PStable semantics for possibilistic logic programs

Mauricio Osorio[1] and Juan Carlos Nieves[2]

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

Abstract. Uncertain information is present in many real applications *e.g.,* medical domain, weather forecast, *etc*. The most common approaches for leading with this information are based on probability however some times; it is difficult to find suitable probabilities about some events. In this paper, we present a possibilistic logic programming approach which is based on possibilistic logic and PStable semantics. Possibilistic logic is a logic of uncertainty tailored for reasoning under incomplete evidence and Pstable Semantics is a solid semantics which emerges from the fusion of non-monotonic reasoning and logic programming; moreover it is able to express answer set semantics, and has strong connections with paraconsistent logics.

1 Introduction

To find a representation of the information under uncertainty has been subject of much debate. For those steeped in probability, there is only one appropriate model for numeric uncertainty, and that is probability. But 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 probability: either one event is more probable than the other, or they have equal probability [4]. In fact, in [7], McCarthy and Hayes pointed out that attaching probabilities to a statement has some objections. For instance:

> The information necessary to assign numerical probabilities is not ordinary available. Therefore, a formalism that required numerical probabilities would be epistemologically inadequate [7].

Hence it is not surprising that many other representations of uncertainty have been considered in the literature. For instance in the MYCIN project which is one of the clearest representatives of the experimental side of Artificial Intelligence (IA) was shown that probability theory has limitations for developing automated assistance for medical diagnosis [2]. In this project, it was adopted a less formal model. This model uses estimates provided by expert physicians that reflect the tendency of a piece of evidence to prove or disprove a hypothesis. The syntax adopted by MYCIN was based on IF-THEN rules with certainty factors. The following is an English version of one of MYCIN's rules:

IF the infection is primary-bacteremia (**a**)
AND the site of the culture is one of the sterile sites (**b**)
AND the suspected portal of entry is the gastrointestinal tract (**c**)
THEN there is suggestive evidence (0.7) that infection is bacteroid (**d**).

The 0.7 is roughly the certainty that the conclusion will be true given the evidence. If the evidence is uncertain the certainties of the bits of evidence will be combined with the certainty of the rule to give the certainty of the conclusion.

John McCarthy pointed out in his seminal paper [6] that the MYCIN's major innovation over many previous expert systems was that it uses measures of uncertainty (not probabilities) for its diagnoses and the fact that it is prepared to explain its reasoning to the physician. We can say that MYCIN was one of the most successful projects at its time; however, it seems that AI's community has taken few lessons from MYCIN's experience for developing new intelligence support systems.

One of the main problems of MYCIN was that it developed a monotonic reasoning about its diagnoses. Then to update the medical knowledge in order to improve its diagnoses was a problem. Nowadays, logic programming and non-monotonic reasoning are solid areas in AI. For instance, during the last two decades, one of the most successful logic programming approaches has been Answer Set Programming (ASP). ASP is the realization of much theoretical work on Non-monotonic Reasoning and Artificial Intelligence applications. It represents a new paradigm for logic programming that allows, using the concept of *negation as failure*, to handle problems with default knowledge and produce non-monotonic reasoning [1].

In [9], it was proposed a possibilistic logic programming framework for reasoning under uncertainty. It is a combination between Answer Set Programming (ASP) [1] and Possibilistic Logic [3]. This framework is able to deal with reasoning that is at the same time non-monotonic and uncertain. Since this framework was defined for normal programs, it was generalized in [10] for capturing possibilistic disjunctive programs and allowing the encoding of uncertain information by using either numerical values or relative likelihoods.

We can accept that the language's expressiveness of the Possibilistic Answer Set Programming approach (PASP) presented in [9, 10] is rich enough for capturing a wide family of problems where one have to confront with incomplete information and uncertain information. For instance, the MYCIN's rule presented previously can be expressed as follows:

$$07 : d \leftarrow a, b, c.$$

where a, b, c, d are propositional atoms whose intended meanings are described in the previous MYCIN's rule.

