Possibilistic Stratified Minimal Model Semantics*

Juan Carlos Nieves
Universitat Politècnica de Catalunya
Software Department
c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu

Mauricio Osorio
Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir
Cholula, Puebla, 72820 México
osoriomauri@googlemail.com

Abstract

In this paper, we introduce a recursive construction of a possibilistic logic programming semantics that we call Possibilistic Stratified Minimal Model Semantics*. We show that this semantics has some interesting properties such as it is always defined and satisfies relevance. One of the main implications of satisfying relevance by this semantics is that it will allow performing top-down queries from a possibilistic knowledge base.*

Keywords*: Non-monotonic reasoning, Possibilistic Reasoning, Minimal Models.*

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 reason about it.

In [22, 24, 25, 27], a common possibilistic framework for reasoning under uncertainty was proposed. This framework is a combination between Answer Set Programming (ASP) and Possibilistic Logic [13]. Possibilistic Logic is based on possibilistic theory where at the mathematical level, degrees of *possibility* and *necessity*[1] are closely related to fuzzy sets. Thanks to the natural properties of possibilistic logic and ASP, this approach allows to deal with reasoning that is at the same time non-monotonic and uncertain. The expressiveness of this approach is rich enough for capturing sophisticated domains such as river basin systems [4, 5] and medical domains [23]. However, a common problem of working with real domains is to find specifications which are inconsistent. Inconsistency in the sense that these specifications can capture contradictory information and hence, these specifications could have no possibilistic models. It is worth mentioning that an expert could want to consult a possibilistic knowledge although it is inconsistent [4].

In this paper, we introduce a new possibilistic semantics, that we call *Possibilistic Stratified Minimal Model Semantics*, for capturing possibilistic logic programs. This semantics is based on the proof theory of possibilistic logic, the idea of stratified minimal models, and a recursive construction. These fundamentals will help to define a possibilistic semantics which will be more tolerant to inconsistences (in the sense of no existence of possibilistic models) than the possibilistic semantics introduced before. The recursive construction of this semantics considers a splitting of the given possibilistic logic program. This splitting supports that our new possibilistic semantics satisfies the property of relevance. In fact, it will allow performing top-down queries from a possibilistic knowledge base.

The rest of the paper is structured as follows: In §2, we will present some basic concepts of the proof theory of possibilistic logic, the basic syntax of non-possibilistic and possibilistic logic programs and the basic concepts for splitting a possibilistic program into its components. In §3, we introduce our new possibilistic semantics for possibilistic programs, as well as, we discuss some of its relevant features. And, in the last section, we present our conclusions.

2 Background

In this section, we define some basic concepts of Possibilistic Logic and the syntaxis of non-possibilistic and possibilistic logic programs.

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

*The authors are named by alphabetic order.

[1] For those steeped in modal logic, we wan to point out that the notion of necessity in possibilistic logic is not the same to the notion of the necessity operator $\Box$ in modal logic. See §2.1, for the formal definition of necessity formulas in possibilistic logic.

measure modeling our possibly incomplete state knowledge [13]. 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*, [13] introduced a formal system for necessity-valued logic which is based in the following axioms schemata (propositional case):

(A1) $(\varphi \rightarrow (\psi \rightarrow \varphi)\ 1)$

(A2) $((\varphi \rightarrow (\psi \rightarrow \xi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \xi))\ 1)$

(A3) $((\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)\ 1)$

Inference rules:

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

(S) $(\varphi\ \alpha) \vdash (\varphi\ \beta)$ if $\beta \leq \alpha$

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

The problem of inferring automatically the necessity-value of a classical formula from a possibilistic base was solved by an extended version of *resolution* for possibilistic logic (see [13] for details).

2.2 Non-Possibilistic Programs and Some Operations

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 and $\bot$ stand for the indecomposable proposition, which we call atoms or atomic propositions. Atoms negated by $\neg$ will be called *extended atoms*. We will use the concept of atom without paying attention if it is an extended atom or not. The negation symbol $\neg$ is regarded as the so called *strong negation* by the ASP's literature and the negation not as the *negation as failure*. A literal is an atom, a, or the negation of an atom $not\ a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $not\ \{a_1, ..., a_n\}$ to denote the set of literals $\{not\ a_1, ..., not\ a_n\}$.

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

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

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

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

We will manage the strong negation ($\neg$), in our logic programs, as it is done in ASP [7]. Basically, each negative atom $\neg a$ is replaced by a new atom symbol a' which does not appear in the language of the program.

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.

