Semantics for Possibilistic Disjunctive Programs

Juan Carlos Nieves[1], Mauricio Osorio[2], and Ulises Cortés[1]

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

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

1 Introduction

As Tversky and Kahneman observed in [12], many decisions that we make in our common life are based on beliefs concerning the likelihood of uncertain events. In fact, we commonly use statements such as "I think that . . . ", "chances are . . . ", "it is *probable* that . . . ", "it is *plausible* that . . . ", *etc.*, for supporting our decisions. In this kind of statements usually we appeal to our experience or our commonsense. It is not surprising to think that a reasoning based on these kind of statements could reach *bias conclusions*. However these conclusions could reflect the experience or commonsense of an expert. Pelletier and Elio pointed out in [8] that people simply have tendencies to ignore certain information because of the (evolutionary) necessity to make decisions quickly. This gives rise to "biases" in judgments concerning what they "really" want to do.

In view of the fact that we know that a reasoning based on statements which are quantified by relative likelihoods could capture our experience or our commonsense, the question is: how could these statements be captured by real application systems like Multi Agent Systems? For those steeped in probability, Halpern has remarked in [6] that probability has its problems. For one thing, the numbers are not always available. For another, the commitment to numbers means that any two events must be comparable in terms of their probabilities: either one event is more probable than the other, or they have equal probability. Now, the question is why not to use explicitly labels like *possible*, *probable*, *plausible*, *etc.*, for capturing the incomplete state of a belief in a logic program when the numerical representations are not available or difficult to get.

In [7], it was proposed a possibilistic 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. Nicolas *et al.*'s approach is based on the concept of *possibilistic stable model* which defines a semantics for possibilistic normal logic programs. One weak point of this approach is that it relies on the expressiveness of normal logic programs and it always depends of a numerical representation for capturing the incomplete state of a belief.

In this paper, we introduce the use of *possibilistic disjunctive clauses* which are able to capture *incomplete information* and *incomplete states of a knowledge base* at the same time. It is important to point out that our approach is not exactly a generalization of Nicolas *et al.*'s approach since our semantics is based on an operator $\mathcal{T}$ which is inspired in partial evaluation [2] and an inference rule of possibilistic logic [3]. Also whereas Nicolas *et al.*'s approach only permits to express the states of a belief by totally ordered sets, our approach permits to consider partially ordered sets for expressing the states of a belief. Moreover we does not adopt to use *strict α-cuts* for handling an inconsistent possibilistic logic program. However our approach in the class of possibilistic normal logic programs coincides with Nicolas *et al.*'s approach when it considers totally ordered sets for capturing the incomplete state of a belief and the possibilistic program is consistent.

By considering partially ordered sets, it is possible to capture the confidence of a claim by using quantifies like the Toulmin's famous "quantifies"[11]. For instances, in [4] Fox and Modgil discuss the expressiveness of these quantifiers for capturing the uncertainty of medical claims. We use relative likelihoods for modeling different quantifiers *e.g., certain*, *confirmed*, *probable*, *plausible*, *supported* and *open*[1], where each quantifier is a possible world/class of beliefs. The user can provide a likelihood ordering for the worlds/classes of beliefs as it is shown in Fig. 1.

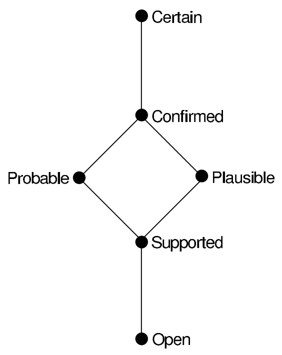

Fig. 1. A lattice where the following relations hold: $Open \preceq Supported$, $Supported \preceq Plausible$, $Supported \preceq Probable$, $Probable \preceq Confirmed$, $Plausible \preceq Confirmed$, and $Confirmed \preceq Certain$

The rest of the paper is divided as follows: In the next section, it is presented the syntax and semantics of our possibilistic framework. In the last section, we present our conclusions.

[1] This set of labels was taken from [4].

2 Possibilistic Disjunctive Logic Programs

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

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. Also we will assume some background on ASP. Mainly we will assume knowledge on the syntax and semantics of *extended disjunctive logic programs* (see [5] for definitions).

