Expressing Extension-Based Semantics based on Stratified Minimal Models

Juan Carlos Nieves[1] Mauricio Osorio[2] and Claudia Zepeda[3]

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

Abstract Extension-based argumentation semantics is a successful approach for performing non-monotonic reasoning based on argumentation theory. An interesting property of some extension-based argumentation semantics is that these semantics can be characterized in terms of logic programming semantics. In this paper, we present novel results in this topic. In particular, we show that one can induce an argumentation semantics (that we call Stratified Argumentation Semantics) based on a logic programming semantics that is based on stratified minimal models. We show that the stratified argumentation semantics overcome some problems of extension-based argumentation semantics based on admissible sets and we show that it coincides with the argumentation semantics CF2.
Keywords: Non-monotonic reasoning, extension-based argumentation semantics and logic programming.

1 Introduction

Argumentation theory has become an increasingly important and exciting research topic in Artificial Intelligence (AI), with research activities ranging from developing theoretical models, prototype implementations, and application studies [3]. 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.

Dung's approach, presented in [6], is a unifying framework which has played an influential role on argumentation research and AI. This approach is mainly orientated to manage the interaction of arguments. The interaction of the arguments is supported by four extension-based argumentation semantics: *stable*

semantics, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of these semantics is the *acceptability of the arguments*. It is worth mentioning that although these argumentation semantics represents different pattern of selection of arguments, all these argumentation semantics are based on the concept of admissible set.

An important point to remark *w.r.t.* the argumentation semantics based on admissible sets is that these semantics exhibit a variety of problems which have been illustrated in the literature [17,2,3]. For instance, let AF be the argumentation framework which appears in Figure 1-a. In this AF there are two arguments: a and b. The arrows in the figure represent conflicts between the arguments. We can see that the argument a is attacked by itself and the argument b is attacked by the argument a. Some authors as Prakken and Vreeswijk [17] suggest in a intuitive way, that one can expect that the argument b can be considered as an acceptable argument since it is attacked by the argument a which is attacked by itself. However, none of the argumentation semantics suggested by Dung is able to infer the argument b as acceptable.

Figure 1. In **a)**, it is presented the graph representation of the argumentation framework: $\langle\{a, b\}, \{(a, a), (a, b)\}\rangle$. In **b)**, it is presented the graph representation of the argumentation framework: $\langle\{a, b, c, d, e\}, \{(a, c), (c, b), (b, a), (a, d), (c, d), (b, d), (d, e)\}\rangle$

Another interesting argumentation framework which has been commented on literature [17,2] is presented in Figure 1-b.

Some authors, as Prakken and Vreeswijk [17], Baroni *et al*[2], suggest that the argument e can be considered as an acceptable argument since it is attacked by the argument d which is attacked by three arguments: a, b, c. Observe that the arguments a, b and c form a cyclic of attacks.

We can recognize two major branches for improving Dung's approach. On the one hand, we can take advantage of graph theory; on the other hand, we can take advantage of logic programming with negation as failure.

With respect to graph theory, the approach suggested by Baroni *et al*, in [2] is maybe the most general solution defined until now for improving Dung's approach. This approach is based on a solid concept in graph theory which is a *strongly connected component* (SCC). Based on this concept, Baroni *et al*, describe a recursive approach for generating new argumentation semantics. For instance, the argumentation semantics CF2 suggested in [2] is able to infer the

argument b as an acceptable argument of the AF of Figure 1-a. Also CF2 regards the argument e as an acceptable argument from the AF of Figure 1-b.

Since Dung's approach was introduced in [6], it was viewed as a special form of logic programming with *negation as failure*. For instance, in [6] it was proved that the grounded semantics can be characterized by the well-founded semantics [8], and the stable argumentation semantics can be characterized by the stable model semantics [9]. Also in [4], it was proved that the preferred semantics can be characterized by the p-stable semantics [16]. In fact, the preferred semantics can be also characterized by the minimal models and the stable models of a logic program [14]. By regarding an argumentation framework in terms of logic programs, it has been shown that one can construct intermediate argumentation semantics between the grounded and preferred semantics [12]. Also it is possible to define extensions of the preferred semantics [15].

When we have a logic program which represents an argumentation framework, it is natural to think that we can split this program into subprograms where each subprogram could represents a part of an argumentation framework. The idea of splitting a logic program into its component, in order to define logic programming semantics, has been explored by some authors in logic programming [5]. For instance, by splitting a logic program, Dix and Müller in [5] combine ideas of the stable model semantics and the well-founded semantics in order to define a skeptical logic programming semantics which satisfies the property of relevance and the general principle of partial evaluation.