PASP could be considered as a good option for developing new intelligence support systems like MYCIN system such that the support systems could perform non-monotonic reasoning. However it is obvious that an intelligence support system always must have an answer to any query to its knowledge base. Since PASP was defined as an ASP's extension, there are some possibilistic logic programs which have no possibilistic answer sets. For instance, a single possibilistic claus as $\alpha : a \leftarrow not\ a$ does

not have a possibilistic answer set. In fact, the existence of one clause of this form will affect all the possibilistic knowledge base such that all the possibilistic knowledge base will not have a possibilistic answer set. It is quite obvious that this situation could not be permitted in an intelligence support system like MYCIN.

In this paper, we define a possibilistic logic programming semantics called *possibilistic pstable semantics*. This semantics is based on pstable semantics [11, 12]. Pstable semantics emerges from the fusion of paraconsistent logics and ASP. This semantics is able to capture ASP; moreover it is less sensitive than the answer set semantics.

Like in possibilistic answer set semantics, possibilistic pstable semantics is based on possibilistic logic. Possibilistic logic is a type of logic of uncertainty tailored for reasoning under incomplete evidence and partially inconsistent knowledge. At the syntactic level it handles formulae of propositional or first-order logic to which are attached degrees of necessity. The degree of necessity (or certainty) of a formula expresses to what extent the available evidence entails the truth of this formula [3]. We argue that possibilistic logic is an excellent approximation of the approach adopted by MYCIN system which showed that it is practical in real applications.

It is worth mentioning that possibilistic pstable semantics is close related to possibilistic logic. For instance, it has the property that given a possibilistic logic program P P, if $P \vdash_{PL} (x\ \alpha)$ then P is equivalent to $P \cup \{(x\ \alpha)\}$ under the possibilistic pstable semantics.

The rest of the paper is divided as follows: In §2, some basic definitions of possibilistic logic and pstable semantics are presented. In §3, the syntax of our possibilistic framework is presented. In §4, the possibilistic pstable semantics is defined. Finally in the last section, we present our conclusions.

2 Background

In this section, we define some basic concepts of Possibilistic Logic and Pstable models. We assume familiarity with basic concepts in classic logic and in semantics of logic programs *e.g.,* interpretation, model, *etc*. A good introductory treatment of these concepts can be found in [1, 8].

2.1 Possibilistic Logic

A necessity-valued formula is a pair $(\varphi\ \alpha)$ where φ is a classical logic formula and $\alpha \in (0, 1]$ is a positive number. The pair $(\varphi\ \alpha)$ expresses that the formula φ is certain at least to the level α, *i.e.* $N(\varphi) \geq \alpha$, where N is a necessity measure modeling our possibly incomplete state knowledge [3]. α is not a probability (like it is in probability theory) but it induces a certainty (or confidence) scale. This value is determined by the expert providing the knowledge base. A necessity-valued knowledge base is then defined as a finite set (*i.e.* a conjunction) of necessity-valued formulae.

Dubois *et al.*[3] introduced a formal system for necessity-valued logic which is based on 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) $((\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)\ 1)$

As in classic logic, the symbols $\neg$ and $\rightarrow$ are considered primitive connectives, then connectives as $\vee$ and $\wedge$ are defined as abbreviations of $\neg$ and $\rightarrow$. Now the inference rules for the axioms are:

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

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

The problem of inferring automatically the necessity-value of a classical formula from a possibilistic base was solved by an extended version of *resolution* for possibilistic logic (see [3] for details).

2.2 Syntaxis: Logic programs

The language of a propositional logic has an alphabet consisting of

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

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

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

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

where $n \geq 0$, a is an atom and each a_i is an atom. When $n = 0$ the clause is an abbreviation of a. An extended normal program P is a finite set of extended normal clauses. By $\mathcal{L}_P$, we denote the set of atoms in the language of P.

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

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

Then replacing the atom $\neg q$ by a new atom symbol q', we will have:

$$a \leftarrow q. \qquad q' \leftarrow r. \qquad q. \qquad r.$$

In order not to allow inconsistent answer sets in the ASP's programs, usually it is added a normal clause of the form $f \leftarrow q, q', f$ such that $f \notin \mathcal{L}_P$. We will omit this clause in order to allow an inconsistent level in our possibilistic pstable models. However the user could add this clause without losing generality.