2.1 Syntax

First of all, we start defining some relevant concepts[2]. A *possibilistic literal* is a pair $l = (a, q) \in L \times Q$, where L is a finite set of literals and $(Q, \leq)$ is a lattice. We apply the projection $*$ over l as follows: $l^* = a$. Given a set of possibilistic literals S, we define the generalization of $*$ over S as follows: $S^* = \{l^* | l \in S\}$. Given a lattice $(Q, \leq)$ and $S \subseteq Q$, $LUB(S)$ denotes the least upper bound of S and $GLB(S)$ denotes the greatest lower bound of S. A possibilistic disjunctive logic program is a finite set of possibilistic disjunctive clauses of the form:

$$r = (\alpha : \ l_1 \vee \ldots \vee l_m \leftarrow l_1, \ldots, l_j, not\ l_{j+1}, \ldots, not\ l_n)$$

where $\alpha \in Q$. The projection $*$ over r is the *extended disjunctive clause* $r^* = l_1 \vee \ldots \vee l_m \leftarrow l_1, \ldots, l_j, not\ l_{j+1}, \ldots, not\ l_n$. $n(r) = \alpha$ is a necessity degree representing the certainty level of the information described by r (see [3] for a formal definition of n). If P is a possibilistic disjunctive logic program, then $P^* = \{r^* | r \in P\}$ is *an extended disjunctive program.* Given an extended disjunctive clause *C*, we denote *C* by $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$, where $\mathcal{A}$ contains all the head literals, $\mathcal{B}^+$ contains all the positive body literals and $\mathcal{B}^-$ contains all the negative body literals.

2.2 Semantics

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

Definition 1 (Reduction P^M). *Let P be a possibilistic disjunctive logic program, M be a set of literals. P reduced by M is the positive possibilistic disjunctive program:*
$P^M := \{(n(r) : \mathcal{A} \cap M \leftarrow \mathcal{B}^+) | r \in P, \mathcal{A} \cap M \neq \emptyset, \mathcal{B}^- \cap M = \emptyset, \mathcal{B}^+ \subseteq M\}$
where r^* is of the form $\mathcal{A} \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$.

Notice that $(P^*)^M$ is not exactly the *Gelfond-Lifschitz reduction*. In fact, our reduction is stronger that Gelfond-Lifschitz reduction when P^* is a disjunctive program [5]. One of the main differences is the condition $\mathcal{A} \cap M \neq \emptyset$ which suggests that any clause that does not have a true head literal is false.

[2] Some concepts presented in this subsection extend some terms presented in [7].

Once a possibilistic logic program P has been reduced by a set of literals M^*, it is possible to test whether M is a possibilistic answer set of the program P. In order to define a possibilistic answer set, we introduce an operator which is inspired in partial evaluation for disjunctive logic programs [2] and an inference rule of Possibilistic Logic [3].

Definition 2. *Let P be a possibilistic logic program. The operator $\mathcal{T}(P)$ is defined as follows:*

$$\mathcal{T}(P) := P \cup \left\{ \begin{array}{ll} r' & \textit{if } (\alpha : \mathcal{A} \leftarrow (\mathcal{B}^+ \cup \{B\}),\ not\ \mathcal{B}^-) \in P \textit{ and} \\ & (\alpha_1 : \mathcal{A}_1 \leftarrow \top) \in P \textit{ such that } B \in \mathcal{A}_1 \end{array} \right\}$$

where $r' := GLB(\{\alpha, \alpha_1\}) : \mathcal{A} \cup (\mathcal{A}_1 \setminus \{B\}) \leftarrow \mathcal{B}^+,\ not\ \mathcal{B}^-$.

Intuitively, the operator $\mathcal{T}$ is an inference rule for possibilistic disjunctive logic programs. For instance, let us consider the lattice of Fig. 1 and the possibilistic clauses: $probable : a \vee b \leftarrow \top$ and $confirmed : e \leftarrow b$. Then by applying the operator $\mathcal{T}$, one can get the new possbilistic clause $supported : e \vee a \leftarrow \top$. Also if we consider the possibilistic clauses: $probable : a \vee b \leftarrow \top$ and $plausible : a \leftarrow b$, one can get the new possibilistic clause $supported : a \leftarrow \top$. An important property of the operator $\mathcal{T}$ is that it always reaches a fix-point.

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

From any possibilistic program, it is possible to identify a set of possibilistic literals which we call Sem_{min}.

