Preferred Extensions as Minimal Models of Clark's Completion Semantics

Mauricio Osorio, Alejandro Santoyo

Universidad de las Américas - Puebla,
Depto. de Actuaría, Física y Matemáticas, Mexico
osoriomauri@googlemail.com, jasrvro@hotmail.com

Abstract. Dung established the connections between several logic programming semantics and various argumentation framework semantics. In this paper we present a characterization of the preferred semantics of argumentation frameworks (which is defined in terms of a maximal admissible set *w.r.t.* set inclusion) in terms of minimal logic models of Clark's completion. Additionally, we make use of integer programming for computing preferred extensions by a method defined by Bell *et al.* [3], which translates a logic program into an integer program which in turn can be solved by an *ad hoc* solver.

Keywords: Argumentation frameworks, preferred extensions, Clark's completion semantics, answer set programming, integer programming.

1 Introduction

Argumentation theory has been a research area in several disciplines such as logic, psychology, philosophy, linguistics, and legal theory. However, even though argumentation theory has a long history, it was just until the arrival of Dung's seminal paper on abstract argumentation theory [7], when this field attracted the attention of many researchers.

Particularly, argumentation theory has become an increasingly important and exciting research topic in Artificial Intelligence (AI). 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.

Currently formal argumentation research has been strongly influenced by Dung's abstract argumentation theory [7]. This approach is mainly oriented to manage the interaction of arguments by introducing a single structure called *Argumentation Framework (AF).* An argumentation framework basically is a tuple of sets: a set of arguments and a set of disagreements between arguments called attacks. Indeed an argumentation framework can be regarded as a directed graph in which the arguments are represented by nodes and the attack relations are represented by arrows.

In [7], four argumentation semantics were introduced: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics.* The central notion of Dung's semantics is the *acceptability of the arguments.* Even though each of these argumentation semantics represents different patterns of selection of arguments, all of them are based on the basic concept of *admissible set*. Informally speaking, an admissible set presents a coherent and defendable point of view in a conflict between arguments.

Dung showed that argumentation can be viewed as logic programming with *negation as failure*. In this setting, he showed that the sets of arguments which can be considered as admissible can be regarded as *logic models* of a given logic program. This result is of great importance because it introduces a general method for generating metainterpreters for argumentation systems and regards argumentation semantics from another point of view in order to identify non-monotonic reasoning features of them. Following this issue, the preferred semantics was characterized by the p-stable semantics [12] in [5]. Moreover, the preferred semantics was characterized by the stable model semantics in [11].

In this work we introduce new results which complete the understanding of Dung's semantics in terms of logic programming semantics with negation as failure. By considering an argumentation framework AF and a uniform mapping of AF into a logic program Π_{AF}, we show that the Clarks's completion minimal models of Π_{AF} characterize the preferred extensions of AF, and also use integer programming fro computing such models.

It is worth mentioning that in section 4 we present some preliminary results of an ongoing research related to find out if integer programming can be used for computing any argumentation framework extension. However, even though we present encouraging results, we still have to be cautious about the conclusions it is possible to draw from them, that is way the section was called "Proof of Concept".

The rest of the paper is divided as follows: In Section 2, we present a basic background about logic programming, Clark's completion semantics, how to compute the minimal models of Clark's completion using mixed integer programming, argumentation theory, and how to map an argumentation framework into a logic program. In Section 3, we present our study about the relationship between minimal models of Clark's completion and preferred extensions. In Section 4 we present a proof of concept experiment for computing preferred extensions using integer programming as part of an ongoing research. In the last section, we outline our conclusions and future work.

2 Background

In this section, we first define the syntax of a valid program, after that the Clark's completion semantics [6] is presented, then the method define by Bell *et al.* for computing the Clark's completion minimal models, then we present some basic concepts of argumentation theory, and finally how to map an argumentation framework to a logic program.

