A possibilistic approach to restore consistency in Answer Set Programming

Pascal Nicolas and **Laurent Garcia** and **Igor Stéphan**
LERIA, University of Angers, France
email: {pascal.nicolas, laurent.garcia, igor.stephan}@univ-angers.fr

Abstract

In Answer Set Programming it is not possible to deduce any conclusion from an inconsistent program (ie: a program that has no model). The same issue occurs in classical logic where there exist some techniques to handle this inconsistency. In this work, we propose to manage inconsistent logic programs in a similar way as possibilistic logic does for classical logic. We compute a consistent subprogram keeping the most important rules of the original program. This importance is described by a necessity degree assigned to each rule.

Introduction

Answer set programming (ASP) is an appropriate formalism to represent various problems issued from Artificial Intelligence and arising when available information is incomplete (non monotonic reasoning, planning, diagnosis...). It is also a very convenient framework to encode and solve $NP{-}complete$ combinatorial problems. From a global view, ASP is a general paradigm covering different declarative semantics for different kinds of logic programs. One is the *stable model* semantics (Gelfond & Lifschitz 1988) for definite logic programs augmented with default negation. If strong negations are allowed in a program, then this one is called an *extended logic program* and the *answer set* semantics (Gelfond & Lifschitz 1991) can be used. Another extension is possible by introducing disjunctions in rule heads to write a *disjunctive logic program* for which answer set semantics is again appropriate (see (Baral & Gelfond 1994; Brewka & Dix 1998) for additional material about these topics). Whatever the used framework, a "model" of a program P is characterized as a literal set S that is called an *answer set* of P. But, it may happen that P has no answer set and in this case we say that P is *inconsistent*. This is done by analogy with classical logic in which we say that a formula set Σ is inconsistent when it has no (classical) model.

Possibilistic logic is issued from Zadeh's possibility theory (Zadeh 1978), which offers a framework for the representation of states of partial ignorance owing to the use of a dual pair of possibility and necessity measures. Possibility theory may be quantitative or qualitative (Dubois & Prade 1988; Dubois & Prade 1998) according to the range of these measures which may be the real interval $[0, 1]$, or a finite linearly ordered scale as well. Possibilistic logic (Dubois, Lang, & Prade 1995) has been developed for more than ten years. It provides a sound and complete machinery for handling qualitative uncertainty with respect to a semantics expressed by means of possibility distributions which rank-order the possible interpretations. Let us mention that in possibilistic logic we deal with uncertainty by means of classical 2-valued (true or false) interpretations that can be more or less certain, more or less possible. We are not concerned by vagueness representation in a multi-valued framework.

One feature of possibilistic logic is its ability to manage inconsistency of a formula set. It proposes a way to restore the consistency of a formula set by deleting some less preferred formulas. In this work, we propose to import in ASP the fundamental principles of possibilistic logic about inconsistency management. Handling inconsistency in ASP using possibilistic logic takes place in a framework called *Possibilistic Answer Set Programming* (PASP) that we have defined and investigated more deeply in another very recent work (Nicolas, Garcia, & Stéphan 2004a). PASP is briefly presented in Appendix at the end of this work.

In the following section, we recall the useful notions to understand our work. Next, we explain how to iteratively restore the consistency of a logic program by a possibilistic approach. We characterize a particular class of programs for which only one iteration of the process is always sufficient. After a conclusion, we show in an appendix how our contribution takes place in a new paradigm of *Possibilistic Answer Set Programming* (Nicolas, Garcia, & Stéphan 2004a; Nicolas, Garcia, & Stéphan 2004b).

Theoretical backgrounds

Answer Set Programming

Since an extended logic program (without head disjunction) under answer set semantics is reducible to a program (without strong negation) under stable model semantics (see (Gelfond & Lifschitz 1991)), we present our work in the stable model paradigm that deals with *general logic programs* (Baral & Gelfond 1994) (also called *normal logic programs* or simply *programs* in the sequel). But all results remain valid if we deal with consistent answer sets of programs with strong negations.