In this paper, we are going to explore the idea of splitting a logic program into its components in order to achieve two main objectives:

1. To explore the definition of candidate argumentation semantics in terms of logic programming semantics. In particular, we define an extension-based argumentation semantics, that we call *stratified argumentation semantics*. This semantics will be induced by the stratified minimal model semantics. We will show that this new argumentation semantics coincides with CF2 which is considered as the most acceptable argumentation semantics introduced in [2].
2. To introduce a recursive construction which define a new logic programming semantics, that we call *stratified minimal models semantics*. Based on the construction of this semantics, we will show that there exists a family of logic programming semantics that are always defined and satisfy the property of relevance.

The rest of the paper is divided as follows: In §2, we present some basic concepts *w.r.t.* logic programming and argumentation theory. In §3, we define the stratified minimal model semantics and introduce our first main theorem. In §4, we introduce the stratified argumentation semantics and present our second main theorem of this paper. Finally in the last section, we present our conclusions.

2 Background

In this section, we define the syntax of the logic programs that we will use in this paper and some basic concepts of logic programming semantics and argumentation semantics.

2.1 Syntax and some operations

A signature $\mathcal{L}$ is a finite set of elements that we call atoms. A *literal* is either an atom a, called *positive literal*; or the negation of an atom $\neg a$, called *negative literal*. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of atoms $\{\neg a_1, ..., \neg a_n\}$. A *normal* clause, C, is a clause of the form

$$a \leftarrow b_1 \wedge \ldots \wedge b_n \wedge \neg b_{n+1} \wedge \ldots \wedge \neg b_{n+m}$$

where a and each of the b_i are atoms for $1 \leq i \leq n+m$. In a slight abuse of notation we will denote such a clause by the formula $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ where the set $\{b_1, \ldots, b_n\}$ will be denoted by $\mathcal{B}^+$, and the set $\{b_{n+1}, \ldots, b_{n+m}\}$ will be denoted by $\mathcal{B}^-$. We define a *normal program* P, as a finite set of normal clauses. If the body of a normal clause is empty, then the clause is known as a *fact* and can be denoted just by: $a \leftarrow$.

We write $\mathcal{L}_P$, to denote the set of atoms that appear in the clauses of P. We denote by $HEAD(P)$ the set $\{a | a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^- \in P\}$.

A program P induces a notion of *dependency* between atoms from $\mathcal{L}_P$. We say that a *depends immediately on* b, if and only if, b appears in the body of a clause in P, such that a appears in its head. The two place relation *depends on* is the transitive closure of *depends immediately on*. The set of dependencies of an atom x, denoted by *dependencies-of*(x), corresponds to the set $\{a \mid x \textit{ depends on } a\}$. We define an equivalence relation $\equiv$ between atoms of $\mathcal{L}_P$ as follows: $a \equiv b$ if and only if $a = b$ or (a *depends on* b and b *depends on* a). We write $[a]$ to denote the equivalent class induced by the atom a.

Example 1. Let us consider the following normal program,

$$S = \{e \leftarrow e,\ \ c \leftarrow c,\ \ a \leftarrow \neg b \wedge c,\ \ b \leftarrow \neg a \wedge \neg e,\ \ d \leftarrow b\}.$$

The dependency relations between the atoms of $\mathcal{L}_S$ are as follows: *dependencies-of*$(a) = \{a, b, c, e\}$; *dependencies-of*$(b) = \{a, b, c, e\}$; *dependencies-of*$(c) = \{c\}$; *dependencies-of*$(d) = \{a, b, c, e\}$; and *dependencies-of*$(e) = \{e\}$. We can also see that, $[a] = [b] = \{a, b\}$, $[d] = \{d\}$, $[c] = \{c\}$, and $[e] = \{e\}$.

We take $<_P$ to denote the strict partial order induced by $\equiv$ on its equivalent classes. Hence, $[a] <_P [b]$, if and only if, b *depends-on* a and $[a]$ is not equal to $[b]$. By considering the relation $<_P$, each atom of $\mathcal{L}_P$ is assigned an order as follows:

- An atom a *is of order* 0, if $[a]$ is minimal in $<_P$.
- An atom a *is of order* $n+1$, if n is the maximal order of the atoms on which a depends.

We say that a program P is of order n, if n is the maximum order of its atoms. We can also break a program P of order n into the disjoint union of programs P_i with $0 \leq i \leq n$, such that P_i is the set of rules for which the head of each clause is of order i (*w.r.t.* P). We say that $P_0, \ldots, P_n$ are the *relevant modules* of P.

