Inferring acceptable arguments with Answer Set Programming

Mauricio Osorio and Claudia Zepeda
Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir
Cholula, Puebla, 72820 México
{josorio,sc098382}@mail.udlap.mx

Juan Carlos Nieves and Ulises Cortés
Universitat Politècnica de Catalunya
Departament de Lleguatges i Sistemes Informàtics
c/Jordi Girona 1-3, E08034, Barcelona, Spain
{jcnieves,ia}@lsi.upc.edu

Abstract

Following the argumentation framework and semantics proposed by Dung, we are interested in the problem of deciding which set of acceptable arguments support the decision making in an agent-based platform called CARREL*. It is an agent-agency which mediates organ transplants. We present two possible ways to infer the stable and preferred extensions of an argumentation framework, one in a declarative way using Answer Set Programming(ASP) and the other one in a procedure way.*

Key words *Argumentation, Logic Programming, Answer Set Programming, Agents.*

1. Introduction

Organ transplants are among the most complex medical procedures performed today. At this time, most donated organs and tissues come from patients who are pronounced brain dead as result of disease or injury but also from non-heart-beating donors, and living donors. Behind these medical triumphs, though, lies a fundamental problem. There are far too few organs available for transplantation: at the time of writing this paper, ten people die daily due to the shortage of transplantable organs.

There are two issues that make transplantation management a very complex issue: (i) scarcity of donors, so it is important to try *to maximize* the number of successful transplants (ii) improve donor/recipient matching, because of the diversity and multiplicity of genetic factors involved in the response to the transplant.

In [16], it was proposed an agent-based architecture called CARREL for carrying out the following tasks involved in managing the vast amount of data to be processed: 1) recipient selection (e.g. from patient waiting lists and patient records), 2) organ/tissue allocation (based on organ and tissue records), 3) ensuring adherence to legislation, 4) following approved protocols, 5) preparing delivery plans (e.g. using train and airline schedules).

Actually, in the process of deciding whether an organ is viable or not, it is involved just the transplant coordination unit which has the potential donor[1]. However, it is not rare that doctors disagree in deciding if an organ is viable or not. For instance, organs from a donor infected with endocarditis are usually discarded, even though in the literature we can find successful transplantation from donors infected with this disease [5].

We introduced argumentation in the transplantation process of CARREL with the idea of maximizing the number of viable organs, we proposed that the transplant coordination units that have a potential recipient of an organ could take part in the decision of organ's viability (see [8] for details).

Now, in this paper we are interested in the necessary process of identifying acceptable arguments that support decision making around an organ in each transplant coordination unit. Having a medical transplantation knowledge base, which is in a symbolic form, we want to generate a set of arguments that supports the decision concerning the viability of an organ.

One of the most important contributions in fundamental argumentation was the theory proposed by Dung in 1995 [10]. His proposal explores ways to implement argumentation on computers. Dung defines the acceptability of an argumentation framework in terms of sets of arguments called extensions. The extensions define the semantics of an argumentation framework. Several proposals have been presented to compute extensions [6, 9, 3]. Some of these proposals are specific algorithms, and some others are based in dialectical proof procedures and model checking.

1 We freely admit that our experience of practice is limited (to this date) to the Spanish and Catalan organizations, however they are among the leaders in organ transplantation in the world.

We propose to use Answer Set Programming (ASP) to represent the medical transplantation knowledge base and the argumentation framework proposed by Dung; therefore, we define CARREL-ASP, namely CARREL extended with ASP to perform decision making based on an argumentation framework.

We need to compute acceptable arguments that support the decision making in a direct way from our ASP knowledge base. With this aim, we present two approaches for decision making process in CARREL. The first one characterizes the stable extensions and the preferred extensions of any argumentation framework in a declarative way. The second one presents an efficient algorithm to compute an preferred extension of any uncontroversial argumentation framework.

The stable extensions were already characterized by Dung in terms of logic programming using a meta-interpreter. In this paper, we present an alternative methodology using a translation of an argument framework (AF) into a logic program (P). In fact, we show that both characterizations are equivalent (see Theorem 1). Using our second approach, we suggest another possibility to compute an stable extension of an argumentation framework in a procedure way. Moreover, this approach takes advantage of well known graph algorithms and rewriting systems.