A logic program P induces a notion of $dependency$ between atoms from $\mathcal{L}_P$. We say that a *depends immediately on* b, if and only if:

- b appears in the body of a clause in P, such that a appears in its head.
- both b and a appear in the head of the same clause C, such that $C \in P$ (observe that in this case a *depends immediately on* b and b *depends immediately on* a).

The two place relation *depends on* is the transitive closure of *depends immediately on*. The set of dependencies of an atom x, denoted by *dependencies-of*(x), corresponds to the set $\{a \mid x\ depends\ on\ a\}$. We define an equivalence relation $\equiv$ between atoms of $\mathcal{L}_P$ as follows: $a \equiv b$ if and only if $a = b$ or ($a\ depends\ on\ b$ and $b\ depends\ on\ a$). We write $[a]$ to denote the equivalent class induced by the atom a.

Example 1 *Let us consider the following normal program P:*

$$\begin{array}{ll} e \leftarrow e. & c \leftarrow c. \\ a \leftarrow\ not\ b, c. & b \leftarrow\ not\ a,\ not\ e. \\ d \leftarrow b. & \end{array}$$

The dependency relations between the atoms of $\mathcal{L}_P$ are as follows:
dependencies-of$(a) = \{a, b, c, e\}$*;* dependencies-of$(b) = \{a, b, c, e\}$*;* dependencies-of$(c) = \{c\}$*;* dependencies-of$(d) = \{a, b, c, e\}$*; and* dependencies-of$(e) = \{e\}$.
We can also see that, $[a] = [b] = \{a, b\}$, $[d] = \{d\}$, $[c] = \{c\}$, *and* $[e] = \{e\}$.

Given a logic program P, we take $<_P$ to denote the strict partial order induced by $\equiv$ on its equivalent classes. Hence, $[a] <_P [b]$, if and only if, b *depends-on* a and $[a]$ is not equal to $[b]$. By considering the relation $<_P$, each atom of $\mathcal{L}_P$ is assigned an order as follows:

- An atom a *is of order* 0, if $[a]$ is minimal in $<_P$.
- An atom a *is of order* $n + 1$, if n is the maximal order of the atoms on which a depends.

We say that a program P is of order n, if n is the maximum order of its atoms. We can also break a program P of order n into the disjoint union of programs P_i with $0 \leq i \leq n$, such that P_i is the set of rules for which the head of each clause is of order i (*w.r.t.* P). We say that $P_0, \ldots, P_n$ are the *relevant modules* of P.

Example 2 *By considering the equivalent classes of the program P in Example 1, the following relations hold:* $\{c, e\} <_P \{a, b\} <_P \{d\}$. *We can see that:* a *is of order* 1, d *is of order* 2, b *is of order* 1, e *is of order* 0, *and* c *is of order* 0. *This means that P is a program of order* 2.

The following table illustrates how the program P can be broken into the disjoint union of the following relevant modules P_0, P_1, P_2:

P	P_0	P_1	P_2
$e \leftarrow e.$	$e \leftarrow e.$		
$c \leftarrow c.$	$c \leftarrow c.$		
$a \leftarrow \ not\ b, c.$		$a \leftarrow \ not\ b, c.$	
$b \leftarrow \ not\ a, \ not\ e.$		$b \leftarrow \ not\ a, \ not\ e.$	
$d \leftarrow b.$			$d \leftarrow b.$

Given a set of interpretations Q and a signature $\mathcal{L}$, we define Q *restricted to* $\mathcal{L}$ as $\{M \cap \mathcal{L} \mid M \in Q\}$. For instance, let Q be $\{\{a, c\}, \{c, d\}\}$ and $\mathcal{L}$ be $\{c, d, e\}$, hence Q *restricted to* $\mathcal{L}$ is $\{\{c\}, \{c, d\}\}$.

Let P be a program and $P_0, \ldots, P_n$ its relevant modules. We say that a semantics S satisfies the property of *relevance* if for every $i, 0 \leq i \leq n$, $S(P_0 \cup \cdots \cup P_i) = S(P)$ restricted to $\mathcal{L}_{P_0 \cup \cdots \cup P_i}$. The interested reader can find in [10] more details *w.r.t.* relevance and other well-accepted properties of logic programming semantics.

2.3 Possibilistic Programs and some Possibilistic Operations

In all the paper, we will consider finite lattices. This convention is taken based on the assumption that in real applications we will rarely have infinite lattices for expressing the incomplete states of a knowledge base.

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 (since the lattice is finite then it is complete). We apply the projection $*$ 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.