Example 2. By considering the equivalent classes of the program S in Example 1, the following relations hold: $\{c, e\} <_S \{a, b\} <_S \{d\}$. We also can see that: a is of order 1, d is of order 2, b is of order 1, e is of order 0, and c is of order 0. This means that S is a program of order 2.

The following table illustrates how the program S can be broken into the disjoint union of the following relevant modules S_0, S_1, S_2:

S	S_0	S_1	S_2
$e \leftarrow e.$	$e \leftarrow e.$		
$c \leftarrow c.$	$c \leftarrow c.$		
$a \leftarrow \neg b \wedge c.$		$a \leftarrow \neg b \wedge c.$	
$b \leftarrow \neg a \wedge \neg e.$		$b \leftarrow \neg a \wedge \neg e.$	
$d \leftarrow b.$			$d \leftarrow b.$

Now we introduce a single reduction for any normal program. The idea of this reduction is to remove from a normal program any atom which has already fixed to some true value. In fact, this reduction is based on a pair of sets of atoms $\langle T; F\rangle$ such that the set T contains the atoms which can be considered as true and the set F contains the atoms which can be considered as false. Formally, this reduction is defined as follows:

Let $A = \langle T; F\rangle$ be a pair of sets of atoms. The reduction $R(P, A)$ is obtained by 4 steps:

1. We replace every atom x that occurs in the bodies of P by 1 if $x \in T$, and we replace every atom x that occurs in the bodies of P by 0 if $x \in F$;
2. we replace every occurrence of $\neg 1$ by 0 and $\neg\ 0$ by 1;
3. every clause with a 0 in its body is removed;
4. finally we remove every occurrence of 1 in the body of the clauses.

We want to point out that this reduction does not coincide with the Gelfond-Lifschitz reduction [9].

Example 3. Let us consider the normal program S of Example 1. Let P be the normal program $S \setminus S_0$, and let A be the pair of sets of atoms $\langle\{c\}; \{e\}\rangle$. This means that we obtain the following programs:

P:
$a \leftarrow \neg b \wedge c.$
$b \leftarrow \neg a \wedge \neg e.$
$d \leftarrow b.$

$R(P, A)$:
$a \leftarrow \neg b.$
$b \leftarrow \neg a.$
$d \leftarrow b.$

2.2 Semantics

From now on, we assume that the reader is familiar with the single notion of *minimal model*. In order to illustrate this basic notion, let P be the normal program $\{a \leftarrow \neg b, \quad b \leftarrow \neg a, \quad a \leftarrow \neg c, \quad c \leftarrow \neg a\}$. As we can see, P has five models: $\{a\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b\}$, $\{a, b, c\}$; however, P has just two minimal models: $\{b, c\}$, $\{a\}$. We will denote by $MM(P)$ the set of all the minimal models of a given logic program P. Usually MM is called *minimal model semantics*.

A semantics SEM is a mapping from the class of all programs into the powerset of the set of (2-valued) models. SEM assigns to every program P a (possible empty) set of (2-valued) models of P. If $SEM(P) = \emptyset$, then we informally say that SEM is undefined for P.

Given a set of interpretations Q and a signature $\mathcal{L}$, we define Q *restricted to* $\mathcal{L}$ as $\{M \cap \mathcal{L} \mid M \in Q\}$. For instance, let Q be $\{\{a, c\}, \{c, d\}\}$ and $\mathcal{L}$ be $\{c, d, e\}$, hence Q *restricted to* $\mathcal{L}$ is $\{\{c\}, \{c, d\}\}$.

Let P be a program and $P_0, \ldots, P_n$ its relevant modules. We say that a semantics S satisfies the property of relevance if for every i, $0 \leq i \leq n$, $S(P_0 \cup \cdots \cup P_i) = S(P)$ restricted to $\mathcal{L}_{P_0 \cup \cdots \cup P_i}$.

2.3 Argumentation basics

Now, we present some basic concepts with respect to extended-based argumentation semantics. The first concept that we consider is the one of *argumentation framework*. An argumentation framework captures the relationships between the arguments.

Definition 1. *[6] An argumentation framework is a pair* $AF = \langle AR, \text{attacks} \rangle$*, 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 $(a, b) \in attacks$ holds. Usually an extension-based argumentation semantics S_{Arg} is applied to an argumentation framework AF in order to infer sets of acceptable arguments from AF. An extension-based argumentation semantics S_{Arg} is a function from $\mathcal{AF}_{AR}$ to 2^{AR}. S_{Arg} can be regarded as a pattern of selection of sets of arguments from a given argumentation framework AF.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, we will say that an argument $a \in AR$ is acceptable, if $a \in E$ such that $E \in S_{Arg}(AF)$.

