Semantics for Possibilistic Disjunctive Programs ∗

JUAN CARLOS NIEVES, ULISES CORTÉS

Universitat Politècnica de Catalunya
Software Department (LSI)
c/Jordi Girona 1-3, E08034, Barcelona, Spain
(*e-mail:* {jcnieves,ia}@lsi.upc.edu)

MAURICIO OSORIO

Universidad de las Américas - Puebla
CENTIA
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
(*e-mail:* josorio@mail.udlap.mx)

3 November 2008

Abstract

We define a possibilistic disjunctive logic programming approach for modeling uncertain, incomplete and inconsistent information. 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. This approach is computable and moreover allows encoding uncertain information by using either numerical values or relative likelihoods. In order to define the semantics of the possibilistic disjunctive programs, three approaches are defined:

1. The first is strictly close to the proof theory of possibilistic logic and answer set models;
2. The second is based on partial evaluation, a fix-point operator and answer set models; and
3. The last is also based on the proof theory of possibilistic logic and pstable semantics.

In order to manage inconsistent possibilistic logic programs, a preference criterion between inconsistent possibilistic models is defined; in addition, the approach of cuts for restoring consistency of an inconsistent possibilistic knowledge base is adopted. The approach is illustrated by a medical scenario.

KEYWORDS: Answer Set Programming, Uncertain Information, Possibilistic Reasoning.

1 Introduction

Uncertain and incomplete information is an unavoidable feature of daily decision-making. In order to deal with uncertain and incomplete information intelligently, we need to be able to represent it and reasoning about it.

∗ This is a revised and improved version of the papers:*Semantics for Possibilistic Disjunctive Programs* (Poster) appeared in C. Baral, G. Brewka and J. Schipf (Eds), Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-07), LNAI 4483. *Semantics for Possibilistic Disjunctive Logic programs* which appears in S. Constantini and W. Watson (Eds), Answer Set Programming: Advantage in Theory and Implementation (ICLP-07 Workshop). *Pstable Semantics for Possibilistic Logic Programs* which appears in A. F. Gelbukh and A.F. Kuri (Eds), 6th Mexican International Conference on Artificial Intelligence (MICAI-07), LNAI 4827.

Answer Set Programming (ASP) is one of the most successful logic programming approaches in Non-monotonic Reasoning and Artificial Intelligence applications (Baral 2003; Gelfond 2008). In (Nicolas et al. 2006), a possibilistic framework for reasoning under uncertainty was proposed. This framework is a combination between ASP and possibilistic logic (Dubois et al. 1994).

Possibilistic Logic is based on possibilistic theory where at the mathematical level, degrees of possibility and necessity are closely related to fuzzy sets (Dubois et al. 1994). Thanks to the natural properties of possibilistic logic and ASP, Nicolas *et al.*'s approach allows 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*.

To say that possibilistic disjunctive logic programs are required for representing incomplete information could be questioned; however, Gelfond and Lifschiz observed in (Gelfond and Lifschitz 1991) that:

> *An important limitation of traditional logic programming as a knowledge representation tool, in comparison with classical logic, is that logic programming does not allow us to deal directly with incomplete information.*

In order to overcome this limitation they suggested to include strong negation and disjunctive clauses in ASP for dealing incomplete information. In possibilistic answer set programming, one can also expect to use strong negation and possibilistic disjunctive clauses for dealing with incomplete information. An important feature of the possibilistic disjunctive clauses is that they are able to capture *incomplete information* and *incomplete states of a knowledge base* at the same time.

Let us consider a medical domain, in order to illustrate a scenario where uncertain and incomplete information is always presented. The particular medical domain that we consider is *human organ transplanting*. In human organ transplanting, one of the most sophisticated and complex processes is the organ donor selection. There are several factors that make this process sophisticated and complex. For instance:

- the transplant acceptance criteria vary ostensibly between transplant teams from the same geographical area and substantially between more distant transplantation teams (López-Navidad et al. 1997). This means that the acceptance criteria applied in one hospital could be invalid or at least arguable in another hospital.
- there are lots of factors that make unpredictable if an organ donor's disease will be diagnosed in the organ recipient. For instance, if an organ donor D has hepatitis, then an organ recipient R could be infected by an organ of D. According to (López-Navidad and Caballero 2003), there are cases where the infection can occur; however the recipient can *spontaneously* clear the infection *e.g.*, hepatitis. This means that an organ donor's infection can be *present* or *not-present* in the organ recipient. It is worth to comment that there are infections which can be prevented by applying a post-transplant treatment to the organ recipient.
- the clinical state of an organ recipient can be affected by several factors *e.g.*, malfunctions of the graft. This means that the clinical state of an organ recipient can

be *stable* or *unstable* after the graft because the graft can have *good graft functions*, *delayed graft functions* and *terminal insufficient functions*[1].

It is worth mentioning that the transplant acceptance criteria may be dependant to the kind of organ (kidney, heart, liver, *etc.*) that will be considered for transplanting and the clinical situation of the potential organ recipients.

Let us consider the particular case of kidney transplantation with organ donors that have a kind of infection *e.g.*, endocarditis, hepatitis. As already stated, the clinical situation of the potential organ recipients is relevant in the organ transplanting process. We will denote the clinical situation of an organ recipient by the predicate $cs(t, T)$, such that t can be *stable*, *unstable*, 0-*urgency* and T denotes a moment in the time. Another important factor that we will consider is the state of the organ's functions. We will denote by the predicate $o(t, T)$ the state of the organ's functions, such that t can be *terminal-insufficient functions*, *good-graft functions*, *delayed-graft functions*, *normal-graft functions* and T denotes a moment in the time. Also, we will consider in our scenario the state of an infection in both the organ recipient and the organ donor, this condition will be denoted by the predicates $r\_inf(present, T)$ and $d\_inf(present, T)$ respectively such that T denotes a moment in the time. The last predicate that we introduce, will be $action(t, T)$ such that t can be *transplant*, *wait*, *post-transplant treatment* and T denotes a moment in the time. This predicate denotes the possible actions of a doctor. In Figure 1[2], a finite state automata is presented — for a formal presentation of automata theory see, (Hopcroft et al. 2007). In this automata each node represents a possible situation where an organ recipient can be found and the arrows represent the possible doctor's actions. Observe that we are assuming that in the initial state the organ recipient is clinically stable and he does not have an infection; however, he has a kidney whose functions are terminal insufficient. From the initial state, the doctor's actions can be either to make a kidney transplantation or just wait[3].

According to Figure 1, an organ recipient can be found in different situations after a graft. In fact, the organ recipient may require another graft and the state of the infection can be unpredictable. Let us introduce extended disjunctive clauses which describe some situations presented in Figure 1

$$r\_inf(present, T2) \vee \neg r\_inf(present, T2) \leftarrow action(transplant, T), \\ d\_inf(present, T), T2 = T + 1.$$

$$o(good\_graft\_funct, T2) \vee o(delayed\_graft\_funct, T2) \vee \\ o(terminal\_insufficient\_funct, T2) \leftarrow action(transplant, T), T2 = T + 1.$$

As syntactic clarification, we want to point out that $\neg$ is regarded as *strong negation* which is not exactly the negation in classical logic. In fact, any atom negated by strong

[1] Usually, when a doctor says that an organ has *terminal insufficient functions*, it means that there exists no clinical treatment for improving the organ's functions.

[2] This finite state automata was built under the supervision of Francisco Caballero M. D. Ph. D. from the Hospital de la Santa Creu I Sant Pau, Barcelona, Spain.

[3] In the automata of Figure 1, we are not considering that there is a waiting list where the organ recipient waits for an organ. This waiting list has different politics for assigning an organ to an organ recipient.

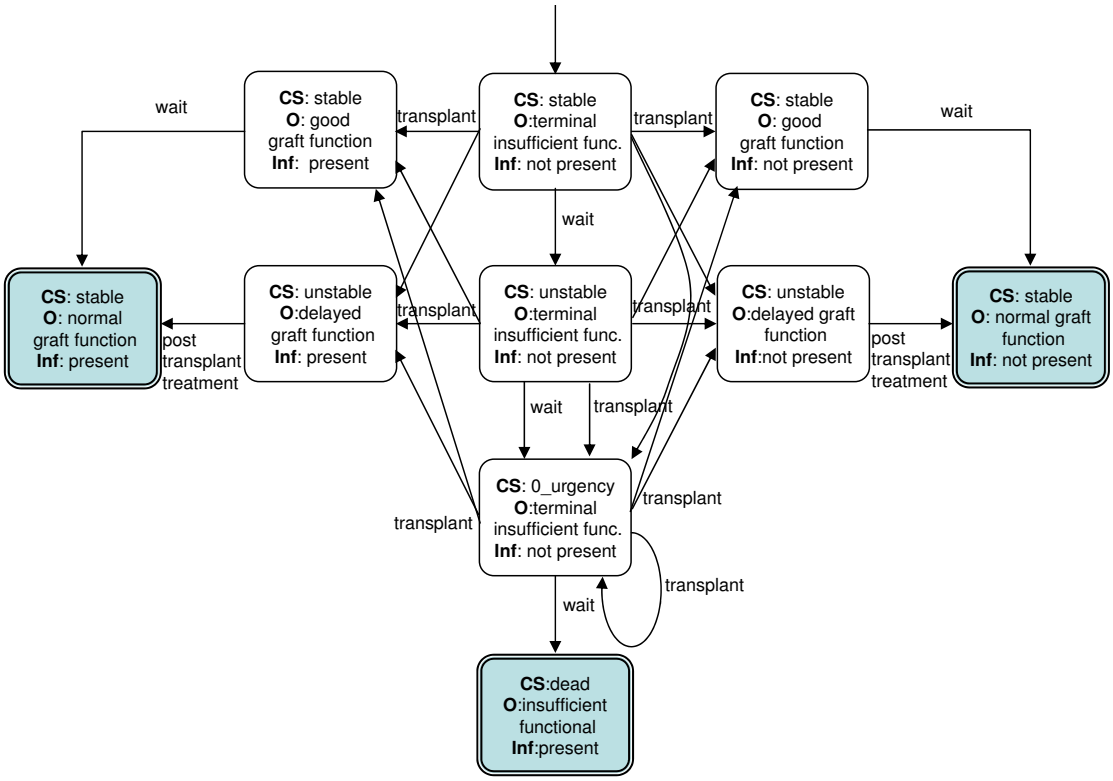

Fig. 1. An automata of states and actions for considering infections in kidney organ transplanting.

negation will be replaced by a new atom as it is done in ASP. This means that $a \vee \neg a$ cannot be regarded as a tautology.

Following with our medical scenario, we can see that the intended meaning of the first clause is that if the organ donor has an infection, then the infection can be *present* or *not-present* in the organ recipient, after the graft, and the intended meaning of the second one is that the graft's functions can be: *good*, *delayed* and *terminal*, after the graft. Observe that these clauses are not capturing the uncertainty that is involved in each statement. For instance, *w.r.t.* the first clause, one can wish to represent an uncertainty degree in order to capture the uncertainty that is involved in this statement — remember that the organ recipient can be infected by the infection of the donor's organ; however, the infection can be *spontaneously* cleared by the organ recipient as it is the case of hepatitis (López-Navidad and Caballero 2003).

In psychology literature, one can find significant observations in order to model uncertain information. For instance, Tversky and Kahneman have observed in (Tversky and Kahneman 1982) that we commonly use statements such as "*I think that . . .*", "*chances are . . .*", "*it is probable that . . .*", "*it is plausible that . . .*", *etc.*, for supporting our decisions. Observe that these statements have as common denominator adjectives which quantify the information. These adjectives are of the form: *probable*, *plausible*, *etc*. Based on this ob-

servation, we propose to use adjectives/labels of the same kind in order to quantify the uncertain information of a knowledge base. The only formal requirement is that this set of adjectives/labels must be a *complete lattice*. For instance, for the case of our medical scenario a transplant coordinator[4] can suggest a set of labels in order to quantify a medical knowledge base and of course to define an order between those labels. Based on those labels we can have possibilistic clauses as:

probable: $r\_inf(present, T2) \vee \neg r\_inf(present, T2) \leftarrow action(transplant, T),$
$d\_inf(present, T), T2 = T + 1.$

Informally speaking, the reading of this clause is: *if the organ donor has an infection, then it is* probable *that the organ recipient can be infected or not after a graft.*

In this paper, we will 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. In order to capture the semantics of possibilistic disjunctive programs, we will present three possible approaches. Two of these approaches will be based on answer set models and the other one will be based on pstable models. As part of our study, we will present some approaches for managing the inconsistency of a possibilistic knowledge. Also, we will take care of the relationship that there is between the approach presented by Nicolas *et al.*, in (Nicolas et al. 2006) and our approach.

The rest of the paper is divided as follows:

In §3, the syntax of our possibilistic framework is presented. In §2 we give all the background and necessary notation. In §4, three approaches for defining the semantics of possibilistic disjunctive programs are presented. In §5, some criteria for managing inconsistent possibilistic logic programs are defined. In §6, we present a small discussion *w.r.t.* related approaches to our work. Finally, in the last section, we present our conclusions and future work.

2 Background

In this section we introduce all the necessary terminology and relevant definitions in order to have a self-contained document. We assume that the reader has familiarity with basic concepts of *classic logic*, *logic programming* and *lattices*.

2.1 Logic programs: Syntaxis

The language of a propositional logic has an alphabet consisting of

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

where $\vee, \wedge, \leftarrow$ are binary-place connectives, $\neg$, not are unary-place connective and $\bot$ is zero-ary connective. The proposition symbols, $\bot$ and $\top$ stand for the indecomposable

[4] A transplant coordinator is an expert in all the process of transplanting (López-Navidad et al. 1997).

propositions, which we call *atoms*, or *atomic propositions*. Atoms negated by $\neg$ will be called *extended atoms*.

Remark 1
We will use the concept of atom without paying attention if it is an extended atom or not.

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 $m \geq 0$, $n \geq 0$, $m + n > 0$, each a_i is an atom[5]. When $n = 0$ and $m > 0$ the clause is an abbreviation of $a_1 \vee \ldots \vee a_m \leftarrow \top$ such that $\top$ is the proposition symbol that always evaluates to true; clauses of these form some times are written just as $a_1 \vee \ldots \vee a_m$. When $m = 0$ the clause is an abbreviation of $\bot \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n$ 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.

Sometimes we denote an extended disjunctive clause *C* by $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$, where $\mathcal{A}$ contains all the head literals, $\mathcal{B}^+$ contains all the positive body literals and $\mathcal{B}^-$ contains all the negative body literals. 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.

