Studying the grounded semantics by using a suitable codification

Juan Carlos Nieves [a] Mauricio Osorio [b] Ulises Cortés [a]

[a] *Technical University of Catalonia, Software Department (LSI), Spain*
[b] *Universidad de las Américas - Puebla, CENTIA, México*

Abstract

One of the most representative semantics of Dung's approach is the grounded semantics. This semantics captures a skeptical approach, this means that given an argumentation framework the grounded semantics always identifies a single set of arguments, called grounded extension. It worth mentioning that the grounded semantics approach is one of the most useful argumentation approaches in real argumentation-based systems

As argumentation can be abstractly defined as the interaction of arguments for and against some conclusion, a reasoning based on an abstract argumentation semantics for describing the interaction arguments is as important as to find an extension of an argumentation framework.

In this paper, we introduce a novel formal argumentation method based on normal programs and rewriting systems which is able to

- describe the interaction of arguments during the process of inferring an extension, and
- define extensions of the grounded semantics based on specific rewriting rules which perform particular kind of reasoning as in reasoning by cases.

Moreover, we point out that our codification of an argumentation framework as a normal program is a suitable codification for studying other abstract argumentation semantics as are the stable semantics and the preferred semantics.

Email addresses: jcnieves@lsi.upc.edu (Juan Carlos Nieves), osoriomauri@googlemail.com (Mauricio Osorio), ia@lsi.upc.edu (Ulises Cortés).

Technical Report: Technical University of Catalonia, *Software Department (LSI)*

1 Introduction

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. During the last years, argumentation has been gaining increasing importance in Multi-Agent Systems (MAS), mainly as a vehicle for facilitating *rational interaction* (*i.e.* interaction which involves the giving and receiving of reasons). A single agent may also use argumentation techniques to perform its individual reasoning because it needs to make decisions under complex preferences policies, in a highly dynamic environment.

Although several approaches have been proposed for argument theory, Dung's approach presented in [13], is a unifying framework which has played an influential role on argumentation research and Artificial Intelligence (AI). 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. The strategy for inferring the set of acceptable arguments is based on abstract argumentation semantics. The kernel of Dung's framework is supported by four abstract argumentation semantics: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics*. From these semantics, the main semantics for collective acceptability are *the grounded semantics* and *the preferred semantics* [22,2]. The first one represents a skeptical approach, since for a given argumentation framework it always identifies a single extension, called grounded extension. The preferred semantics instead represents a credulous approach, since for a given argumentation framework it identifies a set of extensions which are called preferred extensions. It is worth mentioning that the grounded extension is included in all the preferred extensions. This property supports the fact that the grounded semantics is most adequate than the preferred semantics for developing skeptical reasoning [3]. Also we can say that the grounded semantics approach is one of the most useful argumentation approaches in real argumentation-based systems [10,22,1,14].

Since Dung's framework was introduced in [13], it was showed that this approach can be viewed as a special form of logic programming with *negation as failure*. In fact, this result was used for introducing a general logic programming method for generating metainterpreters for argumentation theory. One vital step of this method is to find a flexible and suitable translation of an argumentation framework to a logic program in order to capture the kernel of Dung's framework.

To find suitable translations 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, the codification proposed by Dung in [13] is able to characterize the grounded semantics with the Well-Founded Semantics (WFS [15]) and the stable semantics with answer set models [16]. However it is not able to characterize the preferred semantics.

It is quite obvious that a suitable codification of an argumentation framework should not only permit to characterize abstract argumentation semantics, but also it ought to permit to perform a deep study about an abstract argumentation semantics.

But, what is a suitable codification for argumentation theory? Based on the fact *the grounded semantics* and *the preferred semantics* are the main semantics for the argumentation community [22,2], one can impose that a suitable codification must be able, at least, to characterize both. Also, since the stable semantics could be regarded as an intermediate semantics between the grounded semantics and the preferred semantics, one could expect that a suitable codification must characterize the stable semantics. An extra point about a suitable codification is that since some authors have been pointed out that these semantics have some drawbacks [22,9,4], it is important that a suitable codification must allow to define extensions of these semantics.

The main issue of this paper is to explore a suitable codification which is enough flexible for characterizing the grounded semantics, the stable semantics and the preferred semantics. In particular, we present a frame of extensions of the grounded semantics which are intermediate semantics between the grounded semantics and the preferred semantics. All these semantics have as a common point a suitable logic program and the only difference between them is the logic programming semantics which is applied to the logic program. One of the outstanding properties of these semantics is that they are polynomial time computable.

Also by taking advantage of our suitable codification, we are able to introduce a novel formal argumentation method based on normal programs and rewriting systems which is able to:

- describe the interaction of arguments during the process of inferring an extension and,
- define extensions of the grounded semantics based on specific rewriting rules which perform particular kind of reasoning as in reasoning by cases.

The rest of the paper is structured as follows: In §2, we present some basic concepts of logic programming, argumentation theory and rewriting systems. In §3, we motivate the concept of suitable codification of an argumentation framework. After that in §4, a suitable codification based on normal programs is introduced. In §5, some extensions of the grounded semantics based on

our suitable codification are presented. In §6, we present *how* to visualize the interaction of arguments based on rewriting systems. In §7, we present a small variation of the suitable codification presented in §3. Finally, we present our conclusions and proposals for future work.

2 Background

In this section, we first define the syntax of a valid logic program, after that we define a characterization of Well-Founded Semantics (WFS) in terms of rewriting systems and finally we present some basic concepts of argumentation theory.

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 $a_0 \leftarrow \top$, where $\top$ and$\bot$ are the ever true and ever false propositions respectively. 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 Well-Founded Semantics

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

A partial interpretation based on a signature $\mathcal{L}$ is a disjoint pair of sets $\langle I_1, I_2\rangle$ such that $I_1 \cup I_2 \subseteq \mathcal{L}$. A partial interpretation is total if $I_1 \cup I_2 = \mathcal{L}$. Given two interpretations $I = \langle I_1, I_2\rangle$, $J = \langle J_1, J_2\rangle$, we set $I \leq_k J$ *if, by definition,* $I_i \subseteq J_i$, $i = 1, 2$. Clearly $\leq_k$ is a partial order. We may also see an interpretation $\langle I_1, I_2\rangle$ as the set of literals $I_1 \cup \neg I_2$. When we look at interpretations as sets of literals then $\leq_k$ corresponds to $\subseteq$. A general semantics SEM is a function on $\text{Prog}_{\mathcal{L}}$ which associates with every program a partial interpretation. Given a

signature $\mathcal{L}$ and two semantics SEM_1 and SEM_2, we define $\mathrm{SEM}_1 \leq_k \mathrm{SEM}_2$ if for every program $P \in \mathrm{Prog}_{\mathcal{L}}$, $\mathrm{SEM}_1(P) \leq_k \mathrm{SEM}_2(P)$. What are the minimal requirements we want to impose on a semantics? Certainly we want the to have the requirement that facts, *i.e.* rules with empty bodies, treated as being true. Dually, if an atom does not occur in any head, then its negation should be true. This gives rise to the following definition, which will play an important role later.