The advantage of characterizing the stable extensions is that this semantics gives answer to a wide-ranking of problems. Dung identifies two interesting argumentation framework families which are resolved by stable extensions: uncontroversial and limited controversial .

One of the limitations of the stable semantics (stable extensions) is that it does not exist for some argumentation framework. For instance, an argumentation framework with paradoxes has not stable extensions; however, it has at least a preferred extension. Therefore, it is important for us to characterize the preferred extensions too. In this paper, we take advantage of the Answer Set Programming features to specify the problem of obtaining the preferred extensions of an argumentation framework. This characterization is based on the notion of minimal generalized answer sets of an abductive logic program and the fact that a preferred extension of an argumentation framework is a maximal (w.r.t. inclusion) admissible set. We show how to obtain the preferred extensions from the minimal generalized answer sets of a particular abductive logic program.

The rest of the paper is structured as follows: In §2, we introduce some fundamental definitions of Answer Sets and present the Dung's argumentation framework. In §3, we present a simple scenario in order to show the utility of argumentation in decision making in CARREL. In §4, we present our characterizations of the stable extensions and the preferred extensions of an argumentation framework. Finally, in §5, we present our conclusions.

2. Background

In this section, we introduce some fundamental definitions of Answer Sets and a short clear description that gives the main facts or ideas about the Dung's argumentation framework. Full details of the Dung's argumentation framework can be found in [10].

2.1. Answer Sets

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. Currently, there are several answer set solvers that find the answer sets of a program, such as: DLV[2] and SMODELS[3].

In this paper, logic programs are understood as propositional theories. We will use the language of propositional logic in the usual way, using propositional symbols: $p, q, \ldots$, propositional connectives $\wedge, \vee, \rightarrow, \bot$ and auxiliary symbols: $(,)$. An *atom* is a propositional symbol. A *literal* is either an atom a (a positive literal) or the negation of an atom $\neg a$ (a negative literal). We assume that for any well formed propositional formula f, $\neg f$ is just an abbreviation of $f \rightarrow \bot$ and $\top$ is an abbreviation of $\bot \rightarrow \bot$. In particular, $f \rightarrow \bot$ is called *constraint* and it is also denoted by $\leftarrow f$. The formula $F \leftarrow G$ is just another way of writing $G \rightarrow F$. $G \leftrightarrow F$ is an abbreviation of $(G \leftarrow F) \wedge (F \leftarrow G)$. Following the traditional notation of logic programming, we may use $:-$ instead of $\leftarrow$, *not* instead of $\neg$ and a, b instead of $a \wedge b$. We will define as a *clause* any well formed formula F. A *regular theory* or *logic program* is just a finite set of clauses, it can be called just *theory* or *program* where no ambiguity arises. We want to stress the fact that in our approach, a program is interpreted as a propositional theory. For readers not familiar with this approach, we recommend [15, 13] for further reading. We will restrict our discussion to propositional programs. As usual in answer set programming, we take for granted that programs with predicate symbols are only an abbreviation of the ground program. The signature of a program P, denoted by $\mathcal{L}_P$, is the

2 http://www.dbai.tuwien.ac.at/proj/dlv/

3 http://www.tcs.hut.fi/Software/smodels/

set of the ground atoms that occur in P. In some definitions we use Heyting's *intuitionistic* logic, which will be denoted by the subscript I. For a given set of atoms M and a program P, we will write $P \vdash_{\mathrm{I}} M$ to abbreviate $P \vdash_{\mathrm{I}} a$ for all $a \in M$ and $P \Vdash_{\mathrm{I}} M$ to denote the fact that $P \vdash_{\mathrm{I}} M$ and P is consistent w.r.t. logic I (i.e. there is no formula A such that $P \vdash_{\mathrm{I}} A$ and $P \vdash_{\mathrm{I}} \neg A$).

The stable model semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [11] and it is usually studied in the context of syntax dependent transformations on programs. We follow an alternative approach started by Pearce [15] and studied in more detail by Osorio *et.al.* [13]. A program is just a propositional theory and an answer set is an intuitionistically consistent and complete extension of the theory obtained by adding only negated literals. This definition has been proved to be a correct characterization of answer sets. In this paper we will take the characterization and notation presented in [13].

