Semantics for Possibilistic Disjunctive Programs

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. In this paper by considering an answer set programming approach and some basic ideas from possibilistic logic, we introduce a possibilistic disjunctive logic programming approach that is able to deal with reasoning under uncertainty and incomplete information. Our approach permits to use explicit labels like certain, probable, plausible, *etc.*, for capturing the incomplete state of a belief in a disjunctive logic program.

1 Introduction

Many decisions that we make in our common life are based on beliefs concerning the likelihood of uncertain events. In fact, we commonly use statements such as "I think that ...", "chances are ...", "it is *probable* that ...", "it is *plausible* that ...", *etc.*, for supporting our decisions. In this kind of statements usually we have appealed to our experience or our commonsense. It is not surprising to think that a reasoning based on these kind of statements could reach *bias conclusions*. However these conclusions could reflect the experience or commonsense of an expert. Pelletier and Elio pointed out in [19] that people simply have tendencies to ignore certain information because of the (evolutionary) necessity to make decisions quickly. This gives rise to "biases" in judgments concerning what they "really" want to do. A deep study of the importance of the biassed and heuristics in judgment under uncertainty is presented in the book [12].

In view of the fact that we know that a reasoning based on statements which are quantified by relative likelihoods could capture our experience or our commonsense, the question is: how could these statements be captured by real application systems like Multi Agent Systems? For those steeped in probability, Halpern has remarked in [10] that probability has its problems. For one thing, the numbers are not always available. For another, the commitment to numbers means that any two events must be comparable in terms of their probabilities: either one event is more probable than the other, or they have equal probability. Also in [13], McCarthy and Hayes pointed out that attaching probabilities to a statement has the following objections:

1. It is not clear how to attach probabilities to statements containing quantifiers in a way that corresponds to the amount of conviction people have.

2. The information necessary to assign numerical probabilities is not ordinarily available. Therefore, a formalism that required numerical probabilities would be epistemologically inadequate.

Now, the question is why not to use explicit labels like *possible*, *probable*, *plausible*, *etc.*, for capturing the incomplete state of a belief in a logic program when the numerical representations are not available or difficult to get. For instance, these kind of labels have been explored in argumentation theory for modeling incomplete information of an argument [7, 17].

During the last two decades, one of the most successful logic programming approaches has been Answer Set Programming (ASP). ASP is the realization of much theoretical work on Non-monotonic Reasoning and Artificial Intelligence applications. It represents a new paradigm for logic programming that allows, using the concept of *negation as failure*, to handle problems with default knowledge and produce non-monotonic reasoning [1]. In [15], it was proposed a possibilistic framework for reasoning under uncertainty. It is a combination between ASP and possibilistic logic [6]. This framework is able to deal with reasoning that is at the same time non-monotonic and uncertain. Nicolas *et al.*'s approach is based on the concept of *possibilistic stable model* which defines a semantics for possibilistic normal logic programs. One weak point of this approach is that it relies on the expressiveness of normal logic programs and it always depends on a numerical representation for capturing the incomplete state of a belief. Since Nicolas *et al.*'s approach does not permit disjunctions in the head of a possibilistic clause, there is not a natural way for expressing incomplete information.

In this paper, we introduce the use of *possibilistic disjunctive clauses* which are able to capture *incomplete information* and *incomplete states of a knowledge base* at the same time. It is important to point out that our approach is not exactly a generalization of Nicolas *et al.*'s approach. Whereas Nicolas *et al.*'s approach only permits to express the states of a belief by totally ordered sets, our approach permits to consider partially ordered sets for expressing the states of a belief. Moreover we do not adopt to use *strict* α*-cuts* for handling an inconsistent possibilistic logic program. However in the class of possibilistic normal logic programs, our approach coincides with Nicolas *et al.*'s approach when it considers totally ordered sets for capturing the incomplete state of a belief and the possibilistic program is consistent.

One of our main motivations for considering a generalization of the possibilistic stable models was our necessity for modeling medical knowledge. We have been working in the decision making process for deciding if a human organ is viable or not for being transplanted [21, 16, 20]. Our experience suggests that in our medical domain, we require a *qualitative* theory of default reasoning like ASP for modeling incomplete information and a *quantitative* theory like possibilistic logic for modeling uncertain events.

By considering partially ordered sets, it is possible to capture the confidence of a claim by using qualifiers like the Toulmin's famous "qualifiers"[22]. For instance, in [7] Fox and Modgil discuss the expressiveness of these qualifiers for capturing the uncertainty of medical claims. We use relative likelihoods for modeling different qualifiers *e.g., certain, confirmed, probable, plausible, supported* and *open*[3], where each qualifiers is a possible world/class of beliefs. The user can provide a likelihood ordering for the worlds/classes of beliefs as it is shown in Fig. 1.