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

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

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

$$TOP_Q : \ \bot \leftarrow \mathcal{B}^+, \ not\ \mathcal{B}^-$$

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

3 Possibilistic Stratified Minimal Model Semantics

In this section, we introduce a constructive possibilistic logic programming semantics, called *possibilistic stratified minimal model semantics*, which is based on *possibilistic minimal models*. This semantics has some interesting properties such as: it satisfies the property of relevance, and it agrees with the possibilistic answer set semantics for the class of possibilistic stratified logic programs (a possibilistic logic program P is called a possibilistic stratified logic program if and only if P^* is a stratified logic program. We remember to the reader that the class of stratified logic programs is a class of logic programs where several well-accepted logic programming semantics agree [19]).

Before presenting the definition of our new possibilistic semantics, let us extend the basic concept of modules for possibilistic logic programs. Let PP be a possibilistic logic program, we will say that PP is of order n if PP^* is of

order n. We say that $PP_0, \ldots, PP_n$ are the relevant modules of PP if $PP_0^*, \ldots, PP_n^*$ are the relevant modules of PP^*. In order to illustrate this concepts, let us consider the following example.

Example 3 *Let $PP = \langle(\{0, 0.1, \ldots, 0.9, 1\}, \leq), N\rangle$[2] be a possibilistic program such that PP is composed by the possibilistic clauses :*

PP

$0.5 : e.$
$0.5 : c \leftarrow c.$
$0.7 : a \leftarrow \ not\ b, e.$
$0.5 : b \leftarrow \ not\ a,\ not\ c.$
$0.4 : d \leftarrow b.$

Observe that PP^* has three modules. This means that PP is of order 2; hence, PP can be split into three components: PP_0, PP_1 and PP_2.

PP_0	PP_1	PP_2
$0.5 : e.$ $0.5 : c \leftarrow c.$		
	$0.7 : a \leftarrow \ not\ b, e.$ $0.5 : b \leftarrow \ not\ a,\ not\ c.$	
		$0.4 : d \leftarrow b.$

Since we are going to consider sets of possibilistic atoms as interpretations of a given possibilistic logic program, we are going to introduce some basic operator for sets of possibilistic atoms. These operators were defined in [22].

Given $\mathcal{A}$ a finite set of atoms and (Q,$\leq$) be a lattice, we consider $\mathcal{PS} = 2^{\mathcal{A}\times Q}$ as the finite set of all the possibilistic atoms sets induced by $\mathcal{A}$ and Q. Let $A, B \in \mathcal{PS}$, hence we define the operators $\sqcap$, $\sqcup$ and $\sqsubseteq$ as follows:

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

Given a possibilistic logic program $P = \langle(Q, \leq), N\rangle$ and a set of possibilistic atoms S such that $S^* \subseteq \mathcal{L}_{P^*}$, we define the function $Not\_Compl(S)$ as:

$$Not\_Compl(S) := \{(\sim a, TOP_Q) | a \in (\mathcal{L}_{P^*} \setminus S^*)\}$$

One key concept of the definition of the stratified minimal model semantics is the concept of *possibilistic minimal model*. This concept is the natural generalization of minimal model in terms of possibilistic programs. From now on, we assume that the reader is familiar with the single notion of *minimal model*. In order to illustrate this basic notion, let P be the normal program $\{a \leftarrow \ not\ b;\ \ b \leftarrow \ not\ a;\ \ a \leftarrow \ not\ c;\ \ c \leftarrow \ not\ a\}$. As we can see, P has five models: $\{a\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b\}$, $\{a, b, c\}$; however, P has just two minimal models: $\{b, c\}$, $\{a\}$. By having in mind the notion of minimal model, we define the notion of possibilistic minimal model as follows:

Definition 1 (Possibilistic Minimal Model Semantics) *Let $P = \langle(Q, \leq), N\rangle$ be a possibilistic logic program and S be a set of possibilistic atoms such that $S^* \subseteq \mathcal{L}_{P^*}$. We say that S is a possibilistic minimal model of P if and only if S^* is a minimal model of P^* and $P \cup Not\_Compl(S) \vdash_{PL} S$ and $\nexists S' \in \mathcal{PS}$ such that $S' \neq S$, $P \cup Not\_Compl(S') \vdash_{PL} S'$ and $S \sqsubseteq S'$. We denote by $PMM(P)$ the set of all possibilistic minimal models of P. This set of possibilistic minimal models of P is called possibilistic minimal model semantics*