Sometimes we denote an extended normal clause C by $a \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$, where $\mathcal{B}^+$ contains all the positive body literals and $\mathcal{B}^-$ contains all the negative body literals. When $\mathcal{B}^- = \emptyset$, the clause is called definite clause. A set of definite clauses is called a definite logic program.

2.3 Pstable semantics

First to definite pstable semantics, we define some basic concepts. Logic consequence in classic logic is denoted by $\vdash$. Given a set of proposition symbols S and a theory (a set of well-formed formulae) Γ, if $\Gamma \vdash S$ if and only if $\forall s \in S\ \Gamma \vdash s$. 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 classical negation in classic logic. Given a normal program P, if $M \subseteq \mathcal{L}_P$, we write $P \Vdash M$ when: $P \vdash M$ and M is a classical 2-valued model of P (*i.e.* atoms in M are set to true, and atoms not in M to false; the set of atoms is a classical model of P if the induced interpretation evaluates P to true).

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

Definition 1. *[12] Let P be a normal program and M a set of literals. We define*

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

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

$a \leftarrow\ not\ b,\ not\ c.$
$a \leftarrow b.$
$b \leftarrow a.$

We can see that $RED(P, M)$ is:
$a \leftarrow\ not\ b.$
$a \leftarrow b.$
$b \leftarrow a.$

By considering the reduction RED, it is defined the semantics *pstable* for normal programs.

Definition 2. *[12] 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.* $RED(P_1, M_1) \vdash M_1$. 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. Then we can conclude that $RED(P_1, M_1) \Vdash M_1$. Hence, M_1 is a *pstable model* of P_1.

3 Possibilistic Normal Logic Programs

In this section, we introduce our possibilistic logic programming framework. We shall start by defining the syntax of a valid program and some relevant concepts, after that we shall define the semantics for the possibilistic normal logic programs. In whole paper, we will consider finite lattices. This convention was taken based on the assumption that in real applications rarely we will have an infinite set of labels for expressing the incomplete state of a knowledge base.

3.1 Syntax

First of all, we start defining some relevant concepts[3]. A *possibilistic atom* is a pair $p = (a, q) \in \mathcal{A} \times Q$, where $\mathcal{A}$ is a finite set of atoms and $(Q, \leq)$ is a lattice. We apply the projection $*$ over p as follows: $p^* = a$. Given a set of possibilistic atoms S, we define the generalization of $*$ over S as follows: $S^* = \{p^* | p \in S\}$. Given a lattice $(Q, \leq)$ and $S \subseteq Q$, $LUB(S)$ denotes the least upper bound of S and $GLB(S)$ denotes the greatest lower bound of S. Three basic operations between sets of possibilistic atoms are formalized as follows:

Definition 3. *Let $\mathcal{A}$ be a finite set of atoms and (Q,≤) be a lattice. Consider $\mathcal{PS} = 2^{\mathcal{A}\times Q}$ the finite set of all the possibilistic atom sets induced by $\mathcal{A}$ and Q. $\forall A, B \in \mathcal{PS}$, we define.*

$A \sqcap B = \{(x, GLB\{q_1, q_2\}) | (x, q_1) \in A \wedge (x, q_2) \in B\}$

$A \sqcup B = \{(x, q) | (x, q) \in A\ and\ x \notin B^*\} \cup$
$\{(x, q) | x \notin A^*\ and\ (x, q) \in B\} \cup$
$\{(x, LUB\{q_1, q_2\}) | (x, q_1) \in A\ and\ (x, q_2) \in B\}.$

$A \sqsubseteq B \Longleftrightarrow A^* \subseteq B^*$, *and* $\forall x, q_1, q_2,$
$(x, q_1) \in A \wedge (x, q_2) \in B$ *then* $q_1 \leq q_2$.

[3] Some concepts presented in this subsection extend some terms presented in [9].

Proposition 1. *[10]* $(\mathcal{PS}, \sqsubseteq)$ *is a complete lattice.*

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

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

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

A possibilistic normal logic program P is a tuple of the form $\langle (Q, \leq), N \rangle$, where $(Q, \leq)$ is a lattice and N is a finite set of possibilistic normal clauses. The generalization of the projection $*$ over P is as follows: $P^* = \{r^* | r \in N\}$. Notice that P^* is an extended normal program. When P^* is a definite program, P is called a possibilistic definite logic program.

