Inferring Preferred Extensions by Minimal Models

Juan Carlos Nieves[1], Mauricio Osorio[2], and Ulises Cortés[1]

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

Abstract. We identify that there is a direct relationship between the minimal models of a propositional formula and the preferred extensions of an argumentation framework. Then we show how to infer the preferred extensions of an argumentation framework by using UNSAT algorithms and disjunctive answer set solvers.

1 Introduction

Although several approaches have proposed for argument theory, Dung's approach, presented in [11], is a unifying framework which has played an influential role on argumentation research and Artificial Intelligence (AI). In fact, Dung's approach has influenced subsequent proposals for argumentation systems, *e.g.,* [18, 3]. Besides, Dung's approach is mainly relevant in fields where conflict management plays a central role. For instance, Dung showed that his theory naturally captures the solutions of the theory of n-person game and the well-known stable marriage problem.

Dung defined four argumentation semantics: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of these semantics is the *acceptability of the arguments*. An argument is called *acceptable* if and only if it belongs to a set of arguments which is called *extension*. The main argumentation semantics for collective acceptability are the grounded semantics and the preferred semantics [16, 1]. The first one represents a skeptical approach, since for a given argumentation framework the grounded semantics 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 well-known that the implementation of the decision problem of the grounded semantics is quite straightforward. However, the decision problem of the preferred semantics is hard since it is co-NP-Complete [12]. In the literature, we can find different algorithms for computing the preferred semantics [4, 6, 9,

10, 2]. We have to point out that these algorithms are so specific; they are not really flexible for developing small prototypes.

From the point of view that a proper representation of a given problem is a major step in finding robust solutions to it, we explore a couple of representations of an argumentation framework in order to compute their preferred extensions. In general terms, we identify that there is a direct relationship between the minimal models of a propositional formula and the preferred extensions of an argumentation framework. We show how to infer the *preferred extensions* of an argumentation framework by using *UNSAT algorithms* and *disjunctive answer set solvers e.g.,* DLV [8]. UNSAT is the complement of Satisfiability (SAT), a problem for which very efficient systems have been developed in AI during the last decade. Nowadays, there are fast answer set solvers *e.g.,* DLV [8], SMODELS [17], which have contributed to extend the applications of Answer Set Programming (ASP).

The rest of the paper is divided as follows: In §2, we present some basic concepts of logic programs and argumentation theory. In §3, we present a characterization of the preferred semantics by minimal models. In §4, we present how to compute the preferred semantics by using the minimal models of a positive disjunctive logic program. Finally in the last section, we present our conclusions.

2 Background

In this section, we present the syntax of a valid logic program in ASP, the definition of an answer set, and the definition of the preferred semantics. We will use basic well-known definitions in complexity theory such as co-NP-complete problem. We suggest the reader to consult [7] if s/he needs to read more on such definitions.

2.1 Logic Programs: Syntax

The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: $p_0, p_1, ...$
(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \bot, \top$
(iii) auxiliary symbols : (,).

where $\vee, \wedge, \leftarrow$ are 2-place connectives, $\neg$ is 1-place connective and $\bot, \top$ are 0-place connectives. The proposition symbols and $\bot$ stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. A literal is an atom, a, or the negation of an atom $\neg a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of literals $\{\neg a_1, ..., \neg a_n\}$.

A general clause, C, is denoted by $a_1 \vee \ldots \vee a_m \leftarrow l_1, \ldots, l_n$,[3] where $m \geq 0$, $n \geq 0$, each a_i is an atom, and each l_i is a literal. When $n = 0$ and $m > 0$ the clause is an abbreviation of $a_1 \vee \ldots \vee a_m \leftarrow \top$, where $\top$ is $\neg\bot$. When $m = 0$

[3] $l_1, \ldots, l_n$ represents the formula $l_1 \wedge \cdots \wedge l_n$.

the clause is an abbreviation of $\bot \leftarrow l_1, \ldots, l_n$. Clauses of this form are called constraints (the rest, non-constraint clauses). A general program, P, is a finite set of general clauses. By $\mathcal{L}_P$, we denote the set of atoms that occurs in P. Given a set S and $E \subseteq S$, $\widetilde{E}$ denotes the complement of E *w.r.t.* S.

