Logic Programs with Possibilistic Ordered Disjunction

Roberto Confalonieri, Juan Carlos Nieves, and Javier Vázquez-Salceda

Universitat Politècnica de Catalunya
Dept. Llenguatges i Sistemes Informàtics
C/ Jordi Girona Salgado 1-3
E - 08034 Barcelona
{confalonieri,jcnieves,jvazquez}@lsi.upc.edu

Abstract Logic programs with ordered disjunction have shown to be a flexible specification language able to model common user preferences in a natural way. However, in some realistic scenarios the preferences should be linked to the evidence of the information when trying to reach a single preferred solution. In this paper, we extend the syntax and the semantics of logic programs with ordered disjunction in order to cope with uncertain information. In particular, we define a possibilistic semantics for capturing possibilistic ordered disjunction programs. We use a simple example to explain the approach and outline an application scenario showing the benefits of possibilistic ordered disjunction.

Keywords: Non-monotonic Reasoning, Answer Set Programming, Ordered Disjunction, Possibilistic Reasoning.

1 Introduction

Preference handling is more and more required in many types of decision-making contexts such as online applications, system configuration and design, and complex real-word decision problems [2]. In the case of online application, recommender systems attempt to model user preferences in order to satisfy user requests in a personalised way and new preference handling methods are needed [10]. Classical preference modeling approaches assume in fact a complete specification of an utility function, used to map possible outcomes of decisions to numerical values and select the best. However, in user preference modeling the set of possible decisions tends to be either too large to be described explicitly or information is incomplete, and thus more intelligent approaches are needed.

Nonmonotonic logics have shown to be a potent knowledge representation formalism to reason about preferences. In [4] the relationship between nonmonotonic logics and preferences is discussed. Particularly it describes how extended logic programming, one of the basic formalisms of answer-set programming, can be used to represent user preferences, reasoning in terms of preferred answer sets of a logic program. The ways how preferred answer sets are computed and

selected differ on the formalism chosen. For instance, *logic programs with ordered disjunction* permit to explicitly represent preference information directly into head rules literals [3]. In this way, the language can capture user qualitative preferences by means of disjunction rules, represent choices among different alternatives and specify a preference order between the answer sets through some comparison criteria.

However, in some realistic scenarios the aggregation of new uncertain information can affect the established preference order, preventing the achievement of a single preferred solution. In such cases a mechanism able to consider the *uncertainty* about the new information is needed. As far as we know, a semantics for logic programs with ordered disjunction that considers degrees of uncertainty is lacking. One existing approach to deal with uncertainty degree is the combination of possibilistic logic with answer set semantics, where degrees of possibility and necessity which are closed related to fuzzy sets [6] are associated as uncertainty values to nonmonotonic logic programs clauses [12,13,14].

In this paper, we propose an extension of the semantics of logic programs with ordered disjunction in order to cope with the degree of uncertainty in the reasoning process. In particular, we define a possibilistic semantics for capturing *possibilistic ordered disjunction programs* which is close to the proof theory of possibilistic logic and answer set semantics. We show how by considering the necessity-value of ordered disjunction clauses and using possibilistic inference, in some cases it is possible to reach a single preferred possibilistic answer set, whereas the ordered disjunction program cannot.

The rest of the paper is organized as follows. In the next section we recall the basic notions underlying possibilistic logic and ordered disjunction programs. Section 3 describes our possibilistic extension of ordered disjunction programs, providing their syntax, semantics and three answer sets comparison criteria. In Section 5 we describe an application scenario which shows the benefits of possibilistic ordered disjunction. Throughout the paper, we use a simple example to explain our approach.

2 Background

This section presents the reader with some basic definitions *w.r.t.* extended logic programs, answer set semantics, possibilistic logic, some fundamental definitions of lattice theory and logic programs with ordered disjunction, in order to make this document self contained.

2.1 Extended Logic Programs

We consider extended logic programs which have two kinds of negation, strong negation $\neg$ and default negation *not*. A signature $\mathcal{L}$ is a finite set of elements that we call atoms, where atoms negated by $\neg$ are called extended atoms. Intuitively, *not* a is true whenever there is no reason to believe a, whereas $\neg a$ requires a proof of the negated atom. In the following we use the concept of atom without

paying attention if it is an extended atom or not. A *literal* is either an atom a, called *positive literal*; or the negation of an atom *not* a, called *negative literal*. Given a set of atoms $\{a_1, ..., a_n\}$, we write *not* $\{a_1, ..., a_n\}$ to denote the set of atoms $\{not\ a_1, ...,\ not\ a_n\}$. An extended *normal* rule, r, is a rule of the form

$$a \leftarrow b_1, \ldots, b_n,\ not\ b_{n+1}, \ldots,\ not\ b_{n+m}$$

where a and each of the b_i are atoms for $1 \leq i \leq n+m$. In a slight abuse of notation we will denote such a clause by the formula $a \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$ where the set $\{b_1, \ldots, b_n\}$ will be denoted by $\mathcal{B}^+$, and the set $\{b_{n+1}, \ldots, b_{n+m}\}$ will be denoted by $\mathcal{B}^-$. We denote by $head(r)$ the head a of rule r and by $body(r)$ the $\mathcal{B}^+,\ not\ \mathcal{B}^-$ of the rule r. A *constraint* is an extended rule of the form: $\leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$. We define an *extended logic normal program* P, as a finite collection of extended normal rules and constraints.

If the body of a normal rules is empty, then the clause is known as a *fact* and can be denoted just by a. We write $\mathcal{L}_P$, to denote the set of atoms that appear in the rules of P. We denote by $HEAD(P)$ the set $\{a | a \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^- \in P\}$.