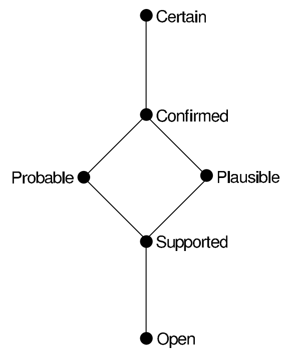

Fig. 1. A lattice where the following relations hold: $Open \preceq Supported$, $Supported \preceq Plausible$, $Supported \preceq Probable$, $Probable \preceq Confirmed$, $Plausible \preceq Confirmed$, and $Confirmed \preceq Certain$.

In general terms, we are proposing a possibilistic disjunctive logic programming approach that is able to deal with reasoning under uncertainty and incomplete information. Moreover, it permits to encode uncertainty by using either numerical values or relative likelihoods. In terms of computability, we observe that our approach is computable.

It worth mentioning that in [18], we presented an alternative semantics for possibilistic logic program. The main difference of this semantics *w.r.t.* the semantics that we are presenting here is that the semantics presented in [18] is based on an operator $\mathcal{T}$ which is inspired in partial evaluation [2] and GMP (an inference rule of possibilistic logic [6]), and the semantics presented in this paper is only based on the proof theory of possibilistic logic.

The rest of the paper is divided as follows: In §2, some basic concepts of possibilistic logic and standard ASP are presented. In §3, the syntax and semantics of our possibilistic framework are presented. In §4, we discuss a little bit the inconsistency of a possibilistic knowledge base. Finally in the last section, our conclusions are presented and the future work is outlined.

2 Background

In this section, we define some basic concepts of Possibilistic Logic and ASP. We assume familiarity with basic concepts in classic logic and in the semantics of logic pro-

[3] This set of labels was taken from [7].

grams *e.g.*, interpretations, models, *etc*. A good introductory treatment of these concepts can be found in [1, 14]

2.1 Possibilistic Logic

A necessity-valued formula is a pair $(\varphi \ \alpha)$ where φ is a classical logic formula and $\alpha \in (0, 1]$ is a positive number. The pair $(\varphi \ \alpha)$ expresses that the formula φ is certain at least to the level α, *i.e.* $N(\varphi) \geq \alpha$, where N is a necessity measure modeling our possibly incomplete state knowledge [6]. α is not a probability (like it is in probability theory) but it induces a certainty (or confidence) scale. This value is determined by the expert providing the knowledge base. A necessity-valued knowledge base is then defined as a finite set (*i.e.* a conjunction) of necessity-valued formulae.

Dubois *et al.*[6] introduced a formal system for necessity-valued logic which is based in the following axioms schemata (propositional case):

(A1) $(\varphi \rightarrow (\psi \rightarrow \varphi) \ 1)$
(A2) $((\varphi \rightarrow (\psi \rightarrow \xi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \xi)) \ 1)$
(A3) $((\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi) \ 1)$

Inference rules:

(GMP) $(\varphi \ \alpha), (\varphi \rightarrow \psi \ \beta) \vdash (\psi \ min\{\alpha, \beta\})$
(S) $(\varphi \ \alpha) \vdash (\varphi \ \beta)$ if $\beta \leq \alpha$

According to Dubois *et al.*, basically we need a complete lattice in order to express the levels of uncertainty in Possibilistic Logic. Dubois *et al.*, extended the axioms schemata and the inference rules for considering partially ordered sets. We shall denote by $\vdash_{PL}$ the inference under Possibilistic Logic without paying attention if the necessity-valued formulae are using either a totally ordered set or a partially ordered set for expressing the levels of uncertainty.

The problem of inferring automatically the necessity-value of a classical formula from a possibilistic base was solved by an extended version of *resolution* for possibilistic logic (see [6] for details).

2.2 Answer Set Programming

Syntaxis The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: $p_0, p_1, ...$
(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \ not, \bot$
(iii) auxiliary symbols : (,).

where $\vee, \wedge, \leftarrow$ are 2-place connectives, $\neg$, not are 1-place connective and $\bot$ is 0-place connective. The proposition symbols, $\bot$, and propositional symbols of the form $\neg p_i \ \ (i \geq 0)$ stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. The negation sign $\neg$ is regarded as the so called *strong negation* by the ASP's literature and the negation not as the *negation as failure*. A literal is an atom, a, or the negation of an atom $not\ a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $not\ \{a_1, ..., a_n\}$ to denote the set of literals $\{not\ a_1, ..., not\ a_n\}$.