3 Stratified Minimal Model Semantics

In this section, we introduce a constructive logic programming semantics, called *stratified minimal model semantics*, which is based on minimal models. This semantics has some interesting properties as: it satisfies the property of relevance, and it agrees with the stable model semantics for the well-known class of stratified logic programs (the proof of this property can be found in [12,13]).

In order to define the stratified minimal model semantics MM^r, we define the operator $*$ and the function $freeTaut$ as follows:

- Given Q and L both sets of interpretations, we define $Q * L := \{M_1 \cup M_2 \mid M_1 \in Q, M_2 \in L\}$.
- Given a logic program P, $freeTaut$ denotes a function which removes from P any tautology.

The idea of the function $freeTaut$ is to remove any clause which is equivalent to a tautology in classical logic.

Definition 2. *Given a normal logic program P, we define the* sstratified minimal model semantics *MM^r as follows: $MM^r(P) = MM^r_c(freeTaut(P) \cup \{x \leftarrow x \mid x \in \mathcal{L}_P \setminus HEAD(P)\}$ such that $MM^r_c(P)$ is defined as follows:*

1. *if P is of order 0, $MM^r_c(P) = MM(P)$.*
2. *if P is of order $n > 0$, $MM^r_c(P) = \bigcup_{M \in MM(P_0)} \{M\} * MM^r_c(R(Q, A))$ where $Q = P \setminus P_0$ and $A = \langle M; \mathcal{L}_{P_0} \setminus M \rangle$.*

We call a model in $MM^r(P)$ a stratified minimal model *of P.*

Observe that the definition of the stratified minimal model semantics is based on a recursive construction where the base case is the application of MM. It is not difficult to see that if one changes MM by any other logic programming semantics S, as the stable model semantics, one is able to construct a relevant version of the given logic programming semantics (see [12,13] for details).

In order to introduce an important theorem of this paper, let us introduce some concepts. We say that a normal program P is basic if every atom x that belongs to $\mathcal{L}_P$, then x occurs as a fact in P. We say that a logic programming semantics SEM is *defined for basic programs*, if for every basic normal program P then $SEM(P)$ is defined.

The following theorem shows that there exists a family of logic programming semantics that are always defined and satisfy the property of relevance.

Theorem 1. *For each semantics SEM that is defined for basic programs, there exists a semantics SEM' that satisfies the following:*

1. *For every normal program P, $SEM'(P)$ is defined.*
2. *SEM' is relevant.*
3. *SEM' is invariant under adding tautologies.*

An instantiation of SEM and SEM' of Theorem 1 are the semantics MM and MM^r respectively. Observe that essentially this theorem is suggesting that given any logic programming semantics SEM, such as MM, that is defined for basic program, one can construct a *relative similar semantic SEM'*, such as MM^r, to SEM satisfying the three properties described in this Theorem 1.

4 Stratified Argumentation Semantics

In this section, we show that by considering the stratified minimal model semantics, one can induce an argumentation semantics. In fact, we show that this new argumentation semantics will take advantage of the properties of the stratified minimal model semantics.

As the stratified minimal model semantics is a semantics for logic programs, we require a function mapping able to construct a logic program from an argumentation framework. Hence, let us introduce a simple mapping to regard an argumentation framework as a normal logic program. In this mapping, we use the predicates $d(x)$, $a(x)$. The intended meaning of $d(x)$ is: "the argument x is defeated" (this means that the argument x is attacked by an acceptable argument), and the intended meaning of $a(X)$ is that the argument X is accepted.

Definition 3. *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework, $P^1_{AF} = \{d(a) \leftarrow \neg d(b_1), \ldots, d(a) \leftarrow \neg d(b_n) \mid a \in AR$ and $\{b_1, \ldots, b_n\} = \{b_i \in AR \mid (b_i, a) \in attacks\}\}$; and $P^2_{AF} = \bigcup_{a\in AR}\{a(a) \leftarrow \neg d(a)\}$. We define: $P_{AF} = P^1_{AF} \cup P^2_{AF}$.*

The intended meaning of the clauses of the form $d(a) \leftarrow \neg d(b_i)$, $1 \leq i \leq n$, is that an argument a will be defeated when anyone of its adversaries b_i is not defeated. Observe that, essentially, P^1_{AF} is capturing the basic principle of *conflict-freeness* (this means that any set of acceptable argument will not contain two arguments which attack each other). The idea P^2_{AF} is just to infer that any argument a that is not defeated is accepted.