2.1 Clark's Completion

The Clark's Completion of a given logic program is an old concept that has been intensively explored in logic programming literature in order to identify basic properties of logic programming semantics with negation as failure [1, 6]. It is defined as follows: Given a normal logic program P, its completion $Comp(P)$ is obtained in two steps:

1. Each normal clause $a_0 \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n \in P$ is replaced with the formula: $a_0 \leftarrow a_1 \wedge \ldots \wedge a_j \wedge \sim a_{j+1} \wedge \ldots \wedge \sim a_n$.
2. For each symbol $a \in \mathcal{L}_P$, let $Support(a)$ denotes the set of all formulae with a in the head. Suppose $Support(a)$ is the set: $\{a \leftarrow Body_1, \ldots, a \leftarrow Body_m\}$, in which each $Body_i (1 \leq i \leq m)$ is of the form $a_1 \wedge \ldots \wedge a_j \wedge \sim a_{j+1} \wedge \ldots \wedge \sim a_n$. Replace $Support(a)$ with the single formula: $a \leftrightarrow Body_1 \vee \ldots \vee Body_m$.
   If $Support(a) = \emptyset$ then replace it by $\sim a$.

2.2 Translating a Logic Program into an Integer Program

In this section we will show a method defined in [3] to translate $Comp(P)$ into a mixed integer program which then is used to compute the minimal models of $Comp(P)$. Thus it is required some definitions.

Definition 1. *[3]*

1. *A variable X is called binary variable if it can only take on a value of either 0 or 1.*
2. *For all $A \in B_{\mathcal{L}}$, let X_A be a binary variable corresponding to A. The set $\{X_A | A \in B_{\mathcal{L}}\}$ is called a binary variable representation of $B_{\mathcal{L}}$.*

If L is a ground literal, we use X_L as short-hand for the binary variable X_A if L is the positive ground atom A, and for the expression $(1 - X_A)$ if L is the negative atom *not* A.

Definition 2. *[3]*

1. *A binary variable assignment is a mapping $S : \{X_A | A \in B_{\mathcal{L}}\} \to \{0, 1\}$.*
2. *Let I be an interpretation. Define the binary variable assignment S_I corresponding to I as follows:*

$$for\ all\ A \in B_{\mathcal{L}}, S_I(X_A) = \begin{cases} 1 & if\ A \in I; \\ 0 & otherwise. \end{cases}$$

Definition 3. *[3] Suppose we consider the ground clause C below:*

$$A \leftarrow B_1, \cdots, B_n, not\ D_1, \cdots, not\ D_m.$$

We use $if(C)$ to denote the linear constraint:

$$X_A \geq 1 - (\sum_{i=1}^{n}(1 - X_{B_i})) - (\sum_{j=1}^{m} X_{D_j}).$$

Given a logic program P, we use the notation $if(P)$ to denote the set $\{if(C) | C \in grd(P)\}$.

Definition 4. *[3] Let P be a normal logic program and C be a formula in $Comp(grd(P))$.*

1. *If C is of the form: $\neg A$, then the constraint version of C, denoted by $lc(C)$, is: $X_A = 0$.*
2. *if C is of the form: $A \leftrightarrow E_1 \vee \cdots \vee E_k$ where $E_i \equiv L_{i,1}$ & $\cdots$ & L_{i,m_i} for all $1 \leq i \leq k$, and none of the E_i's is "true", then the constraint vesion $lc(C)$ of C is given by the set of constraints $\{if(A \leftarrow E_i) | 1 \leq i \leq k\}$ (called "if-constraints") together with the additional set of constraints (called "only-if" constraints):*
$$X_A \leq Y_1 + \cdots + Y_k$$
where, for all $1 \leq i \leq k$, Y_i is a binary variable defined by the following constraints:
$$Y_i \leq X_{A_{i,1}}$$
$$\cdots$$
$$Y_i \leq X_{A_{i,m_i}}$$
$$Y_i \geq 1 - \sum_{j=1}^{m_i} (1 - X_{A_{i,j}})$$
where $E_i = L_{i,1}$ & $\cdots$ & L_{i,m_i}.
3. *For the special case when C is of the same form as above and some E_i, $1 \leq i \leq k$ is empty, $lc(C)$ is simply: $X_A = 1$.*

The solutions to the above set of constraints with binary variables, corresponds to the Herbrand models of $Comp(grd(P))$. The process we just followed leads us to Definition 5 and Theorem 1.

