $acc(d) \leftarrow \neg d(d).$
$acc(e) \leftarrow \neg d(e).$

It is easy to see that P_2 has just one stable model: $\{d(a), d(c), d(e), acc(b), acc(d)\}$. By Theorem 3, we know that the stable models of P_2 characterize the stable extensions of AF, therefore $\{b, d\}$ is the only stable extension of AF. Since by Lemma 2 and Lemma 3, P_2 is stable equivalent to $P_1 \cup P_2$, we can see that the stable models of Ψ_{AF} characterizes the stable extension of AF.

Now, let us introduce the following lemma which will be useful to prove that the pstable models of Ψ_{AF} characterize the preferred extensions of AF.

Lemma 4 *Let P be a normal program such that there exist P_1 and P_2 satisfying:*

1. *$P = P_1 \cup P_2$ and $P_1 \cap P_2 = \{\}$.*
2. *The atoms in the head of P_1 do not occur in P_2.*
3. *The atoms in the body of P_1 do not occur in the head of P_1.*

Then M is a pstable model of P if and only if there exist M_1 and M_2 such that $M = M_1 \cup M_2$, M_2 is a pstable model of P_2 and $M_1 = \{x : x \leftarrow \alpha \in P_1, M_2(\alpha) = 1\}$.

Proof:

⇒ Let us assume that M is a pstable model of P and let us define $M_2 = M \cap \mathcal{L}_{P_2}$ and $M_1 = M \setminus M_2$. It is clear that M_2 models the rules of P_2. Now, from the fact that $RED(P, M) \models$

M_2, it follows that $RED(P_1, M) \cup RED(P_2, M) \models M_2$, and by using the equality $RED(P_2, M) = RED(P_2, M_2)$ we conclude that $RED(P_1, M) \cup RED(P_2, M_2) \models M_2$.
Let us denote by $H(P_1)$ the program consisting of the heads that appear in P_1, *i.e.*, $H(P_1) = \{a \leftarrow | \ a \in HEAD(P_1)\}$. It is clear that each of the rules in $RED(P_1, M)$ follows as a consequence in classical logic from $H(P_1)$.
From these facts we conclude that $H(P_1) \cup RED(P_2, M_2) \models M_2$, and since by hypothesis $H(P_1) \cap \mathcal{L}_{P_2} = \emptyset$, it follows that $RED(P_2, M_2) \models M_2$. This proves that M_2 is a pstable model of P_2.
Now let us consider the set $S = \{x : x \leftarrow \alpha \in P_1, M_2(\alpha) = 1\}$. All the rules in S have bodies that are true with respect to M_2, hence they are true with respect to M; but by hypothesis M models all these rules, then we conclude that $S \subseteq M$, and by condition two in the hypothesis, it follows that $S \subseteq M_1$.
Now it is easy to see that $S \cup M_2$ models all the rules in P, and it is also clear that $RED(P, S \cup M_2) = RED(P_1, S \cup M_2) \cup RED(P_2, S \cup M_2) = RED(P_1, S \cup M_2) \cup RED(P_2, M_2)$ and that $RED(P_1, S \cup M_2) \cup RED(P_2, M_2) \models S \cup M_2$.
We conclude that $S \cup M_2$ is a pstable model for P, and then using the minimality property for pstable models, it follows that $S \cup M_2 = M_1 \cup M_2$, in particular $M_1 = S$. This finishes the first part of the proof.