Example 4 *Let $P = \langle(\{0, 0.1, \ldots, 0.9, 1\}, \leq), N\rangle$ be a possibilistic program such that N is*

$$0.5 : a \vee b. \qquad\qquad 0.8 : a \leftarrow \ not\ c.$$

Let us test if $M_1 = \{(a, 0.5)\}$ is a possibilistic minimal model of P. It is clear that M^* is a minimal model of P^*, then in order to see if M is a possibilistic minimal model of P, we have to prove that $P \cup Not\_Compl(M_1) \vdash_{PL} (a, 0.5)$ where $Not\_Compl(M_1) = \{(\sim b, 1), (\sim c, 1)\}$. The proof of $(a, 0.5)$ from $P \cup Not\_Compl(M_1)$ is as follows:

1. $a \vee b$	*0.5*	***Premise***
2. $\sim c \rightarrow a$	*0.8*	***Premise***
3. $\sim b$	*1*	***Premise***
4. $\sim c$	*1*	***Premise***
6. $\sim b \rightarrow a$	*0.5*	***Logical equivalence from 1.***
7. a	*0.5*	***From 3 and 6 by GMP***

This means that M_1 is a candidate *set to be a possibilistic minimal model of P. However, observe that by considering the premise 2 and 4 of the proof, we can infer the possibilistic atoms $(a, 0.8)$. This means that M_1 cannot be a possibilistic minimal model of P because 0.5 is not the optimal/maximum value that can be inferred from P for the atom a. In fact, we can see that if $M_2 = \{(a, 0.8)\}$, then $M_1 \sqsubseteq M_2$. Observe that M_2 is also a* candidate *set to be a possibilistic minimal model of P because M_2^* is a minimal model of P^* and $P \cup Not\_Compl(M_2) \vdash_{PL} (a, 0.8)$. Since, 0.8 is the optimal/maximum value for the atom a that can be inferred from P. We can conclude that M_2 is a* possibilistic minimal model *of P.*

[2] $\leq$ denotes the standard relation of order between rational number.

Observe that since any extended logic program has at least one minimal model, this suggests that for any possibilistic logic program the possibilistic minimal model semantics is defined. One possible problem of the possibilistic minimal model semantics is that for some programs one can get some unexpected models. For instance, let P be the following set of possibilistic clauses (we assume that these clauses consider the lattice of the program P of Example 4):

$$0.5 : a \leftarrow \ not\ b.$$
$$0.5 : b \leftarrow a,\ not\ a.$$

We can see that $PMM(P) = \{\{(a, 0.5)\}, \{(b, 0.5)\}\}$. However, observe that the atom b is implied by the formula $a \wedge \sim a$; hence, one cannot expect to infer the atom b from P. In fact, any of the possibilistic semantics introduced in [22, 24, 25, 27] does not infer the possibilistic atom $(b, 0.5)$. In order to avoid these kinds of unexpected models, one can remove from P any possibilistic tautology (given a possibilistic clause C, we say that C is a possibilistic tautology if and only if C^* is *possibilistic tautology* in the sense of classical logic). For instance, by removing from P the possibilistic clause $0.5 : b \leftarrow a,\ not\ a$, we get the program P':

$$0.5 : a \leftarrow \ not\ b.$$

We can see that the only possibilistic minimal model of P' is $\{(a, 0.5)\}$. Therefore, for the definition of the possibilistic stratified minimal model semantics (Definition 3), we defined the function $freeTaut$ as follows:

- Given a possibilistic logic program P, $freeTaut$ denotes a function which removes from P any possibilistic tautology.

The idea of the function $freeTaut$ is to remove from a possibilistic program any clause which is equivalent to a possibilistic tautology in order to avoid unexpect possibilistic models which could be inferred by the possibilistic stratified minimal model semantics.

An important part of the definition of the possibilistic stratified minimal model semantics is a single possibilistic reduction. The idea of this reduction is to remove from a possibilistic logic program any atom which has already fixed to some true value. In fact, this reduction is based on a pair of sets of possibilistic atoms $\langle T; F\rangle$ such that the set T contains the atoms which can be considered as true and the set F contains the atoms which can be considered as false. Formally, this reduction is defined as follows:

Definition 2 *Let $P = \langle (Q, \leq), N\rangle$ be a possibilistic logic program and $A = \langle T; F\rangle$ be a pair of sets of possibilistic atoms. The reduction $R(P, A)$ is obtained by 4 steps:*

1. *We replace each possibilistic clause $r = (\alpha_1 : \mathcal{A} \leftarrow \mathcal{B}^+ \cup \{x\},\ not\ \mathcal{B}^-) \in P$ by the possibilistic clause $r' = (\mathcal{GLB}(\{\alpha_1, \alpha_2\}) : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-)$ if $(x, \alpha_2) \in T$.*