Definition 1 (Answer set of a program) *Let P be any theory and M a set of atoms. M is called an answer set for P iff $P \cup \neg(\mathcal{L}_P \setminus M) \cup \neg\neg M \Vdash_{\mathrm{I}} M \cup \neg(\mathcal{L}_P \setminus M)$*

Definition 2 *[2] Let P and P' be a pair of programs such that $P \subseteq P'$. We say that P' is a* conservative extension *of P if the following condition holds: M is an answer set for P iff there is an answer set M' for P' such that $M = M' \cap \mathcal{L}_P$.*

Definition 3 *Let P and P' be a pair of programs. We say that P and P' are equivalent, denoted by $P \equiv P'$, if they have the same answer sets.*

The following definitions are slightly similar to the definitions given in [1].

Definition 4 (Abductive Logic Program) *An abductive logic program is a pair $\langle P, A\rangle$ where P is an arbitrary program and A a set of atoms, called abducibles.*

Definition 5 *Let $\langle P, A\rangle$ be an abductive logic program and $\Delta \subseteq A$. (1) $\langle M, \Delta\rangle$ is an extended generalized answer set of $\langle P, A\rangle$ iff M is an answer set of $P \cup \Delta$. (2) Let $\langle M, \Delta\rangle$ be an extended generalized answer set of $\langle P, A\rangle$ then we say that M is a generalized answer set of $\langle P, A\rangle$.*

Definition 6 *Let $\langle M_1, \Delta_1\rangle$ and $\langle M_2, \Delta_2\rangle$ be extended generalized answer sets of the abductive program $\langle P, A\rangle$, we define $\langle M_1, \Delta_1\rangle < \langle M_2, \Delta_2\rangle$ iff $\Delta_1 \subset \Delta_2$.*

Definition 7 *(1)$\langle M, \Delta\rangle$ is a minimal extended generalized answer set of the abductive program $\langle P, A\rangle$ iff $\langle M, \Delta\rangle$ is an extended generalized answer set of $\langle P, A\rangle$ and it is minimal w.r.t. extended generalized answer set inclusion order. (2) Let $\langle M, \Delta\rangle$ be a minimal extended generalized answer set of the abductive program $\langle P, A\rangle$ then we say that M is a minimal generalized answer set of $\langle P, A\rangle$.*

2.2. Dung's argumentation framework

The key Dung's definition is the concept called Argumentation framework which is defined as follows (see [10]):

Definition 8 *An argumentation framework is a pair $AF = \langle AR, attacks\rangle$, where AR is a set of arguments, and attacks is a binary relation on AR, i.e. $attacks \subseteq AR \times AR$.*

Following Dung's reading, we say that A attacks B (or B is attacked by A) if attacks(A,B) holds. Similarly, we say that a set S of arguments attacks B (or B is attacked by S) if B is attacked by an argument in S.

Definition 9 *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.*

Definition 10 *(1) An argument $A \in AR$ is acceptable with respect to a set S of arguments iff 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 iff each argument in S is acceptable w.r.t. S*

The semantics of an argumentation framework is defined by the notion of preferred extension, which is defined as following:

Definition 11 *A preferred extension of an argumentation framework AF is a maximal (wrt inclusion) admissible set of AF.*

Another relevant concept that Dung introduces is the concept of stable extension.

Definition 12 *A conflict-free set of arguments S is called a stable-extension iff S attacks each argument which does not belong to S.*

Lemma 1 *Every stable extension is a preferred extension, but no vice versa.*

Definition 13 *An argumentation framework AF is said to be coherent if each preferred extension of AF is stable.*

Corollary 1 *Every limited controversial argumentation framework possesses at least one stable extension.*

3. Argumentation for decision making in CARREL

In order to show the utility of argumentation for decision making in CARREL, we shall present a simple scenario[4] where the decision about whether an organ

4 This scenario is following a case of study presented in [5].

from a donor with endocarditis is viable or not should be made.

Let us assume that we have two transplant coordination units, one which is against the viability of the organ (UCT_D) and one which is in favour of the viability of the organ (UCT_R).

UCT_D argues that the organ is not viable, since the donor had endocarditis due to *streptococcus viridans*, then the recipient could be infected by the same microorganism.

In contrast, UCT_R argues that the organ is viable, because the organ presents correct function and correct structure and the infection could be prevented with post-treatment with penicillin, even if the recipient is allergic to penicillin, there is the option of post-treatment with teicoplanin.

