Inferring preferred extensions by Pstable semantics

Juan Carlos Nieves[1] and Mauricio Osorio[2]

[1] Universitat Politècnica de Catalunya
Software Department (LSI)
c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu
[2] Universidad de las Américas, CENTIA,
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@gmail.com

Abstract. Given an argumentation framework AF, we present a normal program Ψ_{AF}, such that the preferred extensions of AF correspond exactly with the pstable models of Ψ_{AF}. Moreover, we motivate the notion of suitable codifications for developing metainterpreters of argumentation theory based on logic programming.

1 Introduction

We are interested in finding suitable translators that maps an argumentation framework into a logic programming in the same line as the metainterpreter proposed by Dung [5]. To find suitable translators for argumentation theory based on logic programming is close related to find suitable codifications of an argumentation framework as logic program. This is because there is a strong relationship between the codification and the logic programming semantics which will be considered for characterizing the abstract argumentation semantics.

Nowadays, logic programming and non-monotonic reasoning (NMR) are solid areas in AI. For instance, during the last two decades, one of the most successful logic programming approaches has been Answer Set Programming (ASP). ASP is the realization of much theoretical work on Non-monotonic Reasoning and Artificial Intelligence applications. It represents a new paradigm for logic programming that allows, using the concept of *negation as failure*, to handle problems with default knowledge and produce non-monotonic reasoning [2].

It has recently been shown that G'_3 logic can be used to express interesting non-monotonic semantics [11]. More generally, two major classes of logics that can be successfully used to model nonmonotonic reasoning are (constructive) intermediate logics and paraconsistent logics [12, 10, 11]. The most well known semantics for NMR is the stable semantics [6]. This semantics provides a fairly general framework for representing and reasoning with incomplete information. There are programs such as P_0:
$a \leftarrow \neg b$.
$a \leftarrow b$.
$b \leftarrow a$:

that do not have stable models. However each atom in $\{a, b\}$ is a classical consequence of P_0 and hence this could be considered as an intended model of our program. This is not exactly a problem of the stable semantics but sometimes we have applications where we need a behavior closer to classical logic such as in the theory of argumentation (as we will soon see). The pstable semantics provides a solution to this situation. This semantics is originally constructed based on paraconsistent logics but a remarkable result is that it can expressed using only classical logic. Our paper follows this alternative simpler presentation.

In this paper we show how to model an abstract argumentation framework AF in terms of NMR. We show in particular the following interesting result. Given an argumentation framework AF, we present a normal program Ψ_{AF}, such that the preferred extensions of AF correspond exactly with the pstable models of Ψ_{AF}. Moreover, we motivate the notion of suitable codifications for developing metainterpreters of argumentation theory based on logic programming.

The rest of the paper is divided as follows: In §2, it is presented some basic definitions of logic programming and pstable semantics. §3 presents the notion of suitable codifications that is the motivation of our work. §4 presents our results. In §5 we present an alternative argumentation semantics to deal with some problems reported about the preferred argumentation semantics. We only present some interesting examples. Finally in the last section, we present our conclusions.

2 Background

In this section, we define some basic concepts of logic programming and Pstable models. We assume familiarity with basic concepts in classic logic and in semantics of logic programs *e.g.*, interpretation, model, *etc*. A good introductory treatment of these concepts can be found in [2, 7].

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, or the negation of an atom $\neg a$. Given a set of atoms $\{a_1, \ldots, a_n\}$, we write $\neg\{a_1, \ldots, a_n\}$ to denote the set of literals $\{\neg a_1, \ldots, \neg a_n\}$. A normal clause is of the form: $a_0 \leftarrow a_1, \ldots, a_j, \neg a_{j+1}, \ldots, \neg a_n$, where a_i is an atom, $0 \leq i \leq n$. When $n = 0$ the normal clause is an abbreviation of the fact a_0. A normal program is a finite set of normal clauses. Sometimes, we denote a clause C by $a \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$, where $\mathcal{B}^+$ 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 definite clause. A definite program is a finite set of definite clauses. We denote by $\mathcal{L}_P$ the signature of P, i.e. the set of atoms that occurs in P. Given a signature $\mathcal{L}$, we write $Prog_{\mathcal{L}}$ to denote the set of all the programs defined over $\mathcal{L}$.

