The stable_abducible argumentation semantics

Mauricio Osorio[1], Juan Carlos Nieves[2], and José Luis Carballido[3]

[1] Universidad de las Américas, CENTIA,
osoriomauri@gmail.com
[2] Universitat Politècnica de Catalunya
jcnieves@lsi.upc.edu
[3] Benemérita Universidad Atónoma de Puebla
jlcarballido7@gmail.com

Abstract We look at a general way of inducing semantics in argumentation theory by means of a mapping defined on the family of 2-valued models of a normal program, which is constructed in terms of the argumentation framework. In this way we define a new argumentation semantics called stable_abducible which lies in between the stable and the preferred semantics. The relevance of this new semantics is that it is non-empty for any argumentation framework, and coincides with the stable argumentation semantics whenever this is non-empty. We study some of the properties of this semantics.
Keywords: Argumentation framework, argumentation semantics, logic programming semantics.

1 Introduction

In the non-monotonic reasoning community, it is well-accepted that the stable model semantics (also called answer set semantics) [7] represents a prominent approach for performing non-monotonic reasoning. In fact it has given place to a new approach of logic programming with *negation as failure*. Usually an answer set program can be seen as a specification of a problem where each stable model of a program P represents possible solutions to the problem. Nowadays, there are efficient solvers such as dlv, clasp, and smodel. The efficiency of these answer set solvers have increased the list of the stable model semantics' applications, *e.g.*, planning, bioinformatics, argumentation theory, etc.

Even though the stable model semantics enjoys a good reputation in the non-monotonic reasoning community, it is also well-known that the stable model semantics does not satisfy some desired properties pointed out by several authors [2,5,10,11]. Among these properties we can mention the existence of intended models in some normal logic programs. In fact, it is not hard to find a normal program which does not have stable models. However, the no existence of stable models for some normal program is not really a weakness because the no existence of stable models can be regarded as the no existence of solutions of a given problem. It all depends on the intended use of our logic programs. In the case of an approach motivated by *argumentation theory* it is desirable that normal

programs (representing a dispute among arguments) always give an answer that infers the *winning* arguments in a dispute among arguments.

In argumentation theory, the philosophy of the stable model semantics is characterized by the so called *stable argumentation semantics* [6]. Like stable model semantics, stable argumentation semantics is undefined in some argumentation scenarios. Hence, it is not strange to find, in the literature, different extensions for both the stable model semantics and the stable argumentation semantics. On the one side, the extensions for the stable models semantics have been supported from different logic-based inference properties. On the other hand, the extensions for the stable argumentation semantics have been motivated mainly from counter-examples. Some authors in argumentation theory [1] have argued for supporting new argumentation semantics mainly by considering inferences reasoning properties. In this setting, argumentation semantics such as the semi-stable has been accepted as a good extension of the stable argumentation semantics. However, semi-stable argumentation still has been weak for satisfying properties such *relevance* [3,8]. Relevance is seemed to be a desired property for dealing with the non-existence of stable extensions.

Since abstract argumentation semantics were introduced [6], it was shown that these semantics can be viewed as a special form of logic programming semantics with negation as failure. Therefore, any extension of the stable argumentation semantics based on logic programming semantics looks as a sound approach for extending argumentation semantics based on logic-based inference.

In this paper, we study an extension of the stable argumentation semantics based on logic-based inference. To this end, we consider the abductive approach introduced in [8]. In [8], an approach for extending logic programming semantics based on an abductive set of atoms was introduced. In this paper, this abductive set of atoms is characterized by classic-logic inference. Hence, by considering this set of abductive atoms, we generalize the stable model semantics by the so called *strong generalized stable models*. We show that these strong generalized stable models characterize a logic programming semantics which is based on paraconsistent logics, the so called p-stable models [9]. It is worth mentioning that the p-stable models capture the stable model semantics.