Example 4. Let AF be the argumentation framework of Figure 1-b. We can see that $P_{AF} = P^1_{AF} \cup P^2_{AF}$ is:

P^1_{AF} :

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

P^2_{AF} :

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

Two relevant properties of the mapping P_{AF} are that the stable models of P_{AF} characterize the stable argumentation semantics and the well founded model of P_{AF} characterizes the grounded semantics [12].

Once we have defined a mapping from an argumentation framework into logic programs, we are going to define a candidate argumentation semantics which is induced by the stratified minimal model semantics.

Definition 4. *Given an argumentation framework A, we define a stratified extension of AF as follows: A_m is a stratified extension of AF if exists a stratified minimal model M of P_{AF} such that $A_m = \{x|a(x) \in M\}$. We write $MM^r_{Arg}(AF)$ to denote the set of stratified extensions of AF. This set of stratified extensions is called stratified argumentation semantics.*

In order to illustrate the stratified argumentation semantics, we are going to presents some examples.

Example 5. Let AF be the argumentation framework of Figure 1-b and P_{AF} be the normal program defined in Example 4. In order to infer the stratified argumentation semantics, we infer the stratified minimal models of P_{AF}. As we can see P_{AF} has three stratified minimal models : $\{d(a), d(b), d(d), a(c), a(e)\}\{d(b), d(c), d(d), a(a), a(e)\}\{d(a), d(c), d(d), a(b), a(e)\}$, this means that AF has three stratified extensions which are: $\{c, e\}$, $\{a, e\}$ and $\{b, e\}$. Observe that the stratified argumentation semantics coincides with the argumentation semantics CF2.

Let us consider another example.

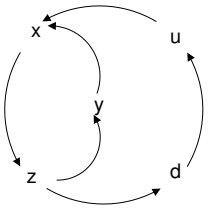

Figure 2. Graph representation of $AF = \langle\{x, y, z, u, d\}, \{(x, z),\ (z, y),\ (y, x),\ (u, x),\ (z, d), (d, u)\}\rangle$.

Example 6. Let us consider the argumentation framework of Figure 2. It is not difficult to obtain its $P_{AF} = P^1_{AF} \cup P^2_{AF}$ where P^1_{AF} and P^2_{AF} correspond to the following programs:

P^1_{AF} :

$d(x) \leftarrow \neg d(y).$
$d(y) \leftarrow \neg d(z).$
$d(z) \leftarrow \neg d(x).$
$d(x) \leftarrow \neg d(u).$
$d(d) \leftarrow \neg d(z).$
$d(u) \leftarrow \neg d(d).$

P^2_{AF} :

$a(x) \leftarrow \neg d(x).$
$a(y) \leftarrow \neg d(y).$
$a(z) \leftarrow \neg d(z).$
$a(d) \leftarrow \neg d(d).$
$a(u) \leftarrow \neg d(u).$

Now let us compute the argumentation semantics MM^r_{Arg}. Since $MM^r(P_{AF}) =$ $\{\ \{d(y), d(z), d(u), a(x), a(d)\},\ \{d(x), d(z), d(d), a(y), a(u)\},\ \{d(u), d(x), d(z),$ $a(y), a(d)\},\ \{d(x), d(y), d(d), a(z), a(u)\}\ \}$ then, $MM^r_{Arg}(AF) = \{\ \{x, d\}, \{y, u\},$ $\{y, d\}, \{z, u\}\ \}$. Notice that MM^r_{Arg} coincides with the argumentation semantics CF2.

We are going to present our second main theorem of this paper. This theorem formalizes that the stratified argumentation semantics and the argumentation semantics CF2 coincide.

Theorem 2. *Given an argumentation framework* $AF = \langle AR, Attacks \rangle$*, and* $E \in AR$*,* $E \in MM^{r}_{Arg}(AF)$ *if and only if* $E \in CF2(AF)$*.*

As final result of this paper, we show an important result *w.r.t.* the decision problem of knowing if a set of arguments is a stratified extension.

Lemma 1. *Given an argumentation framework* $AF = \langle AR, Attacks \rangle$ *and a set of argument* $E \subseteq AR$*, the decision problem of knowing if* E *is a stratified extension of* AF *is polynomial time computable.*

Observe that by this lemma and Theorem 2, one can infer that the decision problem of knowing if a set of arguments belongs to CF2 is polynomial time computable. Recall that on the other hand, the corresponding complexity decision problem for the preferred semantics is CO-NP-Complete [7].