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

Example 1. Let us suppose that a patient suffering from certain symptoms takes a blood test, and that the results show the presence of a bacterium of a certain category in his blood. There are two types of bacteria in this category, and the blood test *does not pinpoint* whether the bacteria present in the blood is either streptococcus viridans or X. The problem is that if the bacterium is streptococcus viridans the patient have to be treated by antibiotics of large spectrum because streptococcus viridans suggests *endocarditis*. However, the doctor tries not to prescribe antibiotics of large spectrum, because they are harmful to the immune system. Then, the doctor in this case must evaluate each potential choice, where each potential choice has different levels of uncertainty[4].

In order to encode this scenario let us consider the lattice $\langle \{Open,\ Supported,\ Plausible,\ Supported,\ Probable,\ Confirmed,\ Certain\}, \preceq \rangle$, where the following relations hold: $Open \preceq Supported$, $Supported \preceq Plausible$, $Supported \preceq Probable$, $Probable \preceq Confirmed$, $Plausible \preceq Confirmed$, and $Confirmed \preceq Certain$. The elements of this lattice represent relative likelihoods which will be used for encoding the uncertainty of our scenario. Then, we could model the doctor's beliefs as follows: One doctor's belief is that it is *confirmed* that the patient has a bacterium of category n. Then, this belief could be encoded by:

$confimed : category\_n.$

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

$certain : streptoccus\_viridans \leftarrow category\_n, not\ bacterium\_x.$
$certain : bacterium\_x \leftarrow category\_n, not\ streptoccus\_viridans.$

[4] This example is an adaptation of Example 6 from [5]

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

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

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

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

One of the main doctor's belief is that he should not use antibiotics of large spectrum if it has not been established that there is not another alternative treatment.

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

We can appreciate the use of relative likelihoods could facilitate the modeling of incomplete states of a belief.

4 Pstable semantics and possibilistic programs

In this section, we define the possibilistic pstable semantics. In order to define pstable semantics for possibilistic normal programs, let us define the following single reduction based on Definition 4.

Definition 4. *Let P be a possibilistic normal program and M a set of literals. We define*

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

Let us consider the following example.

Example 2. First, let S be the set $\{(a, 0.6), (b, 0.7)\}$ and P_1 be the following possibilistic normal program where the possibilistic clauses are built under the lattice $Q := (\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}, \leq)$[5]:

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

Then, the program $PRED(P_1, M)$ is:
$0.7 : a \leftarrow not\ b.$
$0.6 : a \leftarrow b.$
$0.8 : b \leftarrow a.$

[5] $\leq$ is the traditional relation between rational numbers.

Definition 5 (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 3. Let P_1 be the possibilistic program of Example 2 and $S := \{(a, 0.6), (b, 0.7)\}$. 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 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 $PRED(P_1, M_1)$
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 $PRED(P_1, M_1)$
6. a	0.6 From 4 and 5 by GMP

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.

It is worth mentioning that the possibilistic program P_1 is an example where the possibilistic pstable semantics is different to the possibilistic stable semantics [9] and the possibilistic answer set semantics [10]. In fact, P_1 has no possibilistic stable model neither possibilistic answer set.

In order to complete Example 1, we can see that the scenario has two possibilistic pstable models:

1. $\{(category\_n, confimed), (streptoccus\_viridans, certain),$
   $(antibiotics\_large\_spectrum, certain), (\neg alternative\_treatment, plausible)\}$
2. $\{(category\_n, confimed), (bacterium\_x, certain),$
   $(alternative\_treatment, probable), (\neg antibiotics\_large\_spectrum, plausible)\}$

Notice that each one suggests a treatment depending of the bacterium. In this example the possibilistic pstable semantics coincides with the possibilistic answer semantics [10].

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

Lemma 1. *Let P be a possibilistic normal program. If M is a possibilistic answer set of P implies that M is a possibilistic pstable model of P.*

Remark 1. It is worth mentioning that when $P = \langle (Q, \leq), N \rangle$ is a possibilistic program which does not contain extended atoms *i.e.* atoms of form $\neg a$ and $(Q, \leq)$ is a totally ordered set, it will be true that: If M is a possibilistic stable model of P implies that M is a possibilistic pstable model of P.