We point out that whenever we consider logic programs our negation $\neg$ corresponds to the default negation *not* used in Logic Programming. Also, it is convenient to remark that in this paper we are not using at all the so called strong negation used in ASP.

2.2 Answer set semantics

First, to define the answer set semantics, let us define some relevant concepts. Let P be a general program. An interpretation I is a mapping from $\mathcal{L}_P$ to $\{0, 1\}$, where the generalization of I to connectives is as follows: $I(a \wedge b) = min\{I(a), I(b)\}$, $I(a \vee b) = max\{I(a), I(b)\}$, $I(a \leftarrow b) = 0$ if and only if $I(b) = 1$ and $I(a) = 0$, $I(\neg a) = 1 - I(a)$, $I(\bot) = 0$. An interpretation I is called a model of P if and only if for each clause $c \in P$, $I(c) = 1$. Finally, I is a minimal model of P if it does not exist a model I' of P such that $I' \subset I$.

By using answer set programming, it is possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. The answer set semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [13] and it is usually studied in the context of syntax dependent transformations on programs. The following definition of an answer set for general programs generalizes the definition presented in [13] and it was presented in [14].

Let P be any general program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the general program obtained from P by deleting

(i) each rule that has a formula $\neg l$ in its body with $l \in S$, and then
(ii) all formulæ of the form $\neg l$ in the bodies of the remaining rules.

Clearly P^S does not contain $\neg$, then S is an answer set of P if and only if S is a minimal model of P^S.

2.3 Argumentation theory

Now, we define some basic concepts of Dung's argumentation approach. The first one is an argumentation framework. An argumentation framework captures the relationships between the arguments (All the definitions of this subsection were taken from the seminal paper [11]).

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 $\subseteq AR \times AR$.

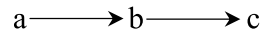

Fig. 1. A single argumentation framework

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

Definition 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.*

By considering conflict-free sets of arguments, it is defined the concept of *admissible set*.

Definition 3. *(1) An argument $A \in AR$ is acceptable with respect to a set S of arguments if and only if for each argument $B \in AR$: If B attacks A then B is attacked by S. (2) A conflict-free set of arguments S is admissible if and only if each argument in S is acceptable w.r.t. S.*

For instance, the argumentation framework of Fig. 1 has two admissible sets: $\{a\}$ and $\{a, c\}$. The (credulous) semantics of an argumentation framework is defined by the notion of preferred extension.

Definition 4. *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*

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

3 Preferred extensions and UNSAT algorithms

In this section, we provide a method for computing preferred extensions. This method is based on model checking and Unsatisfiability (UNSAT). UNSAT is the complement of Satisfiability (SAT), a problem for which very efficient systems have been developed in AI during the last decade.

First of all, we introduce some notations which are used in the rest of the paper. Our representations of an argumentation framework use the predicate *d(X)*, where the intended meaning of *d(X)* is: "the argument X is defeated". Given an argumentation framework $AF := \langle AR, Attacks\rangle$ and $E \subseteq AR$, we define the set $s(E)$ as $\{d(a)|a \in AR \setminus E\}$. Essentially, $s(E)$ expresses the complement of E *w.r.t. AR*. Given $A \in AR$, we define $D(A)$ as $\{B|(B, A) \in Attacks\}$. Intuitively $D(A)$ denotes the set of arguments which attacks A.

Now we definite a mapping from an argumentation framework to a propositional formula.

Definition 5. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, then $\alpha(AF)$ is defined as follows:*

$$\alpha(AF) := \bigwedge_{A\in AR} ((\bigwedge_{B\in D(A)} d(A) \leftarrow \neg d(B)) \wedge (\bigwedge_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C)))$$

In the propositional formula $\alpha(AF)$, we can identify two parts for each argument $A \in AR$:

1. The first part $(\bigwedge_{B\in D(A)} d(A) \leftarrow \neg d(B))$ suggests that the argument A is defeated when one of its attackers is not defeated.
2. The last part $(\bigwedge_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C))$ suggests that the argument A is defeated when all the arguments that defend[4] A are defeated.