2.2 NMR: the Pstable and stable semantics

First to definite pstable semantics (introduced in [11]), we define some basic concepts. Logical inference in classic logic is denoted by $\vdash$. Given a set of proposition symbols S

and a theory (a set of well-formed formulae) Γ, if $\Gamma \vdash S$ 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 is regarded as the standard negation operator in classic 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 is a classical model of P if the induced interpretation evaluates P to true).

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

Definition 1. *[11] Let P be a normal program and M a set of literals. We define*

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

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 $RED(P, M)$ is:
$a \leftarrow \neg b.$
$a \leftarrow b.$
$b \leftarrow a.$

By considering the reduction RED, it is defined the semantics *pstable* for normal programs.

Definition 2. *[11] Let P be a normal program and M 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. $(a \vee b) \rightarrow ((b \rightarrow a) \rightarrow a)$	Tautology
2. $\neg b \rightarrow a$	Premise from $RED(P_1, M_1)$
3. $a \vee b$	From 2 by logical equivalency
4. $(b \rightarrow a) \rightarrow a)$	From 1 and 3 by Modus Ponens
5. $b \rightarrow a$	Premise from $RED(P_1, M_1)$
6. a	From 4 and 5 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.

The well known stable semantics (see [2]) is defined as follows. First, for any theory T, we write $Pos(T)$ to denote the positive formulas of T.

Definition 3. *Let P be a normal program and M a set of atoms. Then M is a stable model of P if M is a minimal model of $Pos(RED(P, M))$.*

2.3 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 the arguments (All the definitions of this subsection were taken from the seminal paper [5]).

Definition 4. *An argumentation framework is a pair $AF := \langle AR, attacks \rangle$, where* AR *is a set of arguments, and* attacks *is a binary relation on* AR*,* i.e. attacks $\subseteq AR \times AR$*.*

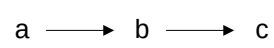

Fig. 1. A single argumentation framework

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 Fig. 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 Fig. 1, $\{a\}$ attacks b.

Definition 5. *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 sets*.

Definition 6. *(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 Fig. 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 7. *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*

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

Definition 8. *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 9. *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 Fig. 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 $\{a, c\}$.

Dung [5] defined some important concepts *w.r.t.* the relationship between arguments when they are taking part of a sequence of attacks.

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

In [5], it was 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) $d(X) \leftarrow attack(Y, X), acc(Y)$

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. Dung uses the predicate *defeat* instead of the predicate *d*. We will use the predicate $d(X)$ for denoting 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*

$$AGU = \{attacks(A, B) \leftarrow |(A, B) \in attacks\}$$

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

$m(E) = AGU \cup \{acc(A)|A \in E\}$
$\cup \{d(B)|B\ is\ attacked\ by\ some\ A \in E\}$

Theorem 1. *[5] Let AF be an argumentation framework and E be an extension of AF*[3]*. Then*

1. *E is a stable extension of AF if and only if $m(E)$ is a stable model of 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 Suitable codifications

We are interested in finding suitable translators that maps an argumentation framework into a logic programming in the same line as the metainterpreter proposed by Dung. To find suitable translators for argumentation theory based on logic programming is close related to find suitable codifications of an argumentation framework as logic program. This is because there is a strong relationship between the codification and the logic programming semantics which will be considered for characterizing the abstract argumentation semantics. For instance, Dung characterized the grounded semantics with *WFS* and the stable semantics with answer set models (see Theorem 1).

Now, what is a suitable codification for generating translators for argumentation theory? Based on the fact *the grounded semantics* and *the preferred semantics* are the main semantics for the argumentation community [13, 1], one can impose that a suitable codification at least must be able to characterize these semantics. Moreover, since some authors have been pointed out that these semantics have some drawbacks [13, 4, 3], it is important that a suitable codification must allow to define extensions of these semantics.

Given an argumentation framework $AF := \langle AR, attacks\rangle$ and a logic program P, we will say that P is a suitable codification of AF if and only if:

1. there is a logic programming semantics *SEM* such that *SEM(P)* characterizes the grounded semantics of AF and
2. there is a logic programming semantics *SEM* such that *SEM(P)* characterizes the preferred semantics.

It is worth mentioning that when we define a suitable codification we are defining a common point between tow kinds of reasonings (skeptical and credulous). In fact, the only switch that it is required for developing a skeptical reasoning or a credulous redulus in a metainterpreters for argumentation theory is to change the logic programming semantics. Also, a suitable codification could be a useful tool for defining intermediate argumentation semantics between the grounded semantics and the preferred semantics. This means that it is possible to define an intermediate reasoning between the grounded semantics and the preferred semantics.