We will manage the strong negation ($\neg$), in our logic programs, as it is done in ASP (Baral 2003). Basically, it is replaced each extended 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:

$$\begin{array}{l} a \leftarrow q. \\ \neg q \leftarrow r. \\ q \leftarrow \top. \\ r \leftarrow \top. \end{array}$$

Then replacing each extended atom by a new atom symbol, we will have:

$$\begin{array}{l} a \leftarrow q. \\ q' \leftarrow r. \\ q \leftarrow \top. \\ r \leftarrow \top. \end{array}$$

[5] Notice that these atoms can be *extended atoms*.

In order to disallow models with complementary atoms *i.e.*, q and $\neg q$, usually it is added a constraint of the form $\bot \leftarrow q, q'$ to the logic program. We will omit this constraint in order to allow models with complementary atoms. However, the user could add this constraint without losing generality.

When we treat a logic program as a theory, each negative literal $not\ a$ is replaced by $\sim a$ such that $\sim$ is regarded as the negation in classic logic. Formulæ are constructed as usual in classic logic by the connectives: $\vee, \wedge, \leftarrow, \sim, \bot, \top$. A theory T is a finite set of formulæ. By $\mathcal{L}_T$, we denote the signature of *T*, namely the set of atoms that occur in *T*.

Given a set of proposition symbols S and a theory Γ in a logic X. If $\Gamma \vdash_X S$ if and only if $\forall s \in S\ \Gamma \vdash_X s$.

2.2 Interpretations and models

In this section we define some relevant concepts *w.r.t.* semantics. The first basic concept that we introduce will be *interpretation*.

Definition 1
Let T be a theory, an interpretation I is a mapping from $\mathcal{L}_T$ to $\{0, 1\}$ meeting the conditions:

1. $I(a \wedge b) = min\{I(a), I(b)\}$,
2. $I(a \vee b) = max\{I(a), I(b)\}$,
3. $I(a \leftarrow b) = 0$ if and only if $I(b) = 1$ and $I(a) = 0$,
4. $I(\sim a) = 1 - I(a)$,
5. $I(\bot) = 0$.
6. $I(\top) = 1$.

It is standard to provide interpretations only in terms of a mapping from $\mathcal{L}_T$ to $\{0, 1\}$. Moreover, it is easy to prove that this mapping is unique by virtue of the definition by recursion (van Dalen 1994). Also, it is standard to use sets of atoms to represent interpretations. The set corresponds exactly to those atoms that evaluate to 1.

An interpretation I is called a (2-valued) model of the logic program P if and only if for each clause $c \in P$, $I(c) = 1$. A theory is consistent if it admits a model, otherwise it is called inconsistent. Given a theory T and a formula α, we say that α is a logical consequence of T, denoted by $T \models \alpha$, if every model I of T holds that $I(\alpha) = 1$. It is a well known result that $T \models \alpha$ if and only if $T \cup \{\sim \alpha\}$ is inconsistent.

We say that a model I of a theory T is a minimal model if there does not exist a model I' of T different from I such that $I' \subset I$. Maximal models are defined in the analogous form.

2.3 Logic programming semantics

In this section, we will define two logic programming semantics: *answer set semantics* and *pstable semantics*. Both semantics represent a two-valued semantics approach.

2.3.1 Answer set semantics

By using ASP, it is possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. It represents one of the most successful approaches of non-monotonic reasoning of the last two decades (Baral 2003). The number of applications of this approach has been increased thanks to the efficient implementations of the answer set solvers that exist.

The answer set semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* (Gelfond and Lifschitz 1988) and it is usually studied in the context of syntax dependent transformations on programs. The following definition of an answer set for extended disjunctive logic programs generalizes the definition presented in (Gelfond and Lifschitz 1988) and it was presented in (Gelfond and Lifschitz 1991): Let *P* be any extended disjunctive logic 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 formulæ 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 called an answer set of *P* if and only if *S* is a minimal model of P^S. In order to illustrate this definition let us consider the following example:

Example 1
Let us consider the set of atoms $S := \{b\}$ and the following normal logic program P:

$$b \leftarrow not\ a. \qquad b \leftarrow \top.$$
$$c \leftarrow not\ b. \qquad c \leftarrow a.$$

We can see that P^S is:

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

Notice that this program has three models: $\{b\}$, $\{b, c\}$ and $\{a, b, c\}$. Since the minimal model amongst these models is $\{b\}$, we can say that S is an answer set of P.

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

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

then, the only answer set of this program is : $\{a, \neg a, b\}$. In Section 5, we will discuss how we will manage the inconsistency in our logic programs.

It is worth mentioning that in literature there are several forms for handling an inconsistency program. For instance, by applying the original definition (Gelfond and Lifschitz 1991) the only answer set of P is: $\{a, \neg a, b, \neg b\}$. On the other hand, the DLV system (DLV 1996) returns no models if the program is inconsistent.

2.3.2 Pstable semantics

Pstable semantics is a recently introduced logic programming semantics which is inspired in paraconsistent logics. This semantics is defined by a fixed point operator in terms of clas-

sical logic. The expressiveness of pstable semantics is at least as the answer set semantics for disjunctive logic program.

First, to define pstable semantics, we will introduce some basic concepts. By $\vdash_C$, we denote logic consequence in classic logic. Given a normal program P, if $M \subseteq \mathcal{L}_P$, we write $P \Vdash M$ when: $P \vdash_C M$ and M is a classical 2-valued model of P.

Pstable semantics is defined in terms of a single reduction which is defined as follows:

Definition 2

(Osorio et al. 2006) Let P be a normal program and M a set of atoms. We define

$$RED(P, M) := \{a \leftarrow \mathcal{B}^+,\ not\ (\mathcal{B}^- \cap M) | a \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^- \in P\}$$

Let us consider the set of atoms $M_1 := \{a, b\}$ and the following normal program P_1:

$$\begin{array}{l} a \leftarrow\ not\ b,\ not\ c. \\ a \leftarrow b. \\ b \leftarrow a. \end{array}$$

We can see that $RED(P_1, M)$ is:

$$\begin{array}{l} a \leftarrow\ not\ b. \\ a \leftarrow b. \\ b \leftarrow a. \end{array}$$

By considering the reduction RED, it is defined the pstable semantics for normal programs as follows:

Definition 3

(Osorio et al. 2006) Let P be a normal program and M a set of atoms. We say that M is a pstable model of P if $RED(P, M) \Vdash M$. We use Pstable to denote the semantics operator of pstable models.

Let us consider again M_1 and P_1 in order to illustrate the definition. We want to verify whether M_1 is a pstable model of P_1. First, we can see that M_1 is a model of P_1, *i.e.*, $\forall\ c \in P_1$, M_1 evaluates c to true. Now, we have to prove each atom of M_1 from $RED(P_1, M_1)$ by using classical inference, *i.e.*, $RED(P_1, M_1) \vdash_C M_1$. Let us consider the proof of the atom a, which belongs to M_1, from $RED(P_1, M_1)$.

1. $(a \vee b) \rightarrow ((b \rightarrow a) \rightarrow a)$	Tautology
2. $\sim b \rightarrow a$	Premise from $RED(P_1, M_1)$
3. $a \vee b$	From 2 by logical equivalency
4. $(b \rightarrow a) \rightarrow a$	From 1 and 3 by Modus Ponens
5. $b \rightarrow a$	Premise from $RED(P_1, M_1)$
6. a	From 4 and 5 by Modus Ponens

Remember that the formula $\sim b \rightarrow a$ corresponds to the normal clause $a \leftarrow\ not\ b$ which belongs to the program $RED(P_1, M_1)$. The proof for the atom b, which also belongs to M_1, is similar to the proof of the atom a. Then we can conclude that $RED(P_1, M_1) \Vdash M_1$. Hence, M_1 is a *pstable model* of P_1.

2.4 Possibilistic Logic

In this section, we will define some basic concepts of possibilistic logic for the case of necessity-valued formulæ.

Possibilistic logic is a weighted logic introduced and developed in the mid-1980s, in the setting of artificial intelligence, with the view to develop a simple an rigorous approach to automated reasoning from uncertain or prioritized incomplete information. Possibilistic logic is especially adapted to automated reasoning when the available information is pervaded with vagueness. In fact, possibilistic logic is a natural extension of classical logic where the notion of total order/partial order is embedded in the logic.

For the case of necessity-valued formulæ, 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 (Dubois et al. 1994). 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 formulæ.

The following properties hold *w.r.t.* necessity-valued formulæ:

$$N(\varphi \wedge \psi) = min(\{N(\varphi), N(\psi)\}) \tag{1}$$

$$N(\varphi \vee \psi) \geq max(\{N(\varphi), N(\psi)\}) \tag{2}$$

$$\text{if } \varphi \vdash \psi \text{ then } N(\psi) \geq N(\varphi) \tag{3}$$

Dubois *et al.*, in (Dubois et al. 1994) 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.*, in (Dubois et al. 1994), 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 formulæ 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 (Dubois et al. 1994) for details).

One of the main basic principle of possibilistic logic is that:

Remark 2
The strength of a conclusion is the strength of the weakest argument used in its proof.

According to Dubois and Prade (Dubois and Prade 2004), the contribution of possibilistic logic setting is to relate this principle (measuring the validity of an inference chain by its weakest link) to fuzzy set-based necessity measures in the framework of Zadeh's possibilistic theory, since the following pattern then holds:

$$N(\sim p \vee q) \geq \alpha \text{ and } N(p) \geq \beta \text{ imply } N(q) \geq min(\alpha, \beta)$$

This interpretive setting provide a semantics justification to the claim that the weight attached to a conclusion should be the weakest among the weights attached to the formulæ involved in the derivation.

2.5 Lattices and order

In mathematics, especially order theory formalizes the intuitive concept of an ordering or an arrangement of the elements of a set. Indeed, the study of order has let to a great unification of results in algebra and logic. More recently, it has infused into theoretical computer science, particularly into programming language semantics (Davey and Priestly 2002).

In this section, we will present some fundamental definitions of lattice theory in order to make this document self contained (see (Davey and Priestly 2002) for more details).

Definition 4
Let $\mathcal{Q}$ be a set. An order (or partial order) on $\mathcal{Q}$ is a binary relation $\leq$ on $\mathcal{Q}$ such that, for all $x, y, z \in \mathcal{Q}$,

(i) $x \leq x$
(ii) $x \leq y$ and $y \leq x$ imply $x = y$
(iii) $x \leq y$ and $y \leq z$ imply $x \leq z$

These conditions are referred to, respectively, as reflexivity, antisymmetry and transitivity.

A set $\mathcal{Q}$ equipped with an order relation $\leq$ is said to be an ordered set (or partial ordered set). It will be denoted by ($\mathcal{Q}$,$\leq$).

Definition 5
Let ($\mathcal{Q}$,$\leq$) be an ordered set and let $S \subseteq \mathcal{Q}$. An element $x \in \mathcal{Q}$ is an upper bound of S if $s \leq x$ for all $s \in S$. A lower bound is defined dually. The set of all upper bounds of S is denoted by S^u (read as 'S upper') and the set of all lower bounds by S^l (read as 'S lower').

If S^u has a least element x, then x is called the least upper bound ($\mathcal{LUB}$) of S. Equivalently, x is the least upper bound of S if

(i) x is an upper bound of S, and
(ii) $x \leq y$ for all upper bound y of S.

The least upper bound of S exists if and only if there exists $x \in \mathcal{Q}$ such that

$$(\forall y \in \mathcal{Q})[((\forall s \in S)s \leq y) \Longleftrightarrow x \leq y],$$

and this characterizes the $\mathcal{LUB}$ of S. Dually, if S^l has a greatest element, x, then x is called the greatest lower bound ($\mathcal{GLB}$) of S. Since least element and greatest element are unique, $\mathcal{LUB}$ and $\mathcal{GLB}$ are unique when they exist.

The least upper bound of S is called the supremum of S and it is denoted by $sup\ S$; the greatest lower bound of S is also called the infimum of S and it is denoted by $inf\ S$.

Definition 6
Let ($\mathcal{Q}$,$\leq$) be a non-empty ordered set.

(i) If $sup\{x, y\}$ and $inf\{x, y\}$ exist for all $x, y \in \mathcal{Q}$, then $\mathcal{Q}$ is called lattice.
(ii) If $sup\ S$ and $inf\ S$ exist for all $S \subseteq \mathcal{Q}$, then $\mathcal{Q}$ is called a complete lattice.

Example 2
Let us consider the set of labels $\mathcal{Q} := \{Certain$, Confirmed, Probable, Plausible, Supported, $Open\}$[6] and let $\preceq$ be a partial order such that the following set of relations holds: $\{Open \preceq Supported$, $Supported \preceq Plausible$, $Supported \preceq Probable$, $Probable \preceq Confirmed$, $Plausible \preceq Confirmed$, $Confirmed \preceq Certain\}$. A graphic representation of S according to $\preceq$ is showed in Figure 2. It is not difficult to see that $(\mathcal{Q}, \preceq)$ is a lattice and even more it is a complete lattice.

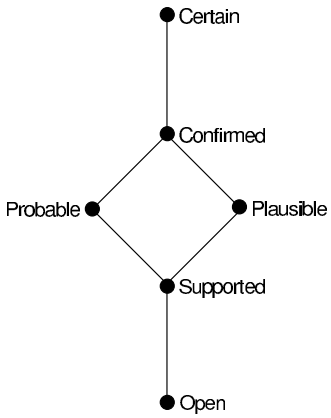

Fig. 2. A graphic representation of a lattice where the following relations holds: $\{Open \preceq Supported$, $Supported \preceq Plausible\ Supported \preceq Probable$, $Probable \preceq Confirmed$, $Plausible \preceq Confirmed$, $Confirmed \preceq Certain\}$.

[6] This set of labels was taken from (Fox and Modgil 2006). In this paper, the authors argue that we can construct a set of labels (they call those: *modalities*) in a way that this set provides a simple scale for ordering the claims of our beliefs. We will use this kind of labels for quantifying the uncertainty degree of a statement.

3 Syntax

In this section, we will define the general syntax for possibilistic disjunctive logic programs. This syntax will be based on the standard syntax of extended disjunctive logic programs (see Section 2.1). First of all, we start defining some relevant concepts[7].