An extended disjunctive clause, *C*, is denoted:

$$a_1 \vee \ldots \vee a_m \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n$$

where $m \geq 0, n \geq 0$, each a_i is an atom. When $n = 0$ and $m > 0$ the clause is an abbreviation of $a_1 \vee \ldots \vee a_m$. When $m = 0$ the clause is an abbreviation of $\bot \leftarrow a_1, \ldots, a_n$ such that $\bot$ is the proposition symbol that always evaluates to false. Clauses of this form are called constraints (the rest, non-constraint clauses). An extended disjunctive program P is a finite set of extended disjunctive clauses. By $\mathcal{L}_P$, we denote the set of atoms in the language of P.

We will manage the strong negation ($\neg$), in our logic programs, as it is done in ASP [1]. Basically, it is replaced each negative atom $\neg a$ by a new atom symbol a' which does not appear in the language of the program. For instance, let P be the normal program:

$a \leftarrow q.$
$\neg q \leftarrow r.$
$q.$
$r.$

Then replacing each negative atom by a new atom symbol, we will have:
$a \leftarrow q.$
$q' \leftarrow r.$
$q.$
$r.$

In order not to allow inconsistent models of the non-possibilistic logic programs, usually it is added a constraint of the form $\leftarrow q, q'$. We will omit this constraint in order to allow complementary literals in a possibilistic answer set. However the user could add this constraint without losing generality.

We denote an extended disjunctive clause *C* by $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$, where $\mathcal{A}$ contains all the head atoms, $\mathcal{B}^+$ contains all the positive body atoms and $\mathcal{B}^-$ contains all the negative body atoms. When $\mathcal{B}^- = \emptyset$, the clause is called positive disjunctive clause. A set of positive disjunctive clauses is called a positive disjunctive logic program. When $\mathcal{A}$ is a singleton set, the clause can be regarded as a normal clause. A normal logic program is a finite set of normal clauses. Finally, when $\mathcal{A}$ is a singleton set and $\mathcal{B}^- = \emptyset$, the clause can be also regarded as a definite clause. A finite set of definite clauses is called a definite logic program.

Given a set of proposition symbols S and a theory (a set of well-formed formulae) Γ in a logic X. If $\Gamma \vdash_X S$ if and only if $\forall s \in S\ \Gamma \vdash_X s$.

Semantics The answer set semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [8] 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 [8] and it was presented in [9]: Let *P* be any extended disjunctive program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the positive program obtained from *P* by deleting

(i) each rule that has a formula $not\ a$ in its body with $a \in S$, and then
(ii) all formulae of the form $not\ a$ in the bodies of the remaining rules.

Clearly P^S does not contain not (this means that P^S is either a positive disjunctive logic program or a definite logic program), hence *S* is an answer set of *P* if and only if *S* is a minimal model of P^S.

In the answer set definition, we are omitting the restriction that if S has a pair of complementary literals then $S := \mathcal{L}_P$. This means that we are allowing that an answer set could have a pair of complementary literals. For instance, let us consider the program P:

$a.$ $\neg a.$ $b.$

then, the only answer set of this program is : $\{a, \neg a, b\}$.

It is worth mentioning that in the literature there are several forms for handling an inconsistency program. For instance, by applying the original definition [9] the only stable model is: $\{a, \neg a, b, \neg b\}$. On the other hand, the DLV system [5] returns no models if the program is inconsistent.

3 Possibilistic Disjunctive Logic Programs

In this section, we introduce our possibilistic logic programming framework. We shall start by defining the syntax of a valid program and some relevant concepts, after that we shall define the semantics for the possibilistict disjunctive logic programs.

3.1 Syntax

First of all, we start defining some relevant concepts[4]. In all the paper, we will consider finite lattices. This convention was taken based on the assumption that in real applications we will rarely have an infinite set of labels for expressing the incomplete state of a knowledge base.

A *possibilistic literal* is a pair $l = (a, q) \in L \times Q$, where L is a finite set of literals and $(Q, \leq)$ is a lattice (since the lattice is finite then it is complete). We apply the projection $*$ as follows: $l^* = a$. Given a set of possibilistic literals S, we define the generalization of $*$ over S as follows: $S^* = \{l^* | l \in S\}$. Given a lattice $(Q, \leq)$ and $S \subseteq Q$, $LUB(S)$ denotes the least upper bound of S and $GLB(S)$ denotes the greatest lower bound of S.