Notice that $\alpha(AF)$ is essentially a propositional formula (just considering the atoms like $d(a)$ as d_a). In order to illustrate the propositional formula $\alpha(AF)$, let us consider the following example.

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

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

It is important to remember that the conjunction of an empty set is the true value ($\top$). Now if one considers the propositional formula *w.r.t.* argument b, we get

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

And the propositional formula *w.r.t.* argument c is

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

Then, $\alpha(AF)$ is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))$$

Essentially $\alpha(AF)$ is a propositional representation of the argumentation framework AF. However $\alpha(AF)$ has the property that its minimal models characterize AF's preferred extensions. In order to formalize this property, let us consider the following proposition which was proved by Besnard and Doutre in [4].

[4] We say that C defends A if B attacks A and C attacks B.

Proposition 1. *[4] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. A set $S \subseteq AR$ is a preferred extension if and only if S is a maximal model of the formula*

$$\bigwedge_{A\in AR}((A \rightarrow \bigwedge_{B\in D(A)} \neg B) \wedge (A \rightarrow \bigwedge_{B\in D(A)} (\bigvee_{C\in D(B)} C)))$$

Notice that $\alpha(AF)$ is related to defeated arguments and the formula of Proposition 1 is related to acceptable arguments. It is not difficult to see that $\alpha(AF)$ is the dual formula of the formula of Proposition 1. For instance, let us consider the argumentation framework AF of Example 1. The formula related to AF, according to Proposition 1, is:

$$(\neg a \leftarrow b) \wedge (\bot \leftarrow b) \wedge (\neg b \leftarrow c) \wedge (a \leftarrow c)$$

If we replace each atom x by the expression $\neg d(x)$, we get:

$$(\neg\neg d(a) \leftarrow \neg d(b)) \wedge (\bot \leftarrow \neg d(b)) \wedge (\neg\neg d(b) \leftarrow \neg d(c)) \wedge (\neg d(a) \leftarrow \neg d(c))$$

Now, if we apply transposition to each implication

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))$$

The latter formula corresponds to $\alpha(AF)$. The following theorem is a straightforward consequence of Proposition 1.

Theorem 1. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is a minimal model of $\alpha(AF)$.*

In order to illustrate Theorem 1, let us consider again $\alpha(AF)$ of Example 1. This formula has three models: $\{d(b)\}$, $\{d(b), d(c)\}$ and $\{d(a), d(b), d(c)\}$. Then, the only minimal model is $\{d(b)\}$, this implies that $\{a, c\}$ is the only preferred extension of AF. In fact, each model of $\alpha(AF)$ implies an admissible set of AF, this means that $\{a, c\}$, $\{a\}$ and $\{\}$ are the admissible sets of AF.

There are several approaches for inferring minimal models from a propositional formula. For instance, it is possible to use UNSAT's algorithms for inferring minimal models. Hence, it is clear that we can use UNSAT's algorithms for computing the preferred extensions of an argumentation framework. This idea is formalized with the following lemma. Let S be a set of well formed formulæ then we define $n(S) := \bigwedge_{c\in S} c$.

Lemma 1. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is a model of $\alpha(AF)$ and $\alpha(AF) \wedge n(\neg\widetilde{s(S)}) \wedge \neg n(s(S))$ is unsatisfiable.*

Proof. It is direct by Theorem 1.

In order to illustrate Lemma 1, let us consider again the argumentation framework AF of Example 1. Let $S = \{a\}$, then $s(S) = \{d(b), d(c)\}$. We have already seen that $\{d(b), d(c)\}$ is a model of $\alpha(AF)$, hence the formula to verify its unsatisfiability is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a)) \wedge \\ \neg d(a) \wedge (\neg d(b) \vee \neg d(c))$$

However, this formula is satisfiable by the model $\{d(b)\}$, then $\{a\}$ is not a preferred extension. Now, let $S = \{a, c\}$, then $s(S) = \{d(b)\}$. As seen before, $\{d(b)\}$ is also a model of $\alpha(AF)$, hence the formula to verify its unsatisfiability is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a)) \wedge \\ \neg d(a) \wedge \neg d(c) \wedge \neg d(b)$$

It is easy to see that this formula is unsatisfiable, therefore $\{a, c\}$ is a preferred extension.