Definition 1 (SEM) *For any logic program P, we define $HEAD(P) = \{a |\ a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^- \in P\}$ — the set of all head-atoms of P. We also define $SEM(P) = \langle P^{true}, P^{false}\rangle$, where $P^{true} := \{p |\ p \leftarrow \top \in P\}$ and $P^{false} := \{p |\ p \in \mathcal{L}_P \backslash HEAD(P)\}$.*

We will use a characterization of *WFS* which is defined in terms of rewriting systems. An *abstract rewriting system* is a pair $\langle S, \rightarrow\rangle$ where $\rightarrow$ is a binary relation on a given set S. Let $\rightarrow^*$ be the reflexive, and transitive closure of $\rightarrow$. When $x \rightarrow^* y$ we say that x *reduces to* y. An *irreducible* element is said to be in *normalform*. We say that a rewriting system is

noetherian: if there is no infinite chain $x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_i \rightarrow x_{i+1} \rightarrow \ldots$, where for all $i$the elements x_i and x_{i+1} are different,
confluent: if whenever $u \rightarrow^* x$ and $u \rightarrow^* y$ then there is a z such that $x \rightarrow^* z$ and $y \rightarrow^* z$,
locally confluent: if whenever $u \rightarrow x$ and $u \rightarrow y$ then there is a z such that $x \rightarrow^* z$ and $y \rightarrow^* z$.

In a noetherian and confluent rewriting system, every element x reduces to a unique normalform that we denote by $norm(x)$.

Now, we define some basic transformation rules for normal logic programs which will be considered for characterizing *WFS*.

Definition 2 (Basic Transformation Rules) *[12] A transformation rule is a binary relation on $Prog_{\mathcal{L}}$. The following transformation rules are called* basic*. Let a program $P \in Prog_{\mathcal{L}}$ be given.*

***RED*$^+$:** *This transformation can be applied to P, if there is an atom a which does not occur in HEAD(P). **RED**$^+$ transforms P to the program where all occurrences of $\neg a$ are removed.*
***RED*$^-$:** *This transformation can be applied to P, if there is a rule $a \leftarrow \top \in P$. **RED**$^-$ transforms P to the program where all clauses that contain $\neg a$ in their bodies are deleted.*
Success: *Suppose that P includes a fact $a \leftarrow \top$ and a clause $q \leftarrow body$ such that $a \in body$. Then we replace the clause $q \leftarrow body$ by $q \leftarrow body \setminus \{a\}$.*
Failure: *Suppose that P contains a clause $q \leftarrow body$ such that $a \in body$ and $a \notin HEAD(P)$. Then we erase the given clause.*

Loop: *We say that P_2 results from P_1 by Loop_A if, by definition, there is a set A of atoms such that 1. for each rule $a \leftarrow body \in P_1$, if $a \in A$, then $body \cap A \neq \emptyset$, 2. $P_2 := \{a \leftarrow body \in P_1 | body \cap A = \emptyset\}$, 3. $P_1 \neq P_2$.*

Let CS_0 be the rewriting system such that contains the transformation rules: RED^+, RED^-, *Success*, *Failure*, and *Loop*. We denote the uniquely determined normal form of a program P with respect to the system $\mathcal{CS}$ by $norm_{\mathcal{CS}}(P)$. Every system $\mathcal{CS}$ induces a semantics $\mathrm{SEM}_{\mathcal{CS}}$ as follows:

$$\mathrm{SEM}_{\mathcal{CS}}(P) := \mathrm{SEM}(norm_{\mathcal{CS}}(P))$$

In order to illustrate the basic transformation rules, let us consider the following example.

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

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

Now, let us apply CS_0 to P. Since $d(a) \notin HEAD(P)$, then, we can apply ***RED***$^+$ *to P. Thus we get:*

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

Notice that we can apply ***RED***$^-$ *to the new program, thus we get:*

$$d(b) \leftarrow \top. \qquad\qquad d(c) \leftarrow d(a).$$

Finally, we can apply ***Failure*** *to the new program, thus we get: $d(b) \leftarrow \top$. This last program is the* normal form *of P w.r.t. CS_0, because none of the transformation rules from CS_0 can be applied.*

We use $P_1 \rightarrow^T P_2$ for denoting that we get P_2 from P_1 by applying the transformation rule T to P_1. All the rewriting systems that we consider in this paper are *confluent* and *noetherian*.

WFS is one of the most acceptable semantics in logic programming. It was introduced in [15] and was characterized in terms of rewriting systems in [7]. This characterization is defined as follows:

Lemma 1 *[7] CS_0 is a confluent rewriting system. It induces a 3-valued semantics that it is the Well-founded Semantics.*

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 [13]).

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 $\subseteq AR \times AR$*. We write* $\mathcal{AF}_{AR}$ *to denote the set of all the argumentation frameworks defined over* AR*.*

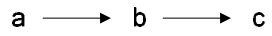

Fig. 1. A single argumentation framework

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 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. (3) An argument* $A \in AR$ *will be called defeated when it is attacked by an acceptable argument.*

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 6 *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 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 Fig. 1. Then

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

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 $\{a, c\}$.

Another interesting semantics which was introduced in [13] is *stable semantics.*

Definition 9 *A conflict-free set of arguments A is called a stable extension if and only if S attacks each argument which does not belong to S.*

Dung showed that this semantics coincides with the notion of stable solutions of n-person games [13]. 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.

Dung [13] defined some important concepts *w.r.t.* the relationship between arguments when they are taking part of a sequence of attacks.

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

In [13], 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 metainterpreter 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. Formally the Dung's metainterpreter is defined as follows:

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, m(E) is defined as follows:

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

Based on P_{AF}, Dung was able to characterize the stable semantics and the grounded semantics.

Theorem 1 *[13] 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 an answer set of P_{AF}
(2) E is a grounded extension of AF if and only if $m(E) \cup \{\neg defeat(A)|A \in E\}$ is the well-founded model of P_{AF}

3 Suitable codifications

The problem of finding suitable translations 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).

What is a suitable codification for argumentation theory? Based on the fact *the grounded semantics* and *the preferred semantics* are the main semantics for the argumentation community [22,2], one can impose that a suitable codification at least must be able to characterize these semantics. Also, since

the stable semantics could be regarded as an intermediate semantics between the grounded semantics and the preferred semantics, one could expect that a suitable codification must characterize the stable semantics. Moreover, since some authors have been pointed out that these semantics have some drawbacks [22,9,4], it is important that a suitable codification must allow to define extensions of these semantics.

Before to introduce the definition of a suitable codification, we will define what we understand when we say that a logic program semantics characterizes an argumentation semantics. Formally, an argumentation semantics arg_SEM is a function from $\mathcal{AF}_{AR} \rightarrow AR^2$.

Definition 11 *Given an argumentation framework $AF := \langle AR, attacks \rangle$, a logic program P, a logic programming semantics SEM, and an argumentation semantics arg_SEM. We say that SEM* of P characterizes *arg_SEM* of *AF if the following conditions hold:*

(1) for each model M inferred by SEM from P, there is an extension E inferred by arg_SEM from AF such that there exists a function f such that $f(M) = E$.
(2) for each extension E inferred by arg_SEM from AF, there is a model M inferred by SEM from P such that there exists a function g such that $g(E) = M$.

Informally speaking, the first condition says that if M is a model of P in a given logic programming semantics, then M represents an extension of AF in a given argumentation semantics. In the same way, the second condition says that if E is an extension of AF in a given argumentation semantics, then E represents a model of P in a given logic programming semantics.