We will manage the strong negation $\neg$, in our logic programs, as it is done in Answer Set Programming (ASP). Basically, each atom $\neg a$ is replaced by a new atom symbol a' which does not appear in the language of the program and we add the constraint $\leftarrow a, a'$ to the program [1]. For managing the constraints in our logic programs, we will replace each rule of the form $\leftarrow \mathcal{B}^+\ not\ \mathcal{B}^-$ by a new rule of the from $f \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-,\ not\ f$ such that f is a new atom symbol which does not appear in the language of the program.

Extended logic programs are very useful for knowledge representation purposes. Two major semantics for extended logic programs have been defined: answer set semantics [9], an extension of stable model semantics, and a version of well-founded semantics [8]. The second approach can be viewed as an efficient approximation of the first.

2.2 Answer set semantics

The answer set semantics was first defined in terms of the so called Gelfond-Lifschitz reduction [9] and it is usually studied in the context of syntax dependent transformations on programs. We will present this definition for the class of programs considered in this paper (extended normal programs).

Definition 1. *(P^M* Reduction*)*
Let P be any extended normal logic program, and M any a set of atoms such that $M \subseteq \mathcal{L}_P$. The M-reduct of P, denoted by P^M, is the positive normal program obtained from P deleting

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

Note that clearly P^M does not contain *not*, *i.e.* it is a positive extended logic program. The following definition allows to check if a set of atoms M is a valid model for P.

Definition 2. *Let P an extended normal logic program and M a set of atoms. Then M is an answer set of P if M is a minimal model of P^M.*

In order to illustrate these definitions let us consider the following example:

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

$b \leftarrow not\ a.$ $\quad\quad$ $b.$

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

We can see that P^M is:

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

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 M is an answer set of P.

2.3 Possibilistic Logic

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

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

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

For the axioms above, the following inference rules are defined:

(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 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 [6] for details).

2.4 Lattices and Order

The order theory studied in mathematics formalizes the intuitive concept of an ordering of the elements of a set. More recently, it has been introduced into theoretical computer science, particularly into programming language semantics [5]. As we will see in Section 3, an order set will be a suitable structure for capturing uncertain information, particularly into answer set programs.

Definition 3. Partial order
Let $\mathcal{Q}$ be a set. A partial order $\leq$ is a binary relations on $\mathcal{Q}$ such that, $\forall x, y, z \in \mathcal{Q}$,

(i) *$x \leq x$ (reflexivity);*
(ii) *If $x \leq y$ and $y \leq x$ then $x = y$ (antisymmetry);*
(iii) *If $x \leq y$ and $y \leq z$ then $x \leq z$ (transivity):*

A set $\mathcal{Q}$ with a partial order relation $\leq$ is called a partial ordered set and is denoted by ($\mathcal{Q}$,$\leq$). An element $x \in \mathcal{Q}$ is the $\mathcal{LUB}$ (*least upper bound*) of $S \subseteq \mathcal{Q}$ if:

(i) $s \leq x$, $\forall s \in S$, *i.e.* x is an upper bound of S and
(ii) $\forall y \in \mathcal{Q}[((\forall s \in S)s \leq y) \Rightarrow x \leq y]$, *i.e.* $x \leq y$ for all upper bound y of S.

The $\mathcal{GLB}$ (*greatest lower bound*) of S is defined dually. $\mathcal{LUB}$ of S is called the *supremum* of S and it is denoted by *sup* S; $\mathcal{GLB}$ of S is also called the infimum of S and it is denoted by $inf\ S$.

Definition 4. Lattice
Let ($\mathcal{Q}$,$\leq$) be a non-empty partially ordered set.

(i) *If $\exists$ sup$\{x, y\}$ and inf$\{x, y\}$ $\forall x, y \in \mathcal{Q}$, then $\mathcal{Q}$ is called lattice.*
(ii) *If $\exists$ sup S and inf S $\forall S \subseteq \mathcal{Q}$, then $\mathcal{Q}$ is called complete lattice.*

2.5 Logic Programs with Ordered Disjunction

Logic programs with ordered disjunction (LPODs) are extended logic programs which allow the use of an ordered disjunction connector $\times$ in the head of rules to express preferences among its literals [3].The rule

$$r = C_1 \times \ldots \times C_n \leftarrow b_1, \ldots, b_m,\ not\ b_{m+1}, \ldots,\ not\ b_{m+k}$$

states that if the body is satisfied then some C_i must be in the answer set, if possible C_1, if not then C_2, and so on, and at least one of them must be true. Each of the C_i can be seen as a preference the user is interested into according to a desired order. One interesting characteristic of LPODs is that they provide a mean to represent preferences among answer sets by considering degrees of satisfaction [3].

Definition 5. *[3] Let M be an answer set of an ordered disjunction program P. Then M satisfies the rule r*

$$C_1 \times \ldots \times C_n \leftarrow b_1, \ldots, b_m, \ not\ b_{m+1} \ldots, \ not\ b_{m+k}$$

- *to degree 1 if $b_j \notin M$ for some j $(1 \leq j \leq m)$, or $b_i \in M$ for some i $(m+1 \leq i \leq m+k)$,*
- *to degree j $(1 \leq j \leq n)$ if all $b_l \in M$ $(1 \leq l \leq m)$, no $b_i \in M$ $(m+1 \leq i \leq m+k)$, and $j = min\{r \mid C_r \in M, 1 \leq r \leq n\}$.*

The satisfaction degree of an answer set M *w.r.t.* a rule, denoted by $deg_M(r)$, provides a ranking of the answer sets of a LPOD, and a preference order on the answer sets can be obtained using some proposed combination strategies. In [3], the authors have proposed three criteria for comparing answer sets, respectively *cardinality*, *inclusion* and *Pareto*. In this paper we show how we maintain the three criteria and extend them for possibilistic answer sets.

LPODs allow to represent preferences which can depend on incomplete knowledge. As LPODs are based on extended nonmonotonic logic incomplete information can be expressed by means of default negation. We provide here a simple example.