In all the document, we will only consider finite lattices — remember that any finite lattice is complete. 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 atom* is a pair $p = (a, q) \in \mathcal{A} \times \mathcal{Q}$, where $\mathcal{A}$ is a finite set of atoms and $(\mathcal{Q}, \leq)$ is a lattice. We apply the projection $*$ to any possibilistic atom p as follows: $p^* = a$. Given a set of possibilistic atoms S, we define the generalization of $*$ over S as follows: $S^* = \{p^* | p \in S\}$.

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

$$\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$$

where $\alpha \in \mathcal{Q}$ and $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$ is an extended disjunctive clause as defined in Section 2.1. 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 C is of the form:

$$\mathcal{TOP}_{\mathcal{Q}} : \bot \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$$

where $\mathcal{TOP}_{\mathcal{Q}}$ is the top of the lattice $(\mathcal{Q}, \leq)$ and $\bot \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$ is a constrain as defined in Section 2.1. As in possibilistic clauses, the projection $*$ for a possibilistic constraint is : $C^* = \bot \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$. Observe that any possibilistic constraint must have the top of the lattice $(\mathcal{Q}, \leq)$, this restriction is motivated on the fact that like a constraint in standard ASP, the purpose of a possibilistic constraint is to eliminate possibilistic models. Hence, it is assumed that there does not exists doubt about the veracity of the information captured by a possibilistic constraint.

A possibilistic disjunctive logic program P is a tuple of the form $\langle (\mathcal{Q}, \leq), N \rangle$, 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 possibilistic positive logic program and so on. A given set of possibilistic disjunctive clauses $\{\gamma, \ldots, \gamma\}$ is also represented as $\{\gamma; \ldots; \gamma\}$ to avoid ambiguities with the use of the comma in the body of the clauses.

Given a possibilistic disjunctive logic program $P = \langle (\mathcal{Q}, \leq), N \rangle$, we define the α-*cut* and the *strict* α-*cut* of P, denoted respectively by P_α and $P_{\overline{\alpha}}$, by

$P_\alpha = \langle (\mathcal{Q}, \leq), N_\alpha \rangle$ such that $N_\alpha = \{c | c \in N \text{ and } n(c) \geq \alpha\}$
$P_{\overline{\alpha}} = \langle (\mathcal{Q}, \leq), N_{\overline{\alpha}} \rangle$ such that $N_{\overline{\alpha}} = \{c | c \in N \text{ and } n(c) > \alpha\}$

[7] Some concepts presented in this section extend some terms presented in (Nicolas et al. 2006).

Example 3
In order to illustrate a possibilistic program, let us go back to our scenario described in Section 1. We will consider the lattice of Example 2; hence, let $(\mathcal{Q}, \preceq)$ be the lattice of Figure 2 such that the relation $A \preceq B$ means that A is less possible that B. The possibilistic program $P := \langle (\mathcal{Q}, \preceq), N \rangle$ will be the following set of possibilistic clauses:
In the introduction, we presented the following possibilistic clause:

probable: $r\_inf(present, T2) \vee \neg r\_inf(present, T2) \leftarrow action(transplant, T),$
$d\_inf(present, T), T2 = T + 1.$

Remember that the intended meaning of the first clause is that if the organ donor has an infection, then it is probable that the organ recipient can be infected or not after a graft.
The intended meaning of the following clause is that it is confirmed that the organ's functions can be: good, delayed and terminal after a graft.

confirmed: $o(good\_graft\_funct, T2) \vee o(delayed\_graft\_funct, T2) \vee$
$o(terminal\_insufficient\_funct, T2) \leftarrow action(transplant, T), T2 = T + 1.$

The intended meaning of the following clause is that it is confirmed that if the organ's functions are terminal insufficient then it is necessary a transplanting.

confirmed: $action(transplant, T) \leftarrow o(terminal\_insufficient\_funct, T).$

The intended meaning of the following clause is that it is plausible that the clinical situation of the organ recipient can be stable if the graft's functions are good.

plausible: $cs(stable, T) \leftarrow o(good\_graft\_funct, T).$

The intended meaning of the following clause is that it is plausible that the clinical situation of the organ recipient can be unstable if the graft's functions are delayed.

plausible: $cs(unstable, T) \leftarrow o(delayed\_graft\_funct, T).$

The intended meaning of the following clause is that it is plausible that the clinical situation of the organ recipient can be of 0-urgency if the graft's functions are terminal insufficient after the graft.

plausible: $cs(\text{0-urgency}, T2) \leftarrow o(terminal\_insufficient\_funct, T2),$
$action(transplant, T), T2 = T + 1.$

The intended meaning of the following possibilistic constraint is that it is certain that the doctor cannot do two actions at the same time.

certain: $\bot \leftarrow action(transplant, T), action(wait, T).$

The intended meaning of the following possibilistic constraint is that it is certain that a transplanting cannot be done if the organ recipient is dead.

certain: $\bot \leftarrow action(transplant, T), cs(dead, T)$.

The initial state of the automata of Figure 1 is captured by the following possibilistic clauses:

certain: $d\_inf(present, 0) \leftarrow \top$.
certain: $\neg r\_inf(present, 0) \leftarrow \top$.
certain: $o(terminal\_insufficient\_funct, 0) \leftarrow \top$.
certain: $cs(stable, 0) \leftarrow \top$.

4 Semantics

In §3, we defined the syntax for any possibilistic disjunctive program. Now, in this section, we will study the semantics for these programs. Essentially, we have explored three approaches for capturing the semantics of possibilistic disjunctive programs. The first two are based on answer set models and the last one is based on pstable models. We start by presenting the approaches which are based on answer set models.

We will consider sets of atoms as interpretations; hence, before to define the possibilistic logic programming semantics, we will introduce two basic operations between sets of possibilistic atoms and a relation of order between them.

Definition 7
Let $\mathcal{A}$ be a finite set of atoms and ($\mathcal{Q}$,$\leq$) be a lattice. Consider $\mathcal{PS} = 2^{\mathcal{A}\times\mathcal{Q}}$ the finite set of all the possibilistic atoms sets induced by $\mathcal{A}$ and $\mathcal{Q}$. $\forall A, B \in \mathcal{PS}$, we define.

$$\begin{array}{ll} A \sqcap B & = \{(x, \mathcal{GLB}(\{q_1, q_2\}) | (x, q_1) \in A \wedge (x, q_2) \in B\} \\ A \sqcup B & = \{(x, q) | (x, q) \in A \; and \; x \notin B^*\} \cup \\ & \{(x, q) | x \notin A^* \; and \; (x, q) \in B\} \cup \\ & \{(x, \mathcal{LUB}(\{q_1, q_2\}) | (x, q_1) \in A \text{ and } (x, q_2) \in B\}. \\ A \sqsubseteq B & \iff A^* \subseteq B^*, \text{ and } \forall x, q_1, q_2, \\ & (x, q_1) \in A \wedge (x, q_2) \in B \text{ then } q_1 \leq q_2. \end{array}$$

This definition is almost the same as Definition 7 presented in (Nicolas et al. 2006). The only difference is that in Nicolas *et al.*'s definition, the operators *min* and *max* are used instead of the operators $\mathcal{GLB}$ and $\mathcal{LUB}$ for defining the operations $\sqcap$ and $\sqcup$[8]. The following proposition is a straightforward consequence of Proposition 6 of (Nicolas et al. 2006).

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

[8] The operators $\sqcap$ and $\sqcup$ were defined as follows in (Nicolas et al. 2006):
$A \sqcap B = \{(x, min(\{q_1, q_2\}) | (x, q_1) \in A \wedge (x, q_2) \in B\}$ and
$A \sqcup B = \{(x, q) | (x, q) \in A \; and \; x \notin B^*\} \cup \{(x, q) | x \notin A^* \; and \; (x, q) \in B\} \cup \{(x, max(\{q_1, q_2\}) | (x, q_1) \in A \text{ and } (x, q_2) \in B\}$.

4.1 A possibilistic semantics based on answer set models

We will define a possibilistic semantics which is strictly close to the proof theory of possibilistic logic and answer set models. Like answer set semantics' definition, our approach has as its base a syntactic reduction. In fact, this reduction is inspired in the Gelfond-Lifschitz reduction.

Definition 8 (*Reduction* P^M)
Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic disjunctive logic program, M be a set of atoms. P reduced by M is the positive possibilistic disjunctive logic program:

$$P^M := \quad \{(n(r) : \mathcal{A} \cap M \leftarrow \mathcal{B}^+) | r \in N, \mathcal{A} \cap M \neq \emptyset, \mathcal{B}^- \cap M = \emptyset, \mathcal{B}^+ \subseteq M\}$$

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. For instance, let us consider the following table.

P :	$P^{\{c,b\}}$:	$(P^*)^{\{c,b\}}$:
$\alpha_1 : a \vee b \leftarrow \top.$	$\alpha_1 : b \leftarrow \top.$	$a \vee b \leftarrow \top.$
$\alpha_2 : c \leftarrow\ not\ a.$	$\alpha_2 : c \leftarrow \top.$	$c \leftarrow \top.$
$\alpha_3 : c \leftarrow\ not\ b.$		

In the first column, the possibilistic program P is defined, in the second one the program P is reduced by $\{c, b\}$ (according to Definition 8) and in the third one the program P^* is reduced by $\{c, b\}$ (according to Gelfond-Lifschitz reduction). Observe that the reduction of Definition 8 removes from the head of the possibilistic disjunctive clauses any atom which does not belong to M. As we will see in Section 4.2, this property will be helpful for defining a possibilistic semantics for possibilistic disjunctive programs based on a fixpoint operator.

Example 4
In order to continue with our medical scenario described in the introduction, let P be a ground instance of the possibilistic program presented in Example 3:

probable: $r\_inf(present, 1) \vee no\_r\_inf(present, 1) \leftarrow action(transplant, 0), d\_inf(present, 0).$
confirmed: $o(good\_graft\_funct, 1) \vee o(delayed\_graft\_funct, 1) \vee o(terminal\_insufficient\_funct, 1) \leftarrow action(transplant, 0).$
confirmed: $action(transplant, 0) \leftarrow o(terminal\_insufficient\_funct, 0).$
plausible: $cs(stable, 1) \leftarrow o(good\_graft\_funct, 1).$
plausible: $cs(unstable, 1) \leftarrow o(delayed\_graft\_funct, 1).$
plausible: $cs(\text{0-urgency}, 1) \leftarrow o(terminal\_insufficient\_funct, 1), action(transplant, 0).$
certain: $\bot \leftarrow action(transplant, 0), action(wait, 0).$
certain: $\bot \leftarrow action(transplant, 0), cs(dead, 0).$
certain: $d\_inf(present, 0) \leftarrow \top.$

certain: $no\_r\_inf(present, 0) \leftarrow \top$.
certain: $o(terminal\_insufficient\_funct, 0) \leftarrow \top$.
certain: $cs(stable, 0) \leftarrow \top$.

Observe that the variables of time T and $T2$ were instantiated with the values 0 and 1 respectively; moreover, observe that the atoms $\neg r\_inf(present, 0)$ and $\neg r\_inf(present, 1)$ were replaced by $no\_r\_inf(present, 0)$ and $no\_r\_inf(present, 1)$ respectively. This change was applied in order to manage the strong negation.

Now, let S be the following possibilistic set:

$$\begin{aligned} S = \quad & \{(d\_inf(present, 0), certain), (no\_r\_inf(present, 0), certain), \\ & (o(terminal\_insufficient\_funct, 0), certain), (cs(stable, 0), certain), \\ & (action(transplant, 0), confirmed), (o(good\_graft\_funct, 1), confirmed), \\ & (cs(stable, 1), plausible), (no\_r\_inf(present, 1), probable)\}. \end{aligned}$$

We can see that P^{S^*} is:

probable: $no\_r\_inf(present, 1) \leftarrow action(transplant, 0), d\_inf(present, 0)$.
confirmed: $o(good\_graft\_funct, 1) \leftarrow action(transplant, 0)$.
confirmed: $action(transplant, 0) \leftarrow o(terminal\_insufficient\_funct, 0)$.
plausible: $cs(stable, 1) \leftarrow o(good\_graft\_funct, 1)$.
plausible: $cs(unstable, 1) \leftarrow o(delayed\_graft\_funct, 1)$.
plausible: $cs(\text{0-urgency}, 1) \leftarrow o(terminal\_insufficient\_funct, 1), action(transplant, 0)$.
certain: $\bot \leftarrow action(transplant, 0), action(wait, 0)$.
certain: $\bot \leftarrow action(transplant, 0), cs(dead, 0)$.
certain: $d\_inf(present, 0) \leftarrow \top$.
certain: $no\_r\_inf(present, 0) \leftarrow \top$.
certain: $o(terminal\_insufficient\_funct, 0) \leftarrow \top$.
certain: $cs(stable, 0) \leftarrow \top$.

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 9 (*Possibilistic answer set*)
Let $P = \langle (\mathcal{Q}, \leq), N \rangle$ be a possibilistic disjunctive logic program and M be a set of possibilistic atoms 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 $M \sqsubseteq M'$.

Example 5
Let P be again the possibilistic program of Example 3 and S be the possibilistic set of atoms introduced in Example 4. First of all, we can see that S^* is an answer set of the extended disjunctive program P^*. Hence, in order to prove that S is a possibilistic answer set of P, we have to verify that $P^{S^*} \vdash_{PL} S$. This means that for each possibilistic atom

$p \in S$, $P^{S^*} \vdash_{PL} p$. We can see that it is straightforward that

$$P^{S^*} \vdash_{PL} \quad \{(d\_inf(present, 0), certain), (no\_r\_inf(present, 0), certain), (o(terminal\_insufficient\_funct, 0), certain), (cs(stable, 0), certain)\}$$

Now let us prove $(cs(stable, 1), plausible)$ from P^{S^*}.

Premises from P^{S^*}	
1. $o(terminal\_insufficient\_funct, 0)$	$certain$
2. $o(terminal\_insufficient\_funct, 0) \rightarrow action(transplant, 0)$	$confirmed$
3. $action(transplant, 0) \rightarrow o(good\_graft\_funct, 1)$	$confirmed$
4. $o(good\_graft\_funct, 1) \rightarrow cs(stable, 1)$.	$plausible$
From 1 and 2 by GMP	
5. $action(transplant, 0)$	$confirmed$
From 3 and 5 by GMP	
6. $o(good\_graft\_funct, 1)$	$confirmed$
From 4 and 6 by GMP	
7. $cs(stable, 1)$.	$plausible$

In this proof, we can also see the inference of the possibilistic atom $(action(transplant, 0), confirmed)$. The proof of the possibilistic atom $(no\_r\_inf(present, 1), probable)$ is similar to the proof of the possibilistic atom $(cs(stable, 1), plausible)$. Therefore, we can say that $P^{S^*} \vdash_{PL} S$ is true. Now, notice that there does not exists a possibilistic set S' such that $S' \neq S$, $P^{(S')^*} \vdash_{PL} S'$ and $S \sqsubseteq S'$; hence, we can conclude that S is a possibilistic answer set of P.