Notice that in this definition, there is a strict relation one to one between the models of P and the extensions of AF. This means that if we are characterizing a *skeptical argumentation semantics* arg_SEM, then it must be characterized by a *skeptical logic programming semantics* SEM. Also when we are charactering a *credulous argumentation semantics* arg_SEM, it must be characterized by a *credulous logic programming semantics*. By having it in mind, we introduce our definition of a suitable codification.

Definition 12 *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:*

(1) there is a logic programming semantics SEM *such that* SEM *of P characterizes the grounded semantics of AF,*
(2) there is a logic programming semantics SEM *such that* SEM *of P characterizes the stable semantics of AF,*
(3) there is a logic programming semantics SEM *such that* SEM *of P char-*

acterizes the preferred semantics of AF,

(4) *the functions f and g of each characterization are polynomial time computable and*

(5) *there exists a polynomial time computable function μ such that $\mu(AF) = P$.*

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 reasoning in a codification for argumentation theory is to change the logic programming semantics which is applied to the suitable codification. Also, a suitable codification could be an 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.

The problem of characterizing abstract argumentation semantics does not only depend on 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 and answer set semantics [16] for characterizing the stable semantics. However, to the best of our knowledge there is not a logic programming semantics which could characterize the preferred semantics by using P_{AF}. Hence we can not say that P_{AF} is a suitable codification.

It is well-known that *WFS* is one of the most acceptable semantics in logic programming. In fact, it is called a *well-behaved semantics* [11]. Also, there is a strong study of *WFS* based on rewriting systems such that it was characterized by rewriting systems (see Lemma 1). Moreover in [12], it was introduced some extensions of *WFS* which are based on rewriting systems. It is quite obvious that these results about *WFS* most have applications in argumentation theory since it is possible to characterize the grounded semantics by *WFS* and a suitable codification. In the following sections, we will see that a suitable codification could be a bridge for carrying WFS' results to argumentation theory. For instance, we will see how to describe the interaction of arguments by using rewriting systems and to define extensions of the grounded semantics based on a particular kind of reasoning *e.g.,* reasoning by cases.

It is worth mentioning that the codification P_{AF} is quite inflexible for describing the interactions of arguments throughout rewriting systems. Moreover, to the best of our knowledge it seems that there is not a suitable logic program-

ming semantics for defining an extension of the grounded semantics based on P_{AF}. In fact, we will see that by applying the WFS' extensions that we consider in this paper to P_{AF} we do not get really an extension of the grounded 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 will denote by $D(A)$ the set of arguments that directly attack the argument A [1]. We define a transformation function *w.r.t.* an argument as follows.

Definition 13 *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 [2] A are defeated.

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

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

$$\Psi_{AF} := \bigcup_{A\in AR} \Psi(A)$$

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

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

Example 2 *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):*

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

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:*

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

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

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

Then, Ψ_{AF} is:

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

Notice that Ψ_{AF} corresponds to the normal program P of Example 1.

There are three remarks at syntactic level which we want to point out about the mapping Ψ_{AF}:

(1) Any argument A which has no adversary will not appear in $HEAD(\Psi_{AF})$, *e.g.,* argument a of Fig. 1. This means that A will always be considered as an acceptable arguments. In fact, for any argumentation framework $AF := \langle AR, Attacks\rangle$ where $Attacks = \emptyset$, Ψ_{AF} will be an empty program.

(2) If A is an argument which is attacked by an argument which has no adversary, then $d(A) \leftarrow \top \in \Psi(A)$, *e.g.,* argument b of Fig. 1. This means that A will always be considered as defeated by any argumentation semantics which is based on Ψ_{AF} and some logic programming semantics.

(3) It is easy to see that the mapping Ψ_{AF} is polynomial time computable.

The flexibility of Ψ_{AF} for charactering the stable semantics and the preferred semantics was proved in [19,18]. In [19], it was proved that the minimal models of Ψ_{AF} correspond to the preferred extensions of AF. Moreover in [18], it was proved that the answer sets of Ψ_{AF} corresponds to the stable extensions of AF and the pstable models of Ψ_{AF} correspond to the preferred extensions of AF.

5 WFS' extensions and the grounded semantics

In this section, we will show that a suitable codification is not only able to characterize the grounded semantics but it is also able to define some extensions of the grounded semantics. First of all, we will show that Ψ_{AF} is a *suitable codification*. Then we will show that Ψ_{AF} is able to characterize the grounded semantics.

We start by presenting some basic terms. Given an argumentation framework $AF := \langle AR, Attacks \rangle$, we understand $f(E) := \{d(a) | a \in E\}$, where $E \subseteq AR$. Then formally the grounded semantics is characterized by Ψ_{AF} and WFS as follows:

Lemma 2 *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. S is the grounded extension of AF if and only if $\exists\ D \subseteq AR$ such that $\langle f(D), f(S) \rangle$ is the well-founded model of Ψ_{AF}.*

Proof. See Appendix A. ■

For instance, we have seen that the program P of Example 1 corresponds to the normal program Ψ_{AF} of the argumentation framework of Fig. 1. Also, we have seen that the *normal form* of Ψ_{AF} is: $d(b) \leftarrow \top$. Then, the well-founded model of Ψ_{AF} is $\langle \{d(b)\}, \{d(a), d(c)\} \rangle$. Then by Lemma 2, this means that $\{a, c\}$ is the grounded extension of the argumentation framework of Fig. 1.

It is worth mentioning that we are using a characterization of WFS which is based on rewriting systems. The advantage of this method is that we are using transformation rules which are actions that represent changes in the graph representation of the argumentation framework AF. For instance, since $SEM(\Psi_{AF}) = \langle \{d(b)\}, \{d(a)\} \rangle$, we know that the argument a is acceptable because a has no adversaries and the argument b is defeated because it is attacked by a. Then if $\Psi_{AF} \rightarrow^{RED^+} \Psi_{AF}^{RED^+}$, the clause $d(b) \leftarrow \neg d(a)$ is reduced to $d(b) \leftarrow \top$. This means that RED^+ recognizes that b is a defeated argument. Therefore, if we apply RED^- to $\Psi_{AF}^{RED^+}$, it will remove the clause

$d(c) \leftarrow \neg d(b)$ which represents an attack of the argument b. This change in the program could be viewed as it is shown in Fig. 2, where the attack/edge of b to c is removed. Since the argument a is acceptable, $Failure$ will remove the clause $d(c) \leftarrow d(a)$ which suggested that the argument c will be defeated in case that a is defeated.

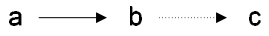

Fig. 2. Visualization of the program $(\Psi_{AF}^{RED^+})^{RED^-}$.

Based on the facts that

- the well founded model of Ψ_{AF} characterizes the grounded semantics (Lemma 2),
- the answer set models of Ψ_{AF} characterizes the stable semantics (Theorem 2 of [18]), and
- the minimal models of Ψ_{AF} characterizes the preferred semantics [19](In fact, it was also proved in [18] that the pstable models of Ψ_{AF} corresponds to the preferred extensions of AF [3]).

we can infer the following corollary.