Definition 5. *[3] Let P be a logic program. The constraint version of P, denoted by $lc(P)$, is the set of constraints versions of all formulas in $Comp(grd(P))$.*

Theorem 1. *[3] Let P be a logic program, let I be an interpretation, and let S_I be the binary variable assignment as defined in Definition 2. Then, S_I is a solution of $lc(P)$ iff I is a model of $Comp(grd(P))$.*

Theorem 1 says that using $lc(P)$ to compute the Herbrand models of $Comp(P)$ (the models that make that all the clauses in $Comp(P)$ hold) is sound and complete, however, Comp(P) might not be consistent, therefore we need the following corollary:

Corollary 1. *[3] Let P be a logic program. Then $Comp(P)$ has an Herbrand model iff lc(P) has a solution.*

Theorem 2. *[3] Let P be a logic program, let M be a model of $Comp(grd(P))$, and let S_M be a binary variable assignment corresponding to M as defined in Definition 2. Then, S_M is an optimal solution of $lc(P)$ minimizing $\sum_{A \in B_{\mathcal{L}}} X_A$, iff M is a card-minimal model of $Comp(grd(P))$.*

Corollary 2. *[3] Let P be a logic program, M be an herbrand interpretation, and S_M be the binary variable assignment corresponding to M as defined in Definition 2. If S_M is an optimal solution of $lc(P)$ that minimizes $\sum_{A \in B_{\mathcal{L}}} X_A$, then M is a minimal (w.r.t. set inclusion) Herbrand model of $Comp(P)$.*

2.3 Argumentation Theory

Now, we define some basic concepts of Dung's argumentation approach. The first one is an argumentation framework.

Definition 6. *[7] 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 $AF := \langle \{a, b\}, \{(a, b), (b, a)\} \rangle$, then AF is represented as it is shown in Figure 1. We say that *a attacks b* (or b is attacked by a) if $attacks(a, b)$ holds.

Let us observe that an argumentation framework is a simple structure which captures the conflicts of a given set of arguments. In order to select *coherent points of views* from a set of conflicts of arguments, Dung introduced a set of *patterns of selection of arguments.* These patterns of selection of arguments were called *argumentation semantics.* Dung defined his argumentation semantics based on the basic concept of *admissible set*:

Definition 7. *[7] 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.*

By considering the concept of admissible set, in [7], Dung introduced four basic argumentation semantics.

Definition 8. *[2] Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \in AR$. We introduce a function $F : 2^{AR} \to 2^{AR}$ such that $F(S) = \{A \mid A\, is\, defended\, by\, S\}$.*

Definition 9. *[2] Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and S be a conflictfree set of argument. S is said to be a complete extension iff $S =$ F(S).*

Definition 10. *[2] [1] Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. An admissible set of argument $S \subseteq AR$ is* preferred *if and only if S is a maximal (w.r.t. inclusion) complete extension of AF.*

[1] This definition differs from the Dung's original, and in fact it is a characterization that it was proved to be equivalent

Since the first argumentation semantics were introduced in [7], Dung's argumentation semantics have given place to different formal studies about the properties of them. One of these formal studies has been to regard them as formal non-monotonic reasoning. In this setting, one can find that the argumentation semantics are closely related to logic programming semantics with *negation as failure*. In the following sections, we will study the preferred extension and a particular relationship with logic programming semantics with negation as failure.

2.4 Mapping from Argumentation Frameworks to Logic Programs

The first step for studying the structure of an argumentation framework as a logic program is to get the logic program that represents such a structure. To this end, we will present the method defined in [13] to map from an argumentation framework into a logic program. Let us observe that this mapping basically is *a declarative representation* of an argumentation framework by having in mind the ideas of *conflict-freeness* and *reinstatement* which are the basic concepts behind the definition of admissible sets.

In this mapping, the predicate $def(x)$ is used, the intended meaning of $def(x)$ is "x is a defeated argument" which means that x cannot be part of an admissible set. A transformation function *w.r.t.* an argument is defined as follows.

Definition 11. *Let $AF := \langle AR, Attacks\rangle$ be an argumentation framework and $a \in AR$. We define the transformation function $\Pi(a)$ as follows:*

$$\Pi(a) = \bigcup_{b:(b,a)\in Attacks} \{def(a) \leftarrow\ not\ def(b)\} \cup$$

$$\bigcup_{b:(b,a)\in Attacks} \{def(a) \leftarrow \bigwedge_{c:(c,b)\in Attacks} def(c)\}$$

The transformation function Π with respect to an argumentation framework AF is defined as follows:

Definition 12. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define its associated normal program as follows:*

$$\Pi_{AF} := \bigcup_{a\in AR} \{\Pi(a)\}.$$

As one can see in Π_{AF}, the language of Π_{AF} only identifies the arguments which can be considered as defeated. By considering *total interpretations*, as the ones suggested by logic programming semantics as stable model semantics [10], we can assume that any argument which is not defeated in a model of Π_{AF} will be acceptable. This means that given an argumentation framework $AF = \langle AR, Attacks\rangle$ if M is a model of Π_{AF}, then any atom $def(x)$ which is false in M will identify an argument x which is acceptable. This assumption suggests a normal clause of the following form:

$$acc(x) \leftarrow not\ def(x).$$

where $acc(x)$ denotes that the argument x can be considered as accepted. This clause essentially fixes as acceptable any argument which is not fixed as defeated in Π_{AF}.