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

The problem of characterizing abstract argumentation semantics does not only depend of the codification but also in the logic programming semantics. In fact, to find a suitable logic programming semantic is as important as to find a suitable codification for characterizing a particular abstract argumentation semantics.

By Theorem 1, we have already seen that by using P_{AF}, *WFS* is a suitable logic programming semantics for characterizing the grounded semantics. However, P_{AF} could not be considered as a suitable codification because there is not a well known logic programming semantics which could characterize the preferred semantics by using P_{AF}.

To the best of our knowledge, the only logic programming semantics that has proposed for characterizing the preferred semantic is based on minimal models [8]. In fact this approach was used for proposing an extension of the preferred semantics [9]. However, we have to accept that by using minimal models, we lose an important property of logic programming which is the use of negation by failure. For instance, let us consider the single argumentation framework $AF := \langle\{a, b\}, \{(a, b)\}\rangle$ and its codification in terms of normal program P_{AF} which is (the process for getting this codification will be described in the next section):

$$d(b) \leftarrow \neg d(a).\ \ d(b) \leftarrow \top.$$

The intended meaning of the first clause is that the argument *b* will be defeated if the argument *a* is not defeated and the last clause says that the argument *b* is defeated. It is clear the only minimal model of this program is $\{d(b)\}$ which means that the argument *b* is defeated. Therefore, by considering the complement of $\{d(b)\}$, it is deduced the only preferred extension of *AF* which is $\{a\}$. A natural extension of the program P_{AF} for inferring directly the acceptable arguments is to consider the following two clauses:

$$acc(a) \leftarrow \neg d(a).\ \ acc(b) \leftarrow \neg d(b).$$

Now, if we consider the minimal models of the program:

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

we will get: $\{acc(a), d(b)\}$ and $\{d(a), d(d)\}$. It quite obvious that the model $\{d(a), d(d)\}$ is not a desire model since this is suggesting that the empty set is a preferred extension. However that is an error.

In the following sections, we will show that *pstable semantics* is able to characterize the preferred semantics by using the suitable codification which was presented in [8]. Moreover, this semantics is able to use *negation by failure* without being affected in the characterization of the preferred semantics.

4 Mapping from argumentation frameworks to normal programs

In order to see an argumentation framework as a normal program, we start by defining a mapping from an argumentation framework to a normal logic program. In our mapping,

we use the predicate $d(X)$, where the intended meaning of $d(X)$ is "X is a defeated argument". Also we use the predicate $acc(X)$, where the intended meaning of $acc(X)$ is "X is an acceptable argument". We will denote by $D(A)$ the set of arguments that directly attacks the argument A[4]. 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:*

$$\Psi(A) := (\bigcup_{B\in D(A)} d(A) \leftarrow \neg d(B)) \cup (\bigcup_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C))$$

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

1. The first part $(\bigcup_{B\in D(A)} d(A) \leftarrow \neg d(B))$ suggests that the argument A is defeated when one of its adversaries is not defeated.
2. The last part $(\bigcup_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C))$ suggests that the argument A is defeated when all the arguments that defend[5] A 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:*

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

Example 1. Let $AF := \langle AR, attacks\rangle$ be the argumentation framework of Fig. 1. We can see that $D(a) = \{\}$, $D(b) = \{a\}$ and $D(c) = \{b\}$. Hence if we consider the normal clauses *w.r.t.* argument a, we obtain (in order to be syntactically clear we use uppercase letters as variables and lowercase letters as constants):

$$(\bigcup_{B\in\{\}} d(a) \leftarrow \neg d(B)) \cup (\bigcup_{B\in\{\}} d(a) \leftarrow \bigwedge_{C\in D(B)} d(C)) \equiv \emptyset \wedge \emptyset \equiv \emptyset$$

It is quite obvious that since the argument a has no attackers in AF, then $d(a) \notin HEAD(\Psi_{AF})$ because a is directly an acceptable argument. Therefore any argument which is attacked by a will be directly a defeated argument *e.g.*, argument b. The normal clauses *w.r.t.* argument b are:

$$(\bigcup_{B\in\{a\}} d(b) \leftarrow \neg d(B)) \cup (\bigcup_{B\in\{a\}} d(b) \leftarrow \bigwedge_{C\in D(B)} d(C)) \equiv$$
$$(d(b) \leftarrow \neg d(a)) \cup (d(b) \leftarrow \bigwedge_{C\in D(a)} d(C)) \equiv (d(b) \leftarrow \neg d(a)) \cup (d(b) \leftarrow \top)$$

[4] Given AF = $\langle AR, Attacks\rangle$ and $A \in AR$. $D(A) := \{B|(B, A) \in Attacks\}$.
[5] We say that C defends A if B attacks A and C attacks B.