Corollary 1 *Ψ_{AF} is a suitable codification.*

The flexibility of a suitable codification could help to meet the results of two approaches of non-monotonic reasoning; logic programming and argumentation theory. In order to show this, we will show that by applying different logic programming semantics to the suitable codification Ψ_{AF} will be enough for defining new argumentation semantics. In particular, we will consider some extensions of *WFS* in order to define extensions of the grounded semantics.

Since *WFS* is a 3-valued logic semantics, where any atom could be *true*, *false*, and *undefined*, we will define the concept of a 3-valued extension, where any argument could be *accepted*, *defeated*, and *undecided*.

Definition 15 (3-valued extension) *Given an argumentation framework $AF := \langle AR, attacks \rangle$, and $S, D \subseteq AR$. A 3-valued extension is a tuple $\langle S, D \rangle$, where $S \cap D = \emptyset$ and S is a conflict-free set. We call an argument a acceptable if $a \in S$, an argument b defeated if $b \in D$, and an argument c undecided if $c \in AR \setminus \{S \cup D\}$.*

[3] It is worth mentioning that there is an implementation of Pstable semantics which could be used for inferring the preferred extensions of an argumentation framework[17].

5.1 $WFS^{LLC'}$ semantics

The first WFS' extension that we will consider is called $WFS^{LLC'}$ and is based on the transformation rule LLC'.

Definition 16 (LLC′) *[12] Let a be an atom that occurs negatively in a program P and also appears in the head of some rule. Let P_1 be the program that results from P by removing $\neg a$ from every clause of P. Let **Success*** denote the reflexive and transitive closure of the relation **Success**. Suppose that P_1 relates to P_2 by **Success*** and $a \in P_2$. In this case, we add $a \leftarrow \top$ to P.*

By considering the transformation rule LLC', it is defined the rewriting system CS_1 as follows: $CS_1 := CS_0 \cup \{LLC'\}$. $WFS^{LLC'}$ is defined as follows:

Lemma 3 *[12] CS_1 is a confluent rewriting system. It induces a 3-valued semantics that we call $WFS^{LLC'}$.*

Now by considering $WFS^{LLC'}$, it is introduced an extension of the grounded semantics.

Definition 17 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S, D \subseteq AR$. $\langle S, D\rangle$ is the $WFS^{LLC'}$-extension of AF if and only if $\langle f(D), f(S)\rangle$ is a $WFS^{LLC'}$- model of Ψ_{AF}.*

The main difference between the grounded extension and the $WFS^{LLC'}$-extension is done by the transformation rule **LLC′**. Based on Ψ_{AF}, we can say that LLC' first removes all the attacks of the argument a from AF; therefore it is reviewed by *Success* whether the argument a is defeated. In case that a appears defeated, it will be assumed that the argument a is defeated. Notice that the only case that a could be defeated after removed its attacks is that a belongs to a cycle of attacks. Let us consider the following example.

Example 3 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c\}$ and $attacks := \{(a, a), (a, b), (b, c), (c, b)\}$ (see Fig. 3). Hence, Ψ_{AF} is:*

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

To infer the AF's $WFS^{LLC'}$-extension, we need to get the Ψ_{AF}'s $WFS^{LLC'}$ model. Then, we apply CS_1 to Ψ_{AF}. We can see in Fig. 3 that the argument a *is a* controversial argument *since it is attacked by itself. Then the transformation rule LLC', first it will remove all the atoms of the form $\neg d(a)$. This means*

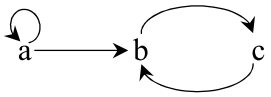

Fig. 3. An argumentation framework with two-length cycle and a self-defeated argument.

that, it will remove all the a*'s attacks of AF. After that, it will view if* a *is defeated. Since* a *appears defeated, it is assumed that the argument* a *is a defeated argument and it is added this assumption* $(d(a) \leftarrow \top)$ *to the program* Ψ_{AF}

$$d(a) \leftarrow \neg d(a). \qquad d(a) \leftarrow d(a). \qquad d(c) \leftarrow \neg d(b).$$
$$d(b) \leftarrow \neg d(a). \qquad d(b) \leftarrow \neg d(c). \qquad d(c) \leftarrow d(c), d(a).$$
$$d(b) \leftarrow d(a). \qquad d(b) \leftarrow d(c). \qquad d(a) \leftarrow \top.$$

If we assume that a *is a defeated argument, then* RED^- *will remove all its attacks (the clauses which will be removed are:* $d(a) \leftarrow \neg d(a)$ *and* $d(b) \leftarrow \neg d(a)$*) and Success will remove all its supports to other arguments (the clause* $d(c) \leftarrow d(c), d(a)$ *is reduced to* $d(c) \leftarrow d(c)$*).*

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

Then applying Success, it is found that the argument b *is a defeated argument:*

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

Therefore applying RED^-*, it removes all the attacks of the argument* b*:*

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

Since the attack of the argument b *to* c *is removed, Loop will remove the clause* $d(c) \leftarrow d(c)$*. Then we get:*

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

Finally, since the argument b *was already fixed as a defeated argument,* RED^+

will remove the attack of the argument c to b which is represented by the clause: $d(b) \leftarrow \neg d(c)$. *Then, the* normal form *of* Ψ_{AF} *is:*

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

Therefore, $WFS^{LLC'}(\Psi_{AF}) := \langle \{d(a), d(b)\}, \{d(c)\} \rangle$, *this means that* $\langle \{c\}, \{a, b\} \rangle$ *is the AF's* $WFS^{LLC'}$*- extension. We can conclude that the argument* c *is an acceptable argument and* a, b *are defeated arguments. Notice that AF has an empty grounded extension, AF has no stable extensions and AF has only one preferred extension which is* $\{c\}$. *In fact, the set of acceptable arguments of the* $WFS^{LLC'}$*- extension corresponds to the only preferred extension of AF.*

As a final comment of this example, let us consider the grounded instance of the program P_{AF} *w.r.t. AF which will be:*

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

It is not difficult to see that $WFS^{LLC'}(P_{AF}) := \langle \{d(a)\}, \{acc(a)\} \rangle$. *This means that the only thing that we can say w.r.t. AF is that the argument a is defeated.*

5.2 WFS^{WK} semantics

Now, let us consider another extension of WFS which is called WFS^{WK}. This semantics is based on the transformation rule *Weak-Cases* which is defined as follows:

Definition 18 (Weak-Cases) *Let P be a program and suppose the following condition holds:* $C_1 \in P$, $C_2 \in P$, C_1 *is of the form* $a \leftarrow l$ *and* C_2 *is of the form* $a \leftarrow \neg l$. *Then the Weak-Cases transformation replaces the clauses* C_1 *and* C_2 *in P by the single clause* $a \leftarrow \top$.

Let CS_2 be a rewriting system which contains the transformation rules $CS_0 \cup \{$ *Weak-Cases* $\}$. Then, WFS^{WK} is defined as follows:

Lemma 4 CS_2 *is a confluent rewriting system. It induces a 3-valued semantics that we call* WFS^{WK}.

Proof. Since *Weak-Cases* is an instance of the transformation rule *T-Weak-Cases*, which is defined in [12], this lemma is straightforward from Theorem

7.11 of [12]. ■

Now, by considering WFS^{WK} semantics, it is defined another extension of the grounded semantics.