5 Conclusions

It is well-accepted that extension-based argumentation semantics is a promising approach for performing non-monotonic reasoning. However, since in the literature of argumentation has been exhibited a variety of problems of some of the existing argumentation semantics, nowadays it has increased the number of new argumentation semantics in the context of Dung's argumentation approach. We have to recognize that many of these new argumentation semantics are only motivated by particular examples, and also these introduced argumentation semantics lack of logic foundations. In this paper, we show that one can induce novel argumentation semantics by considering logic programming semantics. In particular, we introduce a novel argumentation semantics (stratified argumentation semantics) based on a new logic programming semantics (stratified minimal model semantics). In fact, we show that the stratified argumentation semantics coincides with the argumentation semantics CF2 which was introduced in terms of graph theory's terms (Theorem 2). It is worth mentioning that the stratified argumentation semantics is just one of the multiples candidate argumentation semantics that can be induced by the family of logic programming semantics identified by Theorem 1 (for more details about other new candidate argumentation semantics see [12,13]).

An important property of the stratified argumentation semantics is that the decision problem of knowing if a set of arguments is a stratified extension is polynomial time computable. This means that this semantics is computationally less expensive than the preferred semantics. This result also suggests that the decision problem of knowing if a set of arguments belongs to CF2 is polynomial time computable. We believe that the study of argumentation semantics in terms of logic programming semantics could help to explore the non-monotonic properties of the argumentation semantics. The study of non-monotonic properties of an argumentation semantics could suggests some guidelines in order to find suitable argumentation semantics for the applications of these to real domains.

Acknowledgement

We are grateful to anonymous referees for their useful comments. We would like to acknowledge support from the EC founded project ALIVE (FP7-IST-215890). The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

1. J. Bang-Jensen and G. Gutin. Digraphs: Theory, Algorithms and Applications, second Editions, Springer edition, 2008.
2. P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.
3. T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.
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*, 13(41):38–53, 2009.
5. J. Dix and M. Müller. Partial evaluation and relevance for approximations of stable semantics. In *ISMIS*, volume 869 of *Lecture Notes in Computer Science*, pages 511–520. Springer, 1994.
6. 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.
7. P. E. Dunne and T. J. Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141(1):187–203, 2002.
8. 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.
9. 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.
10. A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook in Artificial Intelligence and Logic Programming, Volume 5*, pages 235–324. Oxford University Press, Oxford, 1998.
11. A. C. Kakas and P. Mancarella. Generalized stable models: A semantics for abduction. In *ECAI*, pages 385–391, 1990.
12. J. C. Nieves. *Modeling arguments and uncertain information — A non-monotonic reasoning approach.* PhD thesis, Software Department (LSI), Technical University of Catalonia, 2008.
13. J. C. Nieves and M. Osorio. A General Schema For Generating Argumentation Semantics From Logic Programming Semantics. Research Report LSI-08-32-R, Technical University of Catalonia, Software Department (LSI), http://www.lsi.upc.edu/dept/techreps/buscar.php, 2008.
14. 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.
15. J. C. Nieves, M. Osorio, U. Cortés, I. Olmos, and J. A. Gonzalez. Defining new argumentation-based semantics by minimal models. In *Seventh Mexican International Conference on Computer Science (ENC 2006)*, pages 210–220. IEEE Computer Science Press, September 2006.

16. 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.
17. 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 theorem 1

Proof. The proof is by construction.

First of all, we recall some definitions about the notion of generalized S model.

Let S be a logic programming semantics, P be a logic program and A be a set of atoms (called abductives) such that $A \subseteq \mathcal{L}_P$. We say that M_B is a *generalized S model*[1] of P with respect to A if $M \in S(P \cup B)$ where $B \subseteq A$ and $M \subseteq \mathcal{L}_P$. It is also possible to define a partial order between generalized S models (with respect to A) of a program according to the set inclusion with respect to the subindex B. We say that M is a *minimal generalized S model* of P with respect to A if there exists a set of atoms B, such that M_B is a generalized S model of P with respect to A and M_B is minimal with respect to the partial order just defined.

We write S^* to denote the *minimal generalized S semantics*, where $A = \mathcal{L}_P$. Namely $S^*(P)$ is the collection of minimal generalized S models of P with respect to $\mathcal{L}_P$. Observe that in our definition we are not instantiating the definition to a particular logic programming semantics.

It is immediate to verify that for every semantics S and program P, $S^*(P)$ is defined.

Now, let S be a semantics that is always defined. We define the associate S^r semantics recursively as follow: Given a program P of order 0, $S^r(P) = S(P)$. For a program P of order $n > 0$ we define $S^r(P) = \bigcup_{M \in S(P_0)} \{M\} * S^r(R(Q, A))$ where $Q = P \setminus P_0$ and $A = \langle M; \mathcal{L}_{P_0} \setminus M \rangle$.