Definition 1. *Let L be a finite set of literals and (Q,≤) be a lattice. Consider $\mathcal{PS} = 2^{L \times Q}$ the finite set of all the possibilistic literal sets induced by L and Q. $\forall A, B \in \mathcal{PS}$, we define.*

$$
\begin{aligned}
A \sqcap B &= \{(x, GLB\{q_1, q_2\}) | (x, q_1) \in A \wedge (x, q_2) \in B\} \\
A \sqcup B &= \{(x, q) | (x, q) \in A\ and\ x \notin B^*\} \cup \\
&\quad \{(x, q) | x \notin A^*\ and\ (x, q) \in B\} \cup \\
&\quad \{(x, LUB\{q_1, q_2\}) | (x, q_1) \in A \textit{ and } (x, q_2) \in B\}. \\
A \sqsubseteq B &\Longleftrightarrow A^* \subseteq B^*\textit{, and } \forall x, q_1, q_2, \\
&\quad (x, q_1) \in A \wedge (x, q_2) \in B \textit{ then } q_1 \leq q_2.
\end{aligned}
$$

[4] Some concepts presented in this subsection extend some terms presented in [15].

Proposition 1. *$(\mathcal{PS}, \sqsubseteq)$ is a complete lattice.*

Proof. The proof is straightforward.

Now, we define the syntax of a valid possibilistic logic program. Let $(Q, \leq)$ be a lattice. A possibilistic disjunctive clause is of the form:

$$r = (\alpha : \mathcal{A} \leftarrow \mathcal{B}^{+},\ not\ \mathcal{B}^{-})$$

where $\alpha \in Q$. The projection $*$ for a possibilistic clause is $r^{*} = \mathcal{A} \leftarrow \mathcal{B}^{+},\ not\ \mathcal{B}^{-}$. $n(r) = \alpha$ is a necessity degree representing the certainty level of the information described by r. A possibilistic constraint is of the form:

$$c = (TOP_Q :\ \leftarrow \mathcal{B}^{+},\ not\ \mathcal{B}^{-})$$

where TOP_Q is the top of the lattice $(Q, \leq)$. As in possibilistic clauses, the projection $*$ for a possibilistic constraint is : $c^{*} =\ \leftarrow \mathcal{B}^{+},\ not\ \mathcal{B}^{-}$. A possibilistic disjunctive logic program P is a tuple of the form $\langle (Q, \leq), N \rangle$, where N is a finite set of possibilistic disjunctive clauses and possibilistic constraints. The generalization of $*$ over P is as follows: $P^{*} = \{r^{*} | r \in N\}$. Notice that P^{*} is an extended disjunctive program. When P^{*} is a normal program, P is called a possibilistic normal program. Also when P^{*} is a positive disjunctive program, P is called a positive possibilistic logic program.

In order to illustrate a possibilistic disjunctive logic program, let us consider the following scenario:

Example 1. Let us suppose that a patient suffering from certain symptoms takes a blood test, and that the results show the presence of a bacterium of a certain category in his blood. There are two types of bacteria in this category, and the blood test *does not pinpoint* whether the bacterium presented in the blood is either streptococcus viridans or X. The problem is that if the bacteria is streptococcus viridans the patient have to be treated by antibiotics of large spectrum because streptococcus viridans suggests *endocarditis*. However, the doctor tries not to prescribe antibiotics of large spectrum, because they are harmful to the immune system. Then, the doctor in this case must evaluate each potential choice, where each potential choice has different levels of uncertainty[5].

In order to encode this scenario let us consider the lattice presented in Fig. 1. Then, we could model the doctor's beliefs as follows: First, one doctor's belief is that it is *confirmed* that the patient has a bacterium of category n. Then, this belief could be encoded by:

$confirmed : category\_n.$

Another doctor's belief is that the category n *implies two possible bacteria*. Then it could be encoded by:

[5] This example is an adaptation of Example 3 from [16] and Example 6 from [11]. Part of the medical information was taken from [3].

$certain$: $streptoccus\_viridans \vee bacterium\_x \leftarrow category\_n$.

Now, if the bacterium is $streptococcus\_viridans$, then the patient *have to be* treated by antibiotics of large spectrum.

$certain$: $antibiotics\_large\_spectrum \leftarrow streptococcus\_viridans$.

If the bacteria is x, then the patient *could be* treated without antibiotics of large spectrum.

$probable$: $alternative\_treatment \leftarrow bacterium\_x$.

It is plausible that the doctor does not use antibiotics of large spectrum if it has not been established that there is not another alternative treatment.