A program is a set of rules of the form:

$$c \leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m.\quad n \geq 0,\ m \geq 0$$

where $a_1, \ldots, a_n, b_1, \ldots, b_m$ and c are atoms. A term like $not\ b$ is called a *default negation*. The intuitive meaning of such a rule is: "if you have all the a_i's and no b_j's then you can conclude c". For such a rule r we use the following notations[1]:

- the positive prerequisites of r :
$$body^+(r) = \{a_1, \ldots, a_n\}$$
- the negative prerequisites of r :
$$body^-(r) = \{b_1, \ldots, b_m\}$$
- the conclusion of r :
$$head(r) = c$$
- the positive projection of r :
$$r^+ = head(r) \leftarrow body^+(r).$$

If a program P does not contain any default negation (ie: $body^-(P) = \emptyset$), then P is a *definite program*. In this case, it has one minimal Herbrand model equal to its *deductive closure* $Cn(P)$ that is the smallest atom set *closed* under P. Recall that an atom set X is closed under a definite program P if:

$$\forall r \in P, body^+(r) \subseteq X \Rightarrow head(r) \in X$$

The *reduct* P^X of a program P wrt. an atom set X is the definite program defined by:

$$P^X = \{r^+ \mid r \in P,\ body^-(r) \cap X = \emptyset\}$$

Now, we are able to recall the formal definition of a *stable model*.

Definition 1 *(Gelfond & Lifschitz 1988) Let P be a normal logic program and S an atom set. S is a stable model of P if $S = Cn(P^S)$.*

Note that a program may have one or many stable models and also not at all. In this last case we say that the program is *inconsistent*, otherwise it is *consistent*.

A rule set R is *grounded* if there exists an enumeration $\langle r_i \rangle_{i \in I}$ of R such that

$$\forall i \in I,\ body^+(r_i) \subseteq head(\{r_1, \ldots, r_{i-1}\})$$

The set of *generating rules* of a program P wrt. an atom set A is:

$$GR(P, A) = \left\{ r \in P \;\middle|\; \begin{array}{l} body^+(r) \subseteq A \\ \text{and} \\ body^-(r) \cap A = \emptyset \end{array} \right\}$$

These two last points are central in the alternative characterization of a stable model given by the next result.

Proposition 1 *(Linke 2001) Let P be a normal logic program and S an atom set. S is a stable model of P if and only if $GR(P, S)$ is grounded and $S = Cn(GR(P, S)^+)$*

Example 1 *Let P be the following program*

$$P = \left\{ \begin{array}{lll} a \leftarrow not\ a, not\ b., & d \leftarrow c, not\ d., & \\ e \leftarrow not\ b., & b \leftarrow c., & c. \end{array} \right\}$$

P has no answer set because of the rules $c.$ and $d \leftarrow c, not\ d.$ It is the same for $P \setminus \{c.\}$ because of $a \leftarrow not\ a, not\ b.$ and $P \setminus \{a \leftarrow not\ a, not\ b.\}$ because of the rules $c.$ and $d \leftarrow c, not\ d.$ But $P \setminus \{d \leftarrow c, not\ d.\}$ has one answer set $\{b, c\}$.

[1]These functions are extended to a rule set as usual.

Obviously, when a program is inconsistent it is very easy to extract a consistent subprogram by discarding all rules with a non empty negative prerequisite. Formally $\forall P, \{r \in P \mid body^-(r) = \emptyset\}$ is a definite program, so it is consistent. But it is a drastic simplification of the given program. It is natural to try to restore the consistency of the program by a way allowing us to keep as many rules as possible. Several solutions can be considered but the one we are interested in is to be able to compare the rules for choosing the ones that will be discarded. So, it seems interesting to provide the expert with a framework in which he will be able to indicate which rules he considers the most important to keep in the subprogram. In fact, this is the approach applied in possibilistic logic for finding a consistent subbase of an inconsistent classical formula set as it is described in the next subsection.

Possibilistic logic

At the syntactic level, possibilistic logic handles pairs of the form (p, α) where p is a classical logic formula and α is an element of a totally ordered set. The pair (p, α) expresses that the formula p is certain at least to the level α, or more formally $N(p) \geq \alpha$, where N is the necessity measure associated to the possibility distribution expressing the underlying semantics.