Now what can we say from S about our medical scenario? We can say that if it is *confirmed* that a transplanting is done with a donor with an infection, it is *probable* that the recipient cannot be infected after the transplanting; moreover it is *plausible* that he can be stable. It is worth mentioning that this *optimistic* conclusion is just one of the possible scenarios that we can infer from the program P. In fact, the program P has six possibilistic answer sets where we can find pessimistic scenarios such as it is *probable* that the recipient is infected by the organ donor's infection and; moreover, it is *confirmed* that the recipient needs another transplanting.

Now, let us study some properties of the possibilistic answer set semantics. First, observe that there is an important condition *w.r.t.* the definition of a *possibilistic answer set*. This is that a possibilistic set S cannot be 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 that 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^*. Therefore, S could not be a possibilistic answer set of P. Hence, a straightforward relation between the possibilistic answer semantics and the answer set semantics is formalized by the following proposition.

Proposition 2

Let P be a possibilistic disjunctive logic program. M is a possibilistic answer set of P iff M^* is an answer set of P^*.

When all the possibilistic clauses of a possibilistic program P have as certain 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 $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic disjunctive logic program and $\mathcal{TOP}_{\mathcal{Q}}$ be the top of the lattice $(\mathcal{Q}, \leq)$. If $\forall r \in P$, $n(r) = \mathcal{TOP}_{\mathcal{Q}}$, and M' is an answer set of P^*, then $M := \{(l, \mathcal{TOP}_{\mathcal{Q}}) | l \in M'\}$ is a possibilistic answer set of P.

For the class of possibilistic normal logic programs which are built under a totally ordered set, our definition of possibilistic answer set is closely related to the definition of a *possibilistic stable model* presented in (Nicolas et al. 2006). In fact, both semantics coincide.

Proposition 4
Let $P := \langle(Q, \leq), N\rangle$ be a possibilistic normal program such that $(Q, \leq)$ is a totally ordered set and $\mathcal{L}_P$ has no extended atoms. M is a possibilistic answer set of P if and only if M is a possibilistic stable model of P.

In order to prove that the possibilistic answer set semantics is computable, we will introduce a straightforward generalization of the possibilistic resolution rule introduced in (Dubois et al. 1994):

(R) $(c_1\ \alpha_1)(c_2\ \alpha_2) \vdash (R(c_1, c_2)\ \mathcal{GLB}(\{\alpha_1, \alpha_2\}))$

where $R(c_1, c_2)$ is any classical resolvent of c_1 and c_2 such that c_1 and c_2 are disjunctions of literals. It is worth mentioning that it is easy to transform any possibilistic disjunctive logic program P into a set of possibilistic disjunctions $\mathcal{C}$. Indeed, $\mathcal{C}$ can be obtained as follows:

$$\mathcal{C} := \bigcup\{(a_1 \vee \ldots \vee a_m \vee \sim a_1 \vee \cdots \vee \sim a_j \vee a_{j+1} \vee \ldots, a_n\ \alpha) | \\ (\alpha : a_1 \vee \ldots \vee a_m \leftarrow a_1, \ldots, a_j, not\ a_{j+1}, \ldots, not\ a_n) \in P\}$$

We remember to the reader that whenever we consider a possibilistic program as a theory, each negative literal $not\ a$ is replaced by $\sim a$ such that $\sim$ is regarded as the negation in classic logic — in Example 6, the transformation of a possibilistic program into a set of possibilistic disjunctions is shown.

The following proposition shows that the resolution rule (R) is sound.

Proposition 5
Let $\mathcal{C}$ be a set of possibilistic disjunctions, and $C = (c\ \alpha)$ be a possibilistic clause obtained by a finite number of successive application of *(R)* to $\mathcal{C}$; then $\mathcal{C} \vdash_{PL} C$.

Like the possibilistic rule introduced in (Dubois et al. 1994), (R) is complete for refutation. We will say that a possibilistic disjunctive program P is *consistent* if P has at least a possibilistic answer set. Otherwise P is said to be *inconsistent*. The inconsistency degree of a possibilistic logic program P is $Inc(P) = \mathcal{GLB}(\{\alpha | P_\alpha \text{ is consistent }\})$.

Proposition 6
Let P be a set of possibilistic clauses and $\mathcal{C}$ be the set of possibilistic disjunctions obtained from P; then the valuation of the optimal refutation by resolution from $\mathcal{C}$ is the inconsistent degree of P.

The main implication of Proposition 5 and Proposition 6 is that (R) suggests a method for inferring a possibilistic formula from a possibilistic knowledge base.

Corollary 1
Let $P := \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic disjunctive logic program, φ be literal and $\mathcal{C}$ be a set of possibilistic disjunctions obtained from $N \cup \{(\sim \varphi\ \mathcal{TOP}_{\mathcal{Q}})\}$; then the valuation of the optimal refutation from $\mathcal{C}$ is $n(\varphi)$ *i.e.*$P \vdash_{PL} (\varphi\ n(\varphi))$.

Based on the fact that the resolution rule (R) suggests a method for inferring the necessity value of a possibilistic formula, we can define the following function for computing the possibilistic answer set models of an possibilistic program P.

Function $Poss\_Answer\_Sets(P)$
Let $ASP(P^*)$ be a function that computes the answer set models of the standard logic program P^* *e.g.*, DLV (DLV 1996).
  Poss-ASP := $\emptyset$
  For all $S \in ASP(P^*)$
    Let $\mathcal{C}$ be the set of possibilistic disjunctions obtained from P^S.
    $S' := \emptyset$
    for all $a \in S$
      $C' := \mathcal{C} \cup \{(\sim a\ \mathcal{TOP}_{\mathcal{Q}})\}$
      Search for a deduction of $(R(\Box)\ \alpha)$ by applying repeatedly
      the resolution rule (R) from C', with α maximal.
      $S' := S' \cup \{(a\ \alpha)\}$
    endfor
    Poss-ASP := Poss-ASP $\cup\ S'$
  endfor
return(Poss-ASP).

The following proposition formalizes that the function $Poss\_Answer\_Sets$ computes all the possibilistic answer sets of a possibilistic logic program.

Proposition 7
Let $P := \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic logic program. The set Poss-ASP returned by $Poss\_Answer\_Sets(P)$ is the set of all the possibilistic answer sets of P.

In order to illustrate this algorithm, let us consider the following example:

Example 6
Let $P := \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic program such that $\mathcal{Q} := \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, $\leq$ is the standard relation between rational numbers and N the following set of possibilistic clauses:

$$
\begin{array}{lrl}
0.7: & a \vee b & \leftarrow not\ c. \\
0.6: & c & \leftarrow not\ a, not\ b. \\
0.8: & a & \leftarrow b. \\
0.9: & e & \leftarrow b. \\
0.6: & b & \leftarrow a. \\
0.5: & b & \leftarrow a.
\end{array}
$$

Firs of all, we can see that P^* has two answer sets: $S_1 := \{a, b, e\}$ and $S_2 := \{c\}$. This means that P has two possibilistic answer set models. Let us consider S_1 for our example. Then, one can see that P^{S_1} is:

$$
\begin{array}{lrl}
0.7: & a \vee b & \leftarrow \top. \\
0.8: & a & \leftarrow b. \\
0.9: & e & \leftarrow b. \\
0.6: & b & \leftarrow a. \\
0.5: & b & \leftarrow a.
\end{array}
$$

Then $\mathcal{C} := \{(a \vee b\ 0.7), (a \vee \sim b\ 0.8), (e \vee \sim b\ 0.9), (b \vee \sim a\ 0.6), (b \vee \sim a\ 0.5)\}$. In order to infer the necessity value of the atom a, we add $(\sim a\ 1)$ to $\mathcal{C}$ and a search for finding an optimal refutation is applied. As we can see in Figure 3, there are three refutations, however the optimal refutation is $(\Box\ 0.7)$. This means that the best necessity value for the atom a is 0.7.

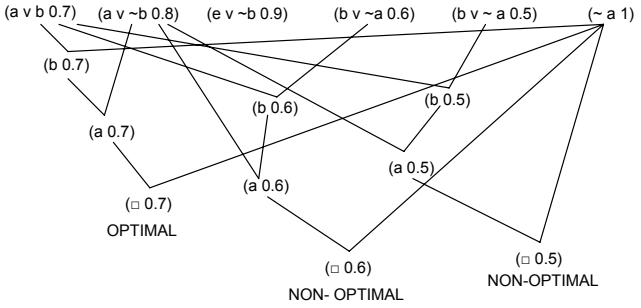

Fig. 3. Possibilistic resolution: Search for an *optimal refutation* for the atom a.

In Figure 4, we can see the optimal refutation search for the atom b. As we can see the optimal refutation is $(\Box\ 0.6)$; hence the best necessity value for the atom b is 0.6.

In Figure 5, we can see that the best necessity value for the atom e is 0.6.

From the process of search, we can infer that a possibilistic answer set of the program P is : $\{(a, 0.7), (b, 0.6), (e, 0.6)\}$.

4.2 Possibilistic answer sets based on partial evaluation

We have defined a possibilistic answer set semantics by considering the formal proof theory of possibilistic logic. However, in standard logic programming there are several frame-

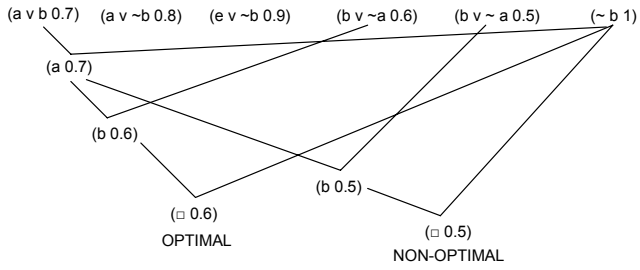

Fig. 4. Possibilistic resolution: Search for an *optimal refutation* for the atom b.

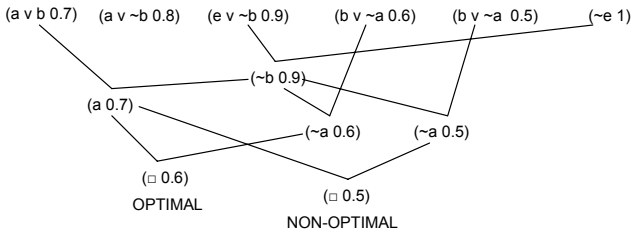

Fig. 5. Possibilistic resolution: Search for an *optimal refutation* for the atom e.

works for analyzing, defining and computing logic programming semantics (Dix 1995a; Dix 1995b). One of these approaches is based on program transformations, in fact there are many studies around this approach *e.g.*, (Brass and Dix 1999; Brass and Dix 1997; Brass and Dix 1998; Dix et al. 2001). For the case of disjunctive logic program, one important transformation is *partial evaluation (also called unfolding)* (Brass and Dix 1999).

In this section, we will see that it is also possible to define a possibilistic disjunctive semantics based on an operator which is a combination between partial evaluation for disjunctive logic programs and the infer rule GMP of possibilistic logic (see Section 2.4). This semantics has the same behavior to the semantics based on the proof theory of possibilistic logic.

We will start this section by defining a version of the general principle of partial evaluation (GPPE) for possibilistic positive disjunctive clauses.

Definition 10 (*Grade-GPPE (G-GPPE)*)
Let r_1 be a possibilistic clause of the form $\alpha : \mathcal{A} \leftarrow \mathcal{B}^+ \cup \{B\}$ and r_2 a possibilistic clause of the form $\alpha_1 : \mathcal{A}_1 \leftarrow \top$ such that $B \in \mathcal{A}_1$, then

$$\text{G-GPPE}(r_1, r_2) = (\mathcal{GLB}(\{\alpha, \alpha_1\}) : \mathcal{A} \cup (\mathcal{A}_1 \setminus \{B\}) \leftarrow \mathcal{B}^+)$$

Observe that one of the possibilistic clauses which is considered by G-GPPE has an empty body. For instance, let us consider the following two possibilistic clauses:

$$r_1 = \ 0.7 : a \vee b \leftarrow \top.$$

$r_2 = \ 0.9 : e \leftarrow b.$

Then G-GPPE$(r_1, r_2) = (0.7 : e \vee a \leftarrow \top)$. Now, by considering G-GPPE, we will define the operator $\mathcal{T}$.

Definition 11

Let P be a possibilistic positive logic program. The operator $\mathcal{T}(P)$ is defined as follows:

$$\mathcal{T}(P) := P \cup \{\text{G-GPPE}(r_1, r_2) | r_1, r_2 \in P\}$$

In order to illustrate the operator $\mathcal{T}$, let us consider the program P^{S_1} of Example 6.

$$\begin{array}{lrl} 0.7: & a \vee b & \leftarrow \top. \\ 0.8: & a & \leftarrow b. \\ 0.9: & e & \leftarrow b. \\ 0.6: & b & \leftarrow a. \\ 0.5: & b & \leftarrow a. \end{array}$$

Hence, $\mathcal{T}(P^{S_1})$ is:

$$\begin{array}{lll} 0.7: & a \vee b \leftarrow \top. & \qquad 0.7: \quad a \leftarrow \top. \\ 0.8: & a \leftarrow b. & \qquad 0.7: \quad e \vee a \leftarrow \top. \\ 0.9: & e \leftarrow b. & \qquad 0.6: \quad b \leftarrow \top. \\ 0.6: & b \leftarrow a. & \qquad 0.5: \quad b \leftarrow \top. \\ 0.5: & b \leftarrow a. & \end{array}$$

Notice that by considering the possibilistic clauses that were added to P^{S_1} by $\mathcal{T}$, one can apply G-GPPE again. For instance, if we consider $0.6 : b \leftarrow \top$ and $0.9 : e \leftarrow b$ from $\mathcal{T}(P^{S_1})$, G-GPPE infers $0.6 : e \leftarrow \top$. Indeed, $\mathcal{T}(\mathcal{T}(P^{S_1}))$ is:

$$\begin{array}{lll} 0.7: & a \vee b \leftarrow \top. & \qquad 0.7: \quad a \leftarrow \top. \qquad\qquad 0.6: \quad a \leftarrow \top. \\ 0.8: & a \leftarrow b. & \qquad 0.7: \quad e \vee a \leftarrow \top. \qquad\qquad 0.5: \quad a \leftarrow \top. \\ 0.9: & e \leftarrow b. & \qquad 0.6: \quad b \leftarrow \top. \qquad\qquad 0.6: \quad e \leftarrow \top. \\ 0.6: & b \leftarrow a. & \qquad 0.5: \quad b \leftarrow \top. \qquad\qquad 0.5: \quad e \leftarrow \top. \\ 0.5: & b \leftarrow a. & \qquad\qquad\qquad\qquad\qquad\qquad\quad 0.6: \quad b \vee e \leftarrow \top. \\ & & \qquad\qquad\qquad\qquad\qquad\qquad\quad 0.5: \quad b \vee e \leftarrow \top. \end{array}$$

An important property of the operator $\mathcal{T}$ is that it always reaches a fix-point.