$plausible$: $\neg antibiotics\_large\_spectrum \leftarrow not\ \neg alternative\_treatment$.

Finally, it is also plausible that the doctor does not use an alternative treatment if it has not been established that antibiotics of large spectrum are not necessary.

$plausible$: $\neg alternative\_treatment \leftarrow not\ \neg antibiotics\_large\_spectrum$.

We can appreciate the use of relative likelihoods could facilitate the modeling of the incomplete state of the knowledge.

3.2 Semantics

The semantics of the possibilistic disjunctive logic programs is defined in terms of a syntactic reduction which is defined as follows:

Definition 2 (Reduction P^M). *Let $P = \langle (Q, \leq), N \rangle$ be a possibilistic disjunctive logic program, M be a set of literals. P reduced by M is the positive possibilistic disjunctive program:*
$P^M := \{(n(r) : \mathcal{A} \cap M \leftarrow \mathcal{B}^+) | r \in N, \mathcal{A} \cap M \neq \emptyset, \mathcal{B}^- \cap M = \emptyset, \mathcal{B}^+ \subseteq M\}$
where r^* is of the form $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$.

Notice that $(P^*)^M$ is not exactly the *Gelfond-Lifschitz reduction*. In fact, our reduction is stronger than Gelfond-Lifschitz reduction when P^* is a disjunctive program.

In order to illustrate the definition, we present a couple of examples.

Example 2. First, let us consider again the possibilistic program P presented in Section 3.1 and the possibilistic set of literals: $S := \{(category\_n, confirmed)$, $(streptoccus\_viridans, certain)$, $(antibiotics\_large\_spectrum, certain)$, $(\neg alternative\_treatment, plausible)\}$. Then, it is easy to see that P^{S^*} is:

$confirmed$: $category\_n$.
$certain$: $streptoccus\_viridans \leftarrow category\_n$.
$certain$: $antibiotics\_large\_spectrum \leftarrow streptococcus\_viridans$.
$plausible$: $\neg alternative\_treatment$.

Notice that in this example, the reduced program has no possibilistic disjunctive clauses. However, it does not always happen in the reduced programs. Let us consider the following example, where the reduced program has possibilistic disjunctive clauses.

Example 3. First, let S be the set $\{(a, 0.7), (b, 0.6)\}$ and P_1 be the following possibilistic logic program where the possibilistic clauses are built under the lattice $Q := (\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}, \leq)$[6]:

$0.7 : a \vee b \leftarrow not\ c.$
$0.6 : c \leftarrow not\ a, not\ b.$
$0.8 : a \leftarrow b.$
$0.6 : b \leftarrow a.$

Then, the program $P_1^{S^*}$ is:
$0.7 : a \vee b.$ $\quad 0.8 : a \leftarrow b.$ $\quad 0.6 : b \leftarrow a.$

Notice that always the reduced program $(P^{M^*})^*$ is either a positive disjunctive logic program or at the best of the cases a definite logic program *e.g.,* P^{S^*} of Example 2.

Once a possibilistic logic program P has been reduced by a set of possibilistic literals M, it is possible to test whether M is a possibilistic answer set of the program P by considering the following definition.

Definition 3 (Possibilistic answer set). *Let $P = \langle (Q, \leq), N \rangle$ be a possibilistic disjunctive logic program and M be a set of possibilistic literals such that M^* is an answer set of P^*. M is a possibilistic answer set of P if and only if $P^{M^*} \vdash_{PL} M$ and $\nexists M' \in \mathcal{PS}$ such that $M' \neq M$, $P^{(M')^*} \vdash_{PL} M'$ and $M \sqsubseteq M'$.*

In order to illustrate the definition, let us consider again the reduced program P^{S^*} of Example 2. By applying successive GMP to P^{S^*}, we can infer S from P^{S^*}. Then, this means that $P^{S^*} \vdash_{PL} S$. It is important to see that S is the only set that we can be infer from P^{S^*}, then S is unique. It is easy to see that S^* is an answer set of P^*, therefore we can say that S is a possibilistic answer set of P.