Note that if S is always defined then S^r is always defined. More over S^r is relevant by construction.

Our final semantics is the following: Let P be a normal program. Let $freetaut(P)$ be program P after removing every tautology. Let $tautp(P) = freetaut(P) \cup \{x \leftarrow x : x \in \mathcal{L}_P\}$. Then

$$S'(P) = S^{*^r}(tautp(P)).$$

Clearly $S'(P)$ is always defined and relevant and invariant under adding tautologies.

[1] The concept of generalized S model is closely related to the semantics of *abductive logic programming* [11,10], in particular to the concept of *generalized answer set*.

Appendix B: Proof of theorem 2

In order to present the proof of Theorem 2, we are going to present some definitions *w.r.t.* the argumentation semantics CF2[2], and we are going to show some lemmas.

Definition 5. *A set S of arguments is said to be conflict-free if there are no arguments a, b in S such that a attacks b.*

We will denote by $max\_conflict\_freeSets(AF)$ the set of maximal conflict free sets (*w.r.t.* set inclusion) of an argumentation framework AF.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, the binary relation of *path-equivalence* between nodes, denoted as $PE_{AF} \subseteq (AR \times AR)$, is defined as follows:
— $\forall \alpha \in AR$, $(\alpha, \alpha) \in PE_{AF}$,
— given two distinct nodes α, $\beta \in AR$, $(\alpha, \beta) \in PE_{AF}$ if and only if there is a path from α to β and a path from β to α.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, the *strongly connected components* of AF are the equivalence classes of nodes under the relation of path-equivalence. The set of the strongly connected components of AF is denoted as $SCCS_{AF}$. Given a node $\alpha \in AR$, the strongly connected component α belongs to is denoted as $SCC_{AF}(\alpha)$.

Now, given an argumentation framework, let $AF = \langle AR, attacks \rangle$, and $S \subseteq AR$, the restriction of AF to S is the argumentation framework $AF \downarrow_S = \langle S, attacks \cap (S \times S) \rangle$.

Considering an argumentation framework, $AF = \langle AR, attacks \rangle$, a set $E \subseteq AR$ and a strongly connected component $S \in SCCS_{AF}$, the set $D_{AF}(S, E)$ consists of the nodes of S attacked by E from outside S, the set $U_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are defended by E (i.e., their defeaters from outside S are all attacked by E), and $P_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are not defended by E (i.e., at least one of their defeaters from outside S is not attacked by E). Finally, $UP_{AF}(S, E) = (S \setminus D_{AF}(S, E)) = (U_{AF}(S, E) \cup P_{AF}(S, E))$.

Here, we define $GF(AF, C)$ for an argumentation framework $AF = \langle AR, attacks \rangle$ and a set $C \subseteq A$, representing the defended nodes of AF: two cases have to be considered in this respect.

If AF consists of exactly one strongly connected component, it does not admit a decomposition where to apply the directionality principle, therefore it has to be assumed that $GF(AF, C)$ coincides in this case with a *base function*, denoted as $BF_S(AF, C)$, that must be assigned in order to characterize a particular argumentation semantics S.

On the other hand, if AF can be decomposed into several strongly connected components, then, $GF(AF, C)$ is obtained by applying recursively GF to each

[2] The details of these definitions are presented in [2].

strongly connected component of AF, deprived of the nodes in $D_{AF}(S, E)$. Formally, this means that for any $S \in SCCS_{AF}$, $(E \cap S) \in GF(AF \downarrow_{UP_{AF}(S,E)}, C')$, where C' represents the set of defended nodes of the restricted argumentation framework $AF \downarrow_{UP_{AF}(S,E)}$. The set C' can be determined taking into account both the attacks coming from outside AF and those coming from other strongly connected components of AF.

Definition 6. *A given argumentation semantics S is* SCC-recursive *if and only if for any argumentation framework $AF = \langle AR, \text{attacks}\rangle$, $E_S(AF) = GF(AF, AR)$, where for any $AF = \langle AR, \text{attacks}\rangle$ and for any set $C \subseteq AR$, the function $GF(AF, C) \subseteq 2^{AR}$ is defined as follows: for any $E \subseteq AR$, $E \in GF(AF, C)$ if and only if*

- *in case $|SCCS_{AF}| = 1$, $E \in BF_S(AF, C)$,*
- *otherwise,$\forall S \in SCCS_{AF}(E \cap S) \in GF(AF \downarrow_{UP_{AF(S,E)}}, U_{AF}(S, E) \cap C)$.*

where $BF_S(AF, C)$ is a function, called base function, that, given an argumentation framework $AF = \langle AR, \text{attacks}\rangle$ such that $|SCCS_{AF}| = 1$ and a set $C \subseteq AR$, gives a subset of 2^{AR}.