Proposition 8

Let P be a possibilistic disjunctive logic program. If $\Gamma_0 := \mathcal{T}(P)$ and $\Gamma_i := \mathcal{T}(\Gamma_{i-1})$ such that $i \in \mathcal{N}$, then $\exists\ n \in \mathcal{N}$ such that $\Gamma_n = \Gamma_{n-1}$. We denote Γ_n by $\Pi(P)$.

Let us consider again the possibilistic program P^{S_1}. We can see that $\Pi(P^{S_1})$ is:

$0.7:$	$a \vee b \leftarrow \top.$	$0.7:$	$a \leftarrow \top.$	$0.6:$	$a \leftarrow \top.$	0.6	$a \vee e \leftarrow \top$
$0.8:$	$a \leftarrow b.$	$0.7:$	$e \vee a \leftarrow \top.$	$0.5:$	$a \leftarrow \top.$	0.5	$a \vee e \leftarrow \top$
$0.9:$	$e \leftarrow b.$	$0.6:$	$b \leftarrow \top.$	$0.6:$	$e \leftarrow \top.$		
$0.6:$	$b \leftarrow a.$	$0.5:$	$b \leftarrow \top.$	$0.5:$	$e \leftarrow \top.$		
$0.5:$	$b \leftarrow a.$			$0.6:$	$b \vee e \leftarrow \top.$		
				$0.5:$	$b \vee e \leftarrow \top.$		

Observe that in $\Pi(PS_1)$ there are possibilistic facts (possibilistic clauses with empty bodies and one atom in their heads) with different necessity value. In order to infer the best necessity value of each possibilistic fact, one can consider the *least upper bound* of these values. For instance the best necessity value for the possibilistic atom a is $\mathcal{LUB}(\{0.7, 0.6, 0.5\}) = 0.7$. Based on this idea, we will define Sem_{min}.

Definition 12
Let P be a possibilistic logic program and $Facts(P, a) := \{(\alpha : a \leftarrow \top) | (\alpha : a \leftarrow \top) \in P\}$. $Sem_{min}(P) := \{(x, \alpha) | Facts(P, x) \neq \emptyset$ and $\alpha := \mathcal{LUB}(\{n(r) | r \in Facts(P, x)\})\}$ where $x \in \mathcal{L}_P$.

It is easy to see that $Sem_{min}(\Pi(P^{S_1}))$ is $\{(a, 0.7), (b, 0.6), (e, 0.6)\}$. Now by considering the operator $\mathcal{T}$ and Sem_{min}, we can define a semantics for possibilistic disjunctive logic programs that will be called possibilistic-$\mathcal{T}$ answer set semantics.

Definition 13
Let P be a possibilistic disjunctive logic program and M be a set of possibilistic atoms such that M^* is an answer set of P^*. M is a possibilistic-$\mathcal{T}$ answer set of P if and only if $M = Sem_{min}(\Pi(P^{M^*}))$.

In order to illustrate this definition, let us consider again the program P of Example 6 and $S = \{(a, 0.7), (b, 0.6), (e, 0.6)\}$. As commented in Example 6, S^* is an answer set of P^*. We have already seen that $Sem_{min}(\Pi(P^{S_1}))$ is $\{(a, 0.7), (b, 0.6), (e, 0.6)\}$, therefore we can say that S is a possibilistic-$\mathcal{T}$ answer set of P. Observe that the possibilistic-$\mathcal{T}$ answer set semantics and the possibilistic answer set semantics coincide. In fact the following proposition guaranties that both semantics are the same.

Proposition 9
Let P be a possibilistic disjunctive logic program and M a set of possibilistic atoms. M is a possibilistic answer set of P if and only if M is a possibilistic-$\mathcal{T}$ answer set of P.

4.3 A possibilistic semantics based on pstable models

We have defined two semantics which capture the semantics for possibilistic disjunctive logic programs. Since these semantics were defined as extensions of the standard answer set semantics, there are some possibilistic logic programs which have no possibilistic answer sets. For instance, a simple possibilistic program as

$$\alpha : a \leftarrow not\ a$$

has no possibilistic answer set. In fact, the existence of one clause of this form will affect

all the possibilistic knowledge base in such a way that the possibilistic knowledge base will not have a possibilistic answer set.

In this section, we present a second approach for capturing the semantics of possibilistic logic programs. This approach is based on *pstable semantics* (see Section 2.3.2). Pstable semantics emerges from the fusion of paraconsistent logics and ASP. This semantics is able to capture ASP; moreover, it is less sensitive (in the sense of inconsistency) than the answer set semantics.

We will start this section defining pstable semantics for *possibilistic normal programs* and after that we will show that this semantics is also able to capture the semantics of possibilistic disjunctive logic programs. Like possibilistic answer set semantics, possibilistic pstable semantics is defined in terms of a syntactic reduction. This reduction is based on Definition 2.

Definition 14
Let P be a possibilistic normal program and M a set of atoms. We define

$$PRED(P, M) := \{(\alpha : a \leftarrow \mathcal{B}^+,\ not\ (\mathcal{B}^- \cap M))|(\alpha : a \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-) \in P\}$$

Let us consider the following example in order to illustrate this definition.

Example 7
First, let S be the set $\{(a, 0.6), (b, 0.7)\}$ and $P_1 := \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic program such that $(\mathcal{Q}, \geq)$ is the lattice of Example 6 and N the following set of possibilistic clauses:

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

Then, we can see that the program $PRED(P_1, S^*)$ is:

$$0.7 : a \leftarrow not\ b.$$
$$0.6 : a \leftarrow b.$$
$$0.8 : b \leftarrow a.$$

By considering the reduction PRED, we define the possibilistic pstable semantics as follows.

Definition 15 (*Possibilistic Pstable Semantics*)
Let P be a possibilistic normal logic program and M be a set of possibilistic atoms such that M^* is a pstable model of P^*. We say that M is a possibilistic pstable model of P if and only if $PRED(P, M^*) \vdash_{PL} M$ and $\nexists M'$ such that $M' \neq M$, $PRED(P, M^*) \vdash_{PL} M'$ and $M \sqsubseteq M'$.

Example 8
Let us continue with Example 7. We have already seen that $PRED(P_1, S^*)$ is:

$$0.7 : a \leftarrow not\ b.$$
$$0.6 : a \leftarrow b.$$
$$0.8 : b \leftarrow a.$$

Then, we want to know if S is a *possibilistic pstable models* of P_1. First of all, we have already seen in Section 2.3.2 that S^* is a pstable models of P_1^*. Hence, we have to construct a proof in possibilistic logic for $(a, 0.6)$ and $(b, 0.7)$. Let us consider the proof for the possibilistic atom $(a, 0.6)$:

1. $(a \vee b) \rightarrow ((b \rightarrow a) \rightarrow a)$	1	**Tautology**
2. $\sim\ b \rightarrow a$	0.7	**Premise from** $PRED(P_1, S)$
3. $a \vee b$	0.7	**From 2 by possibilistic logical equivalency**
4. $(b \rightarrow a) \rightarrow a)$	0.7	**From 1 and 3 by GMP**
5. $b \rightarrow a$	0.6	**Premise from** $PRED(P_1, S)$
6. a	0.6	**From 4 and 5 by GMP**

Observe that the formula $\sim\ b \rightarrow a \quad 0.7$ corresponds to the possibilistic normal clause $0.7 : a \leftarrow\ not\ b$ which belongs to the program $RED(P_1, S^*)$. The proof for $(b, 0.7)$ is similar to the proof of $(a, 0.6)$. Notice that $\nexists\ S'$ such that $PRED(P_1, S^*) \vdash_{PL} S'$ and $S \sqsubseteq S'$. Therefore, we can conclude that S is a *possibilistic pstable models* of P_1.

Observe that the possibilistic program P_1 is an example where the possibilistic pstable semantics is different to both the possibilistic stable semantics (Nicolas et al. 2006) and the possibilistic answer set semantics (Definition 9). In fact, P_1 has no possibilistic stable model neither possibilistic answer set.

Even thought, there are programs where the possibilistic pstable semantics does not coincide with the possibilistic stable semantics neither with the possibilistic answer set semantics, we can identify a relationship between the possibilistic answer set semantics and the possibilistic pstable semantics.

Proposition 10
Let P be a possibilistic normal program. If M is a possibilistic answer set of P, then the following conditions hold:

a) M^* is a pstable model of P^*.
b) there exists a possibilistic pstable mode M' of P such that $M \sqsubseteq M'$ and $M^* = M'^*$.

This proposition points out that whenever a possibilistic normal program has a possibilistic answer set M there exists a possibilistic pstable model M' such that the main differences between M and M' are the necessity-values of their elements. For instance, let us consider the following possibilistic program P:

$$0.3 : a \leftarrow \top.$$
$$0.8 : a \leftarrow\ not\ a.$$

One can see that P has the possibilistic answer set $M := \{(a, 0.3)\}$ and the possibilistic pstable model $M' := \{(a, 0.8)\}$. It is clear that $M^* = M'^*$; however, $M \sqsubseteq M'$.

Remark 3
It is worth to comment that when $P = \langle(\mathcal{Q}, \leq), N\rangle$ is a possibilistic program which does not contain extended atoms *i.e.*atoms of form $\neg a$, and $(\mathcal{Q}, \leq)$ is a totally ordered set, it will be also true that: If M is a possibilistic stable model of P, then the following conditions

hold: $1.-$ M^* is a pstable model of P^* and $2.-$ there exists a possibilistic pstable mode M' of P such that $M \sqsubseteq M'$ and $M^* = M'^*$.

An interesting property of the possibilistic pstable semantics is that this semantics supports a kind of monotony *w.r.t.* the inference under possibilistic logic. In order to formalize this property, we will say that P is *equivalent* to P' *under the possibilistic pstable semantics* if and only if any possibilistic pstable model of P is also a possibilistic pstable model of P' and vice versa.

Proposition 11
Let P be a possibilistic normal program. If $P \vdash_{PL} (x\ \alpha)$ then P is equivalent to $P \cup \{(x\ \alpha)\}$ under the possibilistic pstable semantics.

Notice that neither the possibilistic answer set semantics nor the possibilistic stable semantics (Nicolas et al. 2006) satisfy Proposition 11 *i.e.*if $P \vdash_{PL} (x\ \alpha)$, then P is not equivalent to $P \cup \{(x\ \alpha)\}$ under possibilistic answer set semantics neither under the possibilistic stable semantics. In order to show this, let us consider a simple possibilistic logic program P:

$$\alpha : a \leftarrow not\ a$$

It is clear that $P \vdash_{PL} (a\ \alpha)$. P has no a possibilistic stable model neither a possibilistic answer set. However, $P \cup \{(a\ \alpha)\}$ has a possibilistic stable model and a possibilistic answer set which is the same in both cases and is $\{(a, \alpha)\}$.

The possibilistic answer set semantics was defined for the family of possibilistic disjunctive logic programs. This means that the possibilistic clauses could have a disjunction in their heads. Since the possibilistic pstable semantics was defined for possibilistic normal programs, one can think that the possibilistic pstable semantics is less expressive than the possibilistic answer semantics. However, by considering a simple mapping, one can extend the definition of possibilistic pstable models for possibilistic normal programs in order to define a semantics for possibilistic disjunctive logic programs.

An interesting result is that there exists a relationship between the possibilistic answer sets of a possibilistic disjunctive logic program P and the possibilistic pstable models of the possibilistic normal program $TRAD(P)$ (defined below). In order to formalize this result, we define some basic terms.

Given a possibilistic disjunctive logic program $P := \langle (\mathcal{Q}, \leq), N \rangle$, P_c will denote the set of possibilistic constraints which belong to P and P_N will denote the rest *i.e.*$P_N = P \setminus P_c$.

In order to see a possibilistic disjunctive logic program as a possibilistic normal logic program, we will define a simple mapping of a possibilistic disjunctive logic programs into a possibilistic normal logic programs.

Definition 16
Let P be a possibilistic disjunctive logic program and
$(\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-) \in P_N$. We define

$$R(\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-) := \bigcup_{a \in \mathcal{A}} \{(\alpha : a \leftarrow \mathcal{B}^+,\ not\ (\mathcal{B}^- \cup (\mathcal{A} \setminus \{a\}))\}$$

The generalization of R over P is as follows: $R(P) := \bigcup_{C \in P} R(C)$.

For instance, $R(0.7 : a \vee b \leftarrow not\ c) := \{0.7 : a \leftarrow not\ c, not\ b;\ 0.7 : b \leftarrow not\ c, not\ a\}$

By considering the mapping R, we define the function $TRAD$.

Definition 17
Let P be a possibilistic disjunctive logic program. We define TRAD(P) as:

$$TRAD(P) := R(P_N) \cup P_c$$

Now, by considering the function $TRAD$, we formalize that whenever a possibilistic disjunctive logic program P has a possibilistic answer set M, there exists a possibilistic pstable model M' of the possibilistic normal program $TRAD(P)$ such that the main differences between M and M' are the necessity-values of their elements.

Theorem 1
Let P be a possibilistic disjunctive program. If M is a possibilistic answer set of P, then it implies that

a) M^* is a pstable model of $TRAD(P)^*$.
b) there exists a possibilistic pstable mode M' of $TRAD(P)$ such that $M \sqsubseteq M'$ and $M^* = M'^*$.

Observe that this result is a generalization of the result of Proposition 10. In terms of computability, since there is an algorithm for inferring pstable models (López 2006) and the possibilistic pstable semantics is based on the proof theory of possibilistic logic, the following proposition is a direct consequence of Proposition 7.