An outstanding property of the possibilistic pstable semantics is that this semantics support 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.

Lemma 2. *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 the possibilistic answer set semantics [10] and the possibilistic stable semantics [9] do not satisfy that if $P \vdash_{PL} (x\ \alpha)$, then P is equivalent to $P \cup \{(x\ \alpha)\}$ under either the possibilistic answer set semantics or the possibilistic stable semantics. In order to show this, let us consider the single possibilistic logic program P:

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

It is clear that $P \vdash_{PL} (a\ \alpha)$. P has no possibilistic stable model neither 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 semantics introduced in [10] 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 is defined for possibilistic normal programs, one can think that the possibilistic pstable semantics is less expressive than the possibilistic answer semantics. However, an interesting result is that the possibilistic pstable semantics is the same expressive than the possibilistic answer sets. By lack of space, we will omit the formal presentation of this result.

5 Conclusions

Uncertain information is present in many real applications *e.g.,* medical domain, weather forecast, *etc*. To find a suitable representation of this kind of information has been subject of much debate. The most common approaches for leading with this information are based on probability however some times; it is difficult to find suitable probabilities about some events.

According to the experimental side of Artificial Intelligence, it seem that a good form of capturing uncertain information is to adopted models where they could be close to the common sense of an expert of an area. For instance, MYCIN project [2] used estimates provided by expert physicians that reflect the tendency of a piece of evidence to prove or disprove a hypothesis.

Possibilistic logic is a type of logic of uncertainty tailored for reasoning under incomplete evidence and partially inconsistent knowledge. At the syntactic level it handles formulae of propositional or first-order logic to which are attached degrees of necessity. The degree of necessity (or certainty) of a formula expresses to what extent

the available evidence entails the truth of this formula [3]. We argue that possibilistic logic is an excellent approximation of the approach adopted by MYCIN system which showed that it is practical in real applications.

In this paper, we introduce a new possibilistic logic programming semantics called possibilistic pstable semantics. This semantics is closer to possibilistic logic than the two possibilistic semantics defined until now [9, 10]. Since the possibilistic pstable semantics is less syntactic sensitive than the possibilistic stable semantics [9] and the possibilistic answer set semantics [10], this semantics guaranties the existences of possibilistic pstable models. It is worth mentioning that since the possibilistic pstable semantics is based on pstable semantics which emerges from the fusion of paraconsistent logics ans ASP, the possibilistic pstable semantics is influenced by paraconsistent logic and ASP.

Acknowledgement

We are grateful to anonymous referees for their useful comments. J.C. Nieves wants to thank CONACyT for his PhD Grant.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
2. B. G. Buchanan and E. H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Weslay, 1985.
3. D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pages 439–513. Oxford University Press, Oxford, 1994.
4. J. Y. Halpern. *Reasoning about uncertainty*. The MIT Press, 2005.
5. H. Jakobovits and D. Vermeir. Robust semantics for argumentation frameworks. *Journal of logic and computation*, 9(2):215–261, 1999.
6. J. McCarthy. Some Expert System Need Common Sense. *Annals of the New York Academy of Sciences*, 426(1):129–137, 1984.
7. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.
8. E. Mendelson. *Introduction to Mathematical Logic*. Chapman and Hall/CRC, Fourth edition 1997.
9. P. Nicolas, L. Garcia, I. Stéphan, and C. Lafèvre. Possibilistic Uncertainty Handling for Answer Set Programming. *Annal of Mathematics and Artificial Intelligence*, 47(1-2):139–181, June 2006.
10. J. C. Nieves, M. Osorio, and U. Cortés. Semantics for possibilistic disjunctive programs (poster). In G. B. Chitta Baral and J. Schlipf, editors, *Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-07)*, number 4483 in LNAI, pages 315–320. Springer-Verlag, 2007.
11. M. Osorio, J. A. Navarro, J. R. Arrazola, and V. Borja. Ground nonmonotonic modal logic s5: New results. *Journal of Logic and Computation*, 15(5):787–813, 2005.
12. M. Osorio, J. A. Navarro, J. R. Arrazola, and V. Borja. Logics with Common Weak Completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.