Formally, we have an argumentation framework $AF = \langle AR, attacks \rangle$, where AR is the following set of arguments:

$$
\begin{array}{lcl}
nv & = & \text{“}organ\ is\ non\ viable\text{”} \\
v & = & \text{“}organ\ is\ viable\text{”} \\
cfs & = & \text{“}organ\ has\ correct\ function\ and\text{”} \\
 & & correct\ structure \\
risv & = & \text{“}recipient\ could\ be\ infected\ with \\
 & & streptococcus\ viridans\text{”} \\
pp & = & \text{“}post-treatment\ with\ administer \\
 & & penicillin\text{”} \\
pt & = & \text{“}post-treatment\ with\ administer \\
 & & teicoplanin\text{”} \\
ap & = & \text{“}recipient\ is\ allergic\ to\ penicillin\text{”}
\end{array}
$$

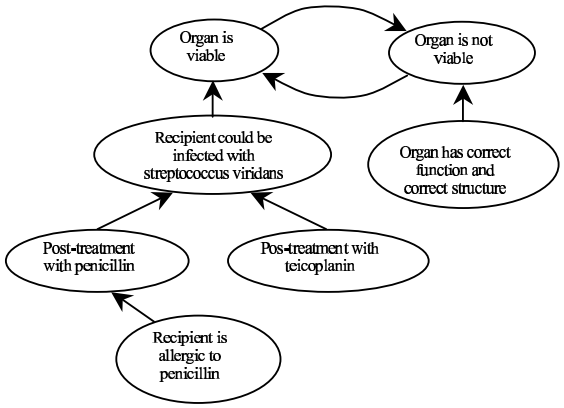

Figure 1. Argumentation for Decision making: A simple scenario.

The relationship between the arguments is shown in Figure 1, where the arrows represent the attacks. If we consider the preferred extension of AF, the acceptable arguments are $\{v, ap, pt, cfs\}$, then we can conclude that the organ is viable. Notice that AF is an uncontroversial argumentation framework which implies that the preferred extension is also a stable extension.

4. Characterization of the Dung's argumentation framework

In this section we present the characterization of the stable extension in terms of answer sets. Then, we present a characterization of the admissible arguments in terms of Answer Set Programming. Finally, we present the characterization of a preferred extension using Answer Sets. This last characterization takes advantage of the Answer Set Programming features to specify the problem of obtaining the preferred extensions given an argumentation framework. The proofs are omitted due to the lack of space[5].

4.1. Characterization of the stable extension

Given an argumentation framework AF, we compute its stable extensions by mapping AF to a normal program P and computing its stable models. This characterization gives the answer to a wide-ranging of argumentation frameworks that Dung called uncontroversial and limited controversial. In particular the limited controversial argumentation framework always possesses at least one stable extension (Corollary 1).

Definition 14 *Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework, $A \in AR$ and $S_A = \{B | (B, A) \in Attacks\}$. We define the transformation function F_t of the argument A as follows:*

$$F_t(A) := A \leftarrow \bigwedge_{B \in S_A} \neg B$$

Notice that if $S_A = \emptyset$ then $F_t(A) := A$.

Definition 15 *Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework. We define its associated argumentation normal program as follow:*

$$P'_{AF} := \{F_t(A) | A \in AR\}$$

Example 1 *Let us consider the argumentation framework of Section 3: $AF = \langle AR, Attacks \rangle$, where*

5 An along version of this paper including the more relevant proofs is [12].

$AR = \{v, nv, risv, cfs, pp, pt, ap\}$ *and* $Attacks = \{(v, nv), (nv, v), (risv, v), (cfs, nv), (pp, risv), (pt, risv), (ap, pp)\}$.

The resulting program P'_{AF} *after applying the transformation* F_t *to each argument of AF is:*

$v \leftarrow \neg nv, \neg risv.$ $\quad nv \leftarrow \neg v, \neg cfs.$
$risv \leftarrow \neg pp, \neg pt.$ $\quad pp \leftarrow \neg ap.$
$ap.$ $\quad pt.$ $\quad cfs.$

The answer set of P'_{AF} *is* $\{v, ap, pt, cfs\}$ *which is the stable extension of the framework AF.*

Formally, we can express this characterization with the following theorem.

Theorem 1 *Let AF be an argumentation framework and E be a set of arguments. E is a STABLE extension of AF iff E is an answer set of* P'_{AF}.