Example 2. We want to express the preferences of a tourist, visiting the city of Barcelona. She is interested in getting restaurant information. She normally prefers Mexican to Italian food if she does not have any information whether Mexican ot Italian restaurants which can be found are goodly or badly reputated. One way to represent the tourist preferences is by means of a LPOD.
Let P be an ordered disjunction program expressing the tourist preferences about restaurants:[1]

$\mathbf{r_1}$: $rest(mex, \_) \times rest(ita, \_) \leftarrow \ not\ info(ita, \_),\ not\ info(mex, \_)$.
$\mathbf{r_2}$: $rest(ita, \_) \times rest(mex, \_) \leftarrow info(ita, \_)$.
$\mathbf{r_3}$: $rest(mex, \_) \times rest(ita, \_) \leftarrow info(mex, \_)$.
$\mathbf{r_4}$:$\leftarrow rest(mex, \_), rest(ita, \_)$.

We can see that there are two answer sets satisfying the program,

$M_1 = \{rest(mex, \_)\}$
$M_2 = \{rest(ita, \_)\}$

and according to their satisfaction degrees $M_1 >_p M_2$ and thus M_1 is preferred to M_2 in all the three comparison criteria.

As shown in the example we have obtained a global order among the user preferences. Sometimes, when defining preferences in ordered disjunction programs, it may happen that conflicting preference rules drive the establishment of more than one preferred answer set.

[1] The syntax of the example is based on the syntax of *psmodels*, an implementation of ordered disjunction using answer set solvers for normal programs [3].

Example 3. Let P be the ordered disjunction program of Example 2 where information about Mexican and Italian restaurant is added:

$\mathbf{r_1} : rest(mex, \_) \times rest(ita, \_) \leftarrow \ not\ info(ita, \_),\ not\ info(mex, \_).$
$\mathbf{r_2} : rest(ita, \_) \times rest(mex, \_) \leftarrow info(ita, \_).$
$\mathbf{r_3} : rest(mex, \_) \times rest(ita, \_) \leftarrow info(mex, \_).$
$\mathbf{r_4} :\leftarrow rest(mex, \_), rest(ita, \_).$
$\mathbf{r_5} : info(ita, \_).$
$\mathbf{r_6} : info(mex, \_).$

We can see that again, there are two answer sets satisfying the program,

$M_1 = \{rest(mex, \_), info(mex, \_), info(ita, \_)\}$
$M_2 = \{rest(ita, \_), info(mex, \_), info(ita, \_)\}$

but now M_1 and M_2 are not comparable anymore in any of the three criteria, as $M_1 \not\succ_c M_2$ and $M_2 \not\succ_c M_1$, $M_1 \not\succ_i M_2$ and $M_2 \not\succ_i M_1$, $M_1 \not\succ_p M_2$ and $M_2 \not\succ_p M_1$.

In such a situation, a mechanism that can express the difference in preference importance is needed. The authors in [3] introduce meta-preferences by defining a relation $\succ$ on rules and incorporate this relation in the comparison criteria. Instead, in our approach we aim at considering the degree of uncertainty as a measure for deciding whether a rule should be considered or not. We will show how our extension with possibilistic disjunction can in some cases overcome the problem considering the necessity-value associated to the possibilistic rules. Before that, we first provide the reader with the formal definition of possibilistic ordered disjunction.

3 Logic Programs with Possibilistic Ordered Disjunction

In this section we propose the syntax and semantics of possibilistic ordered disjunction programs.

3.1 Syntax

The syntax of a possibilistic ordered disjunction program is based on the syntax of ordered disjunction rules (Section 2.5) and of possibilistic logic (Section 2.3). A *possibilistic atom* is a pair $p = (a, q) \in \mathcal{A} \times \mathcal{Q}$ where $\mathcal{A}$ is a set of atoms and $(\mathcal{Q},\leq)$ a finite lattice.[2] The projection $*$ for any possibilistic atom p is defined as: $p^* = a$. Given a set of possibilistic atoms M, the generalization of $*$ over M is defined as: $M^* = \{p^* \mid p \in M\}$. Given $(\mathcal{Q},\leq)$, a possibilistic ordered disjunction rule r is of the form:

$$\alpha : C_1 \times \ldots \times C_n \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$$

where $\alpha \in \mathcal{Q}$ and $C_1 \times \ldots \times C_n \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$ is an ordered disjunction rule as defined in Section 2.5 with $\mathcal{B}^+ = \{b_1, \ldots, b_m\}$ and $\mathcal{B}^- = \{b_{m+1}, \ldots, b_{m+k}\}$.

[2] In the paper we will consider only finite lattices.

The projection $*$ for a possibilistic ordered disjunction rule r, is $r^* = C_1 \times \ldots \times C_n \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$. $n(r) = \alpha$ is a necessity degree representing the certainty level of the information described by r. A possibilistic constraint c is of the form:

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

where $\mathcal{TOP}_{\mathcal{Q}}$ is the top of the lattice $(\mathcal{Q},\leq)$ and $\leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$ is a constraint. Observe that any possibilistic constraint must have the top of the lattice $(\mathcal{Q},\leq)$. This restriction is motivated by the fact that, like constraints in standard Answer Set Programming, the purpose of the possibilistic constraint is to eliminate possibilistic models. Hence, it is assumed that there is no uncertainty about the information captured by a possibilistic constraint. As in possibilistic ordered disjunction rules, the projection $*$ for a possibilistic constraint c is $c^* =\leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$.