On the argumentation side, by considering a declarative specification of admissible sets in terms of logic normal programs and an order between strong generalized stable models, an abductive stable argumentation semantics is introduced. We show that this abductive stable argumentation semantics is an extension of the stable argumentation semantics and has a similar behavior to that of the semi-stable semantics. Moreover, a relationship between the abducible stable semantics and the preferred semantics is proved. In the last part of the paper, we show that the abducible stable argumentation semantics satisfies some intuitions 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 a new logic programming semantics that we call *stable_abducible* based on the concept of *strong_generalized stable models*. In §4, we review the relationships be-

tween two argumentation semantics and the *stable_abducible* logic programming semantics. Theorem 3 is our main contribution. In §5, we review the definition of *weak_relevance* for argumentation frameworks. Theorem 4 is our contribution in this section. In the last section, we present our conclusions.

2 Background

In this section, we review some theory about argumentation semantics and logic programming semantics. We present a short description of Dung's argumentation approaches, and also the definitions of the logic programming semantics *stable* and *p-stable* for normal programs.

2.1 Argumentation theory

We review some basic concepts of the preferred argumentation semantics defined by Dung [6] and some results about how to regard his argumentation approach as logic programming with negation as failure. The basic structure of Dung's argumentation approach is an argumentation framework which captures the relationships between the arguments.

Definition 1. *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.*,* $attacks \subseteq AR \times AR$*.*

Any argumentation framework can be regarded as a directed graph. For instance, if $AF := \langle \{a, b, c, d\}, \{(a, a), (a, c), (b, c), (c, d)\} \rangle$, then AF is represented as in Figure 1. We say that a *attacks* c (or c is attacked by a) if $attacks(a, c)$ holds. Similarly, we say that a set S of arguments attacks c (or c is attacked by S) if c is attacked by an argument in S. For instance in Figure 1, $\{a, b\}$ attacks c.

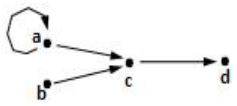

Figure 1. Graph representation of $AF = \langle \{a, b, c, d\}, \{(a, a), (a, c), (b, c), (c, d)\} \rangle$.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, if $A \subseteq AR$ then we write $AF|_A$ as a shorthand for $\langle A, \{(a, b) | (a, b) \in attacks \text{ and } a, b \in A\} \rangle$.

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

Definition 2. *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 argumentation framework of Figure 1 has three admissible sets: { }, $\{b\}$ and $\{b, d\}$.

Definition 3. *A* preferred extension *of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF. The set of preferred extensions of AF, denoted by* preferred_extensions(AF)*, will be referred to as the preferred semantics of AF.*

The only preferred extension of the argumentation framework of Figure 1 is $\{b, d\}$.

Definition 4. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We say that $E \subseteq AR$ is an* stable extension *of AF if E is an admissible set of AF that attacks every argument in $AR \setminus E$. The set of stable extensions of AF, denoted by* stable_extensions(AF)*, will be referred to as the stable semantics of AF.*

The argumentation framework of Figure 1 does not have stable extensions. The argumentation framework defined as $AF_1 = \langle\{a, b, c, d, e\}, \{(a, b), (c, b), (c, d), (d, c), (d, e), (e, d)\}\rangle$ has two stable extensions: $\{a, d\}$ and $\{a, c, e\}$.

2.2 Logic programming semantics

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 $\mathcal{B}^-$. Given a normal clause $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$, denoted by r, we say that $a = H(r)$ is the *head* and $\mathcal{B}^+(r) \cup \neg\mathcal{B}^-(r)$ is the *body* of the clause. If the body of a clause is empty, then the clause is known as a *fact* and can be denoted just by: $a \leftarrow$ or $a \leftarrow \top$. We define a *normal logic program* P, as a finite set of normal clauses. 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 $\{a \mid a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^- \in P\}$. From now on, by *program* we will mean a normal logic program when ambiguity does not arise. We want to point out that our negation symbol, $\neg$, corresponds to "*not*" in the standard use of *Logic Programming*.

From now on, we assume that the reader is familiar with the notion of an *interpretation* and *validity* [12]. An interpretation M is called a (2-valued classical) *model* of P if and only if for each clause $c \in P$, $M(c) = 1$. We say that c

is a logic consequence of P, denoted by $P \models c$, if every model M of P holds that $M(c) = 1$.

In this paper, a logic programming semantics S is a mapping defined on the family of all programs which associates to a given program a subset of its 2-valued (classical) models. We say that M is a *minimal model* of P if and only if there does not exist a model M' of P such that $M' \subset M$, $M' \neq M$ [12].