Proposition 12
Given a possibilistic program $P := \langle (\mathcal{Q}, \leq), N \rangle$ there exists an algorithm that computes the set of possibilistic pstable models of P.

5 Inconsistency in possibilistic logic programs

In this section, we will motivate the relevance of considering inconsistent possibilistic knowledge bases and we will introduce some criteria for managing inconsistent possibilistic logic programs.

5.1 Relevance of inconsistent possibilistic logic programs

Inconsistent knowledge bases usually are regarded as an *epistemic hell* that have to be avoided at all costs. However, many times it is difficult or impossible to stay away of managing inconsistent knowledge bases. There are approaches, as it is the case of Paraconsistent Logics, which allow to infer inconsistent pseudo-models. For instance, in (Bueno 2006), Bueno argues that to pursue inconsistent systems is a useful device for a number of reasons: (1) this is often the only way to explore inconsistent information without arbitrarily rejecting precious data. (2) pursuing inconsistent systems is sometimes the only way to obtain new information (particularly information that conflicts with deeply entrenched

theories). As a result, (3) pursuing inconsistent belief systems allows us to make better *informed decisions* regarding which bits of information to accept or reject in the end.

In order to illustrate a small example, where to explore inconsistent information can be important to make a better informed decision, we will continue with the medical scenario described in Section 1. In Example 4, we have already presented the grounded program $P_{infections}$ of our medical scenario:

probable: $r\_inf(present, 1) \vee no\_r\_inf(present, 1) \leftarrow action(transplant, 0), d\_inf(present, 0).$
confirmed: $o(good\_graft\_funct, 1) \vee o(delayed\_graft\_funct, 1) \vee o(terminal\_insufficient\_funct, 1) \leftarrow action(transplant, 0).$
confirmed: $action(transplant, 0) \leftarrow o(terminal\_insufficient\_funct, 0).$
plausible: $cs(stable, 1) \leftarrow o(good\_graft\_funct, 1).$
plausible: $cs(unstable, 1) \leftarrow o(delayed\_graft\_funct, 1).$
plausible: $cs(\text{0-urgency}, 1) \leftarrow o(terminal\_insufficient\_funct, 1), action(transplant, 0).$
certain: $\bot \leftarrow action(transplant, 0), action(wait, 0).$
certain: $\bot \leftarrow action(transplant, 0), cs(dead, 0).$
certain: $d\_inf(present, 0) \leftarrow \top.$
certain: $no\_r\_inf(present, 0) \leftarrow \top.$
certain: $o(terminal\_insufficient\_funct, 0) \leftarrow \top.$
certain: $cs(stable, 0) \leftarrow \top.$

As commented in Example 4, in this program the atoms $\neg r\_inf(present, 0)$ and $\neg r\_inf(present, 1)$ were replaced by $no\_r\_inf(present, 0)$ and $no\_r\_inf(present, 1)$ respectively. Usually in standard answer set programming, the constraints

$\bot \leftarrow no\_r\_inf(present, 0), r\_inf(present, 0).$
$\bot \leftarrow no\_r\_inf(present, 1), no\_r\_inf(present, 1).$

must be added to the program for avoiding inconsistent answer sets. In order to comment the role of this kind of constraints, let C_1 be the following possibilistic constraints:

certain: $\bot \leftarrow no\_r\_inf(present, 0), r\_inf(present, 0).$
certain: $\bot \leftarrow no\_r\_inf(present, 1), no\_r\_inf(present, 1).$

Also let us consider three new possibilistic clauses (denoted by P_v):

confirmed: $v(kidney, 0) \leftarrow cs(stable, 1), action(transplant, 0).$
probable: $no\_v(kidney, 0) \leftarrow r\_inf(present, 1), action(transplant, 0).$
certain: $\bot \leftarrow not\ cs(stable, 1).$

The intended meaning of the predicate $v(t, T)$ is that the organ t is viable for transplanting and T denotes a moment at the time. Observe that we replaced the atom $\neg v(kidney, 0)$ by $no\_v(kidney, 0)$. The reading of the first clause is that if the clinical situation of the

organ recipient is stable after the graft, then it is *confirmed* that the kidney is viable for transplanting. The reading of the second one is that if the organ recipient is infected after the graft, then it is *plausible* that the kidney is not viable for transplanting. The reading of the possibilistic constraint is that we do not want to consider scenarios where the clinical situation of the organ recipient is not stable. We will also consider the respective possibilistic constrain of the atoms $no\_v(kidney, 0)$ and $v(kidney, 0)$ (denoted by C_2):

certain: $\bot \leftarrow no\_v(kidney, 0), v(kidney, 0).$

Hence we define two programs

$$P := P_{infections} \cup P_v \text{ and } P_c := P_{infections} \cup P_v \cup C_1 \cup C_2$$

Basically, the difference between P and P_c is that P allows inconsistent possibilistic models and P_c does not allow inconsistent possibilistic models.

Now let us consider the possibilistic answer sets of the programs P and P_c. We can see that the program P_c has just one possibilistic answer set:

$\{(d\_inf(present, 0), certain), (no\_r\_inf(present, 0), certain),$
$(o(terminal\_insufficient\_funct, 0), certain), (cs(stable, 0), certain),$
$(action(transplant, 0), confirmed), (o(good\_graft\_funct, 1), confirmed),$
(cs(stable,1), plausible), **(no_r_inf(present,1), probable)**,
(v(kidney,0), plausible)$\}$

This possibilistic answer set suggests that since it is plausible that recipient's clinical situation can be stable after the graft, it is plausible that the kidney is *viable* for transplanting. *Observe that the possibilistic answer sets of P do not warn that the organ recipient could be infected after the graft.*

Let us consider the possibilistic answer set of the program P:

$S_1 := \{(d\_inf(present, 0), certain), (no\_r\_inf(present, 0), certain),$
$(o(terminal\_insufficient\_funct, 0), certain), (cs(stable, 0), certain),$
$(action(transplant, 0), confirmed), (o(good\_graft\_funct, 1), confirmed),$
(cs(stable,1), plausible), **(no_r_inf(present,1), probable)**,
(v(kidney,0), plausible)$\}$

$S_2 := \{(d\_inf(present, 0), certain), (no\_r\_inf(present, 0), certain),$
$(o(terminal\_insufficient\_funct, 0), certain), (cs(stable, 0), certain),$
$(action(transplant, 0), confirmed), (o(good\_graft\_funct, 1), confirmed),$
(cs(stable,1), plausible), **(r_inf(present,1), probable)**,
(v(kidney,0), plausible), **(no_v(kidney,0), probable)**$\}$

P has two possibilistic answer sets: S_1 and S_2. S_1 corresponds to the possibilistic answer set of the program P_c and S_2 is an inconsistent possibilistic answer set — because the atoms **(v(kidney,0), plausible)** and **(no_v(kidney,0), probable)** appear in S_2. Observe that although S_2 is an inconsistent possibilistic answer set, it contains important information

w.r.t. the considerations of our scenario. S_2 suggests that even thought it is plausible that the clinical situation of the organ recipient can be stable after the graft, it is also probable that the organ recipient can be infected by the infection of the donor's organ.

Observe that essentially P_c is unable to infer the possibilistic answer set S_2 by the presence of the possibilistic constraint:

certain: $\bot \leftarrow no\_v(kidney, 0), v(kidney, 0).$

By defining this kind of constrains, we can guarantee that any possibilistic answer set inferred from P_c will be consistent; however, one can omit important considerations *w.r.t.* a decision-making problem. In fact, we agree with Bueno (Bueno 2006) that to consider inconsistent systems, as inconsistent possibilistic answer sets, some times is the only way to explore inconsistent information without arbitrarily rejecting precious data.

5.2 Inconsistency degrees of possibilistic sets

For managing inconsistent possibilistic answer set, it is necessary to define a criterion of preference between possibilistic answer sets. In order to define a criterion between possibilistic answer sets, we will define the concept of *inconsistency degree of a possibilistic set*. We say that a set of possibilistic atoms S is inconsistent (resp. consistent) if and only if S^* is inconsistent (resp. consistent) *i.e.*there exists atom a such that $a, \neg a \in S^*$.

Definition 18
Let $\mathcal{A}$ be a finite set of atoms and extended atoms,($\mathcal{Q}$,$\leq$) be a lattice and $S \in 2^{\mathcal{A}\times\mathcal{Q}}$. The inconsistent degree of S is defined as follows:

$$IconsDegre(S) := \begin{cases} \mathcal{BOT}_{\mathcal{Q}} & \text{if } S^* \text{ is consistent} \\ \mathcal{GLB}(\{\alpha | S_\alpha \text{ is consistent}\}) & \text{otherwise} \end{cases}$$

where $\mathcal{BOT}_{\mathcal{Q}}$ is the bottom of the lattice ($\mathcal{Q}$,$\leq$) and $S_\alpha := \{(a, \alpha_1) \in S | \alpha_1 \geq \alpha\}$.

For instance, the possibilistic answer set S_2 of our example above has an inconsistency degree of *confirmed*. Based on the inconsistency degree of possibilistic sets, we can define a criterion of preference between possibilistic answer sets.

Definition 19
Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic program and M_1, M_2 two possibilistic answer set of P. We say that M_1 is weakest-inconsistent than M_2 if and only if $IconsDegre(M_1) < IconsDegre(M_2)$.

For our example above, it is obvious that S_1 is weakest-inconsistent than S_2. In general terms, we will say that a possibilistic answer set M_1 is preferred than M_2 if and only if M_1 is weakest-inconsistent than M_2. This means that any consistent possibilistic answer set will be preferred than any inconsistent possibilistic answer set.

So far we have commented only the case of inconsistent possibilistic answer set. However, there are possibilistic programs that are inconsistent because they have no possibilistic answer sets neither possibilistic pstable models. For instance, let us consider the following possibilistic program P_{inc} (we are assuming the lattice of Example 6):

$$0.3: \quad a \leftarrow \ not\ b.$$
$$0.5: \quad b \leftarrow \ not\ c.$$
$$0.6: \quad c \leftarrow \ not\ a.$$

Observe that P^*_{inc} has no answer sets neither pstable models; hence, P_{inc} has no possibilistic answer sets neither possibilistic pstable models.

5.3 Restoring inconsistent possibilistic knowledge bases

In order to restore consistency of an inconsistent possibilistic knowledge base, possibilistic logic deletes the set of possibilistic formulæ which are lower than the inconsistent degree of the inconsistent knowledge base. By considering this idea, the authors of (Nicolas et al. 2006) defined the concept of α-cut for possibilistic logic programs. Based on Definition 14 of (Nicolas et al. 2006), we define its respective generalization for our approach.

Definition 20
Let P be a possibilistic logic program

- the strict α-cut is the subprogram $P_{>\alpha} = \{r \in P | n(r) > \alpha\}$
- the consistency cut degree of P:

$$ConsCutDeg(P) := \begin{cases} \mathcal{BOT}_{\mathcal{Q}} & \text{if } P^* \text{ is consistent} \\ \mathcal{GLB}(\{\alpha | P_\alpha \text{ is consistent}\}) & \text{otherwise} \end{cases}$$

where $\mathcal{BOT}_{\mathcal{Q}}$ is the bottom of the lattice ($\mathcal{Q}$,$\leq$).

Notice that the consistency cut degree of a possibilistic logic program identifies the minimum level of certainty for which a strict α-cut of P is consistent. As Nicolas *et al.*, remarked in (Nicolas et al. 2006), by the non-monotonicity of the framework it is not ensure that a higher cut is necessarily consistent.

In order to illustrate these ideas, let us consider again the program P_{inc}. First, we can see that $ConsCutDeg(P_{inc}) = 0.3$; hence, the subprogram $P_{ConsCutDeg(P_{inc})}$ is:

$$0.5: \quad b \leftarrow \ not\ c.$$
$$0.6: \quad c \leftarrow \ not\ a.$$

Observe that this program has a possibilistic answer set which is $\{(c, 0.6)\}$[9]. Hence thanks to the strict α-cut of P, one is able to infer information from P_{inc}

We have commented two kinds of inconsistency in our approach,

- one which arises from the presence of complementary atoms in a possibilistic answer set (or a possibilistic pstable model) and
- the other one which arises from the non-existence of possibilistic answer set (or possibilistic pstable models) of a possibilistic logic program.

[9] Remember that any possibilistic answer set is also a possibilistic pstable model.

For managing the inconsistency of possibilistic answer sets, we have defined a criterium of preference between possibilistic answer sets — of course that this criterium is also applied to possibilistic pstable models. For managing the non-existence of possibilistic answer set (or possibilistic pstable models) of a possibilistic logic program P, we have adopted the approach suggested by Nicolas *et al.*, in (Nicolas et al. 2006) of cuts for getting subprograms of P which are consistent.

It worth to comment that in some cases, it is possible to apply α-cuts in order to avoid inconsistent possibilistic answers. For instance, let P be the following possibilistic program:

$$\begin{array}{ll} 0.9: & a \leftarrow \top. \\ 0.9: & \neg a \leftarrow \top. \\ 0.8: & b \leftarrow \top. \end{array}$$

We can see that $ConsCutDeg(P) = 0.9$; hence, if we apply a strict α-cut to P, we will get an empty program. On the other hand, if we allow an inconsistent possibilistic answer set, we get $\{(a, 0.9), (\neg a, 0.9), (b, 0.8)\}$. As one can see, $0.8 : b \leftarrow \top$ is not involved in the inconsistency of P. Hence, it is not necessary to loss this information. We believe that an inconsistent possibilistic answer set could be more informative answer than a null-possibilistic answer set for an expert.

6 Related Work

Logic programming with uncertainty is an extensively research area. In fact, it has proceeded along various research lines of logic logic programming. An interesting historical recollection in this topic was recently presented by V. S. Subrahmanian in (Subrahmanian 2007). In this recollection he highlights some phases in the evolution of the topic from the viewpoint of a committed researcher.

Research on logic programming with uncertainty has dealt with various approaches of logic programming semantics, as well as different applications. Most of the approaches in the literature employ one of the following formalisms:

- annotated logic programming, *e.g.*, (Kifer and Subrahmanian 1992).
- probabilistic logic, *e.g.*, (Ng and Subrahmanian 1992; Lukasiewicz 1998; Kern-Isberner and Lukasiewicz 2004).
- fuzzy set theory, *e.g.*, (van Emden 1986; Rodríguez-Artalejo and Romero-Dáz 2008; Nieuwenborgh et al. 2007).
- multi-valued logic, *e.g.*, (Fitting 1991; Lakshmanan 1994).
- evidence theoretic logic programming, *e.g.*, (Baldwin 1987).
- possibilistic logic, *e.g.*, (Dubois et al. 1991; Alsinet and Godo 2002; Alsinet and Godo 2000; Alsinet et al. 2008; Nicolas et al. 2006).