$\Leftarrow$ According to the hypothesis, M_2 models in classical logic P_2 and $RED(P_2, M_2) \models M_2$. In order to prove that $M_1 \cup M_2$ is a pstable model of P we only need to show that $M_1 \cup M_2$ models P_1 and that $RED(P_1 \cup P_2, M_1 \cup M_2) \models M_1 \cup M_2$.
The rules of P_1 that define the set M_1 are clearly modeled by M_1, and if we consider a rule in $P_1 : x \leftarrow \alpha$ such that $M_2(\alpha) = 0$ then there are two possibilities, first: $(M_1 \cup M_2)(\alpha) = 0$, in which case the rule is modeled by $M_1 \cup M_2$, and second: $(M_1 \cup M_2)(\alpha) = 1$; but in this case there would exist an atom b in the body of the clause such that $M_1(b) = 1$, but this contradicts the fact that the atoms in the body of P_1 do not appear in the the head of P_1. Thus we conclude that $M_1 \cup M_2$ models P_1 and hence P in the classical sense.
To prove that $RED(P_1 \cup P_2, M_1 \cup M_2) \models M_1 \cup M_2$, we observe that $RED(P_1 \cup P_2, M_1 \cup M_2) = RED(P_1, M_1 \cup M_2) \cup RED(P_2, M_1 \cup M_2) = RED(P_1, M_2) \cup RED(P_2, M_2)$ according to the hypothesis. The conclusion now follows easily.

■

This lemma points out that we can split Ψ_{AF} into two subprograms *e.g.,* P_1 and P_2, such that the pstable models of Ψ_{AF} can be constructed from P_1 and P_2. For instance, let Ψ_{AF} be the normal program of Example 5. A possible instantiation of P_1 and P_2 *w.r.t.* Lemma 4 is:

P_2 :

$d(a) \leftarrow \neg d(b).$ $\quad$ $d(a) \leftarrow d(a).$
$d(b) \leftarrow \neg d(a).$ $\quad$ $d(b) \leftarrow d(b).$
$d(c) \leftarrow \neg d(b).$ $\quad$ $d(c) \leftarrow \neg d(e).$
$d(c) \leftarrow d(a).$ $\quad$ $d(c) \leftarrow d(d).$
$d(d) \leftarrow \neg d(c).$ $\quad$ $d(d) \leftarrow d(b), d(e).$
$d(e) \leftarrow \neg d(d).$ $\quad$ $d(e) \leftarrow d(c).$

P_1 :

$acc(a) \leftarrow \neg d(a).$ $\quad$ $acc(b) \leftarrow \neg d(b).$
$acc(c) \leftarrow \neg d(c).$ $\quad$ $acc(d) \leftarrow \neg d(d).$
$acc(e) \leftarrow \neg d(e).$

We can see that $M_2 = \{d(a), d(c), d(e)\}$ is a pstable model of P_2, hence, we can infer that $M_1 = \{acc(b), acc(d)\}$. This meas that $M = M_2 \cup M_1 = \{d(a), d(c), d(e)acc(b), acc(d)\}$ is a pstable model of Ψ_{AF}.

We have introduced Lemma 4 in order to prove that the pstable models of Ψ_{AF} characterize the preferred extensions of AF; however, observe that this lemma identifies a class of normal programs which can be split in order to construct their pstable models. Hence this results is relevant by itself.

Since our formalization of the characterization of the preferred semantics by the pstable semantics will consider Theorem 4 and this theorem is based on positive disjunctive logic programs and the stable model semantics, we will introduce a lemma which will show how to recover the stable models of a positive disjunctive logic program by considering pstable models.

As we know by Definition 2, the pstable semantics is defined for normal programs. Hence, for a

disjunctive positive rule

$$r = a_1 \vee a_2 \vee ... \vee a_s \leftarrow b_1 \wedge b_2 \wedge \cdots \wedge b_n$$

we define the closure of r as the union of the s rules:

$$a_i \leftarrow \wedge b_k \bigwedge (\wedge \neg a_j)_{\{j \neq i\}}$$

for $i : 1, 2, \ldots, s$ and $k : 1, 2, \ldots, n$. In the case in which the rule r is normal, its closure is the same r. The closure $CL(P)$, of a disjunctive program P, is the union of the closures of its rules. Observe that this program is normal. Our lemma then says:

Lemma 5 *For a positive disjunctive program P, one can recover its stable models from the pstable models of $CL(P)$, specifically, the stable models of P are exactly the pstable models of $CL(P)$.*

Proof: Let us consider along with the given program P, the programs $P_1 = CL(P)$ and $P_2 = \{a' \leftarrow a \mid a \in \mathcal{L}_P\}$ where each a' is a new atom, in other words a' does not appear in $\mathcal{L}_P$. Also let P_3 be the following program:

$$P_3 = \{u \leftarrow a', \neg a : a \in \mathcal{L}_P\} \cup \left\{ \begin{array}{c} x \leftarrow u, \neg y. \\ y \leftarrow u, \neg z. \\ z \leftarrow u, \neg x. \end{array} \right\}$$

Here x, y, z, u are special atoms that do not occur in $\mathcal{L}_P$. According to Theorem 2 (see [17] for details), the stable models of P are found by intersecting the pstable models of $P_3 \cup P_2 \cup P_1$ with the language of P. Formally M is a stable model of P iff there exists a pstable model N of $P_1 \cup P_2 \cup P_3$ such that $M = N \cap \mathcal{L}_P$.

According to Lemma 3.3 of [17], the pstable models of $P_1 \cup P_2 \cup P_3$ correspond to the pstable models of the program (extended with constraints) $P_1 \cup P_2 \cup R$, where R is defined as:

$$R = \{\bot \leftarrow a', \neg a : a \in \mathcal{L}_P\}$$

By Lemma 4, for every pstable model M of $P_1 \cup P_2$, the truth value of each pair of atoms a', a agree. Hence R is redundant. Hence, the pstable models of $P_1 \cup P_2 \cup P_3$ correspond to the pstable models of the program $P_1 \cup P_2$.

Again, by Lemma 4, the pstable models of program $P_1 \cup P_2$ restricted to the language of P correspond to the pstable models of program P_1.

From all this we conclude that the stable models for P correspond to the pstable models of P_1. This finishes the proof.

■

Observe that the result of Lemma 5 is important by itself since it suggests a strict relationship between stable models and pstable models for the class of positive disjunctive logic programs. In fact this lemma is a weak version of Theorem 2.

For the rest of the paper we need the following notation. Let AF be an argumentation framework and E be a set of arguments. We write $tr1(E)$ to denote the set $\{d(a) | a$ is an argument and $a \notin E\}$. For a given set of atoms M, we write $cl(M)$ to denote the set $\mathcal{L}_P \setminus M$. A normal clause which has an atom of the form $acc(x)$ in its head will be called *acc*-rule.

By considering Lemma 4, we will prove that: Given an argumentation framework AF, then the preferred extensions of AF correspond exactly to the pstable models of Ψ_{AF}. More formally:

Theorem 6 *Let AF be an argumentation framework and E be a set of arguments. E is a preferred extension of AF if and only if $tr(E)$ is a pstable model of Ψ_{AF}.*

Proof: Let P be Ψ_{AF} minus the *acc* rules of these translations. Let $CLO(P)$ be $P \cup \{x \leftarrow \neg y : y \leftarrow \neg x \in P\}$. By Theorem 4 and Lemma 5, one can immediately see that E is a preferred extension of AF iff $tr1(E)$ is a pstable model of $CLO(P)$. We will now see that M is pstable model of P iff M is pstable model of $CLO(P)$.

Suppose that M is a pstable of P. It is immediate to see that M is a pstable model of $CLO(P)$.

The interesting case is the converse. Suppose that M is a pstable model of $CLO(P)$ and let us try to prove that M is a pstable model of P. By our assumption and the definition of pstable model, M is a model of $CLO(P)$ and $RED(CLO(P), M) \models M$. Clearly M is a model of P. We only need to prove that $RED(P, M) \models M$. This follows since $RED(P, M) \models RED(CLO(P), M)$ and this finishes this first part of the proof. It is worth to explain the last step in some more detail.