Stable model semantics. The stable model semantics was defined in terms of the so called *Gelfond-Lifschitz reduction* [7] and it is usually studied in the context of syntax dependent transformations on programs. The following definition of a stable model for normal programs was presented in [7].

Let us recall that a normal positive program always has minimal models.

Definition 5. *Let P be a normal program. For a set $S \subseteq \mathcal{L}_P$ we define the program P^S from P by deleting each rule that has a literal $\neg l$ in its body with $l \in S$, and then all literals of the form $\neg l$ in the bodies of the remaining rules. Clearly P^S does not contain $\neg$. S is a* stable model *of P if and only if S is a minimal model of P^S. We will denote by* stable(P) *the set of all stable models of P.*

Example 1. Let $S = \{b\}$ and $P = \{b \leftarrow \neg a,\ \ c \leftarrow \neg b,\ \ b \leftarrow,\ \ c \leftarrow a\}$. Notice that P^S has three models: $\{b\}$, $\{b, c\}$ and $\{a, b, c\}$. Since the minimal model among these models is $\{b\}$, we can say that S is a stable model of P.

p-stable semantics. Logical inference in propositional classical logic is denoted by $\vdash$. Given a set of proposition symbols S and a theory (a set of well-formed formulae) Γ, $\Gamma \vdash S$ if and only if $\forall s \in S$, $\Gamma \vdash s$. When we treat a program as a theory, each negative literal $\neg a$ is regarded as the standard negation operator in classical logic. Given a normal program P, if $M \subseteq \mathcal{L}_P$, we write $P \Vdash M$ when: $P \vdash M$ and M is a classical 2-valued model of P. The p-stable semantics is defined in terms of a single reduction which is defined as follows:

Definition 6. *[9] Let P be a normal program and M be a set of atoms. We define*

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

Now, we define the p-stable semantics for normal programs (introduced in [9]).

Definition 7. *[9] Let P be a normal program and M be a set of atoms. We say that M is a* p-stable model *of P if $RED(P, M) \Vdash M$. We will denote by* p-stable(P) *the set of all* p-stable models of P.

Example 2. Let us consider $P = \{a \leftarrow \neg b \wedge \neg c,\ \ a \leftarrow b,\ \ b \leftarrow a\}$ and $M = \{a, b\}$. We can verify that $RED(P, M)$ is: $\{a \leftarrow \neg b,\ \ a \leftarrow b,\ \ b \leftarrow a\}$. We see that M is a model of P since for each clause C of P we have that M evaluates C to true. We also see that $RED(P, M) \vdash M$. Thus $RED(P, M) \Vdash M$ and M is a *p-stable model* of P.

3 Stable_abducible semantics

In this section, we define a new logic programming semantics with interesting properties in logic programming and argumentation, we call it *stable_abducible* and corresponds to the set of the *strong_generalized stable models* of a given normal program. The strong_generalized stable models correspond to the stable models of the given normal program joined to a particular subset of facts. In this paper, the stable_abducible semantics will help to characterize some argumentation semantics in terms of logic programming.

Definition 8. *Let P be a normal program and X, M_X be sets of atoms. We say that M_X is a* strong_generalized stable model *of P if M_X is a stable model of $P \cup X$ and $X \subseteq \{y \mid RED(P, M_X) \vDash y\}$.*

We define a partial order on the set of strong_generalized stable models of a program.

Definition 9. *Let M_{X_1} and M_{X_2} be two strong_generalized stable models of a program P. We define a* partial order *between these two strong_generalized stable models of P as follows: $M_{X_1} < M_{X_2}$, if $X_1 \subset X_2$.*

Now we present the definition of the minimal strong_generalized stable model of P.

Definition 10. *Let P be a normal program. We say that M_X is a* stable_abducible model *of P, if it is a minimal strong_generalized stable model of P with respect to the partial order $<$.*

We will denote by *stable_abducible*(P) the set of all *stable_abducible* models of P. It is clear that a stable model of P is a stable_abducible model of P since it is a strong generalized stable model of P with $X = \emptyset$.

The following two examples illustrate our new logic programming semantics.