Note: Theorem 1 is less general than Theorem 17 from [10], since Theorem 1 just points out the relationship between stable extensions and answer sets of our characterization.

4.2. Computing preferred arguments

In this section, we present an algorithm to compute the preferred extensions of an uncontroversial argumentation framework taking advantage of the characterization presented in Section 4.1.

We start with the definition of the basic transformation rules. Let $Prog_{\mathcal{L}}$ be the set of all the argumentation normal programs with atoms from $\mathcal{L}$ where $\mathcal{L}$ corresponds to the set of arguments AR of an argumentation framework AF. Let P be an argumentation normal program, and $C \in P$ such that C is of the form $a \leftarrow l_1, \ldots, l_n$; we use $body(C)$ to denote $l_1, \ldots, l_n$ and we define $HEAD(P) = \{a | a \leftarrow l_1, \ldots, l_n \in P\}$. If $body(C)$ is empty, C is called unconditional argument otherwise conditional argument.

Definition 16 *A transformation rule is a binary relation on* $Prog_{\mathcal{L}}$. *The following transformations rules are called basic transformations. Let* $P \in Prog_{\mathcal{L}}$ *be a program.*

RED^+ **:** *If there is an argument a which does not occur in* $HEAP(P)$ *and there is* $b \leftarrow body \in P$ *such that* $\neg a \in body$. RED^+ *reduces the program P to* $P_2 := (P \setminus \{b \leftarrow body\}) \cup \{b \leftarrow (body \setminus \{\neg a\})\}$

RED^- **:** *If there is an unconditional argument* $a \in P$ *and there is* $b \leftarrow body \in P$ *such that* $\neg a \in body$. RED^- *reduces the program P to* $P_2 := (P \setminus \{b \leftarrow body\})$.

Suc **:** *If there is an unconditional argument* $a \in P$ *and there is* $b \leftarrow body \in P$ *such that* $a \in body$. RED^- *reduces the program P to* $P_2 := (P \setminus \{b \leftarrow body\}) \cup \{b \leftarrow (body \setminus \{a\})\}$.

Let $\mathcal{CS}_1$ be the rewriting system which contains exactly the transformations defined in Definition 16.

In order to illustrate $\mathcal{CS}_1$, let us consider the following simple argumentation framework $AF :=< AR, Attacks >$, where $AR := \{a, b, c\}$ and $Attacks := \{(b, a), (c, b)\}$. Then applying F_t to each argument of AF, we get the following argumentation normal program.

$$a \leftarrow \neg b. \quad b \leftarrow \neg c. \quad c.$$

If we apply RED^-, we get the program

$$a \leftarrow \neg b. \qquad c.$$

Now if we apply RED^+, we get the program

$$a. \qquad c.$$

We get a program which is a set of unconditional arguments. Notice that $\{a, c\}$ is the stable extension of AF.

Now we will define some functions which are relevant to our main algorithm.

Definition 17 *We define the function REDU(P,P1,P2) as follows:*
INPUT An argumentation normal program P
OUTPUT Two argumentation programs P1, P2
$P := res\_\mathcal{CS}_1(P)$
$P1 := \{a \leftarrow body | a \leftarrow body \in P \wedge body \neq \emptyset\}$
$P2 := \{a | a \in P\}$
$res\_\mathcal{CS}_1$ *is the reflexive and transitive closure of the application of the transformation rules of* $\mathcal{CS}_1$.

Notice that the function *REDU(P,P1,P2)* is linear, since the transformation rules reduce the size of the program at each step.

Definition 18 *We define the function breakprogram(P, P1, P2) as follows*

INPUT An argumentation normal program P.
OUTPUT Two argumentation programs P1, P2
$G :=$ *get_DAG(P)*
$G^T :=$ *get_transpose(G)*
DFS(G^T*, D[],F[])*
$v :=$ *get_greatest(F[])*
$C :=$ *get_reach_vertices(v,G)*
$P1 := \{r | r : -body \in P \wedge r \in C\}$ $P2 := P \setminus P1$

Where get_DAG gets the direct associated graph G of P, get_transpose gets the transpose of the graph G, DFS is the well known Deft-First Search algorithm which gets two arrays D[] and F[][6]. get_greatest returns the vertex c which has the greatest value in F[]. get_reach_vertices gets the set of vertexes reachable in G from v.

