Extending the Grounded Semantics by Logic Programming Semantics

Juan Carlos NIEVES [a,1], Mauricio OSORIO [b] and Ulises CORTÉS [a]
[a] *KEMLg, Software Department, Universitat Politècnica de Catalunya, Spain*
[b] *Fundación Universidad de las Américas - Puebla, México*

Abstract. We introduce a formal argumentation method based on normal programs and rewriting systems which is able to define extensions of the grounded semantics based on specific rewriting rules which perform particular kind of reasoning as in reasoning by cases. These new argumentation semantics are intermediate argumentation semantics between the grounded and the preferred semantics.

Keywords. Argumentation Theory, Normal Logic Programs, Rewriting Systems.

Introduction

Although several approaches have been proposed for argument theory, Dung's approach presented in [7], 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* [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 [1]. Also we can say that the grounded semantics approach is one of the most useful argumentation approaches in real argumentation-based systems [5,9].

Since Dung's framework was introduced in [7], it was shown that this approach can be viewed as a special form of logic programming with *negation as failure*. In fact,

[1]Correspondence to: Software Department (LSI), Universitat Politècnica de Catalunya. C/Jordi Girona 1-3, E08034, Barcelona, Spain. E-mail: (J.C. Nieves: jcnieves@lsi.upc.edu), (M. Osorio: osoriomauri@googlemail.com), (U. Cortés: ia@lsi.upc.edu)

this result was used for introducing a general logic programming method for generating metainterpreters for argumentation theory.

According to Bench-Capon and Dunne [2], the principal abstract argumentation semantics introduced by Dung exhibit a variety of problems: *Emptiness*, *Non-existence* and *Multiplicity* [2]. Since the grounded semantics is sceptical semantics which always exists, the main problem of it is emptiness. The main issue of this paper is to explore some extension of the grounded 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 and they are intermediate argumentation semantics between the grounded and the preferred semantics. The logic programming semantics which are considered for extending the grounded semantics are extension of the well-founded semantics [6].

The rest of the paper is structured as follows: In §1, some concepts of argumentation theory and logic programming are presented. In §2, a transformation of an argumentation framework into a normal logic program is presented. In 3, our extensions of the grounded semantics are presented. In the last section, our conclusions are discussed.

1. 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.

1.1. 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 $\mathcal{L}$.

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. 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.

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 $P^{false} := \{p|\ p \in \mathcal{L}_P \backslash HEAD(P)\}$.*

1.2. Well-Founded Semantics

We start presenting some basic transformation rules for normal logic programs which will be considered for characterizing *WFS*.

Definition 2 (Basic Transformation Rules) *[6] 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 SEM$_{\mathcal{CS}}$ as follows: SEM$_{\mathcal{CS}}(P) :=$ SEM$(norm_{\mathcal{CS}}(P))$.

WFS is one of the most acceptable semantics in logic programming. It was introduced in [10] and was characterized in terms of rewriting systems in [3]. This characterization is defined as follows:

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

1.3. 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 [7]).

Definition 3 *An argumentation framework is a pair $AF := \langle AR, attacks\rangle$, where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR, i.e. attacks $\subseteq AR \times AR$. *We write $\mathcal{AF}_{AR}$ to denote the set of all the argumentation frameworks defined over AR.*

We say that *a attacks b* (or b is attacked by a) if $attacks(a, b)$ holds. Similarly, we say that a set S of arguments attacks b (or b is attacked by S) if b is attacked by an argument in S.

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

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

Definition 5 *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*

The grounded semantics is defined in terms of a *characteristic function*.

Definition 6 *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 7 *The grounded extension of an argumentation framework AF, denoted by GE_{AF}, is the least fixed point of F_{AF}*

2. Mapping from argumentation frameworks to normal programs

In order to see an argumentation framework as a normal program, we introduce a mapping from an argumentation framework to a normal logic program. This mapping was introduced in [11].

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[2]. We define a transformation function *w.r.t.* an argument as follows.

Definition 8 *Let $AF := \langle AR, Attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Psi(A)$ as follows:*

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

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

[2]Given AF = $\langle AR, Attacks \rangle$ and $A \in AR$. $D(A) := \{B | (B, A) \in Attacks\}$.

Definition 9 *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework. We define its associated normal program as follows: $\Psi_{AF} := \bigcup_{A\in AR} \Psi(A)$.*

3. WFS' extensions and the grounded semantics

In this section, we present the main results of our paper. In particular, we introduce a set of extensions of the grounded semantics by using a set of extensions of the well-founded 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 $E \subseteq AR$.

In [4], it was showed that Ψ_{AF} and *WFS* characterize the grounded semantics 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}.*

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 10 (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 $c \in AR \setminus \{S \cup D\}$.*

3.1. $WFS^{LLC'}$ semantics

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

Definition 11 (LLC') *[6] 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 *[6] 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 12 *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 an 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 1 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where AR :=* $\{a, b, c\}$ *and attacks :=* $\{(a, a), (a, b), (b, c), (c, b)\}$. *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 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 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}*

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

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 $d(c) \leftarrow d(c)$).*

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

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

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

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

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

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.

3.2. 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 13 (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}.

Since *Weak-Cases* is an instance of the transformation rule *T-Weak-Cases*, which is defined in [6], this lemma is straightforward from Theorem 7.11 of [6].

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

Definition 14 *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.

Example 2 *Let* $AF := \langle AR, attacks \rangle$ *be an argumentation framework, where* $AR := \{a, b, c, d\}$ *and* $attacks := \{(a, b), (b, a), (a, c), (b, c), (c, d)\}$. *Then,* Ψ_{AF} *is:*

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

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}.*

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

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

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

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.

3.3. $WFS^{WK+LCC'}$ 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 $CS_3 := CS_0 \cup \{LLC', \textit{Weak-Cases}\}$. 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 $WFS^{WK+LLC'}$.*

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

Definition 15 *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 3 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e,\ f, m, n, p\}$ and $attacks := \{(a, b),\ (b, c),\ (c, a),\ (a, d),\ (d, e),\ (e, f),\ (m, e),\ (n, m), (n, p), (p, m),\ (p, n)\}$. 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.*

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

3.4. Formalizing the extensions of the grounded 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 1 *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.*

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 stronger than *WFS* and weaker than the pstable semantics (the formal definition of pstable semantics is presented in [12]). Remember that the pstable models of Ψ_{AF} correspond to the preferred extensions of AF [4].

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 2 *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 $D \subseteq E'$.*

4. Conclusions and Future Work

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 particular in this paper, we showed that by using extension of the well-founded semantics, it is possible to define an intermediate reasoning between the grounded semantics and the preferred semantics.

A fundamental step in our future research is to explore the relation between the extensions of the grounded semantics introduced in this paper and the ideal semantics [8]. In fact, we can see that at least with the examples of this paper and the new extensions of the grounded semantics coincide with the ideal semantics.

References

[1] 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.

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

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

[4] J. L. Carballido, J. C. Nieves, and M. Osorio. Inferring Preferred Extensions by Pstable Semantics. *Iberoamerican Journal of Artificial Intelligence (Inteligencia Artificial) ISSN: 1137-3601, (doi: 10.4114/ia.v13i41.1029)*, 13(41):38–53, 2009.

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

[6] 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.

[7] 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.

[8] P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(issues 10-15):642–674, 2007.

[9] 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.

[10] 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.

[11] J. C. Nieves, M. Osorio, and U. Cortés. Preferred Extensions as Stable Models. *Theory and Practice of Logic Programming*, 8(4):527–543, July 2008.

[12] 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.