Example 3. Let $P = \{a \leftarrow b, \quad b \leftarrow \neg a\}$. We can verify that P does not have stable models.
a) Let $M = \{a\}$ and $X = \{a\}$. In this case, $P \cup X = \{a \leftarrow, \quad a \leftarrow b, \quad b \leftarrow \neg a.\}$ So, M is a stable model of $P \cup X$, $RED(P, M) = \{a \leftarrow b, \quad b \leftarrow \neg a\}$, $A = \{y \mid RED(P, M) \vDash y\} = \{a\}$, and $X \subseteq A$. Hence M is a strong_generalized stable model of P.
b) Let $M = \{a, b\}$, and $X = \{b\}$. In this case, $P \cup X = \{b \leftarrow, \quad a \leftarrow b, \quad b \leftarrow \neg a\}$ So, M is a stable model of $P \cup X$, $RED(P, M) = \{a \leftarrow b, \quad b \leftarrow \neg a.\}$, $A = \{y \mid RED(P, M) \vDash y\} = \{a\}$, however X is not a subset of A. Hence M is not a strong_generalized stable model of P.
Finally, since $\{a\}$ is minimal, the *stable_abducible*$(P) = \{\{a\}\}$.

Next we present a theorem that shows that the strong generalized stable models of a normal program are the same as its p-stable models. First we present some lemmas and theorems useful to prove our main theorem.

Definition 11. *[4] Two programs P_1 and P_2 are* equivalent*, denoted by $P_1 \equiv P_2$, if P_1 and P_2 have the same 2-valued models. Equivalently, if their sets of rules are equivalent in classical logic.*

Lemma 1. *Let P be a normal program and let X, M be sets of atoms. If $X \subseteq \{y \mid RED(P, M) \vDash y\}$ then $RED(P, M) \equiv RED(P \cup X, M)$.*

Proof. $RED(P \cup X, M) = RED(P, M) \cup RED(X, M)$. Hence, it is enough to show that $RED(P, M) \vDash RED(X, M)$. But this is equivalent to the hypothesis $X \subseteq \{y \mid RED(P, M) \vDash y\}$. □

Theorem 1. *If M is a strong_generalized stable model of P then M is a p-stable model of P.*

Proof. If M is a strong_generalized stable model of P then there exists X such that
(1) M is a stable model of $P \cup X$.
(2) $X \subseteq \{y \mid RED(P, M) \vDash y\}$.
From (1) it follows that

(i) M is a classical model of P, since M models in classical logic $P \cup X$ and in particular it models P.
(ii) M is a p-stable model of $P \cup X$ since for normal programs the stable semantics is a subset of the p-stable semantics

From (ii) it follows that
(a) $RED(P \cup X, M) \vDash M\}$.
From (a) and Lemma 1 we obtain
(b) $RED(P \cup X, M) \equiv RED(P, M)$.
From the last two relations (b) and (a) we obtain:
(c) $RED(P, M) \vDash M$.

From (i) and (c) it follows that M is a p-stable model of P. □

Lemma 2. *Let P be a normal program and M be a set of atoms. Then $R(P \cup M, M) \vDash RED(P, M)$, where $R(T, M)$ denotes the Gelfond and Lifschitz reduction of T with respect to M.*

Proof. We consider two types of rules.

(1) The positive rules of $RED(P, M)$ are also rules of $R(P \cup M, M)$.
(2) The negative rules of $RED(P, M)$ (the interesting case) are of the form $r := x \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$, where $\mathcal{B}^- \subseteq M$. Hence $M \vDash \mathcal{B}^-$ and $\mathcal{B}^- \vDash r$. Hence $M \vDash r$ but $M \subseteq R(P \cup M, M)$. Therefore $R(P \cup M, M) \vDash r$.

□

Theorem 2. *Let M be a p-stable model of the normal program P, then M is a strong_generalized stable model of P.*

Proof. Let us assume that M is a p-stable model of P. Then

(1) M is a classical model of P, and also of $R(P, M)$
(2) $RED(P, M) \vDash M$.
It is clear that
(i) $R(P \cup M, M) \vDash M$.
From (1) and (i) it follows that M is a minimal model of $R(P \cup M, M)$.
Hence,
(ii) M is a stable model of $P \cup M$.
Also,
(iii) $M \subseteq \{y \mid RED(P, M)\}$ by definition of p-stable model.
From (ii) and (iii) we conclude that M is a strong_generalized stable model of P (where M is the set M_X in the Definition 10 with $X = M$).