2. *We replace each possibilistic clause $r = (\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^- \cup \{x\}) \in P$ by the possibilistic clause $r' = (\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-) \in P$ if $x \in F^*$.*

3. *We remove each possibilistic clause $r = (\alpha : \mathcal{A} \leftarrow \mathcal{B}^+ \cup \{x\},\ not\ \mathcal{B}^-) \in P$ if $x \in F^*$*

4. *We remove each possibilistic clause $r = (\alpha : \mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^- \cup \{x\}) \in P$ if $x \in T^*$*

In order to illustrate this definition, let us consider the program PP_1 which is a module of the program PP introduced in Example 3 and $A = \langle \{(e, 0, 5)\}; \{(c, 1)\}\rangle$. We can see that $R(P, A)$ is the program:

$$0.5 : a \leftarrow \ not\ b. \qquad\qquad 0.5 : b \leftarrow \ not\ a.$$

Since the definition of the possibilistic stratified minimal model semantics requires an especial union of sets of possibilistic atoms, we introduce the operator $\times$ as follows:

- Given Q and L both sets of possibilistic atoms, we define $Q \times L := \{M_1 \sqcup M_2 \mid M_1 \in Q, M_2 \in L\}$.

The operator $\times$ will be useful for making the union of the models of two different possibilistic programs. For instance, let us consider the following two possibilistic programs:

P_1 :	P_2 :
$0.5 : e.$	$0.7 : a \leftarrow \ not\ b$
$0.5 : c \leftarrow c.$	$0.5 : b \leftarrow \ not\ a$

We can see that $PMM(P_1) = \{(e, 0.5)\}$ and $PMM(P_2) = \{\{(a, 0.7)\}, \{(b, 0.5)\}\}$; hence, we can see that

$$PMM(P_1) \times PMM(P_2) = \{\{(e, 0.5), (a, 0.7)\}, \{(e, 0.5), (b, 0.5)\}\}$$

Now, by considering the possibilistic minimal model semantics we define **the possibilistic stratified minimal model semantics** as follows:

Definition 3 *Given a possibilistic logic program $P = \langle (Q, \leq), N\rangle$, we define the* possibilistic stratified minimal model semantics *PMM^r as follows: $PMM^r(P) = PMM^r_c(freeTaut(P) \cup \{TOP_Q : x \leftarrow x \mid x \in \mathcal{L}_{P^*} \setminus HEAD(P^*)\}$ such that $PMM^r_c(P)$ is defined as follows:*

1. *if P is of order 0, $PMM^r_c(P) = PMM(P)$.*

2. *if* P *is of order* $n > 0$, $PMM_c^r(P) = \bigcup_{M \in PMM(P_0)}\{M\} \times PMM_c^r(R(Q, A))$ *where* $Q = P \setminus P_0$ *and* $A = \langle M, \{(a, TOP_Q) | a \in (\mathcal{L}_{P_0^*} \setminus M^*)\}\rangle$.

We call a model in $PMM^r(P^*)$ *a* possibilistic stratified minimal model *of* P.

Observe that the definition of the Possibilistic Stratified Minimal Model Semantics is based on a recursive function where its base case is essentially the possibilistic minimal model semantics. An interesting feature of this definition is that, if we consider a different semantics for the base case of the recursive function, such as the possibilistic answer set semantics [22, 24, 25] and the possibilistic p-stable semantics [27], one is able to construct different possibilistic semantics which will satisfy the property of relevance.

In order to illustrate the definition of the possibilistic stratified minimal model semantics, let us consider the possibilistic program PP introduced in Example 3. As we saw PP is of order 2. Following Definition 3, we can see that

$$PMM^r(PP_0) = \{(e, 0.5)\}$$

By considering $PMM^r(PP_0)$, we can reduce PP_1 by $A = \langle\{(e, 0.5)\}, \{(c, 1)\}\rangle$; hence, $R(PP_1, A)$ is:

$$0.5 : a \leftarrow \ not\ b. \qquad\qquad 0.5 : b \leftarrow \ not\ a.$$

We can see that $PMM^r(PP_0) \times PMM^r(R(PP_1, A)) = \{\{(e, 0.5), (a, 0, 5)\}, \{(e, 0.5), (b, 0.5)\}\}$. This means that there are two different sets, $\{(e, 0.5), (a, 0, 5)\}$ and $\{(e, 0.5), (b, 0.5)\}$, for reducing PP_2:

1. Let $M = \{(e, 0.5), (a, 0, 5)\}$, we can see that PP_2 reduced by $A = \langle\{(e, 0.5), (a, 0, 5)\}, \{\}\rangle$ is the same program. Hence $PMM^r(R(PP_2, A)) = \emptyset$.