3 Preferred Extensions as Logic Programming Semantics

This section presents the characterization of the preferred extension in terms of the minimal models of Clark's completions, which allows us to propose that integer programming could be used for computing the preferred extension of an argumentation framework, and even other argumentation framework semantics such as the complete extensions.

Please note that while argumentation frameworks extensions are defined in terms of accepted arguments, this work was developed in terms of the defeated ones.

3.1 Preferred Extensions as Minimal Models of Clark's Completion

The first step is to map a given argumentation framework AF into a logic program Π_{AF}, and then characterize it as complete extensions in terms of *logic models* introduced in [4], to this end consider the following proposition:

Proposition 1. *[4] Let $\langle AR, attacks\rangle$ be an argument system. A set $S \subseteq AR$ is a complete extension iff S is a model of the formula*

$$\bigwedge_{a \in AR} ((a \rightarrow \bigwedge_{b:(b,a)\in attacks} \neg b) \wedge (a \leftrightarrow \bigwedge_{b:(b,a)\in attacks} (\bigvee_{c:(c,b)\in attacks} c))).$$

Using this proposition and the following definition:

Definition 13. *[13] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR$, then $m(A) = \{def(x) \mid x \in AR \setminus A\}$.*

It was proved the correspondence between complete extensions and models of $Comp(\Pi_{AF})$. This relationship was formalized by the following Theorem:

Theorem 3. *[13] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR, A$ is a complete extension of AF, iff $m(A)$ is a model of $comp(\Pi_{AF})$.*

Once more note that $m(A)$ is defined in terms of defeated arguments instead of the accepted ones, *i.e.* if $m(A)$ is a model of $comp(\Pi_{AF})$, then it is not directly a complete extension but its complement. Thus, we have the following corollary:

Corollary 3. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR, A$ is a maximal complete extension of AF, iff $m(A)$ is a minimal model of comp(Π_{AF}).*

Proof. Since $m(A)$ is defined in terms of defeated arguments instead of the accepted ones, then to a maximal complete extension corresponds a minimal model of comp(Π_{AF}).

Now, considering Corollary 3 we have the following theorem:

Theorem 4. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. Let $A \subseteq AR$, A is a preferred extension of AF, iff $m(A)$ is a minimal (w.r.t. set inclusion) model of comp(Π_{AF}).*

Proof. .

- *From Theorem 3 we know that A is a complete extension of AF, iff $m(A)$ is a model of comp(Π_{AF}).*
- *From Definition 10 we know that a model A is a preferred extension, iff A is also a maximal (w.r.t. inclusion) complete extension of AF*
- *From Corollary 3 we know that A is a maximal complete extension of AF, iff $m(A)$ is a minimal model of comp(Π_{AF}).*
- *Therefore, A is a preferred extension of AF, iff $m(A)$ is a minimal (w.r.t. set inclusion) model of comp(Π_{AF}).*

Example 1. In order to illustrate the proof, we will determine the preferred extensions of the argumentation framework depicted in Figure 1.
1.- Mapping this argumentation framework AF into a logic program, we get Π_{AF} as follows:
$defeated(b) \leftarrow not\ defeated(a).$ $\quad defeated(b) \leftarrow \top.$
$defeated(a) \leftarrow not\ defeated(b).$ $\quad defeated(a) \leftarrow \top.$
2.- Then we get *comp*(Π_{AF}):
$defeated(a) \leftrightarrow not\ defeated(b) \vee defeated(a)$
$defeated(b) \leftrightarrow not\ defeated(a) \vee defeated(b)$
3.- The models of *comp*(Π_{AF}) are: $\{a\}, \{b\}, \{\}$, therefore they also are *complete extensions* of AF (Theorem 3).
4.- A maximal (*w.r.t.* inclusion) *complete extensions* is also a preferred extension (Definition 10), therefore the models {a}, {b} are also preferred extensions.
5.- However, considering that we worked with defeated arguments, we should take into account the minimal models of *comp*(Π_{AF}) which are the maximal models' complement (Corollary 3). In our case the complement of {a} is {b}, and the complement of {b} is {a}.
6.- Therefore {b} and {a} are the preferred extension of AF (Theorem 3).