The relevance of Lemma 1 is that UNSAT is the prototypical and best-researched co-NP-complete problem. Hence, Lemma 1 opens the possibilities for using a wide variety of algorithms for inferring the preferred semantics.

4 Preferred extensions and general programs

In Section 3, we presented a representation of an argumentation framework in terms of a propositional formula for inferring preferred extensions. Another option for computing the preferred semantics is by considering a straightforward mapping from an argumentation framework to a *general program*. This approach is an *elegant* and *short* form for inferring the preferred extensions of an argumentation framework. The only system that we need for inferring the preferred extensions of an argumentation framework is any disjunctive answer set solver *e.g.,* DLV [8].

We start this section by defining a simple mapping from an argumentation framework to a positive disjunctive logic program.

Definition 6. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:*

$$\Gamma(A) := \Big(\bigwedge_{B \in D(A)} (d(A) \vee d(B)) \Big) \wedge \Big(\bigwedge_{B \in D(A)} \Big(d(A) \leftarrow \bigwedge_{C \in D(B)} d(C) \Big) \Big)$$

The generalization of the function Γ is defined as follows:

Definition 7. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define its associated general program as follows:*

$$\Gamma_{AF} := \bigwedge_{A \in AR} \Gamma(A)$$

Remark 1. Notice that $\alpha(AF)$ (see Definition 5) is similar to Γ_{AF}. The main syntactic difference of Γ_{AF} *w.r.t.* $\alpha(AF)$ is the first parte of Γ_{AF} which is $(\bigwedge_{B \in D(A)}(d(A) \vee d(B)))$; however this part is logical equivalent to the first part of $\alpha(AF)$ which is $(\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B))$. In fact, the main difference is their behavior *w.r.t.* answer set semantics. In order to illustrate this difference, let us consider the argumentation framework $AF := \langle AR, attacks \rangle$, where $AR := \{a\}$ and $attacks := \{(a, a)\}$. Then we can see that

$$\Gamma_{AF} := (d(a) \vee d(a)) \wedge (d(a) \leftarrow d(a))$$

and

$$\alpha(AF) := (d(a) \leftarrow \neg d(a)) \wedge (d(a) \leftarrow d(a))$$

It is clear that both formulæ have a minimal model which is $\{d(a)\}$, however $\alpha(AF)$ has no answer sets. In fact both formulæ are logically equivalent in classic logic but not in answer set semantics.

In the following theorem we formalize a characterization of the preferred semantics in terms of positive disjunctive logic programs and answer set semantics.

Theorem 2. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is an answer set of Γ_{AF}.*

Proof. S is a preferred extension of AF if and only if *s(S)* is a minimal model of $\alpha(AF)$ (by Theorem 1) if and only if $s(S)$ is a minimal model of Γ_{AF} (since Γ_{AF} is logically equivalent to $\alpha(AF)$ in classical logic) if and only if *s(S)* is an answer set of Γ_{AF} (since Γ_{AF} is a positive disjunctive program and for every positive disjunctive program P, M is an answer set of P if and only if M is a minimal model of P).

Let us consider the following example.

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

$$\begin{array}{ll} d(a) \vee d(b). & d(a) \leftarrow d(a). \\ d(b) \vee d(a). & d(b) \leftarrow d(b). \\ d(c) \vee d(b). & d(c) \vee d(e). \\ d(c) \leftarrow d(a). & d(c) \leftarrow d(d). \\ d(d) \vee d(c). & d(d) \leftarrow d(b), d(e). \\ d(e) \vee d(d). & d(e) \leftarrow d(c). \end{array}$$

Γ_{AF} has two answer sets which are $\{d(a), d(c), d(e)\}$ and $\{d(b), d(c), d(e), d(d))\}$, therefore $\{b, d\}$ and $\{a\}$ are the preferred extensions of AF.

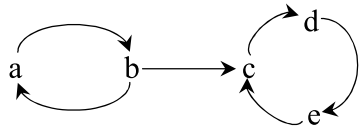

Fig. 2. An argumentation framework.

An alternative form for computing the preferred extensions of an argumentation framework, without considering the predicate *d(X)*, is taking advantage of *default negation*. It is possible by considering a new dual symbol for each argument of the argumentation framework. This means that we can infer the acceptable arguments directly from the answers sets of the logic program.