Example 4. Let P_1 be the possibilistic program from Example 3 and $S := \{(a, 0.7), (b, 0.6)\}$. We have already seen that $P_1^{S^*}$ is:
$0.7 : a \vee b.$ $\quad 0.8 : a \leftarrow b.$ $\quad 0.6 : b \leftarrow a.$
Then, we want to know if S is a *possibilistic answer set* of P_1. First of all, it is easy to see that S^* is an answer set of P_1^*. Hence, we have to construct a proof in possibilistic logic for $(a, 0.7)$ and $(b, 0.6)$ by considering $P_1^{S^*}$. Let us consider the proof for the possibilistic atom $(a, 0.7)$:

1. $((a \vee b) \rightarrow ((b \rightarrow a) \rightarrow a)$	$1)$	Tautology
2. $(a \vee b$	$0.7)$	Premise from P_1^S
3. $((b \rightarrow a) \rightarrow a)$	$0.7)$	From 1 and 2 by GMP
4. $(b \rightarrow a$	$0.8)$	Premise from P_1^S
5. $(a$	$0.7)$	From 3 and 4 by GMP

[6] $\leq$ is the standard relation between rational numbers.

The proof for $(b, 0.6)$ is similar to the proof of $(a, 0.7)$. Notice that $\nexists\ S'$ such that $P_1^{(S')^*} \vdash_{PL} S'$ and $S \sqsubseteq S'$. Therefore, we can conclude that S is a *possibilistic answer set* of P_1.

We have to notice that there is an important condition *w.r.t.* the definition of the *possibilistic answer sets*. This is that a possibilistic set S is not a possibilistic answer set of a possibilistic logic program P if S^* is not an answer set of the extended logic program P^*. This condition guarantees any clause of P^* is satisfied by M^*. For instance, let us consider the possibilistic logic program P:

$0.4 : a. \qquad 0.6 : b.$

and the possibilistic set $S = \{(a, 0.4)\}$. We can see that $P^{S^*} \vdash_{PL} S$. However, S^* is not an answer set of P^*. Then S could not be a possibilistic answer set of P. Hence, a straightforward relation between the possibilistic answer semantics and answer set semantics is formalized by the following proposition.

Proposition 2. *Let P be a possibilistic disjunctive logic program. If M is a possibilistic answer set of P, then M^* is an answer set of P^*.*

Proof. The proof is straightforward by the possibilisitic answer set's definition.

When all the possibilistic clauses of a possibilistic program P have as certainty level the top of the lattice that was considered in P, the answer sets of P^* can be directly generalized to the possibilistic answer sets of P.

Proposition 3. *Let $P = \langle (Q, \leq), N \rangle$ be a possibilistic disjunctive logic program and TOP_Q be the top of the lattice $(Q, \leq)$. If $\forall r \in P$, $n(r) = TOP_Q$, and M' is an answer set of P^*, then $M := \{(l, TOP_Q) | l \in M'\}$ is a possibilistic answer set of P.*

Proof. (Sketch) We know that if M is a possibilistic answer set of P, then M^* is an answer set of P^* (by Proposition 2) and $N^{M^*} \vdash_{PL} M$. Now, since (GMP) $(\varphi\ TOP_Q), (\varphi \rightarrow \psi\ TOP_Q) \vdash_{PL} (\psi\ TOP_Q)$, then any formula inferred from P by GMP will have TOP_Q as necessity-value. Then, if M' is an answer set of P^*, then $\{(l, TOP_Q) | l \in M'\}$ will be a possibilistic answer set of P.

For the class of possibilistic normal logic programs, our definition of possibilistic answer set is closely related to the definition of possibilistic stable model presented in [15]. In fact, both semantics coincide.

Proposition 4. *Let P be a possibilistic normal logic program. M is a possibilistic answer set of P if and only if M is a possibilistic stable model of P.*

Proof. (Sketch) It is not difficult to see that when P is a possibilistic normal program, then the syntactic reduction of Definition 2 and the syntactic reduction of Definition 10 from [15] coincide. Then the problem is reduced to possibilistic definite programs. But, this case is straightforward, since essentially GMP is applied for inferring the possibilistic models of the program in both approaches.

In terms of computability, since there is an extended version of *resolution* for possibilistic logic ([6]) it is not difficult to see that the there exists an algorithm that computes the set of possibilistic answer sets of any possibilistic disjunctive logic program.

Proposition 5. *Given a possibilistic program P there exists an algorithm that computes the set of possibilistic answer sets of P.*

Proof. This results follows from the generalization of resolution for Possibilistic Logic [6] and the answer solvers that there are in the literature [1].

4 Inconsistent possibilistic logic programs

As we commented in the background section, there are several approaches for handling an inconsistent program in ASP. Most of them do not give a useful answer. However, frequently we have to confront with inconsistent knowledge bases and it is needed a concrete answer.

Let us consider again Example 1. By continuing with the described situation, now we will suppose that the patient fell in dead brain and now the patient is considered as a potential donor of organs. The problem is that if the donor had endocarditis due to streptococcus viridans, then the recipient could be infected by the same microorganism. Therefore, the organs from this donor could not be viable for transplanting.