Given a formula set Σ, the basic element of possibility theory is the possibility distribution π_Σ which is a mapping from Ω, the interpretation set, to the interval $[0, 1]$. $\pi_\Sigma(\omega)$ represents the degree of compatibility of the interpretation ω with Σ that represents the available information (or beliefs) about the real world. By convention, $\pi_\Sigma(\omega) = 0$ means that ω is impossible, and $\pi_\Sigma(\omega) = 1$ means that nothing prevents ω from being the real world (ω is a model of Σ). When $\pi_\Sigma(\omega) > \pi_\Sigma(\omega')$, ω is a preferred candidate to ω' for being the real state of the world.

Formally, possibilistic knowledge base is a finite set of weighted formulas

$$\Sigma = \{(p_i, \alpha_i), i = 1, \ldots, n\}$$

where α_i is understood as a lower bound on the degree of necessity $N(p_i)$. Formulas with degree zero are not explicitly represented in the knowledge base (only beliefs which are somewhat accepted are explicitly represented). The higher is the weight, the more certain is the formula. We call the α-cut (resp. strict α-cut) of Σ, denoted by $\Sigma_{\geq\alpha}$ (resp. by $\Sigma_{>\alpha}$), the set of classical formulas in Σ having a certainty degree at least equal to (resp. strictly greater than) α.

Σ is said to be *consistent* if its classical support, obtained by forgetting the weights, is classically consistent. We denote the inconsistency degree of Σ by: $Inc(\Sigma) = max\{\alpha_i : \Sigma_{\geq\alpha_i}$ is inconsistent$\}$. $Inc(\Sigma) = 0$ means that $\Sigma_{\geq\alpha_i}$ is consistent for all α_i. Furthermore, the inconsistency degree can be computed by :

$Inc(\Sigma) = 1 - max\{\pi_\Sigma(\omega) \mid \omega \in \Omega\}$ where
if ω is a model of Σ
then $\pi_\Sigma(\omega) = 1$
else $\pi_\Sigma(\omega) = min\{1 - \alpha_i \mid \omega \not\models p_i, (p_i, \alpha_i) \in \Sigma\}$

Such a π_Σ is the least specific possibility distribution induced by the necessity values attributed to the formulas in Σ. If Σ has no model, then, by discarding formulas which

necessity degree is lower than the inconsistency degree, it defines an α-cut $\Sigma_{>Inc(\Sigma)}$ that is consistent. It is clear that this cut may eliminate some formulas that are not involved in the inconsistency. Nevertheless, $Inc(\Sigma)$ defines a plausibility level under which information is no more pertinent. So, it is justified to eliminate all the formulas representing this piece of knowledge.

Furthermore, a syntactic possibilistic entailment has been defined as follows. Let Σ be a possibilistic knowledge base and (p, α) a possibilistic formula, p is entailed from Σ to degree α denoted by $\Sigma \vdash (p, \alpha)$ if and only if $\Sigma_{>\alpha}$ entails p. In other words, (p, α) is a possibilistic consequence of Σ if and only if $Inc(\Sigma \cup \{(\neg p, 1)\}) > Inc(\Sigma)$ and $Inc(\Sigma \cup \{(\neg p, 1)\}) \geq \alpha$.

Example 2 *Let Σ be the possibilistic base*

$$\Sigma = \left\{ \begin{array}{l} (a, 0.5), (\neg d, 0.6), (\neg e, 0.9), (\neg a \vee b, 0.7), \\ (b \vee c, 0.8), (\neg b \vee d, 0.3), (\neg b \vee e, 0.7) \end{array} \right\}$$

By applying the resolution principle

$$\frac{(a \vee B, \alpha) \ \ (\neg a \vee C, \beta)}{(B \vee C, min\{\alpha, \beta\})}$$

that is still valid in possibilistic logic, we obtain

$$\Sigma \vdash (\bot, 0.3) \text{ and } \Sigma \vdash (\bot, 0.5)$$

but $\neg\exists\alpha, \alpha > 0.5, \Sigma \vdash (\bot, \alpha)$. Thus $Inc(\Sigma) = 0.5$ and then

$$\Sigma_{>0.5} = \left\{ \begin{array}{l} (\neg d, 0.6), (\neg e, 0.9), (\neg a \vee b, 0.7), \\ (b \vee c, 0.8), (\neg b \vee e, 0.7)\} \end{array} \right\}$$

is a consistent subbase of Σ. $\Sigma_{>0.5}$ has a unique model $\{\neg a, \neg b, c, \neg d, \neg e\}$ which is the preferred interpretation of Σ since $\pi_\Sigma(\{\neg a, \neg b, c, \neg d, \neg e\}) = min\{1 - 0.5\} = 0.5$ that is the greatest value of π_Σ over Ω.