□

4 Mapping from argumentation to logic programming

In this section, we review the relationships between two argumentation semantics (the stable argumentation semantics and the preferred argumentation semantics) and the *stable_abducible* logic programming semantics. These relationships are based on the proposal introduced in [8] which consists of the following steps: First, given an argumentation framework AF, we use a particular mapping that associates to AF a normal program denoted by P_{AF}. Second, we obtain the *stable_abducible* models of P_{AF}. Finally, we use a second mapping, called f, to obtain the stable argumentation semantics of AF. The mapping f assigns to each *stable_abducible* model of P_{AF} a set of arguments from the argumentation framework AF.

Now, we present the definitions of the two mappings and the theorem that specifies the relationship between the argumentation semantics and the *stable_abducible* logic programming semantics.

The first mapping uses the predicate *d(x)* to represent that "the argument x is defeated". This mapping also includes clauses such as $d(x) \leftarrow \neg d(y)$ to capture the idea of argument x is defeated when anyone of its adversaries y is not defeated.

Definition 12. *[8] Let $AF = \langle AR, attacks \rangle$ be an argumentation framework,* $P^D_{AF} = \bigcup_{x \in AR} \{d(x) \leftarrow \neg d(y) \mid (y, x) \in attacks\}$ *and* $P^E_{AF} = \bigcup_{x \in AR} \{\cup_{y:(y,x) \in attacks} \{d(x) \leftarrow \wedge_{z:(z,y) \in attacks} d(z)\}\}$. *We define:*

$$P_{AF} = P^D_{AF} \cup P^E_{AF}$$

For a given atom x in the definition of P^E_{AF} there may not be a z as described, in that case the corresponding conjunction $\wedge_{z:(z,y) \in attacks} d(z)$ is empty leaving the fact $d(x) \leftarrow$ in P^E_{AF}. The reader familiar with argumentation theory can observe that essentially, P^D_{AF} captures the basic principle of *conflict-freeness* [8].

Example 4. [3] Now, we illustrate the mapping P_{AF}, let AF be the argumentation framework of Figure 1. We can see that $P_{AF} = P^D_{AF} \cup P^E_{AF}$ where,

P^D_{AF} :
$d(a) \leftarrow \neg d(a).$ $d(c) \leftarrow \neg d(a).$
$d(c) \leftarrow \neg d(b).$ $d(d) \leftarrow \neg d(c).$

P^E_{AF} :
$d(a) \leftarrow d(a).$ $d(d) \leftarrow d(a) \wedge d(b).$
$d(c) \leftarrow d(a).$ $d(c) \leftarrow .$

Now, we see how the second mapping, called f, can induce an argumentation semantics of an argumentation framework AF based on the *stable_abducible* logic programming semantics. The mapping f assigns an extension (a set of accepted arguments) of AF to each *stable_abducible* model of P_{AF}. Specifically, it assigns the set of arguments that does not appear as defeated arguments in the *stable_abducible* model of P_{AF}. Each of these extensions will be called *stable_abducible* extension. The induced *argumentation semantics* (the set of *stable_abducible* extensions) of AF will be denoted by $Ext_{s\_a}(AF)$.

Definition 13. *Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. The* stable_abducible argumentation semantics *of AF, denoted by Ext_{s_a}, is defined as follows:*

$$Ext_{s\_a}(AF) = \{N \mid N = f(M), M \in stable\_abducible(P_{AF})\}$$

where f is the mapping from $2^{\mathcal{L}_{P_{AF}}}$ to 2^{AR}, such that

$$f(M) = \{a \mid d(a) \in (\mathcal{L}_{P_{AF}} \setminus M)\}$$

Example 5. Let us consider the argumentation framework AF of Figure 1 and its normal program P_{AF} previously obtained in Example 4. We are going to obtain the *stable_abducible argumentation semantics of AF*, namely $Ext_{s\_a}(AF)$. First of all, we have to obtain the *stable_abducible* semantics of P_{AF}. It is not difficult to verify that *stable_abducible*(P_{AF}) corresponds to the following set: $\{ \{d(a), d(c)\} \}$. Hence, $Ext_{s\_a}(AF) = \{\{d, b\}\}$.