Definition 19 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S, D \subseteq AR$. $\langle S, D\rangle$ is the WFS^{WK}-extension of AF if and only if $\langle f(D), f(S)\rangle$ is a WFS^{WK}- model of Ψ_{AF}.*

The main difference between the characterizations of the grounded semantics and the WFS^{WK}-extension is made by the transformation rule *Weak-Cases*. It is worth mentioning that essentially the transformation rule *Weak-Cases* deploys a reasoning by cases. In order to illustrate the WFS^{WK}-extension, let us consider the following example.

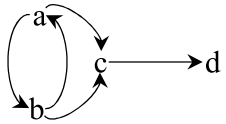

Fig. 4. An argumentation framework with a floating argument.

Example 4 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c, d\}$ and $attacks := \{(a, b), (b, a), (a, c), (b, c), (c, d)\}$ (see Fig. 4). Then, Ψ_{AF} is:*

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

In order to infer the WFS^{WK}-extension of AF, it is applied CS_2 to Ψ_{AF}. First of all, we can see that the argument a *is controversial w.r.t. the argument* c *because* a *is attacking to* c *($d(c) \leftarrow \neg d(a)$) and also* a *is defending to* c *($d(c) \leftarrow d(a)$). Therefore, if* a *is fixed as an acceptable argument, then* c *will be a defeated argument. Moreover, if* a *is fixed as a defeated argument, then* c *also will be a defeated argument. Under this situation, the transformation rule* Weak-Cases *will assume that the argument* c *is defeated, then it will remove the clauses $d(c) \leftarrow \neg d(a)$ and $d(c) \leftarrow d(a)$ from Ψ_{AF} and the clause $d(c) \leftarrow \top$ is added to Ψ_{AF}. Notice that the argument* b *is also controversial w.r.t.* c*. Then the clauses $d(c) \leftarrow \neg d(b)$ and $d(c) \leftarrow d(b)$ are removed from Ψ_{AF}.*

$$d(a) \leftarrow \neg d(b). \qquad d(a) \leftarrow d(a). \qquad d(d) \leftarrow \neg d(c).$$
$$d(b) \leftarrow \neg d(a). \qquad d(b) \leftarrow d(b). \qquad d(d) \leftarrow d(b), d(a).$$
$$d(c) \leftarrow \top.$$

Since the argument c *was assumed as to be a defeated argument, the* RED^{-} *will remove* c*'s attacks. Hence, we get:*

$$d(a) \leftarrow \neg d(b). \qquad d(a) \leftarrow d(a). \qquad d(d) \leftarrow d(b), d(a).$$
$$d(b) \leftarrow \neg d(a). \qquad d(b) \leftarrow d(b). \qquad d(c) \leftarrow \top.$$

Since this program is the formal form of Ψ_{AF}, $WFS^{WK}(\Psi_{AF}) := \langle \{d(c)\}, \{\}\rangle$. *Hence* $\langle \{\}, \{c\}\rangle$ *is the* WFS^{WK}*-extension of AF. This means that the argument c is defeated.*

Notice that the grounded extension of AF is the empty set, there are two stable extensions which are $\{a, d\}$ *and* $\{b, d\}$*, and there are two preferred extensions which coincide with the stable extensions:* $\{a, d\}$ *and* $\{b, d\}$*. It is worth mentioning that usually any argument which does not belong to a preferred/stable extension is considered defeated. Then we can see that both preferred/stable extensions of AF coincide that the argument c is a defeated argument. Therefore we can appreciate that the* WFS^{WK}*-extension coincides with the preferred/stable extensions that the argument c is defeated.*

Now let us consider the grounded instance of the program P_{AF} *w.r.t. AF in order to see the behavior of* P_{AF} *w.r.t.* WFS^{WK}.

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

We can see that $WFS^{WK}(\Psi_{AF}) := \langle \{\}, \{\}\rangle$. *This means that like the grounded extension, the argumentation semantics defined by* WFS^{WK} *and* Ψ_{AF} *is empty.*

5.3 $WFS^{WK+LCC'}$ semantics

We have defined two extensions of the grounded semantics based on two extensions of WFS, where the main support of these extensions is the use of

the transformation rules: $Weak - Cases$ and LLC'. Now the combination of these transformation rules also suggests another extension of the grounded semantics.

Let $CS_3 := CS_0 \cup \{LLC',\ Weak\text{-}Cases\}$. Obviously, CS_3 also defines an extension of WFS which is defined as follows:

Lemma 5 *CS_3 is a confluent rewriting system. It induces a 3-valued semantics that we call $WFS^{WK+LLC'}$.*

Proof. It is straightforward from Theorem 7.13 of [12]. ■

Then by considering $WFS^{WK+LLC'}$, we define an extension of the grounded semantics.

Definition 20 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S, D \subseteq AR$. $\langle S, D\rangle$ is the $WFS^{WK+LLC'}$- extension of AF if and only if $\langle f(D), f(S)\rangle$ is a $WFS^{WK+LLC'}$- model of Ψ_{AF}.*

None of both $WFS^{LLC'}$ and WFS^{WK} extensions is the same to $WFS^{WK+LLC'}$-extension. In order to illustrate this difference let us consider the following example.

Example 5 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e, f, m, n, p\}$ and attacks $:= \{(a, b),\ (b, c),\ (c, a),\ (a, d),\ (d, e),\ (e, f),\ (m, e),\ (n, m),\ (n, p), (p, m),\ (p, n)\}$ (see Fig. 5). It is not difficult to see that $WFS^{LLC'}$-extension $:= \langle\{\}, \{a, b, c, d, e\}\rangle$, WFS^{WK}-extension $:= \langle\{\}, \{m\}\rangle$, $WFS^{WK+LLC'}$-extension $:= \langle\{\}, \{a, b, c, d, e, m\}\rangle$, and the grounded extension is empty.*

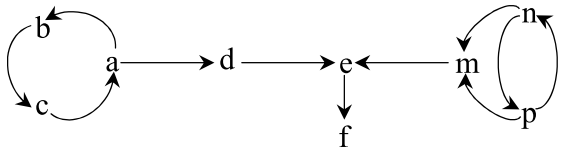

Fig. 5. Example

This argumentation framework has no stable extensions and has tow preferred extensions: $\{n\}$ and $\{p\}$.

5.4 Formalizing the extensions of the grounded semantics

We have introduced three new abstract argumentation semantics, all of them have as common point a suitable codification which is Ψ_{AF} and the only difference between them is the logic programming semantics which it is applied

to Ψ_{AF}. In the following table, we summarize the abstract argumentation semantics and the argumentation frameworks which we have commented.