2. On the other hand, by considering $M = \{(e, 0.5), (b, 0.5)\}$, we can see that PP_2 reduced by $A = \langle(e, 0.5), (b, 0.5)\}, \{\}\rangle$ we get the program $R(PP_2, A)$:

$$0.4 : d.$$

Therefore $PMM^r(R(PP_2, A)) = \{\{(d, 0.4)\}\}$.

This means that

$$PMM^r(PP) = \{\{(e, 0.5), (a, 0, 5)\}, \{(e, 0.5), (b, 0.5), (d, 0.4)\}\}$$

Observe that the answer sets [15] of PP^* and $PMM^r(PP)^*$ coincide for this program. In particular, for the class of stratified programs[3] we can ensure the following proposition.

[3]The reader can see [7] for a formal definition of stratified logic program.

Proposition 1 *Let* P *be a possibilistic logic program. If* P^* *is a stratified logic program, then the set of answer sets of* P^* *coincide with* $PMM^r(P)^*$.

In the literature, we can find at lest two approaches for capturing the semantics of possibilistic logic programs: one based on the answer set semantics[22, 24, 25] and another one based on pstable model semantics [27]. All of these approaches are based on possibilistic logic inference; hence, we can say that they follow the philosophy of possibilistic logic of looking for optimal/maximum uncertain degrees for the possibilistic atoms inferred from a possibilistic knowledge base. A nice property of the possibilistic stratified minimal model semantics is that it is able to infer more optimal/maximum uncertain degrees than the approach based on answer set semantics and at least as optimal/maximum that the approach based on pstable semantics. For instance, let us consider the following program $P = \langle(\{0, 0.1, \ldots, 0.9, 1\}, \leq), N\rangle$ such that N is:

$$0.7 : a \leftarrow b. \qquad 0.7 : a \leftarrow \ not\ b.$$
$$0.7 : b \leftarrow a. \qquad 0.1 : b \leftarrow \ not\ c.$$

Following any of the possibilistic semantics based on the answer set semantics we can see that this program P has just one possibilistic answer set $M_1 = \{(a, 0.1), (b, 0.1)\}$. On the other hand, observe that P has just one possibilistic stratified minimal model $M_2 = \{(a, 0.7), (b, 0.7)\}$. It is clear that $M_1 \sqsubseteq M_2$. In general, we can ensure that if $PSEM$ is the possibilistic answer set semantics defined in [24][4] the following implication is true:

> If $(a, \alpha_1) \in M$ such that $M \in PSEM(P)$ then $\exists\ M' \in PMM^r(P)$ such that $(a, \alpha_2) \in M'$ and $\alpha_1 \leq \alpha_2$.

In fact, this implication remains true if one instantiates $PSEM$ with the possibilistic semantics introduced in [27].

A relevant property of the possibilistic stratified minimal models semantics that is supported by its recursive construction is that it satisfies the property of relevance.

Proposition 2 *Let* P *be a possibilistic disjunctive logic program such that* P *is of order* n. *Then for every* i , $0 \leq i \leq n$, $PMM^r(P_0 \cup \cdots \cup P_i) = PMM^r(P)$ *restricted to* $\mathcal{L}_{P_0 \cup \cdots \cup P_i}$.

Observe that since PMM^r satisfies the property of relevance, one can decide if an atom is inferred by a possibilistic knowledge based by considering only the relevant clauses of the given atom. For instance, let $P = \langle(\{0, 0.1, \ldots, 0.9, 1\}, \leq), N\rangle$ such that N is:

[4]It is worth mentioning that the possibilistic semantics defined in [24] extends the semantics defined in [22] for possibilistic disjunctive logic programs and it is equivalent to the semantics defined in [25].

$0.6 : q$ $\quad\quad$ $0.7 : m \leftarrow \ not\ n, q.$
$0.8 : r \leftarrow m.$
$0.5 : b \leftarrow \ not\ a, q.$ $\quad\quad$ $0.5 : a \leftarrow \ not\ b, r.$

Now, let us suppose that we want to known if there is a possibilistic stratified minimal model M such that r belongs to M^*. One possible strategy for knowing if there exists M is to apply PMM^r to the whole program P. However, since PMM^r satisfies relevance, we can consider only the clauses that are relevant for inferring r. Hence, we can only consider the clauses that their head-atoms belong to *dependencies-of*(r). Thefore, for inferring r from P, we can only consider the following subprogram P' of P:

$0.6 : q$ $\quad\quad$ $0.7 : m \leftarrow \ not\ n, q.$
$0.8 : r \leftarrow m.$

It is easy to see that $MPP^r(P') = \{\{(q, 0, 6), (m, 0.6), (r, 0.6)\}\}$ and $MPP^r(P) = \{\{(q, 0, 6), (m, 0.6), (r, 0.6), (a, 0.5)\}, \{(q, 0, 6),\ (m, 0.6),\ (r, 0.6),\ (a, 0.5)\}\}$. Observe that by considering the relevant clauses of r we are reducing the search space for looking if there exists M. In general, one can answer a query *w.r.t.* an atom a by considering its relevant clauses that are suggested by *dependencies-of*(a).

As last important property of MPP^r, we formalize that for any possibilistic logic program MPP^r is defined.

Proposition 3 *Let P be a possibilistic disjunctive logic program. Hence $MPP^*(P) \neq \emptyset$.*

4 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 [30]. 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.*, [17].
- probabilistic logic, *e.g.*, [21, 20, 16].
- fuzzy set theory, *e.g.*, [31, 29, 32].
- multi-valued logic, *e.g.*, [14, 18].
- evidence theoretic logic programming, *e.g.*, [6].
- possibilistic logic, *e.g.*, [12, 3, 2, 1, 22].

Basically, these approaches differ in the underlying notion of uncertainty and how uncertainty values, associated to clauses and facts, are managed. Of course, our approach falls in the approaches based on possibilistic logic. One of the relevant properties that we can point out from our approach is that this approach allows dealing with reasoning that is at the same time non-monotonic and uncertain. This property is not easy for being satisfied for other approaches which capture uncertain information.

5 Conclusions and Future Work

In this paper, we have introduced a new approach for defining semantics of possibilistic logic program. This approach is based on the idea of splitting a possibilistic logic program into its components. In particular, by considering the components of a given possibilistic program we define a recursive definition of a possibilistic semantics that we call *possibilistic stratified minimal model semantics* PMM^r. An interesting feature of PMM^r's construction is that it suggests a general approach for defining new possibilistic semantics based on any possibilistic semantics defined before [22, 24, 25, 27]. In fact, the new semantics induced by this construction will satisfy the property of *relevance*.

Some of the main features of PMM^r that we can point out are

- PMM^r is able to reduce the search space for performing top-down queries from a possibilistic knowledge base. This is because the PMM^r allows to infer partial possibilistic models.
- PMM^r is able to infer more optimal/maximun uncertain degrees for any atom inferred from a possibilistic knowledge base than the possibilistic semantics introduced before. This means that this semantics is more close to the possibilistic logic philosophy than the possibilistic semantics introduced before.
- PMM^r is more tolerant to inconsistencies (in the sense of no existence of possibilistic models) than the possibilistic semantics introduced before.
- PMM^r satisfies the property of relevance.

The idea of splitting a logic program into its component, in order to define logic programming semantics as it is done in PMM^r's construction, has been explored by some authors in logic programming [11, 28]. For instance, by splitting a logic program, Dix and Müller in [11] combine ideas of the stable model semantics and the well-founded semantics in order to define a skeptical logic programming semantics which satisfies the property of relevance and the general principle of partial evaluation.

Recently in [26], it was shown that by considering a similar construction to PMM^r for normal programs, it is possible to define a logic programming semantics which is able to characterize a recently introduced argumentation semantics called $CF2$ [8]. This suggests that by considering PMM^r one is able to explore a possibilistic version of CF2 for argumentation approaches where the relationship between arguments depends on the evidence of each argument as in [9].

One of the main issues in our future work is to do a deep analysis of complexity of PMM^r. By the moment, we can observe that there are three main factors which can impact in the computational complexity of PMM^r

1. the computational cost of computing minimal models,
2. the computational cost of computing optimal possibilistic degrees for each atom which appears in the possibilistic models, and
3. the computational cost of the possibilistic reduction of Definition 2.

We belief that after the analysis of complexity of PMM^r we will be able to implement an efficient solver of PMM^r in order to apply PMM^r in real domains as river basin systems [4, 5].

Acknowledgement

We are grateful to anonymous referees for their useful comments. This research has been partially supported by the EC founded project ALIVE (FP7-IST-215890). The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

[1] T. Alsinet, C. I. Chesñevar, L. Godo, and G. R. Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems*, 159(10):1208–1228, 2008.