Now, we present the relationship between the *stable_abducible argumentation semantics* and two well known argumentation semantics: the *stable argumentation semantics* and the *preferred argumentation semantics*. It is important to remark that the semi-stable argumentation semantics defined by Caminada in [3] also has a similar relationship with these two argumentation semantics. As future work, it would be interesting to explore the relation between the semi-stable semantics and the stable_abducible argumentation semantics.

The following theorem shows four facts: there exists at least one *stable_abducible* extension; a stable extension is also a *stable_abducible* extension; a *stable_abducible* extension is also a preferred extension; and if there exists at least one stable extension, then the *stable_abducible* extensions coincide with the stable extensions.

Let us point out that, according to the next result, *stable_abducible* extensions always exists, and in case the family of stable extensions of a program is non-empty, the two argumentation semantics coincide.

Theorem 3. *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework and $L \subseteq AR$, the following four statements hold:*
(1) $Ext_{s\_a}(AF) \neq \emptyset$
(2) if $L \in stable\_extensions(AF)$ then $L \in Ext_{s\_a}(AF)$
(3) if $L \in Ext_{s\_a}(AF)$ then $L \in preferred\_extensions(AF)$
(4) when AF has at least an stable extension, it holds that

$$Ext_{s\_a}(AF) = stable\_extensions(AF)$$

Proof. In [6] it is proven that the preferred semantics of an AF is not empty, hence the p-stable semantics of P_{AF} is not empty. Since, the $stable\_abducible(P_{AF})$ is defined as the set of p-stable models of P_{AF} that satisfies certain minimality conditions, then $stable\_abducible(P_{AF}) \neq \emptyset$. Therefore, $Ext_{s\_a}(AF) \neq \emptyset$. This proves **(1)**.

It is clear from the definition of stable_abducible logic programming semantics that if P is a normal program and $M \in stable(P_{AF})$ then $M \in stable\_abducible(P_{AF})$, and that $stable\_abducible(P_{AF}) \subseteq p - stable(P_{AF})$. In fact, let us observe that if $stable(P_{AF}) \neq \emptyset$ then $stable(P_{AF}) \neq stable\_abducible(P_{AF})$, for if $M \in stable(P_{AF})$, it is known that $M \in$*p-stable(P_{AF})*, and therefore M satisfies the definition of $stable\_abducible(P_{AF})$ with $X = \emptyset$.
We then have: $stable(P_{AF}) \subseteq stable\_abducible(P_{AF}) \subseteq$ *p-stable(P_{AF}).*
Since the stable semantics logic programming corresponds to the stable argumentation semantics and the p-stable semantics corresponds to the preferred argumentation semantics according to the mapping f of Definition 13, then we obtain *stable_extensions(AF)*$\subseteq Ext_{s\_a}(AF) \subseteq$ *preferred_extensions(AF).* This proves **(2)** and **(3)**.

To prove **(4)**, we observe that if AF has at least one stable extension then P_{AF} has at least one stable model, then $stable(P_{AF}) = stable\_abducible(P_{AF})$. We conclude that *stable_extensions(AF)*$= Ext_{s\_a}(AF)$. □

The converse of Theorem 3 does not hold. That is, it is not the case that each extension in $Ext_{s\_a}(AF)$ is also a stable extension, see below Example 6; and it is not the case that each preferred extension is also in $Ext_{s\_a}(AF)$, see below Example 7.

Example 6. Let S be the *stable_abducible semantics*. Let us consider the argumentation framework AF of Figure 1. In Example 5, we show that $Ext_{s\_a}(AF) = \{\{d, b\}\}$. We can see that $Ext_{s\_a}(AF)$ does not coincides with the set of stable extensions of AF, i.e., $Ext_{s\_a}(AF) \neq stable\_extensions(AF)$.

Example 7. Let us consider the argumentation framework $AF = \langle AR, attacks\rangle$ where $AR = \{a, b, c, d, e\}$ and $attacks = \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}$. We can see that $L = \{a\}$ is a preferred extension and $L \notin Ext_{s\_a}(AF)$. The only stable_abducible extension is $\{b, d\}$.

5 Weak relevance

Now, we present the definition of *weak_relevance* for argumentation frameworks which is called *relevant* by Caminada in [3].