Observe that Definition 6 does not define any particular semantics, essentially it defines a general schema for defining argumentation semantics. In particular, when $BF_S(AF, C)$ is instantiated by the function which returns the maximal conflict free sets of AF *w.r.t.* C, Definition 6 defines $CF2$.

The following lemma shows that the number of strongly connected components of an argumentation framework AF is the same to the number of components of the normal logic program P_{AF}.

Lemma 2. *Let AF be an argumentation framework. If $P_{AF} = P^1_{AF} \cup P^2_{AF}$ such that P^1_{AF} is of order n, then $|SCC_{AF}| = n + 1$.*

Proof. (sketch) Since the number of components of P^1_{AF} depends on the number of equivalent classes of atoms in $\mathcal{L}_{P_{AF}}$ and the number of strongly connected components depends on the number of equivalent classes of nodes in PE_{AF}, the proof follows from that fact that:

- The number of equivalent classes of atoms induced by the relation *depends on* in $\mathcal{L}_{P_{AF}}$ is the same to the number of classes of nodes induces by the relation *path-equivalence* in PE_{AF}.

Lemma 3. *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. If $P_{AF} = P^1_{AF} \cup P^2_{AF}$ such that P^1_{AF} is of order 0, then $E \in max\_conflict\_freeSets(AF)$ if and only if $\{a(a)|a \in E\} \cup \{d(a)|a \in AR \setminus E\}$ is a minimal model of P_{AF}.*

Proof. Observations:

1. M is a minimal model of P_{AF} if and only if there exists M_1 and M_2 such that $M = M_1 \cup M_2$, M_1 is a minimal model of P^1_{AF} and $M_2 = \{a(a)|a(a) \leftarrow \neg d(a) \in P^2_{AF}, d(a) \notin M_1\}$.

2. If E is a conflict free set of AF, then $M = \{d(a)|a \in AR \setminus E\}$ is a model of P^1_{AF}.
3. If M is a model of P^1_{AF}, then $E = \{a|a(a) \in M\}$ is a conflict free set of AF.

=> If E is a maximal conflict free set of AF, then, by Proposition 1 of [14] and Observation 2, $M_1 = \{d(a)|a \in AR \setminus E\}$ is a minimal model of P^1_{AF}. Hence, by Observation 1, $M_1 \cup \{a(a)|a \in E\}$ is a minimal model of P_{AF}.

<= If $M = M_1 \cup M_2$ such that $E \subseteq AR$, $M_1 = \{d(a)|a \in AR \setminus E\}$, $M_2 = \{a(a)|a \in E\}$ and M is a minimal model of P_{AF}, hence by Observation 1, M_1 is a minimal model of P^1_{AF}. Therefore, by Observation 3 and Proposition 1 of [14], E is a maximal conflict free of AF.

Given the set of strongly connected components $SCC(AF)$, we denote by $\leq_{SCC}$ the partial order between strongly connected components defined in [2]. This partial order is induced by the so called directionality principle and the relation of attack between set of arguments.

Main proof

Proof. **Theorem 2** (sketch) Since the construction of both semantics is recursive, the proof is by induction *w.r.t.* the number of components n of the normal logic program P_{AF}.

Base Step If $n = 0$, then AF has just one strongly connected component (Lemma 2); hence, $MM^r_{Arg}(AF) = CF2(AF)$ by Lemma 3.

Inductive Step If $n > 0$, then the proof follows from the following observations:

1. The partial order $\leq_{SCC}$ and the partial order $<_P$ define equivalent classes of sets of arguments of AF and atoms in $\mathcal{L}_{P_{AF}}$ respectively.
2. The base function for the construction of MM^r_{Arg} and $CF2(AF)$ are equivalent (Lemma 3).

Appendix C: Proof of Lemma 1

Proof. **Lemma 1** (sketch)

The proof follows from the following observations:

1. By the definition of the stratified argumentation semantics, the decision of knowing if the set of arguments E is a stratified extension of AF is reduced to the decision problem of knowing if a given set of atoms M is a minimal model of a logic program P (it is a consequence of the base case of the recursive function $MM^r_c(P)$, see Definition 2).
2. Since there is a relationship between minimal models and logic consequence (see Lemma 1 of [14]), the decision problem of knowing if $M \subseteq \mathcal{L}_{P_{AF}}$ is a minimal model of P_{AF} can be reduced to the decision problem of 2-UNSAT.
3. It is known that the decision problem of 2-UNSAT is polynomial time computable [1].