The characterization of preferred extensions of an argumentation framework in terms of minimal models of Clark's completions, and the method defined by Bell *et al.* for computing such minimal models can be put together in a method for computing preferred extensions by integer programming, as stated in the following corollary:

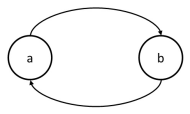

Fig. 1. Graph representation of $AF := \langle \{a, b\}, \{(a, b), (b, a)\} \rangle$.

Corollary 4. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework, let P be the logic program mapped from AF, and let M be an Herbrand interpretation, and S_M be a binary variable assignment corresponding to M as defined in Definition 2. If S_M is an optimal solution of $lc(P)$ that minimizes $\sum_{A \in B_{\mathcal{L}}} X_A$, then M is a minimal (w.r.t. set inclusion) Herbrand model of $comp(P)$, and a preferred extension of AF.*

Proof. Direct from Definition 2 and Theorem 4.

4 Computing Argumentation Framework Extensions with Integer Programming: An Ongoing Research

In this section we present a technical result in a proof of concept of Corollary 4, however, the purpose of this section is not to make a detailed description of the experiment we made, but to make a little reflection about a new possible method for computing argumentation framework extensions. However, even though the results are not conclusive and the experiment is not finished, it is worth commenting this experiment.

In order to have a measure of the performance of the method based on integer programming for computing the preferred extension of an argumentation framework, we considered that it would be important to compare it against other method, and we chose the method based on Answer Set Programming (ASP) encodings.

The integer programming method used the *ad-hoc* Xpress[2] solver to solve the integer program associated to each argumentation frameworks instance, and for the method based on ASP it was used DLV[3].

4.1 Computing Preferred Extension using Mixed Integer Programming

To carry out the experiment it was necessary to develop some pieces of software, which were used according to the following procedure:

1. We developed a java program to map AF_i instances into logic programs Π_{AF_i}.

[2] http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx
[3] http://www.dlvsystem.com/

2. We also developed a C++ program to compute the completion of each Π_{AF_i} generated in previous step, to get $comp(\Pi_{AF_i})$.
3. The same C++ program was used to generate $lc(\Pi_{AF_i})$.
4. The same C++ program also generated, for each $lc(\Pi_{AF_i})$, a *.mos* program which is the language used by Xpress for mathematical programming.
5. Within each *.mos* program it was included the algorithm described after this procedure for computing all the minimal models of each $lc(\Pi_{AF_i})$.
6. Each *.mos* program was executed by Xpress, and the execution times were recorded within each program.

In order to compute all the minimal models of each integer program it was used the following algorithm, and the execution times were recorded immediately after computing all of them.

Algorithm: Minimal Models of $Comp(P)$
Let P be a logic program and $lc(P)$ be constructed as described in Section **??**. *In the following, S is intented to contain all minimal models of $Comp(P)$ and AC is a set of additional constraints.*

```
1. Set $S$ and $AC$ to $\emptyset$.
2. Solve the integer program: Minimize $\sum_{A \in B_{\mathcal{L}}} X_A$ subject to $lc(P) \cup AC$.
3. If no optimal solution can be found, halt and return $S$ as the set of minimal models.
4. Otherwise, let $M$ be the model corresponding to the optimal solution found in step 2. Add $M$ to $S$.
5. Add the constraint $\sum_{A \in M} X_A \leq (k-1)$ to $AC$, where $k$ is the cardinality of $M$. Then go to step 2.
```

The sixty instances used for the experiment ranged from 20 to 50 nodes, and they were taken from the Database and Artificial Intelligence Group Web Page[4].

4.2 Computing Preferred Extension using Answer Set Programming Encodings

For the method based on ASP, which is a declarative programming paradigm under the stable models semantics [9], we used the encodings in [8], where Egly *et al.* presented the system ASPARTIX for reasoning problems in different types of argumentation frameworks by means of computing the answer sets of a datalog program.

ASPARTIX was developed in DLV and is capable to compute several extensions, among them the preferred extensions. For space reasons we can not show the answer set program used to compute the preferred extensions, however the program is available at ASPARTIX's web page[5].

With regard to the experiment, the DLV solver was used for processing each instance, and the execution times were recorded after computing all the minimal models associated to each instance by a C++ program.