In order to prove that $RED(P,M) \models RED(CLO(P),M)$, let r_1 be any rule that belongs to $CLO(P)$ such that it does not belong to P. Then r_1 must be of the form $x \leftarrow \neg y$. Clearly r_2 (of the form $y \leftarrow \neg x$) belongs to both P and $CLO(P)$. Let s_1 be the result of r_1 after the reduction RED. Similarly, let s_2 be the result of r_2 after the reduction RED. So, $s_2 \in (RED(P,M) \cap RED(CLO(P),M))$ and $s_1 \in RED(CLO(P),M)$. We have two cases:

A) $r_1 = s_1$. Then clearly $s_2 \models s_1$. Hence $RED(P,M) \models s_1$.

B) Suppose that r_1 is different from s_1. Then $r_2 = s_2$. In addition s_1 should be the fact x which belongs to $RED(CLO(P),M)$. Necessarily $y \notin M$. Let $x \leftarrow \neg z_1$, ...,$x \leftarrow \neg z_m$ be all the non definite rules of P such that x is in the head. Then $y \leftarrow z_1,...,z_m \in P$. Since M is a model of P then there exists $z \in \{z_1,...,z_m\}$ such that $z \notin M$. Hence $x \leftarrow \neg z \in P$ and so x is a fact of $RED(P,M)$. Hence, $x \models s_1$ and so $RED(P,M) \models s_1$.

From both cases $RED(P,M) \models s_1$, so it follows that $RED(P,M) \models RED(CLO(P),M)$.

As a consequence of the above reasoning E is a preferred extension of AF iff $tr1(E)$ is a pstable model of P.

Finally, by Lemma 4 one can extend our final result from $tr1, P$ to tr, Ψ_{AF} respectively to obtain our desired result, namely that E is a preferred extension of AF iff $tr(E)$ is a pstable model of Ψ_{AF}. ■

Example 6 *In order to illustrate this theorem, let Ψ_{AF} be the normal program of Example 5. As we can see Ψ_{AF} has two pstable models:*

$$\{d(a), d(c), d(e), acc(b), acc(d)\}$$
$$\{d(b), d(c), d(d), d(e), acc(a)\}$$

This means that AF has two preferred extensions: $\{\{b,d\},\{a\}\}$.

We can formalize the fact that the well-founded model of Ψ_{AF} characterizes the grounded extensions of AF.

Theorem 7 *Let AF be an argumentation framework and E be a set of arguments. Then E is the grounded extension of AF if and only if $tr(E)$ is the well-founded model of Ψ_{AF}.*

Proof: The proof is direct by Lemma 2 of [16]. ■

As we can see by Theorem 5, Theorem 6 and Theorem 7, the mapping Ψ_{AF} is a suitable codification able to characterize the grounded, stable and preferred semantics. These results extend the results of Theorem 17 of [7]. It is worth to comment that, to the best of our knowledge, Ψ_{AF} is the first codification able to infer three argumentation semantics of Dung's framework by considering three different logic programming semantics with negation as failure.

5. Conclusions

When 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 model semantics respectively. This result is important because it defines a general method for generating metainterpreters for argumentation systems [7]. Concerning this issue, Dung did not give any characterization of the preferred semantics in terms of logic programming semantics.

In this paper, we present an extension of the results presented in Theorem 17 of [7]. In our case, we prove that by considering an argumentation framework as a logic program, it is possible to identify a unique codification (Ψ_{AF}) such that

- the well-founded model of Ψ_{AF} characterizes the grounded extension of AF (Theorem 7);
- the stable models of Ψ_{AF} characterize the stable extensions of AF (Theorem 5); and
- the pstable models of Ψ_{AF} characterize the preferred extensions of AF (Theorem 6).