Inconsistency in ASP

Inconsistency degree of a program

First of all we have to assign a certainty degree to every rule of a program. For this, we consider given a finite set of atoms $\mathcal{X}$ and a finite set of decimal necessity values $\mathcal{N} \subset]0, 1]$ and we define a *possibilistic normal logic program* (p.n.l.p.) as a set of *possibilistic rules* of the form:

$(c \leftarrow a_1, \ \ldots, \ a_n, \ not\, b_1, \ \ldots, \ not\, b_m., \ \alpha)\ n \geq 0, m \geq 0$

where $\{c, a_1, \ \ldots, \ a_n, \ b_1, \ \ldots, \ b_m\} \subseteq \mathcal{X}$ and $\alpha \in \mathcal{N}$. By this way the expert is able to state that certain rules are more necessary (or sure, suitable, plausible, ...) than some others. If r is such a possibilistic rule, we denote

- the classical projection of r:
$$r^* = c \leftarrow a_1, \ \ldots, \ a_n, \ not\, b_1, \ \ldots, \ not\, b_m.$$
- the necessity degree of r:
$$n(r) = \alpha$$

For a p.n.l.p. P we note $P^* = \{r^* \mid r \in P\}$ the set of possibilistic rules without their necessity degree.

As in possibilistic logic, the necessity values affected to the rules of a p.n.l.p. P induce a possibility distribution π_P on the set $2^{\mathcal{X}}$ that evaluates at which level an atom set $X \in 2^{\mathcal{X}}$ can be a stable model of P. This point is formalized in the next two definitions.

Definition 2 *Let $X \in 2^{\mathcal{X}}$ be an atom set and r a rule (not a possibilistic one). X satisfies r, denoted $X \models r$, if $body^+(r) \subseteq X \wedge body^-(r) \cap X = \emptyset \Rightarrow head(r) \in X$.*

Definition 3 *Let P be a p.n.l.p. π_P is a possibility distribution if $\pi_P : 2^{\mathcal{X}} \longmapsto [0, 1]$ such that*

if $GR(P^*, X)$ is not grounded, then $\pi_P(X) = 0$, i)
else if $X \not\subseteq head(GR(P^*, X))$, then $\pi_P(X) = 0$, ii)
else if $\forall r \in P, X \models r^*$, then $\pi_P(X) = 1$, iii)
else $0 \leq \pi_P(X) < 1$ and
$\pi_P(X) = \min_{r \in P}\{1 - n(r) \mid X \not\models r^*\}$ iv)

This definition relies on the characterization of a stable model given by proposition 1 and captures the notion of minimum specificity. In cases i) and ii), the set X is not possible at all since it is impossible to find a correct set of generating rules for it. In case iii), the set X is a stable model so its possibility is total. In case iv), the possibility of X is determined by the degree of the most certain rule that is not satisfied by X. The more X does not satisfy sure rules, the less X is possible. Furthermore, in this last case $\pi_P(X)$ can not be equal to 1 since P is not supposed to contain a rule with a necessity equal to 0.

Proposition 2 *Let P be a p.n.l.p. and $X \in 2^{\mathcal{X}}$, then*
X is a stable model of $P^* \iff \pi_P(X) = 1$

Proof 2 *1) If X is a stable model of P^*, then by proposition 1 $GR(P^*, X)$ is grounded and $X = Cn(GR(P^*, X)^+)$. It gives $X \subseteq head(GR(P^*, X))$ and $\forall r \in P, X \models r^*$, thus $\pi_P(X) = 1$.*
2) If $\pi_P(X) = 1$ then $GR(P^*, X)$ is grounded and $X \subseteq head(GR(P^*, X))$. Furthermore, $\forall r \in P, X \models r^*$, that is $\forall r \in P^*, body^+(r) \subseteq X \wedge body^-(r) \cap X = \emptyset \Rightarrow head(r) \in X$. So, by definition of GR, we have $head(GR(P^*, X)) \subseteq X$ and then $X = head(GR(P^*, X))$.
By groundedness of $GR(P^*, X)$, $X = GR(P^*, X) = Cn(GR(P^*, X))$ and then by proposition 1 X is a stable model of P^*. □

This possibility distribution allows us to define the *inconsistency degree* of a p.n.l.p.