A Logic Program with Possibilistic Ordered Disjunction (LPPOD) is a tuple of the form $P := \langle (Q, \leq), N\rangle$ such that N is a finite set of possibilistic ordered disjunction rules and possibilistic constraints. The generalization of $*$ over P is defined as follows: $P^* := \{r^* \mid r \in N\}$. Notice that P^* is an ordered disjunction logic program. Given a possibilistic ordered disjunction program $P := \langle (Q, \leq), N\rangle$, we define the α-cut of P denoted by P_α as [6]:

$$P_\alpha := \{r \mid r \in P, n(r) \geq \alpha\}$$

Example 4. Let $P = \langle (Q, \leq), N\rangle$ be a possibilistic ordered disjunction program such that $\mathcal{Q} = (\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\},\ \leq)$, $\leq$ be the standard relation between rational numbers, and N be the set of possibilistic rules of the program of Example 2 with possibilistic values $\alpha \in \mathcal{Q}$ associated to each preference rule and constraint:

$r_1 = \mathbf{0.5} : rest(mex, \_) \times rest(ita, \_) \leftarrow not\ info(ita, \_),\ not\ info(mex, \_).$
$r_2 = \mathbf{0.3} : rest(ita, \_) \times rest(mex, \_) \leftarrow info(ita, \_).$
$r_3 = \mathbf{0.1} : rest(mex, \_) \times rest(ita, \_) \leftarrow info(mex, \_).$
$r_4 = \mathbf{1} :\ \leftarrow rest(mex, \_), rest(ita, \_).$
$r_5 = \mathbf{0.9} : info(ita, \_).$
$r_6 = \mathbf{0.6} : info(mex, \_).$

The way the uncertainty values of the rules are generated is described in Section 5.

3.2 Semantics

Our semantics are based on the definition of answer set semantics for extended normal programs (see Section 2.2). We will consider sets of possibilistic atoms as interpretations. Hence, before defining the possibilistic ordered disjunction logic programming semantics, we introduce basic operations between sets of possibilistic atoms and a relation of order between them.

Definition 6. *Given $\mathcal{A}$ a finite set of atoms and $(\mathcal{Q},\leq)$ be a lattice, we consider $\mathcal{PS} = 2^{\mathcal{A}\times\mathcal{Q}}$ as the finite set of all the possibilistic atoms sets induced by $\mathcal{A}$ and $\mathcal{Q}$. Let $A, B \in \mathcal{PS}$, hence we define the operators $\sqcap$, $\sqcup$ and $\sqsubseteq$ as follows:*

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

We will define a possibilistic ordered disjunction semantics which is close to the proof theory of possibilistic logic and answer set semantics. As in answer set semantics definition, our approach is based on a syntactic reduction. The following reduction is a possibilistic extension of the reduction defined in [3].

Definition 7. (Reduction $r_\times^M$)
Let $r = \alpha : C_1 \times \ldots \times C_n \leftarrow \mathcal{B}^+$, not $\mathcal{B}^-$ be a possibilistic ordered disjunction clause and M be a set of atoms. The $\times$-possibilistic reduct $r_\times^M$ is defined as follows:

$$r_\times^M := \{\alpha : C_i \leftarrow \mathcal{B}^+ | C_i \in M \text{ and } M \cap (\{C_1, \ldots, C_{i-1}\} \cup \mathcal{B}^-) = \emptyset\}$$

.

Definition 8. (Reduction $P_\times^M$)
Let $P = \langle (Q, \leq), N\rangle$ be a possibilistic ordered disjunction program and M be a set of atoms. The $\times$-possibilistic reduct $P_\times^M$ is defined as follows:

$$P_\times^M = \bigcup_{r \in N} r_\times^M$$

Example 5. Let P be the possibilistic ordered disjunction of Example 4 and let M be a possibilistic set, M=$\{(rest(ita,\_), 0.3), (info(mex,\_), 0.6), (info(ita,\_), 0.9)\}$. We can see that $P_\times^M$ is:

$\mathbf{0.3} : rest(ita,\_) \leftarrow info(ita,\_)$. $\mathbf{0.1} : rest(ita,\_) \leftarrow info(mex,\_)$. $\mathbf{0.9} : info(ita,\_)$.
$\mathbf{0.6} : info(mex,\_)$.

Observe that the program $P_\times^M$ is a possibilistic positive extended logic program.[3] Once a possibilistic ordered disjunction program P has been reduced by a set of possibilistic atoms 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 $P = \langle (Q, \leq), N\rangle$ be a possibilistic ordered disjunction program and M be a set of possibilistic atoms such that M^* is an answer set of $(P_\times^{M^*})^*$. M is a possibilistic answer set of P if and only if $P_\times^{M^*} \vdash_{PL} M$ and $\nexists M' \in \mathcal{PS}$ such that $M' \neq M$, $P_\times^{(M')^*} \vdash_{PL} M'$ and $M \sqsubseteq M'$.

[3] A positive program is a program without negated as failure atoms.

Example 6. Let P be the possibilistic program of Example 4 and M the possibilistic set of atoms introduced in Example 5. First of all we can see that M^* is an answer set of the reduced extended positive program $(P_\times^{M^*})^*$ as $M^* = \{rest(ita, \_), info(mex, \_), info(ita, \_)\}$ is answer set of

$(P_\times^{M^*})^*$:
$rest(ita, \_) \leftarrow info(ita, \_).$
$rest(ita, \_) \leftarrow info(mex, \_).$
$info(ita, \_).$
$info(mex, \_).$

Hence in order to prove that M is a possibilistic answer set of P, we have to verify that $P_\times^{M^*} \vdash_{PL} M$. This means that, for each possibilistic atom $p \in M$, $P_\times^{M^*} \vdash_{PL} p$. It is straightforward to see that

$$P_\times^{M^*} \vdash_{PL} \{(rest(ita, \_), 0.3), (info(mex, \_), 0.6), (info(ita, \_), 0.9)\}$$

Let us prove (*rest(ita_)*,**0.3**) from $P_\times^{M^*}$.