A short representation of this medical situation could be the following program P_2 (again in this program we consider the relative likelihoods of Figure 1):

$$\begin{array}{ll} probable: & endocarditis \leftarrow \top. \\ probable: & \neg endocarditis \leftarrow \top. \\ confirmed: & non\_viable \leftarrow endocarditis \\ plausible: & viable \leftarrow not\ \neg endocarditis. \end{array}$$

The first two rules say that the doctor does not know if the donor has endocarditis. The third one says that if the donor has endocarditis then his organs are not viable for transplanting and the last one says that if the donor explicitly does not have endocarditis then it is plausible that his organs are viable for transplanting.

Notice that P_2 is an inconsistent possibilistic program. Nicolas *et al.*, in [15] suggested to consider an inconsistent degree for eliminating the formulae involved in the inconsistency. Formally, this process is called *strict α-cut*. However to apply a *strict α-cut* to an inconsistent program could eliminate important information in order to support/infer conclusions. In order to illustrate this problem, let suppose that the strict α-cut just deletes from P_2 the possibilistic clauses:

$$\begin{array}{l} probable: endocarditis \leftarrow \top. \\ probable: \neg endocarditis \leftarrow \top. \end{array}$$

It is worth mentioning that the strict α-cut was defined in terms of totally ordered sets. Hence we are applying a hypothetical cut to P_2 where we are supposing that $probable \leq plausible$ and $plausible \leq confirmed$ and the inconsistent degree of P_2 is $probable$. Then the consistent program P_2' that we get after applying the strict α-cut to P_2 is:

$$\begin{array}{ll} confirmed: & non\_viable \leftarrow endocarditis \\ plausible: & viable \leftarrow not\ \neg endocarditis. \end{array}$$

The only possibilistic answer set of P'_2 is: $\{(viable, plausible)\}$. Then this possibilistic answer set suggests that the organs of the donor could be viable for transplanting. But this is a *dangerous* conclusion, because we are omitting the premises that the donor could be infected by streptococcus viridans.

Now, since we are allowing that an answer set could have a pair of complementary literals, we can apply directly the definition of possibilistic answer set to P_2. Then the only possibilistic answer set of P_2 is: $\{(endocarditis, probable), (\neg endocarditis, probable), (non\_viable, probable)\}$. This possibilistic answer set suggests that the donor's organs probable are not viable for transplanting. Notice that this conclusion is a more cautious conclusion.

In general, we believe that to eliminate information for handling inconsistent knowledge bases could carry unexpected results. In addition, not to have a concrete answer from an inconsistent knowledge base could be an expected result when we have to make or support a decision.

5 Conclusions and future work

We have been working in the decision making process for deciding if a human organ is viable or not for being transplanted [21, 16, 20]. Our experience suggests that in our medical domain, we require a *qualitative* theory of default reasoning like ASP for modeling incomplete information and a *quantitative* theory like possibilistic logic for modeling uncertain events.

This paper describes a possibilistic disjunctive logic programming approach which considers ideas from ASP and possibilistic logic. This approach introduces the use of possibilistic disjunctive clauses which are able to capture *incomplete information* and *incomplete states of a knowledge base* at the same time. In fact, one of the main motivations of our approach is to define a description language and a reasoning process where the user could consider relative likelihoods for modeling different levels of uncertainty *e.g.,* Toulmin's "qualifiers"[22]: *possible*, *probable*, *plausible*, *supported* and *open*, where each likelihood is a possible world/class of beliefs. We know that this kind of representation of uncertainty could reach *bias conclusions*. However, we have to accept that this form of reasoning is commonly performed by ordinary people. In fact, these kind of bias are many times well-accepted since they could reflect the experience or commonsense of an expert in a field [12].

The approach of possibilistic logic programming is not new in the literature [4, 23, 15]. However, to the best of our knowledge all of the well-known approaches suggested until now do not include possibilistic disjunctive programs. Moreover, our approach is enough flexible for using lattices for expressing incomplete states of a knowledge base and is close related to possibilistic logic.

In general terms, we are proposing a possibilistic disjunctive logic programming framework able to deal with reasoning under uncertainty and incomplete information. This framework permits to use explicitly labels like *possible*, *probable*, *plausible*, *etc.*, for capturing the incomplete state of a belief in a disjunctive logic program when the numerical representations are not available or difficult to get.

In terms of computability, we observe that our approach is computable. By the moment, we do not have results *w.r.t.* complexity of our approach; however it is an issue for our future work. Also we have observed that the possibilistic normal logic programs can be expressed by standard logic programs. In fact, we have an experimental approach for mapping possibilistic normal programs into standard logic programs. However it seems that to express possibilistic disjunctive logic programs into standard logic programs is not straightforward.