Logic Semantics	Argumentation Semantics	Fig. 1	Fig. 3	Fig. 4	Fig. 5
WFS	GE_{AF}	$\{a, c\}$	$\{\}$	$\{\}$	$\{\}$
Answer Set Semantics	Stable semantics	$\{a, c\}$	-	$\{a, d\}, \{b, d\}$	-
Min. Models/Pstable Sem.	Preferred Sem.	$\{a, c\}$	$\{c\}$	$\{a, d\}, \{b, d\}$	$\{n\}, \{p\}$
$WFS^{LLC'}$	$WFS^{LLC'}$-ext.	$\langle\{a, c\}, \{b\}\rangle$	$\langle\{c\}, \{a, b\}\rangle$	$\langle\{\}, \{\}\rangle$	$\langle\{\}, \{a, b, c, d, e\}\rangle$
WFS^{WK}	WFS^{WK}-ext.	$\langle\{a, c\}, \{b\}\rangle$	$\langle\{c\}, \{a, b\}\rangle$	$\langle\{\}, \{c\}\rangle$	$\langle\{\}, \{m\}\rangle$
$WFS^{WK+LLC'}$	$WFS^{WK+LLC'}$-ext.	$\langle\{a, c\}, \{b\}\rangle$	$\langle\{c\}, \{a, b\}\rangle$	$\langle\{\}, \{c\}\rangle$	$\langle\{\}, \{a, b, c, d, e, m\}\rangle$

As can be seen in the table, we can identify a direct relationship between abstract argumentation semantics and logic programming semantics.

Once we have defined a direct relationship between abstract argumentation semantics and logic programming semantics, it is possible to understand the behavior of some abstract argumentation semantics based on the properties of the logic programming semantics. For instance, since the grounded semantics is characterized by Ψ_{AF} and *WFS*, we can infer that the $WFS^{LLC'}$-extension, the WFS^{WK}-extension and the $WFS^{WK+LLC'}$-extension are extensions of the grounded semantics and are polynomial time computable. This is essentially because the semantics $WFS^{LLC'}$, WFS^{WK} and $WFS^{WK+LLC'}$ are extensions of *WFS* and are polynomial time computable. This result is formalized with the following theorem:

Theorem 2 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and E be the grounded extension of AF. Then*

a) *(1) If $\langle S, D\rangle$ is the $WFS^{LLC'}$-extension of AF then $E \subseteq S$.*
(2) If $\langle S, D\rangle$ is the WFS^{WK}-extension of AF then $E \subseteq S$.
(3) If $\langle S, D\rangle$ is the $WFS^{WK+LLC'}$-extension of AF then $E \subseteq S$.
b) *(1) The $WFS^{LLC'}$-extension of AF is polynomial time computable.*
(2) The WFS^{WK}-extension of AF is polynomial time computable.
(3) The $WFS^{WK+LLC'}$-extension of AF is polynomial time computable.

Proof.

a) It is direct by Lemma 2.
b) It is not difficult to see that the mapping of Definition 14 is polynomial time computable; moreover, the rewriting systems CS_1, CS_2, and CS_3 are polynomial time computable [12].

■

Another property that can be formalized *w.r.t.* the new argumentation semantics is that they are intermediate logic between the grounded semantics and the preferred semantics. This is essentially because the semantics $WFS^{LLC'}$,

WFS^{WK} and $WFS^{WK+LLC'}$ are strongest than WFS and weakest than the pstable semantics (the formal definition of pstable semantics is presented in [20,21]). Remember that the pstable models of Ψ_{AF} correspond to the preferred extensions of AF [18].

In order to show that our new argumentation semantics are intermediate semantics between the grounded semantics and the preferred semantics, we will show that they are contained in the preferred semantics.

Theorem 3 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, E be a preferred extension of AF, and $E' := AR \setminus E$. Then,*

(1) If $\langle S, D\rangle$ is the $WFS^{LLC'}$-extension of AF then $S \subseteq E$ and $D \subseteq E'$.
(2) If $\langle S, D\rangle$ is the WFS^{WK}-extension of AF then $S \subseteq E$ and $D \subseteq E'$.
(3) If $\langle S, D\rangle$ is the $WFS^{WK+LLC'}$-extension of AF then $S \subseteq E$ and $D \subseteq E'$.

Proof. (sketch)

This theorem follows from the facts that

(1) Pstable semantics satisfies the basic transformation RED^+, RED^-, $Success$, $Failure$, $Loop$, WK, and LLC' (this means that any atom inferred by the basic transformations is also inferred by Pstable semantics. This fact is essentially because the basic transformation are closed under classic logic),
(2) the pstable models of Ψ_{AF} correspond to the preferred extensions of AF (By Theorem 3 of [18]).

■

6 Rewriting systems and the interaction between arguments

One of the purposes of argumentation theory is to provide tools for supporting decision making processes. For instance, argumentation theory is able to suggest arguments in favour of a decision. Usually, argumentation theory is adequate for supporting decision-making in scenarios where the information is uncertain and incomplete. In this section, we will present a small example where we will show that each time that it is applied a transformation rule to Ψ_{AF} the reduced program will suggest an approximation of the argumentation semantics that we are inferring. These approximations together with the reduced program will be describing the interaction of the arguments of the argumentation framework.

We start introducing a scenario in the medical domain, where the decision

about whether an organ from a donor with endocarditis is viable or not for being transplanted should be made [4]. Let us assume that we have two transplant coordination units, one which is against the viability of the organ (UCT_D) and one which is in favour of the viability of the organ (UCT_R).

- UCT_D argues that the organ is not viable, since the donor had endocarditis due to *streptococcus viridans*, then the recipient could be infected by the same microorganism.
- In contrast, UCT_R argues that the organ is viable, because the organ presents correct function and correct structure and the infection could be prevented with post-transplanted-treatment with penicillin, even if the recipient is allergic to penicillin, there is the option of post-transplanted-treatment with teicoplanin.

Formally, we have an argumentation framework $AF := \langle AR, attacks \rangle$, where AR has the following arguments:

a.- organ is non viable.
b.- organ is viable.
c.- organ has correct function and correct structure.
d.- recipient could be infected with streptococcus viridans.
e.- post-transplanted-treatment with administer penicillin.
f.- post-transplanted-treatment with administer teicoplanin.
g.- recipient is allergic to penicillin.

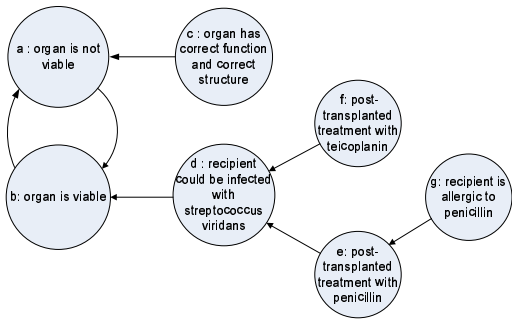

Fig. 6. A simple scenario.

and $attacks := \{(a,b),(b,a),(c,a),(d,b),(e,d),(f,d),(g,e)\}$ (The graphic representation of AF is shown in Fig. 6). Then Ψ_{AF} is:

[4] The medical information was taken from [8].

$$d(a) \leftarrow \neg d(b). \qquad d(a) \leftarrow d(a), d(d).$$
$$d(a) \leftarrow \neg d(c). \qquad d(a) \leftarrow \top.$$
$$d(b) \leftarrow \neg d(a). \qquad d(b) \leftarrow d(b), d(c).$$
$$d(b) \leftarrow \neg d(d). \qquad d(b) \leftarrow d(e), d(f).$$
$$d(d) \leftarrow \neg d(e). \qquad d(d) \leftarrow d(g).$$
$$d(d) \leftarrow \neg d(f). \qquad d(d) \leftarrow \top.$$
$$d(e) \leftarrow \neg d(g). \qquad d(e) \leftarrow \top.$$

A good question is: What can we say from Ψ_{AF} about the argumentation framework AF at this moment? We have not applied any transformation to Ψ_{AF} yet; however we can see that $SEM(\Psi_{AF}) := \langle \{d(a), d(d), d(e)\}, \{d(c), d(f), d(g)\} \rangle$. This means that in the discussion between UCT_D and UCT_R there is not doubt that the arguments $\{c, f, g\}$ are acceptable. Hence the arguments that are attacked by them are defeated which are $\{a, d, e\}$. However, at this moment $SEM(\Psi_{AF})$ cannot say nothing about the argument b (see Fig. 7).