This means that the main patterns of inference in argumentation semantics can be totally captured by logic programming with *negation as failure*. These results suggest that one can explore argumentation semantics by considering the results of logic programming with *negation as failure*. For instance, since the pstable semantics can

be characterized by paraconsistent logics (as the Cw and G'_3 logics [18]) or modal logics (as the S5 modal logic [20]), one can explore argumentation constructions in terms of paraconsistent logics or modal logics.

In general we believe that our results *w.r.t.* argumentation semantics are relevant for at least two reasons: First, it shows a very close relation between two well-known Non Monotonic Reasoning approaches. Second, it gives mechanisms to compute argumentation semantics since there are implementations of WFS, stable models ([6, 24]) and pstable semantics [13].

In the context of logic programming semantics, it is worth to comment that we identify a pair of stable-equivalences between normal programs (Lemma 2 and Lemma 3). We also identify a class of normal programs which can be split to construct their pstable models (Lemma 4) and finally a strict relationship between stable models and pstable models for the class of positive disjunctive program is formalized (Lemma 5).
Finally, as future work we plan on exploring the relation between supported models and pstable models for normal and disjunctive programs.

Acknowledgement

We are grateful to anonymous referees for their useful comments. J.C. Nieves thanks to CONACyT for his PhD Grant.

Referencias

[1] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press, Cambridge, 2003.

[2] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.

[3] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.

[4] Carlos Iván Chesñevar, Ana Gabriela Maguitman, and Ronald Prescott Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.

[5] Jürgen Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundam. Inform.*, 22(3):257–288, 1995.

[6] System DLV. Vienna University of Technology. http://www.dbai.tuwien.ac.at/proj/dlv/, 1996.

[7] Phan Minh 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.

[8] Paul E. Dunne and Trevor J. M. Bench-Capon. Complexity in value-based argument systems. In *JELIA*, volume 3229 of *LNCS*, pages 360–371. Springer, 2004.

[9] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[10] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[11] Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[12] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Trans. Comput. Log.*, 2(4):526–541, 2001.

[13] Alejandra López. Implementing pstable. In Rogelio Dávila, Mauricio Osorio, and Claudia Zepeda, editors, *Workshop in Logic, Language and Computation*, volume 220. CEUR Workshop Proceedings, 2006.

[14] Elliott Mendelson. *Introduction to Mathematical Logic.* Chapman and Hall/CRC, Fourth edition 1997.

[15] Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés. Preferred extensions as stable models. *Theory and Practice of Logic Programming*, 8(4):527–543, July 2008.

[16] Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés. Studying the grounded semantics by using a suitable codification. Research report LSI-08-6-R, Universitat Politècnica de Catalunya, Software Department (LSI), Barcelona, Spain, January 2008.

[17] Mauricio Osorio, José R. Arrazola, and José Luis Carballido. Logical Weak Completions of Paraconsistent Logics. *Journal of Logic and Computation*, doi: 10.1093/logcom/exn015, 2008.

[18] Mauricio Osorio and Jose Luis Carballido. Brief study of G'3 logic. *Journal of Applied Non-Classical Logics*, 18(4):79–103, 2008.

[19] Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Safe beliefs for propositional theories. *Journal of Pure and Applied Logic*, 2005.

[20] Mauricio Osorio, Juan Antonio Navarro, José R. Arrazola, and Verónica Borja. Ground Nonmonotonic Modal Logic S5: New Results. *Journal of Logic and Computation*, 15(5):787–813, 2005.

[21] Mauricio Osorio, Juan Antonio Navarro, José R. Arrazola, and Verónica Borja. Logics with Common Weak Completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.

[22] Henry Prakken and Gerard A. W. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, Dordrecht/Boston/London, second edition, 2002.

[23] Iyad Rahwan and Peter McBurney. Argumentation technology: Introduction to the special issue. *IEEE Intelligence Systems*, 22(6):21–23, 2007.

[24] System SMODELS. Helsinki University of Technology. http://www.tcs.hut.fi/Software/smodels/, 1995.

[25] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.