Nowadays, we are interested in supporting decision making in the medical domain. Especially in the process of organ transplanting. In this issue, we have been defining an argumentation framework for building arguments that support a given decision [17]. We have seen that the use of arguments could help to infer consistent information from an inconsistent knowledge base. In [17], it is described a process for inferring consistent information from an inconsistent possibilistic knowledge base. This process consists mainly of three steps:

1. To infer the possibilistic stable models from the possibilistic knowledge base,
2. To build arguments, and
3. Selection of arguments.

In [17], the knowledge base only could be expressed by possibilistic normal programs. Now, as future work we will define an argumentation framework based on the possibilistic answer sets for inferring consistent information from an inconsistent knowledge base.

Acknowledgements

We are grateful to anonymous referees for their useful comments. J.C. Nieves wants to thank 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. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
2. S. Brass and J. Dix. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999.
3. F. Caballero, A. López-Navidad, M. Perea, C. Cabrer, L. Guirado, and R. Solá. Successful liver and kidney transplantation from cadaveric donor with left-sided bacterial endocarditis. *American Journal of Transplantation*, 5:781–787, 2005.
4. C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In *ECSQARU*, volume 2143 of *Lecture Notes in Computer Science*, pages 748–759. Springer, 2001.
5. S. DLV. Vienna University of Technology. http://www.dbai.tuwien.ac.at/proj/dlv/, 1996.
6. D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pages 439–513. Oxford University Press, Oxford, 1994.

7. J. Fox and S. Modgil. From arguments to decisions: extending the Toulmin view. In *Arguing on the Toulmin model: New essays on argument analysis and evaluation*. Argumentation Library series published by Kluwer Academic, Currently in press.
8. 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.
9. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
10. J. Y. Halpern. *Reasoning about uncertainty*. The MIT Press, 2005.
11. H. Jakobovits and D. Vermeir. Robust semantics for argumentation frameworks. *Journal of logic and computation*, 9(2):215–261, 1999.
12. D. Kahneman, P. Slovic, and A. Tversky. *Judgment under uncertainty:Heuristics and biases*. Cambridge Univertisy Press, 1982.
13. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.
14. E. Mendelson. *Introduction to Mathematical Logic*. Chapman and Hall/CRC, Fourth edition 1997.
15. P. Nicolas, L. Garcia, I. Stéphan, and C. Lafèvre. Possibilistic Uncertainty Handling for Answer Set Programming. *Annal of Mathematics and Artificial Intelligence*, 47(1-2):139–181, June 2006.
16. J. C. Nieves, M. Osorio, and U. Cortés. Supporting decision making in organ transplating using argumentation theory. In *LANMR 2006: 2nd Latin American Non-Monotonic Reasoning Workshop*, pages 9–14, 2006.
17. J. C. Nieves, M. Osorio, and U. Cortés. Modality-based argumentation using possibilistic stable models. In R. Kibble, F. Grasso, and C. Reed, editors, *7th Workshop on Computational Models of Natural Argument (CMNA VII)*, pages 35–41, Hyderabad, India, January 2007.
18. J. C. Nieves, M. Osorio, and U. Cortés. Semantics for possibilistic disjunctive programs. In G. B. Chitta Baral and J. Schlipf, editors, *Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-07)*, number 4483 in LNAI, pages 315–320. Springer-Verlag, 2007.
19. F. J. Pelletier and R. Elio. *Scope of Logic, Methodology and Philosophy of Science*, volume 1 of *Synthese Library*, chapter Logic and Computation, pages 137–156. Dordrecht: Kluwer Academic Press, 2002.
20. P. Tolchinsky, U. Cortés, S. Modgil, F. Caballero, and A. López-Navidad. Increasing Human-Organ Transplant Availability: Argumentation-Based Agent Deliberation. *IEEE Intelligent Systems: Special Issue on Intelligent Agents in Healthcare*, 21(5):30–37, November/December 2006.
21. P. Tolchinsky, U. Cortés, J. C. Nieves, A. López-Navidad, and F. Caballero. Using arguing agents to increase the human organ pool for transplantation. In *Proc. of the Third Workshop on Agents Applied in Health Care (IJCAI 2005)*, 2005.
22. S. E. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.
23. G. Wagner. Negation in fuzzy and possibilistic logic programs. In T. Martin and F. Arcelli, editors, *Logic programming and Soft Computing*. Research Studies Press, 1998.