Premises from $P_\times^{M^*}$
1. $info(ita, \_) \rightarrow rest(ita, \_)$ **0.3**
2. $info(mex, \_) \rightarrow rest(ita, \_)$ **0.1**
3. $info(ita, \_)$ **0.9**
4. $info(mex, \_)$ **0.6**

From 4 and 2 by GMP
5. $rest(ita, \_)$ **0.1**

From 3 and 1 by GMP
6. $rest(ita, \_)$ **0.3**

From 5 and 6 by S
7. $rest(ita, \_)$ **0.3**

The inference of the possibilistic atoms (*info(mex,_)*,**0.6**) and (*info(ita,_)*,**0.9**) is trivial. Therefore, we can say that $P_\times^{M^*} \vdash_{PL} M$ is true. Notice that it does not exist a possibilistic set M' such that $M' \neq M$, $P_\times^{(M')^*} \vdash_{PL} M'$ and $M \sqsubseteq M'$, hence we can conclude that M is a possibilistic answer set of P.

From Definition 9, we can observe that there is an important condition *w.r.t.* the definition of a *possibilistic answer set* of a possibilistic ordered disjunction program: a possibilistic set S cannot be a possibilistic answer set of a possibilistic ordered disjunction program P, if S^* is not an answer set of the positive extended program $(P_\times^{M^*})^*$. The following proposition shows that our semantics is a generalization of the semantics of ordered disjunction programs introduced by Brewka in [3].

Proposition 1. *Let $P = \langle (Q, \leq), N \rangle$ be a possibilistic ordered disjunction program and M be a set of possibilistic atoms. If M is a possibilistic answer set of P then M^* is a answer set of P^*.*

Proof. (Sketch)
First of all we can observe that if M is a possibilistic answer set of P then, by Definition 9, M^* is answer set of $(P_\times^{M^*})^*$. By Proposition 1 of Brewka[4], we can prove that M^* is answer set of P^* if the following conditions are satisfied:

1. $M^* = Cn((P_\times^{M^*})^*)$
2. M^* is consistent, *i.e.* if $a \in M^*$ then $\neg a \notin M^*$
3. M^* satisfies every rule $r \in P^*$

Condition (1) is satisfied if we can show that the possibilistic reduction we defined in Defintion 7 and 8 is equivalent to the reduction of Brewka, *i.e.* $(P_\times^M) = (P_\times^{M^*})^*$. This is straightforward to see observing that our reduction is based on the reduction of Brewka and that the possibilistic value of the rule does not affect the reduction itself.
Condition (2) is verified by the way we manage the strong negation $\neg$ in our logic programs (see Section 2.1). Basically, each atom $\neg a$ is replaced by a new atom symbol a' which does not appear in the language of the program and we add the constraint to the program $\leftarrow a, a'$. In this way M^* is consistent.
Condition (3) follows from the definition of answer set of M^*. M^* in fact satisfies every rule $r^* \in (P_\times^{M^*})^*$ and thus satisfies every rule $r^* \in P^*$ as well.

□

4 Possibilistic Preferred Answer Sets

To distinguish between preferred possibilistic answer sets, we take the definition and the notation of the satisfaction degree of M *w.r.t.* a rule r as $deg_M(r)$ (see Section 2.5). In the following we define three comparison criteria (as in [3]), adapted to possibilistic answer sets. We prove how the three relations maintain their implication property and we show how sometimes possibilistic ordered disjunction programs can reach a single preferred solution, when ordered disjunction program could not. We first define the set of possibilistic atoms M satisfying a degree i as follows:

Definition 10. *Let M be a set of possibilistic atoms and P be a possibilistic ordered disjunction logic program. Then $M^{i,\alpha}(P) = \{r \in P \mid deg_M(r) = i$ and $n(r) \geq \alpha\}$.*

Given a set of possibilistic atoms M, $n(M)$ is defined as $min\{\alpha \mid (a, \alpha) \in M\}$. We can now define the three preference relations. The possibilistic version of cardinality-based preference can be defined as follows:

Definition 11. *Let M_1 and M_2 be possibilistic answer sets of a possibilistic ordered disjunction logic program P. M_1 is possibilistic cardinality-preferred to M_2, $(M_1 >_{pc} M_2)$ iff $\exists$ i such that $\mid M_1^{i,\alpha}(P) \mid > \mid M_2^{i,\alpha}(P) \mid$ and $\forall j < i$, $\mid M_1^{j,\alpha}(P) \mid = \mid M_2^{j,\alpha}(P) \mid$, where $\alpha = min\{n(M_1), n(M_2)\}$.*

[4] Let P be an LPOD and M a set of atoms. Then, M is answer set of P iff the following three conditions hold: (a) $M = Cn(P_\times^M)$; (b) M is consistent; (c) M satisfies every rule $r \in P$ [3].

We define the inclusion-based preference as:

Definition 12. *Let M_1 and M_2 be possibilistic answer sets of a possibilistic ordered disjunction logic program P. M_1 is possibilistic inclusion-preferred to M_2, $(M_1 >_{pi} M_2)$ iff $\exists\, k$ such that $M_2^{k,\alpha}(P) \subset M_1^{k,\alpha}(P)$ and $\forall\, j < k$, $M_1^{j,\alpha}(P) = M_2^{j,\alpha}(P)$, where $\alpha = min\{n(M_1), n(M_2)\}$.*

Lastly, the possibilistic Pareto-based preference is:

Definition 13. *Let M_1 and M_2 be possibilistic answer sets of a possibilistic ordered disjunction logic program P. M_1 is possibilistic pareto-preferred to M_2, $(M_1 >_{pp} M_2)$ iff $\exists\, r \in P$ such that $deg_{M_1}(r) < deg_{M_2}(r)$, and $\nexists r' \in P$ such that $deg_{M_1}(r') > deg_{M_2}(r')$, and $n(r) \geq max\{n(M_1), n(M_2)\}$.*

Although we adapted the three relations to possibilitic answer sets, the three relations maintain their inclusion property.