It is not difficult to see that the function *break-program* is a linear function too.

Definition 19 *We define the function get_solution(P) as following*

INPUT An argumentation normal program P
OUTPUT A set of literals S
G := get_DAG(P)
G' := two_coloring_graph(G)
return get_one_color(G')

Where get_DAG gets the direct associated graph of P, two_coloring_graph is the well known two coloring graph algorithm, and get_one_color returns a set of vertices which have the same color.

Since the function *get_solution* uses the two coloring graph which is well known that is linear, the function get_solution is linear too.

Now, we define our algorithm.

Definition 20 *We define the function Calc_mod(P) as following:*
INPUT An uncontroversial argumentation normal program P
OUTPUT An preferred extension of P
REDU(P, P', F)
If $P' = \emptyset$ return F
break-program(P', P1, P2)
M := get_solution(P1)
If $P2 = \emptyset$ return $(F \cup M)$
return $(F \cup$ calc_mod$(P2 \cup M))$

Lemma 2 *Given an argumentation framework AF which is uncontroversial and P'_{AF} its associated argumentation normal program, then Calc_mod(P'_{AF}) computes a stable extension of AF.*

4.3. Characterization of the admissible arguments

In Definition 10 is indicated when a conflict-free set of arguments S is admissible given an argumentation framework $AF = \langle AR, attacks \rangle$. In this subsection we are going to present an encoding using ASP, denoted by $\Pi(AF)$, to obtain from the answer sets of $\Pi(AF)$ the conflict-free sets of arguments of AR that are admissible. In this encoding, we use the predicates $argument(a_i)$, $argument(a_j)$ and $attacked(a_i, a_j)$ to represent that the argument a_j is attacked by the argument a_i. We enumerate the possibility space which specifies that each argument a_j may or may not be admissible. Then, we use the elimination constraints to force that each admissible argument cannot be attacked by an admissible argument, and an admissible argument is an acceptable argument. An argument a_j is acceptable, if it is attacked by an argument a_i such that a_i is attacked by an admissible argument. Finally, the sets of arguments a_i of the predicate $admissible(a_i)$ for each answer set of program $\Pi(AF)$ correspond to the conflict-free sets of AR that are admissible. The rules of the program $\Pi(AF)$ with their intuitive meaning are as follows:

1.The domain specifications.
$argument(a_1) \leftarrow . \ \ldots \ argument(a_m) \leftarrow .$
$attacked(a_i, a_j) \leftarrow . \ \ldots \ attacked(a_k, a_l) \leftarrow .$

2.For each argument X, either X is admissible or **not**.
$admissible(X) \leftarrow \neg\, not\_admissible(X), argument(X).$
$not\_admissible(X) \leftarrow \neg\, admissible(X), argument(X).$

3. An admissible argument Y cannot be attacked by an admissible argument X
$\leftarrow admissible(X), admissible(Y), attacked(X, Y).$

4. An admissible argument X cannot be a *not_acceptable* argument.
$\leftarrow admissible(X), not\_acceptable(X), argument(X).$

5. An argument X is *not_acceptable* if it is attacked by an argument Y such that Y is **not** *attacked_by_an_admissible* argument.
$not\_acceptable(X) \leftarrow attacked(Y, X),$
$\neg\, attacked\_by\_pref(Y), argument(X), argument(Y).$
$attacked\_by\_pref(Y) \leftarrow argument(Y),$
$admissible(X), attacked(X, Y).$

Now we can give a definition and a lemma to obtain the admissible conflict-free sets of an argumentation framework.

Definition 21 *Let $AF = \langle AR, attacks \rangle$ be an argumentation framework and $Adm = \{admissible(X) \mid X \in AR\}$. Let f_{adm} be a function from AR onto Adm such that $f_{adm}(X) = admissible(X)$.*

We define a straightforward generalization of f_{adm} over a set $S \subseteq AR$ as follows: $f_{adm}(S) = \{f_{adm}(s) \mid s \in S\}$. Furthermore, f_{adm} is an invertible function, then

6 See [7] for details.

the inverse function f_{adm}^{-1} from Adm onto AR is defined as follows if $admissible(X) = f_{adm}(X)$ then $f_{adm}^{-1}(admissible(X)) = X$.