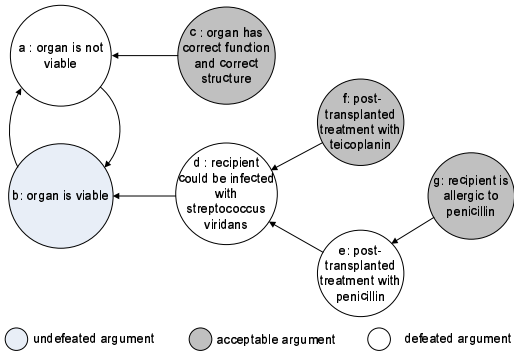

Fig. 7. Interpretation of Ψ_{AF} and $SEM(\Psi_{AF})$.

Now, let us apply the transformation rule **Failure** to Ψ_{AF} as many times as we can, then we get the following reduced program:

$$d(a) \leftarrow \neg d(b). \qquad d(a) \leftarrow d(a), d(d).$$
$$d(a) \leftarrow \neg d(c). \qquad d(a) \leftarrow \top.$$
$$d(b) \leftarrow \neg d(a).$$
$$d(b) \leftarrow \neg d(d).$$
$$d(d) \leftarrow \neg d(e).$$
$$d(d) \leftarrow \neg d(f). \qquad d(d) \leftarrow \top.$$
$$d(e) \leftarrow \neg d(g). \qquad d(e) \leftarrow \top.$$

Notice that the rules that were removed by Failure were assuming that the argument b could be defeated in case that the arguments c and f were defeated, but we have already known that c and f are acceptable arguments, hence those rules are irrelevant.

Now, let us apply the transformation rule $\mathbf{RED}^{-}$ to the reduced program as many times as we can, then the new reduced program Ψ'_{AF} is:

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

Now notice that the rules which were removed from the program were the rules which were representing the attackers of the arguments $\{a, d, e\}$. As we have already known that $\{a, d, e\}$ are defeated arguments, then their adversaries did not have sense for being represented in the program. Moreover, at this moment $SEM(\Psi'_{AF}) := \langle\{d(a), d(d), d(e)\}, \{d(c), d(f), d(g), d(b)\}\rangle$, this means that $SEM(\Psi_{AF})$ is able to suggest that the argument b is acceptable. Applying the transformation rule *Success* as many times as we can, we get the following new reduced program.

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

We can observe that in this reduced program, all the attackers of the defeated arguments were removed from the program. We can visualize this status as is shown in Fig. 8. Finally, applying the transformation rule Red^{+}, we get the normal form of Ψ_{AF} which is:

$$d(a) \leftarrow \top. \qquad d(d) \leftarrow \top. \qquad d(e) \leftarrow \top.$$

Since, the argument b is acceptable and this argument suggests that *the organ is viable* even that the donor has infected by *streptococcus viridans* and the arguments which support this decision are c, f, and g. It is possible to suggest that the transplant coordination unit UCT_R is the winner. This means that it is possible to assume that the organ could be considered for transplanting.

Notice that the grounded extension of the argumentation framework AF is: $\{b, c, f, g\}$, since all the new semantics presented in this paper are extensions

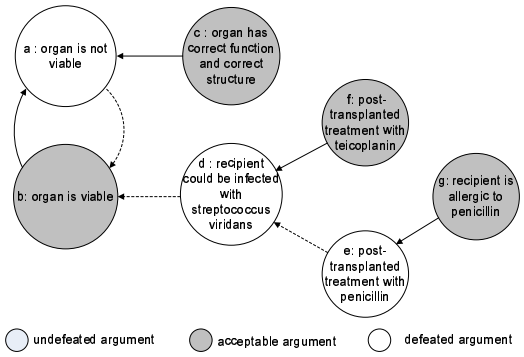

Fig. 8. Interpretation of Ψ_{AF} and $SEM(\Psi_{AF})$.

of the grounded semantics, then all the semantics presented in this paper has the behavior described in this section.

We can say that the interaction between rewriting systems and normal programs which represent an argumentation framework could describe the interaction of arguments of an argumentation framework. This allows to visualize the process of selecting acceptable arguments from an argumentation framework.

7 Acceptable arguments by default negation

In all the paper, we have described the features that could have a suitable codification of an argumentation framework as logic program. In fact, we showed that Ψ_{AF} is a suitable codification. However, the reader could think that Ψ_{AF} is a tedious codification because it does not infer the acceptable arguments of AF in a straightforward form.

An alternative form for computing the acceptable arguments in Ψ_{AF}, without considering the predicate *d(X)*, is taking advantage of *default negation*. This is possible by considering a new dual symbol for each argument of the argumentation framework. This means that we can infer the acceptable arguments directly from the models of Ψ_{AF}. In order to define a small variation of Ψ_{AF}, let us present some definitions.

Definition 21 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define the function η as $\eta : AR \rightarrow AR'$. Where AR' has the same cardinality to AR such that $AR \cap AR' = \emptyset$.*

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

Definition 22 *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:*

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

Definition 23 *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define its associated general program as follows:*

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

Notice that $\Psi(A)$ and $\Lambda(AF)$ are equivalent (module notation) and the main difference between Φ_{AF} and Λ_{AF} is the rule $A \leftarrow \neg A'$ for each argument.

The characterization of the grounded semantics by WFS and $\Lambda(AF)$ is formalized by a small variation of Lemma 2.

Lemma 6 *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. S is the grounded extension of AF if and only if $\exists\ D \subseteq AR \cup AR'$ such that $\langle S', D \rangle$ is the well-founded model of Λ_{AF} and $S = S' \cap AR$.*

Proof. The proof is straightforward from Lemma 2 and the semantics of default negation. ■

In order to illustrate this lemma, let us consider the argumentation framework of Fig. 1. Then the program Λ_{AF} is:

$$b' \leftarrow \neg a'. \qquad b' \leftarrow \top. \qquad c' \leftarrow \neg b'. \qquad c' \leftarrow a'.$$
$$a \leftarrow \neg a'. \qquad b \leftarrow \neg b' \qquad c \leftarrow \neg c'.$$

We can see that the $WFS(\Lambda_{AF}) := \langle \{a, b', c\}, \{\} \rangle$. Then $\{a, c\} = \{a, b', c\} \cap AR$, this means that $\{a, c\}$ is the grounded extension of AF.

8 Related work

We have discussed an approach for studying abstract argumentation semantics. Especially, we pay attention to the strong relationship that exists between abstract argumentation semantics and logic programming semantics. We define a declarative codification of an argumentation framework in terms of normal logic programs. Based on it, we are able to characterize abstract

argumentation semantics in terms of logic programming semantics. This approach is not new in fact since Dung's framework was introduced in [13], it was showed that this approach can be viewed as a special form of logic programming with *negation as failure*.