Definition 3. *Let P be a possibilistic logic program and $Facts(P, A) := \{(\alpha : A \leftarrow \top) | (\alpha : A \leftarrow \top) \in P\}$. $Sem_{min}(P) := \{(x, \alpha) | Facts(P, x) \neq \emptyset$ and $\alpha := LUB_{r \in Facts(P,x)}(n(r))\}$ where $x \in \mathcal{L}_P$.*

Notice that if a possibilistic literal is obtained by different possibilistic clauses, then the possibilistic part of the literal will be obtained by LUB. Now by considering the operator $\mathcal{T}$ and Sem_{min}, we define a posibilistic answer set of a possibilistic program as follows:

Definition 4 (Possibilistic answer set). *Let P be a possibilistic disjunctive logic program and M be a set of possibilistic literals such that M^* is an answer set of P^*. M is a possibilistic answer set of P if and only if $M = Sem_{min}(\Pi(P^{M^*}))$.*

We have to notice that there is an important condition *w.r.t.* the definition of the *possibilistic answer sets*. This is that a possibilistic set S is not a possibilistic answer set of a possibilistic logic program P if S^* is not an answer set of the extended logic program P^*. This condition guarantees that any clause of P^* is satisfied by M^*. In fact, when all the possibilistic clauses of a possibilistic program P have as certainty level the top of the lattice that was considered in P, the answer sets of P^* can be directly generalized to the possibilistic answer sets of P.

In the class of possibilistic normal logic programs[3], our definition of possibilistic answer set is closely related to the definition of possibilistic stable model presented in [7]. In fact, both semantics coincide.

Proposition 2. *Let P be a possibilistic normal logic program. M is a possibilistic answer set of P if and only if M is a possibilistic stable model.*

In terms of computability, we can observe that $\Pi(P)$ is computable.

Proposition 3. *Let P be a finite possibilistic disjunctive logic program. Suppose also that $(Q, \leq)$ is a finite lattice. Then $\Pi(P)$ is computable.*

The main implication of Proposition 3 is that the possibilistic answer sets of a possibilistis logic program are computable.

Proposition 4. *Given a possibilistic program P there exists an algorithm that compute the set of possibilistic answer sets of P.*

3 Conclusions

We have been working in the decision making process for deciding if a human organ is viable or not for being transplanted [10,9]. Our experience suggests that in our medical domain, we require a *qualitative* theory of default reasoning like ASP for modeling incomplete information and a *quantitative* theory like possibilistic logic for modeling uncertain events which always exist in the medical domain.

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

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

In conclusion, the possibilistic disjunctive logic programs define a possibilistic approach able to capture *incomplete information* and *incomplete states of a knowledge*

[3] A possibilistic logic program P is called possibilistic normal logic program if P^* is a normal program.

base. To the best of our knowledge this approach is the first one that deals with disjunctive programs and partially ordered sets in order to define a possibilistic disjunctive semantics.

Acknowledgement. We are grateful to anonymous referees for their useful comments. J.C. Nieves thanks to CONACyT for his PhD Grant. J.C. Nieves and U. Cortés were partially supported by the grant FP6-IST-002307 (ASPIC). The views expressed in this paper are not necessarily those of ASPIC consortium.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
2. S. Brass and J. Dix. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999.
3. 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. Fox and S. Modgil. From arguments to decisions: extending the toulmin view. In *Arguing on the Toulmin model: New essays on argument analysis and evaluation*. Argumentation Library series published by Kluwer Academic, Currently in press.
5. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
6. J. Y. Halpern. *Reasoning about uncertainty*. The MIT Press, 2005.
7. P. Nicolas, L. Garcia, I. Stéphan, and C. Lafèvre. Possibilistic Undertainty Handling for Answer Set Programming. *Annal of Mathematics and Artificial Intelligence*, 47(1-2):139–181, June 2006.
8. F. J. Pelletier and R. Elio. *Scope of Logic, Methodology and Philosophy of Science*, volume 1 of *Synthese Library*, chapter Logic and Computation, pages 137–156. Dordrecht: Kluwer Academic Press, 2002.
9. P. Tolchinsky, U. Cortés, S. Modgil, F. Caballero, and A. López-Navidad. Increasing Human-Organ Transplant Availability: Argumentation-Based Agent Deliberation. *IEEE Intelligent Systems: Special Issue on Intelligent Agents in Healthcare*, 21(5):30–37, November/December 2006.
10. P. Tolchinsky, U. Cortés, J. C. Nieves, A. López-Navidad, and F. Caballero. Using arguing agents to increase the human organ pool for transplantation. In *Proc. of the Third Workshop on Agents Applied in Health Care (IJCAI 2005)*, 2005.
11. S. E. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.
12. A. Tversky and D. Kahneman. *Judgment under uncertainty:Heuristics and biases*, chapter Judgment under uncertainty:Heuristics and biases, pages 3–20. Cambridge Univertisy Press, 1982.