Proposition 2. *Let M_1 and M_2 be possibilistic answer sets of a possibilistic ordered disjunction logic program P. Then $(M_1 >_{pp} M_2)$ implies $(M_1 >_{pi} M_2)$ and $(M_1 >_{pi} M_2)$ implies $(M_1 >_{pc} M_2)$.*

Proof. $(M_1 >_{pp} M_2) \Rightarrow (M_1 >_{pi} M_2)$. If $(M_1 >_{pp} M_2)$ holds then the set of rules with degree i of M_1 contains at least one element more than the set of rules with degree i of M_2. Thus $(M_1 >_{pi} M_2)$ follows.
$(M_1 >_{pi} M_2) \Rightarrow (M_1 >_{pc} M_2)$. If $(M_1 >_{pi} M_2)$ holds, then the number of the elements of the set of rules with degree i of M_1 is bigger than the number of the element of the set of rules with degree i of M_2. Thus $(M_1 >_{pc} M_2)$ follows.
□

In Section 2.5, we have presented (by means of Example 3) a situation where two preferred answer set are generated by an ordered disjunction program. Using the above comparison criteria, now we want to show how a possibilistic ordered disjunction program can manage to select one single preferred answer set.

Example 7. Let P be the possibilistic ordered disjunction program of Example 4. We have seen in the previous section that P has two possibilistic answer sets:

$M_1 = \{(rest(mex, \_), 0.3), (info(mex, \_), 0.6), (info(ita, \_), 0.9)\}$
$M_2 = \{(rest(ita, \_), 0.3), (info(mex, \_), 0.6), (info(ita, \_), 0.9)\}$

We can see that $n(M_1) = 0.3$ and $n(M_2) = 0.3$. Let us define now the $M_1^{i,0.3}(P)$ and $M_2^{i,0.3}(P)$, *i.e.* we consider only the rules of program P where $n(r) \geq 0.3$. In this way we obtain:
$M_1^{1,0.3}(P) = \{r_1, r_4, r_5, r_6\} \qquad M_2^{1,0.3}(P) = \{r_1, r_2, r_4, r_5, r_6\}$

We can now conclude that $(M_2 >_{pc} M_1)$, $(M_2 >_{pi} M_1)$ and $(M_2 >_{pp} M_1)$ applying the possibilistic version of the comparison criteria one by one.

In this case we have obtained an order between the answer set of the possibilistic ordered disjunction program P, cutting one of the conflicting preference

rules. However, generally speaking, we have been able to cover only some cases. Particularly only the cases where the uncertainty values of the conflicting preference rules are lower than the value of the uncertain information added.

Nevertheless, we can observe that there is an important property *w.r.t.* the comparison criteria for *possibilistic answer sets* of a possibilistic ordered disjunction program: a possibilistic set M is comparable if M^* is comparable in the ordered disjunction program P^*. Therefore our extension maintains the preference relation between answer sets of ordered disjunction programs.

Proposition 3. *Let M_1 and M_2 be possibilistic answer sets of a possibilistic ordered disjunction logic program P. Then*

$$\text{If } M_1^* >_c M_2^* \text{ then } M_1 >_{pc} M_2 \qquad (1)$$
$$\text{If } M_1^* >_i M_2^* \text{ then } M_1 >_{pi} M_2 \qquad (2)$$
$$\text{If } M_1^* >_p M_2^* \text{ then } M_1 >_{pp} M_2 \qquad (3)$$

Proof. (Sketch) We will prove (1). The proofs for (2) and (3) are similar. To start let us observe that:

(a) $M_1^* >_c M_2^*$ iff $\exists\ i$ such that $\mid M_1^{*^i}(P^*) \mid > \mid M_2^{*^i}(P^*) \mid$ and $\forall j < i$, $\mid M_1^{*^j}(P^*) \mid = \mid M_2^{*^j}(P^*) \mid$ by Definition 9 in [3]

(b) if M_1 and M_2 are possibilistic answer sets of P, then by Proposition 1, M_1^* and M_2^* are answer sets of P^*

(c) $\forall r \in P$, $deg_{M_1^*}(r^*) = deg_{M_1}(r)$ and $deg_{M_2^*}(r^*) = deg_{M_2}(r)$

Let us assume that (1) is false. Then by contradiction, we have two cases: either (i) $M_1 \not>_{pc} M_2$ or (ii) $M_2 >_{pc} M_1$.

Case 1. $M_1 \not>_{pc} M_2$. If $M_1 \not>_{pc} M_2$ holds then it means that $\exists\ i$ such that $\mid M_1^{i,\alpha}(P) \mid = \mid M_2^{i,\alpha}(P) \mid$ and $\forall j < i$, $\mid M_1^{j,\alpha}(P) \mid = \mid M_2^{j,\alpha}(P) \mid$ where $\alpha = min\{n(M_1), n(M_2)\}$. By (a) this can only happen if $\exists r' \in P$ and $r' \notin P_\alpha$, *i.e.* a rule has been cut. But this it would mean that $deg_{M_1}(r') < deg_{M_2}(r')$ and $n(r') < min\{n(M_1), n(M_2)\}$. But for observation (c) it is absurde.

Case 2. $M_2 >_{pc} M_1$. If $M_2 >_{pc} M_1$ holds then it means that $\exists\ i$ such that $\mid M_1^{i,\alpha}(P) \mid < \mid M_2^{i,\alpha}(P) \mid$ and $\forall j < i$, $\mid M_1^{j,\alpha}(P) \mid = \mid M_2^{j,\alpha}(P) \mid$ where $\alpha = min\{n(M_1), n(M_2)\}$. By (a) this can only happen only if $\exists r', r'' \in P$ and $r', r'' \notin P_\alpha$, *i.e.* two rules have been cut. But this it would mean that $deg_{M_1}(r') < deg_{M_2}(r')$, $deg_{M_1}(r'') < deg_{M_2}(r'')$ and $n(r')$, $n(r'') < min\{n(M_1), n(M_2)\}$. We have already shown in the former case that we can reach an absurde. Then (1) is true.