Definition 14. *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. An argument $a \in AR$ is* weakly_relevant *w.r.t. an argument $b \in AR$ if and only if there exists an undirected path between a and b.*

If in Definition 14 we replace *an undirected path between a and b* with *a directed path from a to b*, we would be talking about a relevance relation which is *stronger* and which is used in the context of logic programming.

Example 8. Let us consider the argumentation framework $AF = \langle AR, attacks\rangle$ where $AR = \{a, b, c, d\}$ and $attacks = \{(a, a), (b, c), (c, d)\}$. In this AF, the arguments b, c, and d are *weakly_relevant* with respect to each other, and argument a is not *weakly_relevant* with respect to b, c and d. Yet, argument a is the reason why there is no stable extension containing b and d.

The following theorem indicates that irrelevant arguments do not determine whether an argument is justified under the stable_abducible argumentation semantics.

Theorem 4. *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework and let $a \in AR$ and $A \subset AR$ the set of all arguments that are weakly_relevant w.r.t. a.*

1. *There exists a stable_abducible extension of AF iff there exists a stable_abducible extension of $AF|_A$.*
2. *a is in every stable_abducible extension of AF iff a is in every stable_abducible extension of $AF|_A$.*

Proof. **Sketch.**
To prove this theorem first we should prove two things:

(1) If L is a stable_abducible extension of AF then $L \cap A$ is a stable_abducible extension of $AF|_A$.

(2) If L is a stable_abducible extension of $AF|_A$ then there exists a stable_abducible extension L' of AF with $L' \cap A = L$.

Finally, the theorem follows directly from these two results. □

6 Conclusions

Argumentation theory and logic programming with negation as failure are two approaches non-monotonicity which have been intensively explored the last years. Moreover, these approaches share some common behaviors in their inference process; in particular, by the relationships there exist between argumentation semantics and logic programming semantics. In this paper, on the one hand, an extension of the stable model semantics is introduced by the so called strong generalized stable models. It is shown that the strong generalized stable models capture the p-stable models (Theorem 2). By considering an order between strong generalized stable models, the stable_abducible semantics is defined. On the other hand, by considering the stable_abducible semantics, the stable_abducible argumentation semantics is defined. This new argumentation semantics is an intermediate argumentation semantics between the stable argumentation semantics

and the preferred semantics (Theorem 3). It is shown that the stable argumentation semantics is always defined and satisfies the notion of relevance introduced by Caminada [3].

The results of Theorem 3 suggest that the stable abducible argumentation semantics has similar behavior to semi-stable semantics. Part of our future work will be to formalize the relationship between the stable abducible argumentation semantics and the semi-stable semantics. We want to point out that the stable abducible argumentation semantics does not characterize exactly the semi-stable semantics; because the stable abducible argumentation semantics satisfies a notion of relevance which semi-stable semantics does not satisfies.

7 Funding

This work was supported by the CONACyT [CB-2008-01 No.101581].

References

1. P. Baroni and M. Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence.*, 171(10-15):675–700, 2007.
2. G. Brewka. An abductive framework for generalized logic programs. In *LPNMR*, pages 349–364, 1993.
3. M. Caminada. Semi-stable semantics. In P. E. Dunne and T. J. M. Bench-Capon, editors, *COMMA*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 121–130. IOS Press, 2006.
4. J. L. Carballido, M. Osorio, and J. Arrazola. Equivalence for the G'$_3$-stable models semantics. *J. Applied Logic*, 8(1):82–96, 2010.
5. J. Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundam. Inform.*, 22(3):257–288, 1995.
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. 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.
8. J. C. Nieves, M. Osorio, and C. Zepeda. A schema for generating relevant logic programming semantics and its applications in argumentation theory. *Fundamenta Informaticae*, 106(2-4):295–319, 2011.
9. 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.
10. L. M. Pereira and A. M. Pinto. Layer supported models of logic programs. In *LPNMR 2009*, volume 5753 of *Lecture Notes in Computer Science*. Springer, 2009.
11. J. S. Schlipf. Formalizing a logic for logic programming. *Ann. Math. Artif. Intell.*, 5(2-4):279–302, 1992.
12. D. van Dalen. *Logic and structure*. Springer-Verlag, Berlin, 3rd., augmented edition, 1994.