It is important to remember that the conjunction of an empty set is the true value $\top$, then $d(b) \leftarrow \bigwedge_{C \in D(a)} d(C) \equiv d(b) \leftarrow \top$. The clause $d(b) \leftarrow \top$ suggests that the argument b is defeated. Now, the normal clauses *w.r.t.* argument c are

$$(\bigcup_{B \in \{b\}} d(c) \leftarrow \neg d(B)) \cup (\bigwedge_{B \in \{b\}} d(c) \leftarrow \bigwedge_{C \in D(B)} d(C)) \equiv$$
$$(d(c) \leftarrow \neg d(b)) \cup (d(c) \leftarrow d(a))$$

Then, Ψ_{AF} is:

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

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:

$$tr(E) = \{acc(A) | A \in E\}$$
$$\cup \{d(B) | B \; is \; an \; argument \; and \; B \notin E\}$$

One important result of our work is the following.

Theorem 2. *Let AF be an argumentation framework and E a set of arguments. Then E is a stable extension of AF iff $tr(E)$ is a stable model of Ψ_{AF}.*

Proof (sketch). Let P be Ψ_{AF} minus the positive rules of this translations. By the results by Dung, one can immediately see that E is a stable extensions of AF iff $tr(E)$ is a stable model of P. Then one can verify that $Pos(RED(P, M))$ and $Pos(RED(\Psi_{AF}, M))$ have the same minimal models. The result is then immediate.

The main result of the paper is the following. Given an argumentation framework AF, then the preferred extensions of AF correspond exactly with the pstable models of Ψ_{AF}. More formally:

Theorem 3. *Let AF be an argumentation framework and E a set of arguments. Then E is a preferred extension of AF iff $tr(E)$ is a pstable model of Ψ_{AF}.*

Proof (sketch). Let $tr1(E) = \{d(A) | A \; is \; an \; argument \; and \; A \notin E\}$. Let P be Ψ_{AF} minus the acc rules of this translations. By the results by [8], one can immediately see that E is a stable extensions of AF iff $tr1(E)$ is a minimal model of P. It is well known that every pstable model is a minimal model of normal program (see [11]). Considering the restricted syntax of this programs one can prove by contradiction that every minimal model is also a pstable model. As a consequence E is a stable extensions of AF iff $tr1(E)$ is a pstable model of P. Finally, by the simple negation as failure property of pstable 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}

5 Alternative Semantics

It is well-known that the preferred semantics has some problems *w.r.t.* the treatment of cycles [13, 3]. The authors in [13] underline:

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

Thus, it is an open research issue to *find* an appropriate argumentation semantics which could treat cycles without being affected by the length of the cycles. Based on the fact that Ψ_{AF} and pstable models characterize the preferred semantics, we will present an extension of the preferred semantics. This semantics is based on an alternative codification of an argumentation framework as a logic program. In particular, we will identify the arguments which do not belong to a cycles of attacks. These arguments will be called *acyclic argument*.

Definition 13. *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework and $A \in AR$. A is an acyclic argument if there is not a sequence of attacks $A_0, \ldots, A_n$ such that 1) $A = A_0$ and $A = A_n$, and 2) for each i, $0 \leq i \leq n-1$, A_{i+1} attacks A_i.*

Essentially an acyclic argument is an argument which does not belong to a cycles of attacks of *AF*. By considering the concept of acyclic argument, it is defined a variation of the normal program $\Psi(A)$ which makes a distinction between the arguments which are acyclic and which are not.