□

5 An Application Scenario

Although logic programs with ordered disjunction are a flexible specification language able to capture common users preferences, in some realistic scenarios the preferences should be linked in some way to the evidence of the information

in order to cope with uncertainty when trying to reach a single preferred solution. In this section we describe an application scenario of possibilistic ordered disjunction logic programs which can be incorporated to the user recommender system described in [10]. We show how the scenario can benefit of possibilistic ordered disjunction programs presented in this paper. Though the scenario is settled in the context of user recommendation, it can be generalized as the problem of viewing the impact the evidence has about agent preferences or beliefs.

5.1 Architecture

Figure 1 shows a simple multi-agent system supporting our scenario. The system consists essentially of three types of agents.

Personal Assistant Agents (PAAs) assist users, interpreting their preferences and generating LPODs. They are responsible of user preference elicitation. In our system, we represent only one PAA which communicates with a Broker Agent. In the following we assume to have already the user's preferences encoded as a LPOD.

Broker Agents are responsible of processing user preference received from PAAs and retrieving information from Crawler Agents. They are cognitive agents with a possibilistic program generator module and their decision making is extended to reason about possibilistic ordered disjunction logic programs. We consider only one instance of this type of agent.

Crawler Agents are basically information gatherers. They can be assigned to recommendation services in the Web and respond to Broker Agents queries, retrieving recommendation lists. In this scenario we consider only one Crawler Agent assigned to a recommendation service such as TripAdvisor[5].

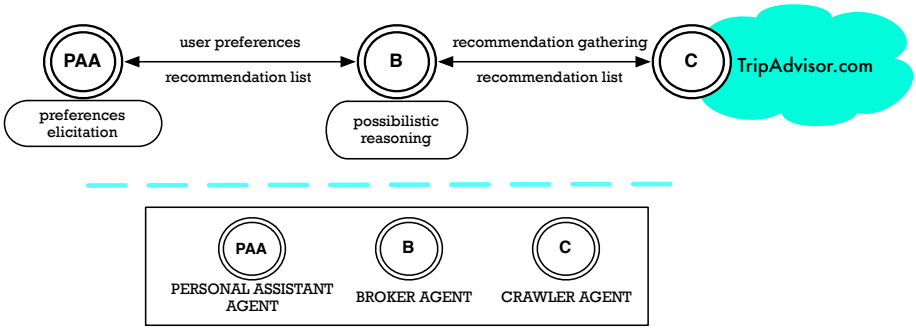

Figure 1. General Architecture

The actors of the system interact as follows:

[5] `http://www.tripadvisor.com/`

1. The Personal Assistant Agent builds the Ordered Disjunction Logic Program representing the user preferences and sends it to the Broker Agent.
2. The Broker Agent receives ordered disjunction programs and queries the Crawler agent according to the user preference list obtained.
3. The Crawler Agent processes queries and retrieves a list of recommendations.
4. The Broker Agent interprets the result of the Crawler, generates a possibilistic ordered disjunction program, executes it and sends the recommendation to the Personal Assistant Agent.
5. The User Agent processes the result and presents the recommendation to the user.
6. Restart the process from 1 if the end-user decides to specify another recommendation search.

The components of this multi-agent system can be deployed in distributed heterogeneous environments and integrated with distributed technologies, such as Web Services [11,10]. We assume that the Broker, the Personal Assistant and the Crawler agents share some domain knowledge and ontology. The agents communicate using a well defined communication FIPA-based specification [7] composed of performatives and protocols.

5.2 Possibilistic Ordered Disjunction Reasoning

Until now we have presented the generic architecture of the scenario, saying that the Broker Agent is able to create and to process possibilistic ordered logic programs. We show now how we intend to generate a possibilistic ordered disjunction program and exploit the possibilistic semantics we defined in Section 3.2.

Figure 2 shows the meta-architecture of the Broker. The Broker internally consists of four components: a *LPOD Reasoner*, a *Recommendation Gatherer*, a *Possibilistic LPOD Generator*, and a *Possibilistic LPOD Reasoner*. In the following we describe the functionality of each components referring to the examples defined in the paper. Let us consider the LPOD representing user preferences about restaurants in Example 2.

The *LPOD Reasoner* executes the program (using *psmodels*[6] for instance), infers the answer sets and applies the Pareto comparison. In this way $M_1 = \{rest(mex, \_)\}$ and $M_2 = \{rest(ita, \_)\}$, where $M_1 >_p M_2$ are obtained. This preference order is kept when searching for restaurant information.

The *Recommendation Gatherer* component processes the preference list according to the order relation they have. First it tries to fulfil the most preferred ones and then the less preferred. For each answer set, the Recommendation Gatherer queries the Crawler Agent, where each query is made using the literals which compound the answer set as parameter list. For each query, a result about recommendation of restaurant places is returned, such that every record is associated with a recommendation value. The recommendation value spans between

[6] `http://www.tcs.hut.fi/Software/smodels/priority/`

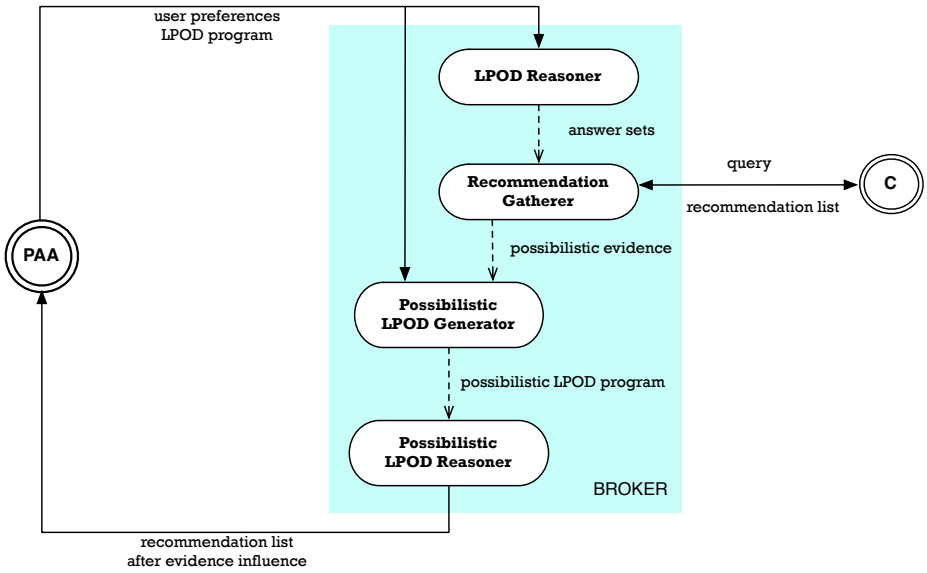

Figure 2. The Possibilistic Broker conceptual architecture and interaction with other component in the system