Lemma 3 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework. Let M be an answer set of $\Pi(AF)$ such that $M \cap Adm \neq \emptyset$. Then $f_{adm}^{-1}(M \cap Adm)$ is an admissible conflict-free set of AF.*

Example 2 *Let $AF = \langle AR, Attacks\rangle$ be the argumentation framework of the Example 1. Then, the domain specifications of the program $\Pi(AF)$ is defined according to AF.*

The answer sets of the program $\Pi(AF)$ are twelve, we show only some of them after intersecting them with the set Adm:

$\{\}$, $\{admissible(ap)\}$, $\{admissible(cfs)\}$, $\{admissible(pt), admissible(ap)\}, \ldots$

Then, the result of applying f_{adm}^{-1} to the twelve answer sets of the program $\Pi(AF)$ (after the intersection with the set Adm) corresponds to the twelve conflict-free sets of AR that are admissible: $\{\}$, $\{ap\}$, $\{cfs\}$, $\{pt\}$, $\{cfs, pt\}$, $\{v, pt\}$, $\{cfs, ap\}$, $\{pt, ap\}$, $\{v, pt, ap\}$, $\{v, cfs, pt\}$, $\{cfs, pt, ap\}$ *and* $\{v, cfs, pt, ap\}$.

4.4. Characterization of a preferred extension

In this subsection, we are going to present the characterization of a preferred extension using Answer Sets.

We need a preliminary definition about a bijective and restricted function defined on a subset of the signature of a program. This function assigns to each element of the subset of the signature an element that does not occur in the signature of the original program. Moreover, this function will help us to define an abductive logic program.

Definition 22 *Let $\mathcal{L}_P$ be the signature of a program P. Let $\mathcal{T}_P$ be a signature of the same cardinality of $\mathcal{L}_P$ such that $\mathcal{L}_P \cap \mathcal{T}_P = \emptyset$. Let $\mathcal{L}_P^\bullet$ any fixed bijective function from $\mathcal{L}_P$ onto $\mathcal{T}_P$.*

We shall denote the image of a under $\mathcal{L}_P^\bullet$ as $a^\bullet$, namely, $\mathcal{L}_P^\bullet(a) = a^\bullet$. We define the straightforward generalization of $\mathcal{L}_P^\bullet$ over $A \subseteq \mathcal{L}_P$ as follows: $\mathcal{L}_P^\bullet(A) = \{\mathcal{L}_P^\bullet(a) \mid a \in A\}$. Abusing of the notation, we let $A^\bullet$ represent $\mathcal{L}_P^\bullet(A)$.

The characterization of a preferred extension is introduced as a corollary of Lemma 4: Corollary 2. This Corollary takes advantage of the correspondence between the definition of a preferred extension as a maximal (w.r.t. inclusion) admissible set of AF (see Definition 11) and the definition of a maximal answer set. Moreover, Corollary 2 says that the preferred extension of an argumentation framework AF is obtained by getting the minimal generalized answer sets of an abductive logic program. The abductive logic program corresponds to a particular translation of the program $\Pi(AF)$. Program $\Pi(AF)$ was defined in the previous subsection. Hence, we are going to present some definitions about maximal answer sets and the particular abductive logic program used to obtain the preferred extension.

Definition 23 *Let $\{S_i : i \in I\}$ be a collection of subsets of U such that $\bigcup_{i\in I} S_i = U$ and $A \subseteq U$. We say that S_i is a maximal set w.r.t. A among the collection $\{S_i : i \in I\}$ iff there is no S_j with $j \neq i$ such that $(S_i \cap A) \subset (S_j \cap A)$.*

Definition 24 *Let P be a consistent program and $\{M_i : i \in I\}$ be the collection of answer sets of P. Let $A \subseteq \mathcal{L}_P$. We say that M_i is a maximal answer set w.r.t. A iff M_i is an answer set of P such that M_i is a maximal set w.r.t. A among the collection of answer sets of P.*

The translation of a program w. r. t. a set of atoms consists in adding a set of constraints to the original program as follows:

Definition 25 *Let P be a program and $A \subseteq \mathcal{L}_P$. We define the translation of program P w.r.t. A as $P \cup Cons_A$ where $Cons_A = \{\leftarrow \neg a, \neg a^\bullet \mid a \in A\}$.*