Basically, these approaches differ in the underlying notion of uncertainty and how uncertainty values, associated to clauses and facts, are managed.

As stated on §1, we are interested on modeling quantitative expressions such that these

expressions could capture the available information especially when this information is incomplete, uncertain and inconsistent. As far of this paper we have defined a logic programming approach with uncertainty which captures uncertain values by considering complete lattices. The use of lattices for capturing uncertain values is not new, maybe one of the most influential approach in this context was suggest by Fitting in (Fitting 1991). In (Fitting 1991), Fitting showed that interlaced bilattices provide a simple and elegant setting for the consideration of logic programming extensions allowing for incomplete or contradictory answers. On the theoretical level he showed that his approach is a considerable unification of several approaches.

An interesting observation of Fitting is that in the abstract level all interlaced bilattices are quite natural; however not all are appropriate for computer implementation. By Proposition 4.1 of (Fitting 1991), we know that given two complete lattices C and B, $\mathcal{B}(C, D)$ is an interlaced bilattice[10]. It is not difficult to see that essentially the semantics of a possibilisitic disjunctive logic program $P = \langle(\mathcal{Q}, \leq), N\rangle$ is defined in the domain of the interlaced bilattice $\mathcal{B}(\{0, 1\}, \mathcal{Q})$. Since the possibilisitic semantics defined in this paper are computable, our approach is restricted to computable interlaced bilattices. Observe that by considering a complete lattice Q' different to $\{0, 1\}$, we can explore new logic programming semantics for our approach by considering multi-valued logics defined under Q' and Fitting's approach. Of course that this issue requires a deep analysis to understand how Fitting's approach and our approach are related. It is worth to comment that in (Alsinet and Godo 2000), a possibilistic logic programming approach is defined over the many-valued $G\ddot{o}del$ logic. The syntax of this approach is restricted to a Horn-clause sublanguage of the many-valued $G\ddot{o}del$ logic; hence it is unable to capture default negation and disjunctive clauses.

To prioritize logic clauses, as it is done in our possibilistic approach, can be also regarded as a preference relation between rules. In fact, by considering the certainty degrees as *preferences*, it was defined two criteria for restoring inconsistent possibilistic knowledge bases in §5. Observe that these criteria are based on the notion of maximal consistent subsets of premises. In other words, we try to recover the maximal consistent subset of possibilistic clauses from an inconsistent possibilistic program to infer consistent information. The use of *qualitative preferences* in logic programming has been suggested by authors as G. Brewka in (Brewka 2004). The Brewka's approach is also motivated from the fact that a variety of applications numerical information is hard to obtain. To have a correct understanding of the relationship between Brewka's approach and our approach requires a deep analysis.

7 Conclusions and future work

The most common forms for modeling knowledge are based on symbolic logic. Even thought, the diversity of formal languages is wide and the question of how to model uncertain information has caused much heated debate. Maybe, the most common form of representing uncertain information is based on probability theory (Halpern 2005). In fact, we can

[10] See (Fitting 1991) for details.

find successful approaches based on probability theory as Bayesian Networks. However, there are several authors which disagree with probability theory for modeling uncertain information.

- McCarthy and Hayes in (McCarthy and Hayes 1969) pointed out that attaching probabilities to a statement has objections. For example, they say that

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

- Halpern has remarked in (Halpern 2005) 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.
- Dubois and Prade in (Dubois and Prade 2004) have pointed that there are at least three worth noticing difficulties when casting the probability calculus into a logic framework for handling uncertain information.

Since probability has its problems for modeling uncertain information, it is not surprising that many other approaches of uncertainty have been considered in computer science literature.

In the mid-1980s, it was introduced a logic framework called *Possibilistic Logic* (Dubois et al. 1994). Possibilistic logic is a logic of uncertainty tailored for reasoning under incomplete evidence and partially inconsistent knowledge. In this approach all the formulæ are attached by degrees of uncertain. These degrees are quantifications of necessity or possibility of the corresponding possibilistic formulæ. At the mathematical level, degrees of possibility and necessity are closely related to fuzzy set and, possibilistic logic is especially well adapted to automated reasoning when the available information is pervaded with vagueness. In general terms, we can say that possibilistic logic is a tool for reasoning under uncertainty based on the idea of ordering rather than counting, on the contrary to probabilistic logic (Dubois et al. 1994).

An important feature of possibilistic logic is that the degrees of uncertainty of a possibilistic formula do not belong necessarily to a totally ordered set. This feature allows to explore a diversity of uncertain degrees *e.g.non-numerical uncertain degrees*. In psychology literature, we can find significant observations which are worth mentioning when we are designing an approach for modeling uncertain information. Tversky and Kahneman have observed in (Tversky and Kahneman 1982) [11], that 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 appeal to our experience or our commonsense. It is not surprising to think that a reasoning based on these kind of statements could reach to *biased conclusions*. However, these conclusions could reflect an expert's experience or commonsense. Pelletier

[11] It is worth mentioning that Kahneman (an author of (Kahneman et al. 1982)) is the winner of the 2002 Nobel Prize in Economics for having integrated insights from psychological research into economic science, especially concerning human judgment and decision-making under uncertainty

and Elio pointed out in (Pelletier and Elio 2002) 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.

In this paper, we introduced a possibilistic disjunctive logic programming approach. 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 particular, we have defined three approaches for capturing the semantics of the possibilistic disjunctive programs:

- the first is strictly close to the proof theory of possibilistic logic and *answer set models*;
- the second is based on *partial evaluation* and a fix-point operator; and
- the last is also based on the proof theory of possibilistic logic and *pstable semantics*.

Based on the flexibility of possibilistic logic for defining degrees of uncertainty, we have illustrated in this paper that it is possible to consider non-numerical degrees for capturing uncertain information. In particular, we have discussed the use of non-numerical degrees of uncertainty in a medical scenario.

For managing the inconsistency of possibilistic models, we have defined a criterium of preference between possibilistic answer sets. Also, for managing the non-existence of possibilistic answer set (or possibilistic pstable models) of a possibilistic logic program P, we have adopted the approach suggested by Nicolas *et al.*, in (Nicolas et al. 2006) of cuts for getting subprograms of P which are consistent.

As part of our future work, we have considered to define an extension of our possibilistic approach in order to deal reasoning about actions under uncertainty. In fact in (Nieves et al. 2007), we have already defined our first ideas in order to define an action language which is called $\mathcal{A}^{Poss}$.

References

ALSINET, T., CHESÑEVAR, C. I., GODO, L., AND SIMARI, G. R. 2008. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems 159,* 10, 1208–1228.

ALSINET, T. AND GODO, L. 2000. A Ccomplete Calculus for Possibilistic Logic Programming with Fuzzy Propositional Variable. In *Proceedings of the Sixteen Conference on Uncertainty in Artificial Intelligence*. ACM Press, 1-10.

ALSINET, T. AND GODO, L. 2002. Towards an automated deduction system for first-order possibilistic logic programming with fuzzy constants. *Int. J. Intell. Syst. 17,* 9, 887–924.

BALDWIN, J. F. 1987. Evidential support logic programming. *Fuzzy Sets and Systems 24,* 1 (October), 1–26.

BARAL, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge.

BRASS, S. AND DIX, J. 1997. Characterizations of the disjunctive stable semantics by partial evaluation. *J. Log. Program. 32,* 3, 207–228.

BRASS, S. AND DIX, J. 1998. Characterizations of the disjunctive well-founded semantics: Confluent calculi and iterated gcwa. *J. Autom. Reasoning 20,* 1, 143–165.

BRASS, S. AND DIX, J. 1999. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming 38(3)*, 167–213.

BREWKA, G. 2004. Answer sets: From constraint programming towards qualitative optimization. In *Logic Programming and Nonmonotonic Reasoning, 7th International Conference, LPNMR 2004, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings*, V. Lifschitz and I. Niemelä, Eds. Lecture Notes in Computer Science, vol. 2923. Springer, 34–46.

BUENO, O. 2006. *Knowledge and Inquiry : Essays on the Pragmatism of Isaac Levi*. Cambridge Studies in Probability, Induction and Decision Theory. CAMBRIDGE UNIVERSITY PRESS, Chapter Why Inconsistency Is Not Hell: Making Room for Inconsistency in Science, 70–86.

DAVEY, B. A. AND PRIESTLY, H. A. 2002. *Introduction to Lattices and Order*, Second ed. Cambridge University Press.

DIX, J. 1995a. A classification theory of semantics of normal logic programs: I. strong properties. *Fundam. Inform. 22,* 3, 227–255.

DIX, J. 1995b. A classification theory of semantics of normal logic programs: II. weak properties. *Fundam. Inform. 22,* 3, 257–288.

DIX, J., OSORIO, M., AND ZEPEDA, C. 2001. A general theory of confluent rewriting systems for logic programming and its applications. *Ann. Pure Appl. Logic 108,* 1-3, 153–188.

DLV, S. 1996. Vienna University of Technology. http://www.dbai.tuwien.ac.at/proj/dlv/.

DUBOIS, D., LANG, J., AND PRAD, H. 1991. Towards possibilistic logic programming. In *ICLP*, K. Furukawa, Ed. The MIT Press, 581–595.

DUBOIS, D., LANG, J., AND PRADE, H. 1994. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, D. Gabbay, C. J. Hogger, and J. A. Robinson, Eds. Oxford University Press, Oxford, 439–513.

DUBOIS, D. AND PRADE, H. 2004. Possibilistic logic: a retrospective and prospective view. *Fuzzy Sets and Systems 144,* 1, 3–23.

FITTING, M. 1991. Bilattices and the semantics of logic programming. *Journal of Logic Programming 11,* 1&2, 91–116.

FOX, J. AND MODGIL, S. 2006. From arguments to decisions: Extending the Toulmin view. In *Arguing on the Toulmin model: New essays on argument analysis and evaluation*, D. Hitchcock and B. Verheij, Eds. Springer Netherlands, 273–287.

GELFOND, M. 2008. *Handbook of Knowledge Representation*. Elsevier, Chapter Answer Sets, 285–316.

GELFOND, M. AND LIFSCHITZ, V. 1988. The Stable Model Semantics for Logic Programming. In *5th Conference on Logic Programming*, R. Kowalski and K. Bowen, Eds. MIT Press, 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing 9*, 365–385.

HALPERN, J. Y. 2005. *Reasoning about uncertainty*. The MIT Press.

HOPCROFT, J. E., MOTWANI, R., AND ULLMAN, J. D. 2007. *Introduction to Automata Theory, Languages and Computation, 3/E*. Addison Wesley Higher Education.

KAHNEMAN, D., SLOVIC, P., AND TVERSKY, A. 1982. *Judgment under uncertainty:Heuristics and biases*. Cambridge Univertisy Press.

KERN-ISBERNER, G. AND LUKASIEWICZ, T. 2004. Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence 157,* 1-2, 139–202.

KIFER, M. AND SUBRAHMANIAN, V. S. 1992. Theory of generalized annotated logic programming and its applications. *J. Log. Program. 12,* 3&4, 335–367.

LAKSHMANAN, L. V. S. 1994. An epistemic foundation for logic programming with uncertainty. In *FSTTCS*. 89–100.

LÓPEZ, A. 2006. Implementing pstable. In *Workshop in Logic, Language and Computation*, R. Dávila, M. Osorio, and C. Zepeda, Eds. Vol. 220. CEUR Workshop Proceedings.

LÓPEZ-NAVIDAD, A. AND CABALLERO, F. 2003. Extended criteria for organ acceptance: Strategies for achieving organ safety and for increasing organ pool. *Clinical Transplantation, Blackwell Munksgaard 17*, 308–324.

LÓPEZ-NAVIDAD, A., DOMINGO, P., AND VIEDMA, M. A. 1997. Professional characteristics of the transplant coordinator. In *XVI International Congress of the Transplantation Society*. Transplantation Proceedings, vol. 29. Elsevier Science Inc, 1607–1613.

LUKASIEWICZ, T. 1998. Probabilistic logic programming. In *ECAI*. 388–392.

MCCARTHY, J. AND HAYES, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, 463–502. reprinted in McC90.

NG, R. T. AND SUBRAHMANIAN, V. S. 1992. Probabilistic logic programming. *Inf. Comput. 101,* 2, 150–201.

NICOLAS, P., GARCIA, L., STÉPHAN, I., AND LAFÈVRE, C. 2006. Possibilistic Uncertainty Handling for Answer Set Programming. *Annal of Mathematics and Artificial Intelligence 47,* 1-2 (June), 139–181.

NIEUWENBORGH, D. V., COCK, M. D., AND VERMEIR, D. 2007. An introduction to fuzzy answer set programming. *Ann. Math. Artif. Intell. 50,* 3-4, 363–388.

NIEVES, J. C., OSORIO, M., CORTÉS, U., CABALLERO, F., AND LÓPEZ-NAVIDAD, A. 2007. Reasoning about actions under uncertainty: A possibilistic approach. In *In proceedings of CCIA*, C. Angulo and L. Godo, Eds.

OSORIO, M., ARRAZOLA, J. R., AND CARBALLIDO, J. L. 2008. Logical Weak Completions of Paraconsistent Logics. *Journal of Logic and Computation doi: 10.1093/logcom/exn015*.

OSORIO, M., NAVARRO, J. A., ARRAZOLA, J. R., AND BORJA, V. 2005. Ground Nonmonotonic Modal Logic S5: New Results. *Journal of Logic and Computation 15,* 5, 787–813.

OSORIO, M., NAVARRO, J. A., ARRAZOLA, J. R., AND BORJA, V. 2006. Logics with Common Weak Completions. *Journal of Logic and Computation 16,* 6, 867–890.

PELLETIER, F. J. AND ELIO, R. 2002. *Scope of Logic, Methodology and Philosophy of Science*. Synthese Library, vol. 1. Dordrecht: Kluwer Academic Press, Chapter Logic and Computation, 137–156.

RODRÍGUEZ-ARTALEJO, M. AND ROMERO-DÁZ, C. A. 2008. Quantitative Logic Programming revisited. In *9th International Symposium, FLOPS*, J. Garrigue and M. Hermenegildo, Eds. LNCS, vol. 4989. Springer-Verlag Berlin Heidelberg, 272–288.