Definition 4 *Let P be a p.n.l.p., its inconsistency degree is*
$$InconsDeg(P) = 1 - \max_{X \in 2^{\mathcal{X}}}\{\pi_P(X)\}$$

Proposition 3 *Let P be a p.n.l.p., then*
P^* is consistent $\iff InconsDeg(P) = 0$

Proof 3 *1) P^* is consistent $\Longrightarrow \exists X \in 2^{\mathcal{X}}$ such that X is a stable model of $P^* \Longrightarrow \exists X \in 2^{\mathcal{X}}, \pi_P(X) = 1$ by proposition 2, and then $InconsDeg(P) = 0$.*
2) $InconsDeg(P) = 0 \Longrightarrow \exists X \in 2^{\mathcal{X}}$ such that $\pi_P(X) = 1 \Longrightarrow \exists X \in 2^{\mathcal{X}}$ such that X is a stable model of P^* by proposition 2, and then P^* is consistent. □

Given a p.n.l.p. P and a value $\alpha \in \mathcal{N}$, the strict α-cut of P is the subprogram $P_{>\alpha} = \{r \in P \mid n(r) > \alpha\}$. The α-cut is used to determine a subprogram that allows to reason from the initial information. This subprogram may be a consistent subprogram as illustrated by the program P_1 in example 3. Unfortunately, unlike possibilistic logic, if $InconsDeg(P) > 0$, then $P_{>InconsDeg(P)}$ is not necessary consistent as it is illustrated by program P_2 in example 3.

Example 3 *Let P_1 and P_2 be two possibilistic versions of the program in example 1.*

$$P_1 = \left\{ \begin{array}{ll} (a \leftarrow not\ a, not\ b., 0.5), & (e \leftarrow not\ b., 0.8), \\ (d \leftarrow c, not\ d., 0.6), & (b \leftarrow c., 0.5), \\ (c., 1) & \end{array} \right\}$$

Its inconsistency degree is $InconsDeg(P_1) = 0.6$ and

$$P_{1_{>0.6}} = \{\ (e \leftarrow not\ b., 0.8),\quad (c., 1)\ \}$$

Then,$P^*_{1_{>0.6}}$ is consistent, it has one stable model $\{c, e\}$. We can observe that the rule $(b \leftarrow c., 0.5)$ has been discarded even if this was not necessary to restore the consistency. But the necessity degree of this rule was too low to be pertinent with respect to the program and should not be kept in the program.

Let us change the certainty degree of the first rule, we have.

$$P_2 = \left\{ \begin{array}{ll} (a \leftarrow not\ a, not\ b., 0.7), & (e \leftarrow not\ b., 0.8), \\ (d \leftarrow c, not\ d., 0.6), & (b \leftarrow c., 0.5), \\ (c., 1) & \end{array} \right\}$$

Its inconsistency degree is $InconsDeg(P_2) = 0.6$ and

$$P_{2_{>0.6}} = \left\{ \begin{array}{ll} (a \leftarrow not\ a, not\ b., 0.7), & (c., 1), \\ (e \leftarrow not\ b., 0.8) & \end{array} \right\}$$

Then $InconsDeg(P_{2_{>0.6}}) = 0.7$ and $P^*_{2_{>0.6}}$ is still inconsistent.

The result observed for P_2 in example 3 is not surprising because it is inherent to the non monotonic nature of ASP. In fact, the non existence of a stable model is a consequence of a circularity between some rules (as it is established for default logic (Linke & Schaub 2000)). Then, in whole generality, only a global study taking into account the whole program, can solve the problem. But, our inconsistency degree computation and our program reduction are only local processes, since they take rules one by one. So, $P^*_{>InconsDeg(P)}$ is not necessary a consistent program. Nevertheless the following iterative process is always possible to restore the consistency of a program.