Lemma 4 *Let P and M be a program and an answer set of P respectively. Let $A \subseteq \mathcal{L}_P$. Then M is a maximal answer set of P w.r.t A iff $M \cup A^\bullet$ is a minimal generalized answer set of the abductive logic program $\langle (P \cup Cons_A), A^\bullet\rangle$.*

Corollary 2 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework and $A = f_{adm}(AR)$. Let $P = \Pi(AF)$ be the program obtained from AF and M be an answer set of P. Then $f_{adm}^{-1}(M \cap A)$ is a preferred extension of AF iff $M \cup A^\bullet$ is a minimal generalized answer set of the abductive logic program $\langle (P \cup Cons_A), A^\bullet\rangle$.*

It is worth mentioning that the preferred extension of the argumentation framework of the Example 1 when we apply to it Corollary 2 coincides with the result given in section 3: $\{v, ap, pt, cfs\}$ [7].

In order to compute the preferred extension of an argumentation framework AF is possible to use the characterization of minimal generalized answer sets in terms of ordered disjunction programs presented in [14]. In this characterization an abductive program can be represented using ordered disjunction [4]. Hence, in

7 Similarly, since Example 1 is uncontroversial we could also use Definition 15 obtaining the same preferred extension.

[14] is defined a translation of an abductive logic program into an ordered program. Then by running the translated program in *Psmodels* [4], we can obtain the different inclusion preferred answer sets. Finally, we can obtain from each preferred answer set the preferred extensions of the argumentation framework AF.

5. Conclusions

We present two approaches for the support of decision making process in CARREL. The first characterizes the stable and preferred semantics of an argumentation framework in a declarative way. The characterization of preferred semantics is based on the notion of minimal generalized answer sets of an abductive logic program. We think that our approach can be useful not only to characterize the preferred semantics, but also other problems. The second approach presents an efficient algorithm to compute a preferred extension of an uncontroversial argumentation framework taking advantage of the translation of an argument framework into a logic program.

Acknowledgements

U. Cortés and J. C. Nieves were supported in part by the Grant FP6-IST-002307 (ASPIC).

References

[1] M. Balduccini and M. Gelfond. Logic Programs with Consistency-Restoring Rules. In P. Doherty, J. McCarthy, and M.-A. Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, March 2003.

[2] C. Baral. *Knowledge Representation, reasoning and declarative problem solving with Answer Sets*. Cambridge University Press, Cambridge, 2003.

[3] 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.

[4] G. Brewka, I. Niemelä, and T. Syrjänen. Implementing Ordered Disjunction Using Answer Set Solvers for Normal Programs. In *Proceedings of the 8th European Workshop Logic in Artificial Inteligence JELIA 2002*. Springer, 2002.

[5] F. Caballero, A. López-Navidad, M. Perea, C. Cabrer, L. Guirado, and R. Sòla. Successful liver and kidney transplantation from cadaveric donor with left-sided bacterial endocarditis. *American Journal of Transplantation*, 5:781–787, 2005.

[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] U. Cortés, P. Tolchinsky, J. C. Nieves, A. López-Navidad, and F. Caballero. Arguing the discard of organs for transplantation in CARREL. In *CATAI 2005*, pages 93–105, 2005.

[9] Y. Dimopoulos, B. Nebel, and F. Toni. Preferred Arguments are Harder to Compute than Stable Extensions. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 36–41, 1999.

[10] 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.

[11] 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.

[12] J. C. Nieves, M. Osorio, C. Zepeda, and U. Cortés. The appropriateness of ASP for argumentation. Technical report, Departament de Lleguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2005.

[13] M. Osorio, J. A. Navarro, and J. Arrazola. Applications of Intuitionistic Logic in Answer Set Programming. *Theory and Practice of Logic Programming (TPLP)*, 4:325–354, May 2004.

[14] M. Osorio, M. Ortiz, and C. Zepeda. Using cr-rules for evacuation planning. In G. D. I. Luna, O. F. Chaves, and M. O. Galindo, editors, *IX Ibero-american Workshops on Artificial Inteligence*, pages 56–63, 1994.

[15] D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.

[16] J. Vázquez-Salceda, U. Cortés, J. Padget, A. López-Navidad, and F. Caballero. Extending the carrel system to mediate in the organ and tissue allocation processes: A first approach. *Artificial Intelligence in Medicine*, 3:233–258, 2003.