This idea is formalized with the following lemma. First, let us present some definitions.

Definition 8. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define the function η as $\eta : AR \to AR'$. Where AR' has the same cardinality to AR such that $AR \cap AR' = \emptyset$.*

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

Definition 9. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:*

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

Definition 10. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define its associated general program as follows:*

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

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

Lemma 2. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if there is an answer set M of Λ_{AF} such that $S = M \cap AR$.*

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

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

Example 3. Let $AF := \langle AR, attacks \rangle$ be the argumentation framework of Example 2. So Λ_{AF} is

$$
\begin{array}{ll}
a' \vee b'. & a' \leftarrow a'. \\
b' \vee a'. & b' \leftarrow b'. \\
c' \vee b'. & c' \vee e'. \\
c' \leftarrow a'. & c' \leftarrow d'. \\
d' \vee c'. & d' \leftarrow b', e'. \\
e' \vee d'. & e' \leftarrow c'. \\
a \leftarrow \neg a'. & b \leftarrow \neg b'. \\
c \leftarrow \neg c'. & d \leftarrow \neg d'. \\
e \leftarrow \neg e'. &
\end{array}
$$

Γ_{AF} has two answer sets which are $\{a', c', e', b, d\}$ and $\{b', c', e', d', a\}$, hence $\{b, d\}$ and $\{a\}$ are the preferred extensions of AF.

5 Conclusions

The preferred semantics is regarded as the most satisfactory argumentation semantics of Dung's argumentation approach. For instance, John Pollock made preferred semantics one of the key ingredients of his revised formalism [15]. Also, it has been shown that some non-monotonic logic programming semantics can be viewed as a special form of this abstract argumentation semantics [5, 11].

It is well-known that the decision problem of the preferred semantics is co-NP-Complete. Then, to have different approaches for inferring this semantics could help to develop argumentation systems based on the preferred semantics. The inference of minimal models from a propositional formula and a logic program is a widely explored problem. Therefore, by defining a relationship between the preferred semantics and minimal models, we identify a wide family of algorithms for inferring the preferred semantics. In particular in this paper, we show how to infer the preferred extensions of an argumentation framework by using UNSAT algorithms and disjunctive answer set solvers.

Acknowledgement

We are grateful to anonymous referees for their useful comments. J.C. Nieves thanks to CONACyT for his PhD Grant. J.C. Nieves and U. Cortés were partially supported by the grant FP6-IST-002307 (ASPIC). The views expressed in this paper are not necessarily those of ASPIC consortium.

References

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

2. ASPIC:Project. ASPIC: Argumentation engine demo. `http://aspic.acl.icnet.uk/`, 2006.
3. T. Bench-Capon. Value-based argumentation frameworks. In *Proceedings of Non Monotonic Reasoning*, pages 444–453, 2002.
4. P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004),*, pages 59–64, June 2004.
5. A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.
6. C. Cayrol, S. Doutre, and J. Mengin. On Decision Problems related to the preferred semantics for argumentation frameworks. *Journal of Logic and Computation*, 13(3):377–403, 2003.
7. T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.
8. S. DLV. Vienna University of Technology. `http://www.dbai.tuwien.ac.at/proj/dlv/`, 1996.
9. S. Doutre and J. Mengin. An algorithm that computes the preferred extensions of argumentation frameworks. In *ECAI'2000, Third International Workshop on Computational Dialectics (CD'2000)*, pages 55–62, Aug. 2000.
10. S. Doutre and J. Mengin. Preferred Extensions of Argumentation Frameworks: Computation and Query Answering. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 272–288. Springer-Verlag, 2001.
11. 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.
12. P. E. Dunne and T. J. M. Bench-Capon. Complexity in value-based argument systems. In *JELIA*, volume 3229 of *LNCS*, pages 360–371. Springer, 2004.
13. 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.
14. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
15. J. L. Pollock. *Cognitive Carpentry: a blueprint for how to build a person*. The MIT Press, May 4, 1995.
16. 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.
17. S. SMODELS. Helsinki University of Technology. `http://www.tcs.hut.fi/Software/smodels/`, 1995.
18. G. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90(1-2):225–279, 1997.