[4] http://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/#download
[5] http://www.dbai.tuwien.ac.at/research/project/argumentation/systempage/

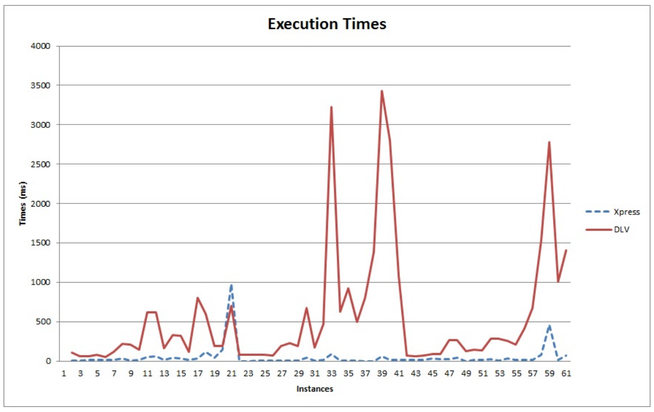

Fig. 2. Graph with Xpress and DLV execition times.

4.3 Proof of Concept Results

Figure 4.3 depicts the results we had in this proof of concept stage, and even though we could not include the table with detailed information, the figure is pretty clear, Xpress outperformed DLV. Remember that the main point at this stage was to find out if an *ad-hoc* integer programming solver can be used for computing preferred extensions of given argumentation framework, therefore the results are not conclusive since the research is not ended.

Thus, at this stage, these results allows us to think that we can go further, in order to be able to draw definitive conclusions.

Maybe the reader can ask why a new method and why integer programming. Well, we should consider that the integer programming method provide a mathematical representation of a given problem, and that integer programming solvers have a long history of development and achievements. Therefore we can infer that this method could become a good alternative for computing argumentation framework extensions.

On the other hand, even if we got just encouraging results, we think that this work represents a good starting point, since it also constitutes a new possible method for computing preferred extensions of argumentation frameworks.

Please, note the importance of this work lies not on the number of theorems or the difficult to reach them, but to make note that the known connection between logic programming and mathematical programming can be used for argumentation frameworks.

5 Conclusions and Future Work

Since Dung introduced his abstract argumentation approach, he proved that his approach can be regarded as a special form of logic programming with negation as failure. In this paper we have showed that preferred extensions can be characterized in terms of minimal models of Clark's completion semantics, by considering a unique mapping of an argumentation framework AF into a logic program.

It is worth mentioning again, that these kind of results also help to understand the close relationship between two successful approaches of nonmonotonic reasoning: argumentation theory and logic programming with negation as failure.

On the other hand, these results helped to build a bridge between argumentation framework semantics and integer programming, but it is required to make an exhaustive experiment to determine the real potential of this method.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Cambridge (2003)
2. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowledge Eng. Review 26(4), 365–410 (2011)
3. Bell, C., Nerode, A., Ng, R.T., Subrahmanian, V.S.: Mixed integer programming methods for computing nonmonotonic deductive databases. Journal of the ACM 41(6), 1178–1215 (1994)
4. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004). pp. 59–64 (June 2004)
5. Carballido, J.L., Nieves, J.C., Osorio, M.: Inferring Preferred Extensions by Pstable Semantics. Iberoamerican Journal of Artificial Intelligence (Inteligencia Artificial) ISSN: 1137-3601, (doi: 10.4114/ia.v13i41.1029) 13(41), 38–53 (2009)
6. Clark, K.L.: Logic and Databases, chap. Negation as Failure, pp. 293–322. Plenum Press (1978)
7. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–358 (1995)
8. Egly, U., Alice Gaggl, S., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument & Computation 1(2), 147–177 (2010), http://www.tandfonline.com/doi/abs/10.1080/19462166.2010.486479
9. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. pp. 1070–1080. MIT Press (1988)
10. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
11. Nieves, J.C., Osorio, M., Cortés, U.: Preferred Extensions as Stable Models. Theory and Practice of Logic Programming 8(4), 527–543 (July 2008)
12. Osorio, M., Navarro, J.A., Arrazola, J.R., Borja, V.: Logics with Common Weak Completions. Journal of Logic and Computation 16(6), 867–890 (2006)
13. Osorio, M., Nieves, J.C., Santoyo, A.: Complete extensions as clark's completion semantics. Accepted paper in ENC 2013 (2013)