Definition 14. *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Phi(A)$ as follows:*
If $|D(A)| = 1$ and A is an acyclic argument:

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

otherwise

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

Notice that when an argument is acyclic and it has just one attack, it is omitted the positive clause. The generalization of the transformation function Φ to an argumentation framework is defined as follows:

Definition 15. *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework. We define its associated normal program as follows:*

$$\Phi_{AF} := \bigcup_{A \in AR} \Phi(A) \cup acc(A) \leftarrow \neg d(A).$$

The extension of the preferred semantics will be defined as follows:

Definition 16. *Let AF be an argumentation framework and E a set of arguments. Then E is an acyclic preferred extension of AF iff $tr(E)$ is a pstable model of Φ_{AF}.*

In order to illustrate the acyclic preferred semantics, let us consider the following examples:

Example 2. Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where AR := $\{a, b, c, d, e\}$ and attacks := $\{(a, c), (c, b), (b, a), (a, d), (b, d), (c, d), (d, e)\}$ (see Fig. 2). AF is a widely discussed argumentation framework [13, 3]. The interesting point *w.r.t.* AF is that, intuitively, we can expect to get *e* as an accepted argument. However, none of the Dung's semantics could infer *e* as an accepted argument. The grounded extension is empty and the only preferred extension is empty.
Let us consider the normal program Φ_{AF} which is:

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

We can see that Φ_{AF} has a pstable models which is $\{d(a), d(b), d(c), d(d), acc(e)\}$. This means that the argumentation framework *AF* has an *acyclic preferred extension* which is $\{e\}$.

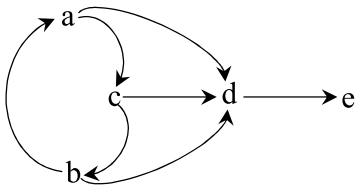

Fig. 2. An argumentation framework with a three-length cycle.

In following example, it is presented another interesting argumentation framework where there is a controversial argument.

Example 3. Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where AR := $\{a, b\}$ and attacks := $\{(a, a), (a, b)\}$ (see Fig. 3). We can observe that the argument *a* is a controversial argument. In this argumentation framework, we can expect to infer the argument b as an acceptable argument, since the only argument that attacks *b* is *a*, but this argument is self defeated. Notice that the only preferred extension of AF is the empty set.
In this case the program Φ_{AF} is:

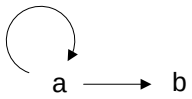

Fig. 3. An argumentation framework with a self defeated argument.

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

The only pstable models of this program Φ_{AF} is $\{acc(b), d(b)\}$. This means that the acyclic preferred extension of the argumentation framework AF is $\{b\}$.

6 Conclusiones

Given an argumentation framework AF, we present a normal program Ψ_{AF}, such that the preferred extensions of AF correspond exactly with the pstable models of Ψ_{AF}. Our result is relevant for al least two reasons. First, it shows a very close relation between two well known NMR approaches. Second, it gives as a mechanism to compute preferred extensions since it exists already an implementation of the pstable semantics. We also presented an alternative argumentation semantics to deal with some problems reported about the preferred argumentation semantics. We only present some interesting examples and this is an open problem, namely how does behave our alternative semantics?

References

1. ASPIC:Project. *Deliverable D2.2:Formal semantics for inference and decision-making*. Argumentation Service Plarform with Integrated Components, 2005.
2. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
3. P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.
4. M. Caminada. Contamination in formal argumentation systems. In *BNAIC 2005 - Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence, Brussels, Belgium, October 17-18*, pages 59–65, 2005.
5. 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.
6. M. Gelfond and V. 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.
7. E. Mendelson. *Introduction to Mathematical Logic*. Chapman and Hall/CRC, Fourth edition 1997.

8. J. C. Nieves, M. Osorio, and U. Cortés. Inferring preferred extensions by minimal models. In G. Simari and P. Torroni, editors, *Argumentation and Non-Monotonic Reasoning (LPNMR-07 Workshop)*, pages 114–124, Arizona, USA, 2007.
9. J. C. Nieves, M. Osorio, U. Cortés, I. Olmos, and J. A. Gonzalez. Defining new argumentation-based semantics by minimal models. In *Seventh Mexican International Conference on Computer Science (ENC 2006)*, pages 210–220. IEEE Computer Science Press, September 2006.
10. M. Osorio, J. A. Navarro, J. R. Arrazola, and V. Borja. Ground nonmonotonic modal logic s5: New results. *Journal of Logic and Computation*, 15(5):787–813, 2005.
11. 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.
12. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
13. H. Prakken and G. 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.