The Dung' semantics have characterized from several points of view. For instance in [5], the authors present a set of equations in order to decide whether a set of arguments is acceptable under a given semantics. In particular, they are able of charactering stable, preferred and complete semantics. They leave out the grounded semantics of their study, because it is not captured by a set-equation. An interesting point of the equation checking approach presented in [5] is that it is able of defining intermediate semantics between stable semantics and the preferred semantics. Since they are only interested on answering yes/not questions, their approach is unable of describing the interactions of arguments. The same authors in [6] presented a set of techniques for checking the acceptability of a set of arguments. Mainly they introduce some techniques based on model checking and satisfiability checking which are supported by a set of codifications of an argumentation framework in terms of propositional formulae. The problem of that codifications is that they are built having in mind just a particular argumentation semantics, then they are far of satisfying the conditions of a suitable codification. Moreover, they are not really flexible for describing the interactions of arguments.

9 Conclusions

To find suitable codifications for argumentation theory based on logic programming could help to close the wide separation between argumentation theory and argumentation systems. It is quite obvious that a suitable codification of an argumentation framework should not only permit to compute abstract argumentation semantics, but also it ought to permit to perform a deep study about an abstract argumentation semantics.

Our experience in the interaction between argumentation semantics and logic programming semantics suggests that the correct understanding of the behavior of one side helps to understand the behavior of the other side. For instance, thanks to the deep study that there is on the well-founded semantics is easy to understand the behavior of any extension of the grounded semantics which is based on an extension of the well-founded semantics.

In this paper, we motivate some basic principles which must satisfy any codification of an argumentation framework as logic program. We show 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 reasoning in a metainterpreter for argumentation theory is to change the logic programming semantics which is applied to the suitable codification. Also, a suitable codification could be an 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.

Also, we show that rewriting systems not only permit to study abstract argumentation semantics but also they are useful tools for describing the interaction of arguments of an argumentation framework.

Acknowledgement

J.C. Nieves thanks to CONACyT for his PhD Grant. J.C. Nieves and U. Cortés were partially supported by the grant FP6-IST-002307 (ASPIC). Authors want to acknowledge the advice of Dr. Antonio López-Navidad and Dr. Francisco Caballero in the design of the medical example. The views expressed in this paper are not necessarily those of ASPIC consortium.

References

[1] L. Amgoud and H. Prade. Using arguments for making decisions: A possibilistic logic approach. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 10–17, Arlington, Virginia, 2004. AUAI Press.

[2] ASPIC:Project. *Deliverable D2.2:Formal semantics for inference and decision-making*. Argumentation Service Plarform with Integrated Components, 2005.

[3] P. Baroni and M. Giacomin. Evaluating argumentation semantics with respect to skepticism adequacy. In *ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings*, volume 3571 of *LNCS*, pages 329–340. Springer, 2005.

[4] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.

[5] P. Besnard and S. Doutre. Characterization of semantics for argument systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 183–193. AAAI Press, 2004.

[6] P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004),*, pages 59–64, June 2004.

[7] S. Brass, U. Zukowski, and B. Freitag. Transformation-based bottom-up computation of the well-founded model. In *NMELP*, pages 171–201, 1996.

[8] F. Caballero, A. López-Navidad, M. Perea, C. Cabrer, L. Guirado, and R. Solá. Successful liver and kidney transplantation from cadaveric donor with left-sided bacterial endocarditis. *American Journal of Transplantation*, 5:781–787, 2005.

[9] 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.

[10] C. I. Chesñevar, A. G. Maguitman, and R. P. Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.

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

[12] J. Dix, M. Osorio, and C. Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Ann. Pure Appl. Logic*, 108(1-3):153–188, 2001.

[13] 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.

[14] A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.

[15] A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[16] 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.

[17] A. López. Implementing pstable. In R. Dávila, M. Osorio, and C. Zepeda, editors, *Workshop in Logic, Language and Computation*, volume 220. CEUR Workshop Proceedings, 2006.

[18] J. C. Nieves and M. Osorio. Inferring preferred extensions by pstable semantics. Research Report LSI-07-26-R, Technical University of Catalonia, Software Department (LSI), http://www.lsi.upc.edu/dept/techreps/buscar.php, 2007. Accepted paper in the third Latin American Workshop on Non-Monotonic Reasoning 2007 (LANMR07).

[19] 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.

[20] 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.

[21] 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.

[22] 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.

Appendix A : Proof of Lemma 2

Proof. The proof is by induction on the minimum number of steps N to get the Ψ_{AF}'s formal form. Let F_{AF} be the characteristic function of AF. It is well-known that given any normal program P, $WFS(P) := SEM(norm_{CS_0}(P))$. So, let $\langle f(D), f(S)\rangle$ be the well-founded model of Ψ_{AF}.

Base Step If $N = 0$, then Ψ_{AF} is in its normal form. By the definition of Ψ_{AF}, if $A \in AR$ such that $D(A) = \emptyset$, then $d(A) \notin HEAD(\Psi_{AF})$ this means $d(A) \in f(S)$ and $A \in F^0_{AF}$. And also, it is easy to see that F^0_{AF} is the fix point of F_{AF} and $f(F^0_{AF}) = f(S)$. So, S is the grounded extension of AF

Inductive step Now, let us suppose that Ψ_{AF} is not in its normal form, so we need N steps to get its normal form. Let $\langle f(D'), f(S')\rangle$ be $SEM(\Psi_{AF})$ and $\langle f(D''), f(S'')\rangle$ be $SEM(\Psi''_{AF})$ such that $\Psi_{AF} \rightarrow^T \Psi''_{AF}$ and $T \in CS_0$. If $A \in AR$ such that $D(A) = \emptyset$, then $d(A) \notin HEAD(\Psi_{AF})$, $d(A) \in f(S')$, $A \in F^m_{AF}$, and $m \geq 0$. There are two relevant cases *w.r.t.* argument A:

(1) If $B \in AR$ such that B is attacked by A, then there is a rule $r_1 \in \Psi_{AF}$ of the form $r_1 : d(B) \leftarrow \neg d(A)$; therefore, if $T = RED^+$, then $d(B) \leftarrow \top \in \Psi''_{AF}$ and $d(B) \in f(D'')$. This means B is a defeated argument and $B \notin F^m_{AF}$.

(2) If B is defended by A, then there is a rule $r_2 \in \Psi_{AF}$ of the form $r_2 : d(B) \leftarrow d(X_1), \ldots, d(A), \ldots, d(X_n)$, where $X_i \in AR$ such that X_i defends B; therefore, r_2 is deleted by $Failure$. This means, if $T = Failure$, then $r_2 \notin \Psi''_{AF}$. Notice that, if $d(B) \notin HEAD(\Psi''_{AF})$, then $d(B) \in f(S'')$ and $B \in F^m_{AF}$.

One can see that the application of CS_0 over Ψ_{AF} will remove from Ψ_{AF} any rule $r \in \Psi_{AF}$ such that r's head is an atom of the form $d(A)$ and A is an acceptable argument. So, by inductive hypothesis, it is easy to see that if $\langle f(D), f(S)\rangle$ is the well-founded model of Ψ_{AF} then S is the grounded extension of AF.

■