0 and 10, and represents the computed mean reputation value of the restaurant according to other users votes.

The *Possibilistic LPOD Generator* interprets this value as the evidence that a given restaurant is goodly or badly reputated. It takes the maximum value of each of the result set (each set is about a restaurant type), and represents it as a possibilistic logic program. According to the syntax in Section 3.1, we obtain as result:

$\mathbf{0.6} : info(mexican,' LaCoronela').$
$\mathbf{0.9} : info(italian,' LaBellaNapoli').$

where the uncertainty values are normalized to the interval to which α belongs, *i.e.* $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. The Possibilistic LPOD generator builds then the possibilistic version of the LPOD which looks like the possibilistic ordered disjunction program of Example 4. The uncertainty degree associated to the preference rules can be the value computed by some reinforcement learning algorithms in which previous executions are considered.

The *Possibilistic LPOD Reasoner* computes the answer sets of the Possibilistic LPOD program. According to the semantics defined in Section 3.2 and applying the possibilistic version of the Pareto-based criteria, the reasoning infers that

$M_1 = \{(rest(mex,$'La Coronela'$), 0.3), info(mex,$'La Coronela'$), 0.6), (info(ita,$'La Bella Napoli'$), 0.9)\}$

$M_2 = \{(rest(ita,$'La Bella Napoli'$), 0.3), info(mex,$'La Coronela'$), 0.6), (info(ita,$'La Bella Napoli'$), 0.9)\}$

where $(M_2 >_{pp} M_1)$, as shown in Example 7.

This result is returned as output of the possibilistic reasoning to the PAA, which shows the recommendation result to the user.

6 Conclusions

In this paper we have introduced possibilistic ordered disjunction programs as an extension of ordered disjunction programs [3]. This approach is able to capture incomplete information and incomplete states of a knowledge base at the same time. We have shown how realistic scenarios require to cope with the degree of uncertain knowledge and how it is possible to handle it with a combination of possibilistic logic with answer set semantics. We have described an application scenario as an example where such uncertainty is present and have shown how the scenario can benefit from our approach.

We have formulated some important definitions and propositions for possibilistic ordered disjunction programs. In particular, we guarantee by Proposition 1 that valid models of our extended version are valid solutions for no possibilistic programs as well, showing how our semantics is a generalization of the original semantics of ordered disjunction programs. We have generalized the comparison criteria presented in [3] for preferred possibilistic answer sets (Definition 11, 12, and 13), taking into account the cut determined by the uncertainty degree of the possibilistic answer sets according to the possibilistic answer set semantics, our extension is based on. By considering the minimum uncertainty values of possibilistic answer sets in the comparison criteria, we have been able to show how in some cases excluding possibilistic rules behind this cut, we are able to reach a single preferred possibilistic answer set, whereas the ordered disjunction program cannot.

However, as we already said, we have not reached a general solution. In fact it can happen that the uncertainty degree of possibilistic answer sets does not produce any significant cut. Applying different well-studied semantics for possibilistic logic programs such as, possibilistic pstable model semantics [14], we aim to produce a different cut and explore its impacts. Moreover we think that a recursive cut algorithm which considers the influence the uncertainty has on preference rules to apply recursive cuts, can eventually reach a global order between valid answer sets. Therefore we are especially interested to work out the influence the uncertainty degree of the new information can play in the preference order establishment. In fact our intuition tells us that the uncertainty value of the information added should affect the uncertainty value of the preference rules where such information appears in rules heads.

Acknowledgements

This work has been funded mainly by the European Commission Framework 7 funded project ALIVE (FP7-215890). Javier Vázquez-Salceda's work has been also partially funded by the Ramón y Cajal program of the Spanish Ministry of Education and Science. The opinions of the authors do not reflect the opinions of the European Commission.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
2. R. Brafman and C. Domshlak. Preference Handling - An Introductory Tutorial. *AI Magazine*, 30(1), 2009.
3. G. Brewka, I. Niemelä, and T. Syrjänen. Logic Programs with Ordered Disjunction. *Computational Intelligence*, 20(2):333–357, 2004.
4. G. Brewka, I. Niemelä, and M. Truszczyński. Preferences and Nonmonotonic Reasoning. *AI Magazine*, 29(4):69–78, 2008.
5. B. A. Davey and H. A. Priestly. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002.
6. D. Dubois, J. Lang, and H. Prade. Possibilistic Logic. In D. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pages 439–513. Oxford University Press, Oxford, 1994.
7. FIPA. Foundation for Intelligent Physical Agents. http://www.fipa.org.
8. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.
9. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
10. I. Gómez-Sebastià, M. Palau, J. C. Nieves, J. Vázquez-Salceda, and L. Ceccaroni. Dynamic Orchestration of Distributed Services on Interactive Community Displays: The ALIVE approach. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, volume 55 of *Advances in Intelligent and Soft Computing*, 2009.
11. M. N. Huhns. Agents as Web Services. *IEEE Internet Computing*, 6(4):93–95, 2002.
12. 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.
13. 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 (ICLP-07 Workshop)*, pages 271–284, 2007.
14. 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.