[2] T. Alsinet and L. Godo. A Ccomplete Calculus for Possibilistic Logic Programming with Fuzzy Propositional Variable. In *Proceedings of the Sixteen Conference on Uncertainty in Artificial Intelligence*, 1-10, 2000. ACM Press.

[3] T. Alsinet and L. Godo. Towards an automated deduction system for first-order possibilistic logic programming with fuzzy constants. *Int. J. Intell. Syst.*, 17(9):887–924, 2002.

[4] M. Aulinas. *Management of industrial wastewater discharges through agents' argumentation*. PhD thesis, PhD on Environmental Sciences, University of Girona, to be presented.

[5] M. Aulinas, J. C. Nieves, M. Poch, and U. Cortés. Supporting Decision Making in River Basin Systems Using a Declarative Reasoning Approach. In M. Finkel and P. Grathwohl, editors, *Proceedings of the AquaTerra Conference (Scientific Fundamentals for River Basic Management), ISSN 0935-4948*, page 75, March 2009.

[6] J. F. Baldwin. Evidential support logic programming. *Fuzzy Sets and Systems*, 24(1):1–26, Octuber 1987.

[7] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.

[8] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.

[9] T. Bench-Capon. Value-based argumentation frameworks. In *Proceedings of Non Monotonic Reasoning*, pages 444–453, 2002.

[10] J. Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundam. Inform.*, 22(3):257–288, 1995.

[11] J. Dix and M. Müller. Partial evaluation and relevance for approximations of stable semantics. In *ISMIS*, volume 869 of *Lecture Notes in Computer Science*, pages 511–520. Springer, 1994.

[12] D. Dubois, J. Lang, and H. Prad. Towards possibilistic logic programming. In K. Furukawa, editor, *ICLP*, pages 581–595. The MIT Press, 1991.

[13] 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.

[14] M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11(1&2):91–116, 1991.

[15] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[16] G. Kern-Isberner and T. Lukasiewicz. Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence*, 157(1-2):139–202, 2004.

[17] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Log. Program.*, 12(3&4):335–367, 1992.

[18] L. V. S. Lakshmanan. An epistemic foundation for logic programming with uncertainty. In *FSTTCS*, pages 89–100, 1994.

[19] J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1987.

[20] T. Lukasiewicz. Probabilistic logic programming. In *ECAI*, pages 388–392, 1998.

[21] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Inf. Comput.*, 101(2):150–201, 1992.

[22] P. Nicolas, L. Garcia, I. Stéphan, and C. Lafèvre. Possibilistic Uncertainty Handling for Answer Set Programming. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):139–181, June 2006.

[23] J. C. Nieves. *Modeling arguments and uncertain information — A non-monotonic reasoning approach*. PhD thesis, Software Department (LSI), Technical University of Catalonia, 2008.

[24] J. C. Nieves, M. Osorio, and U. Cortés. Semantics for possibilistic disjunctive programs. In S. Costantini and R. Watson, editors, *Answer Set Programming: Advances in Theory and Implementation*, pages 271–284, 2007.

[25] J. C. Nieves, M. Osorio, and U. Cortés. Semantics for possibilistic disjunctive programs. In C. Baral, G. Brewka, 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.

[26] J. C. Nieves, M. Osorio, and C. Zepeda. Expressing Extension-Based Semantics based on Stratified Minimal Models. In H. Ono, M. Kanazawa, and R. de Queiroz, editors, *Proceedings of WoLLIC 2009, Tokyo, Japan*, volume 5514 of *FoLLI-LNAI subseries*, pages 305–319. Springer Verlag, 2009.

[27] M. Osorio and J. C. Nieves. Pstable semantics for possibilistic logic programs. In *MICAI 2007: Advances in Artificial Intelligence, 6th Mexican International Conference on Artificial Intelligence*, number 4827 in LNAI, pages 294–304. Springer-Verlag, 2007.

[28] L. M. Pereira and A. M. Pinto. Layered Models Top-Down Querying of Normal Logic Programs. In *PADL*, Lecture Notes in Computer Science, pages 254–268. Springer, 2009.

[29] M. Rodríguez-Artalejo and C. A. Romero-Díaz. Quantitative Logic Programming revisited. In J. Garrigue and M. Hermenegildo, editors, *9th International Symposium, FLOPS*, volume 4989 of *LNCS*, pages 272–288. Springer-Verlag Berlin Heidelberg, 2008.

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

[31] M. H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 3(1):37–53, 1986.

[32] D. Van-Nieuwenborgh, M. D. Cock, and D. Vermeir. An introduction to fuzzy answer set programming. *Ann. Math. Artif. Intell.*, 50(3-4):363–388, 2007.