SUBRAHMANIAN, V. S. 2007. Uncertainty in logic programming. *Association for Logic Programming (ALP), Newsletter 20,* 2 (May/June).

TARSKI, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics 5,* 2, 285–309.

TVERSKY, A. AND KAHNEMAN, D. 1982. *Judgment under uncertainty:Heuristics and biases*. Cambridge Univertisy Press, Chapter Judgment under uncertainty:Heuristics and biases, 3–20.

VAN DALEN, D. 1994. *Logic and structure*, 3rd., aumented edition ed. Springer-Verlag, Berlin.

VAN EMDEN, M. H. 1986. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming 3,* 1, 37–53.

Appendix: Proofs

In this annex we give the proof of most of the results given in this paper.

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

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

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

Proof
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) $(\varphi\ \mathcal{TOP}_{\mathcal{Q}}), (\varphi \rightarrow \psi\ \mathcal{TOP}_{\mathcal{Q}}) \vdash_{PL} (\psi\ \mathcal{TOP}_{\mathcal{Q}})$, then any formula inferred from P by GMP will have $\mathcal{TOP}_{\mathcal{Q}}$ as necessity-value. Then, if M' is an answer set of P^*, then $\{(l, \mathcal{TOP}_{\mathcal{Q}}) | l \in M'\}$ will be a possibilistic answer set of P. □

Proposition 4 *Let $P := \langle(Q, \leq), N\rangle$ be a possibilistic normal program such that $(Q, \leq)$ is a totally ordered set and $\mathcal{L}_P$ has no extended atoms. 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 8 and the syntactic reduction of Definition 10 from (Nicolas et al. 2006) coincide. Then the proof 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. □

Proposition 5 *Let $\mathcal{C}$ be a set of possibilistic disjunctions, and $C = (c\ \alpha)$ be a possibilistic clause obtained by a finite number of successive application of* (R) *to $\mathcal{C}$; then $\mathcal{C} \vdash_{PL} C$.*

Proof
(The proof is similar to the proof of Proposition 3.8.2 of (Dubois et al. 1994)) Let us consider two possibilistic clauses: $C_1 = (c_1\ \alpha_1)$ and $C_2 = (c_2\ \alpha_2)$, the application of R yields $C' = (R(c_1, c_2)\ \mathcal{GLB}(\{\alpha_1, \alpha_2\}))$. By classic logic, we known that $R(c_1, c_2)$ is sound; hence the key point of the proof is to show that $n(R(c_1, c_2)) \geq \mathcal{GLB}(\{\alpha_1, \alpha_2\})$.

By definition of necessity-valued clause, $n(c_1) \geq \alpha_1$ and $n(c_2) \geq \alpha_2$, then $n(c_1 \wedge c_2) = \mathcal{GLB}(\{n(c_1), n(c_2)\}) \geq \mathcal{GLB}(\{\alpha_1, \alpha_2\})$. Since $c_1 \wedge c_2 \vdash_C R(c_1, c_2)$, then $n(R(c_1, c_2)) \geq n(c_1 \wedge c_2)$ (because if $\varphi \vdash_{PL} \psi$ then $N(\psi) \geq N(\varphi)$). Thus $n(R(c_1, c_2)) \geq \mathcal{GLB}(\{\alpha_1, \alpha_2\})$; therefore (R) is sound. Then by induction any possibilistic formula inferred by a finite number of successive applications of (R) to $\mathcal{C}$ is a logical consequence of $\mathcal{C}$. □

Proposition 6 *Let P be a set of possibilistic clauses and $\mathcal{C}$ be the set of possibilistic disjunctions obtained from P; then the valuation of the optimal refutation by resolution from $\mathcal{C}$ is the inconsistent degree of P.*

Proof
(The proof is similar to the proof of Proposition 3.8.3 of (Dubois et al. 1994)) By possibilistic logic, we know that $\mathcal{C} \vdash_{PL} (\bot\ \alpha)$ if and only if $(\mathcal{C}_\alpha)^*$ is inconsistent in the sense of classic logic. Since (R) is complete in classic logic, then there exists a refutation $R(\Box)$ from $(\mathcal{C}_\alpha)^*$. Thus considering the valuation of the refutation $R(\Box)$, we obtain a refutation from $\mathcal{C}_\alpha$ such that $n(R(\Box)) \geq \alpha$. Then $n(R(\Box)) \geq Inc(\mathcal{C})$. Since (R) is sound then $n(R(\Box))$ cannot be strictly greater than $Inc(\mathcal{C})$. Thus $n(R(\Box))$ is equal to $Inc(\mathcal{C})$. According to Proposition 3.8.1 of (Dubois et al. 1994), $Inc(\mathcal{C}) = Inc(P)$, thus $n(R(\Box))$ is also equal to $Inc(P)$. $\Box$

Proposition 7 *Let $P := \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic logic program. The set Poss-ASP returned by $Poss\_Answer\_Sets(P)$ is the set of all the possibilistic answer sets of P.*

Proof
The result follows from the following facts:

1. The function ASP computes all the answer set of P^*.
2. If M is a possibilistic answer set of P iff M^* is an answer set of P^* (Proposition 2).
3. By Corollary 1, we know that the possibilistic resolution rule R is sound and complete for computing optimal possibilistic degrees.

$\Box$

Proposition 8 *Let P be a possibilistic disjunctive logic program. If $\Gamma_0 := \mathcal{T}(P)$ and $\Gamma_i := \mathcal{T}(\Gamma_{i-1})$ such that $i \in \mathcal{N}$, then $\exists\ n \in \mathcal{N}$ such that $\Gamma_n = \Gamma_{n-1}$. We denote Γ_n by $\Pi(P)$.*

Proof
It is not difficult to see that the operator $\mathcal{T}$ is monotonic, then the proof is direct by Tarski's Lattice-Theoretical Fixpoint Theorem (Tarski 1955). $\Box$

Proposition 9 *Let P be a possibilistic disjunctive logic program and M a set of possibilistic atoms. M is a possibilistic answer set of P if and only if M is a possibilistic-$\mathcal{T}$ answer set of P.*

Proof
Two observations:

1. By definition, it is straightforward that if M_1 is a possibilistic answer set of P, then there exists a possibilistic-$\mathcal{T}$ answer set M_2 of P such that $M_1^* = M_2^*$ and viceversa.
2. Since G-GPPE can be regarded as a macro of the possibilistic rule (R), we can conclude by Proposition 5 that G-GPPE is sound.

Let M_1 be a possibilistic answer set of P and M_2 be a possibilistic-$\mathcal{T}$ answer set of P. By Observation 1, the central point of the proof is to prove that if $(a, \alpha_1) \in M_1$ and $(a, \alpha_2) \in M_2$ such that $M_1^* = M_2^*$, then $\alpha_1 = \alpha_2$.

The proof is by contradiction. Let us suppose that $(a, \alpha_1) \in M_1$ and $(a, \alpha_2) \in M_2$ such that $M_1^* = M_2^*$ and $\alpha_1 \neq \alpha_2$. Then there are two cases $\alpha_1 < \alpha_2$ or $\alpha_1 > \alpha_2$

$\alpha_1 < \alpha_2$: Since G-GPPE is sound (Observation 2), then α_1 is not the optimal necessity-value for the atom a, but this is false by Corollary 1.

$\alpha_1 > \alpha_2$: If $\alpha_1 > \alpha_2$ then there exists a possibilistic claus $\alpha_1 : \mathcal{A} \leftarrow \mathcal{B}^+ \in P^{(M_1)^*}$ that belongs to the optimal refutation of the atom a and it was not reduced by G-GPPE. But this is false because G-GPPE is a macro of the resolution rule (R).

□

Proposition 10 *Let P be a possibilistic normal program. If M is a possibilistic answer set of P, then the following conditions hold:*

a) *M^* is a pstable model of P^*.*

b) *there exists a possibilistic pstable mode M' of P such that $M \sqsubseteq M'$ and $M^* = M'^*$.*

Proof

a) The proof is straightforward by Theorem 4.4 of (Osorio et al. 2006) (The Theorem 4.4 of (Osorio et al. 2006) says that given a normal logic program P and a set of atoms M, if M is an answer set of P then M is a pstable model of P).

b) First of all observe that the following relation is true:

$$P^{M^*} \subseteq PRED(P, M) \tag{4}$$

By a), it is direct that if M is a possibilistic answer set of P, then there exists a possibilistic pstable model M' such that $M^* = M'^*$. Hence, if $(a, \alpha_1) \in M$, then $a \in M'^*$ such that $(a, \alpha_2) \in M'$. Therefore, the relevant part of prove is to show that $\alpha_1 \leq \alpha_2$.
The proof is by contradiction: Let us suppose that $\alpha_2 < \alpha_1$, by definition of possibilistic answer set and pstable model, $P^{M^*} \vdash_{PL} (a, \alpha_1)$ and $PRED(P, M) \vdash_{PL} (a, \alpha_2)$ such that α_1 and α_2 are optimal. Since $P^{M^*} \subseteq PRED(P, M)$, hence then $PRED(P, M) \vdash_{PL} (a, \alpha_1)$. Therefore, α_2 is not the optimal value of a *w.r.t.* $PRED(P, M)$. This is a contradiction, because α_2 is the optimal value of a *w.r.t.* $PRED(P, M)$ by definition of the possibilistic pstable semantics.

□

Proposition 11 *Let P be a possibilistic normal program. If $P \vdash_{PL} (x\ \alpha)$ then P is equivalent to $P \cup \{(x\ \alpha)\}$ under the possibilistic pstable semantics.*

Proof

Some observations;

a) By Theorem 7.11 of (Osorio et al. 2005) and Theorem 5.1 of (Osorio et al. 2006), we can see that: If $P^* \vdash_C x$ then P^* is equivalent to $P^* \cup \{x\}$ under the pstable semantics *i.e.*$Pstable(P^*) = Pstable(P^* \cup \{x\})$

b) By definition of the possibilistic pstable semantics: M is a possibilistic pstable model of P then M^* is a pstable model of P^*.

c) By definition of the syntactic reduction $PRED$, it is easy to see that: Given a possibilistic normal program P and a set of atoms M : $PRED(P \cup (a, \alpha), M) = PRED(P, M) \cup \{(a, \alpha)\}$.

d) In (Dubois et al. 1994), it was proved that: $P \vdash_{PL} (x\ \alpha)$ iff $P_\alpha \vdash_{PL} (x\ \alpha)$.

We use $poss\_Pstable$ to denote the semantics operator of the possibilistic pstable semantics. Then we have to prove that

$$poss\_Pstable(P) = poss\_Pstable(P \cup \{(x\ \alpha)\})$$

$=>$ We have to prove that if $M \in poss\_Pstable(P)$ then $M \in poss\_Pstable(P \cup \{(x\ \alpha)\})$.
Proof: $M \in poss\_Pstable(P)$ iff $M^* \in Pstable(P^*)$ (by b) iff $M^* \in Pstable(P^* \cup \{x\})$ (by a). Hence, there exists $M' \in poss\_Pstable(P \cup \{(x\ \alpha)\})$ such that $M^* = M'^*$.
Let us suppose that $M \neq M'$, this means that there exists $(a, \alpha_1) \in M$ and $(a, \alpha_2) \in M'$ such that $\alpha_1 \neq \alpha_2$.
If $\alpha_1 \neq \alpha_2$, then there two cases:

- $\alpha_1 > \alpha_2$**:** If $\alpha_1 > \alpha_2$, then $PRED(P, M^*)_{\alpha_1} \subset PRED(P \cup \{(a, \alpha)\}, M'^*)_{\alpha_2}$ (remember that $PRED(P \cup (a, \alpha), M) = PRED(P, M) \cup \{(a, \alpha)\}$). Since $PRED(P, M^*)_{\alpha_1} \vdash_{PL} (a\ \alpha_1)$ and $PRED(P \cup (a, \alpha), M'^*)_{\alpha_2} \vdash_{PL} (a\ \alpha_2)$, hence α_2 is not the optimal necessity value of a inferred from $PRED(P \cup \{(a, \alpha)\}, M'^*)_{\alpha_2}$. This is a contradiction, because M' is a possibilistic Pstable model of $P \cup (a, \alpha)$.
- $\alpha_1 < \alpha_2$**:** If $\alpha_1 < \alpha_2$, then $PRED(P, M'^*)_{\alpha_2} \subset PRED(P, M^*)_{\alpha_1}$ and $\alpha < \alpha_1$. Hence $\alpha_2 = \alpha$. Then $P_{\alpha_2} \subset P_{\alpha_1}$. Since $P_{\alpha_2} \vdash_{PL} (x\ \alpha_2)$, then $P_{\alpha_2} \vdash_{PL} (a\ \alpha_2)$. Therefore $P_{\alpha_1} \vdash_{PL} (a\ \alpha_2)$. Then α_1 is not the optimal necessity value for a inferred from $PRED(P, M)_{\alpha_1}$. This is a contradiction, because M is a possibilistic Pstable model of P.

Therefore $\alpha_1 = \alpha_2$, this means that $M = M'$.
$<=$ We have to prove that if $M \in poss\_Pstable(P \cup \{(x\ \alpha)\})$ then $M \in poss\_Pstable(P)$.
The proof is similar to the previous case.

□

Theorem 1 *Let P be a possibilistic disjunctive program. If M is a possibilistic answer set of P, then it implies that*

a) *M^* is a pstable model of $TRAD(P)^*$.*
b) *there exists a possibilistic pstable mode M' of $TRAD(P)$ such that $M \sqsubseteq M'$ and $M^* = M'^*$.*

Proof
a) The proof is straightforward by Theorem 5.3 of (Osorio et al. 2008) (The Theorem 5.3 of (Osorio et al. 2008) says that if M is an answer set of P^* then M is a pstable model of $(TRAD(P))^*$. The authors of (Osorio et al. 2008) use the concept of "closed under d-shift"; but this concept is nothing else that the consideration of the mapping $TRAD$ without the possibilistic values).
b) Direct by a) and Proposition 10.

□

Proposition 12 *Given a possibilistic program $P := \langle (\mathcal{Q}, \leq), N \rangle$ there exists an algorithm that computes the set of possibilistic pstable models of P.*

Proof
The algorithm is the same to the algorithm presented in the proof of Proposition 7. The only difference is that instead of using an algorithm for computing the answer sets of P^*, it is used an algorithm for computing the pstable models of P^* *e.g.*, the algorithm presented in (López 2006). □