Proposition 4 *Let P be a p.n.l.p., and cut the function defined by*

$$\begin{array}{l} \mathbf{cut}(P)\{ \\ \textit{if } InconsDeg(P) = 0 \textit{ return } P \\ \textit{else return } cut(P_{>InconsDeg(P)})\ \} \end{array}$$

Then, $cut(P)^*$ is a consistent subprogram of P^*.

Proof 4 *The correction of the function cut is ensured by proposition 3. Furthermore, it is obvious that cut always ends after a finite number of calls since*

- *we consider only finite programs,*
- *$cut(P_{>InconsDeg(P)}) \subset P$ because of the strict α-cut*
- *the empty program is consistent because $\emptyset$ is a stable model for it.*

□

For instance $cut(P_2)^* = \{e \leftarrow not\ b.,\ c.\}$ is the consistent subprogram of program P_2 given in example 3.

Characterization for possibilistic bases

In this section, we focus our attention to programs that encode classical possibilistic bases.

Let $\mathcal{A}$ be an atom set from which we build a classical propositional base. Recall that every propositional base Σ can be encoded in a clause set. So, without loss of generality, we consider here only clause sets. On its turn, such a clause set Σ can be translated in a normal logic program $P(\Sigma)$ as following (a similar process is exposed in (Simons 2000)).

First, we give a translation of a clause in a rule:

$$\begin{array}{c} \forall cl = (\neg a_1 \vee \cdots \vee \neg a_n \vee b_1 \vee \cdots \vee b_m), \\ P(cl) = f \leftarrow a_1, \ldots, a_n, b'_1, \ldots, b'_m. \end{array}$$

and $$\begin{array}{ll} P(\Sigma) = & \{P(cl) \mid cl \in \Sigma\} \\ & \cup\{x \leftarrow not\ x'., x' \leftarrow not\ x. \mid x \in \mathcal{A}\} \\ & \cup\ \{bug \leftarrow f, not\ bug.\} \end{array}$$

The intuition behind this translation stands on the following remarks.

- x' is a new atom encoding the negative literal $\neg x$
- Rules $x \leftarrow not\ x'.$ and $x' \leftarrow not\ x.$ allow to generate all possible classical propositional interpretations by doing an exclusive choice between x and $\neg x$ for each atom x in $\mathcal{A}$.
- The goal of each rule $P(cl)$ is to conclude f (a new symbol for *false*) if the choice of atoms (x and $\neg x$) corresponds to an interpretation that does not satisfy the clause cl. By this way, if there exists a stable model not containing f, then it corresponds to an interpretation of Σ (since every clause is satisfied).
- The goal of special rule $bug \leftarrow f, not\ bug.$, where bug is a new symbol, is to discard every stable model containing f. Since bug appears in the head and in the negative body of this rule and nowhere else, if a stable model exists then it may not contain f.

By this way there is a one to one correspondence between the propositional models of Σ and the stable models of $P(\Sigma)$. But, as stated in (Niemelä 1999) there is no modular mapping from program to set of clauses, only a modular transformation from set of clauses to program exists. So, in a way, ASP has better knowledge representation capabilities than propositional logic and it is interesting to study how it can be extended to the possibilistic case in particular when there is an inconsistency.

The interesting point is that in this case we are able to restore the consistency of a program in only one step as it can be summarized in the figure 1.

To reach our goal, we first extend the transformation of a clause set to the possibilistic case in a natural way. If $(p, \alpha) \in \Sigma$, then all clauses encoding p keep the same necessity degree α and it is the same for each corresponding rule in $P(\Sigma)$. A necessity value equal to 1 is assigned to all the other rules (the "technical" ones).

Definition 5 *Let $\Sigma = \{(cl_i, \alpha_i), i = 1, \ldots, n\}$ be a possibilistic base (in CNF), its encoding in a p.n.l.p. is given by*

$$\begin{array}{ll} P(\Sigma) = & \{(P(cl_i), \alpha_i) \mid (cl_i, \alpha_i) \in \Sigma\} \\ & \cup\{(x \leftarrow not\ x'., 1), (x' \leftarrow not\ x., 1) \mid x \in \mathcal{A}\} \\ & \cup\ \{(bug \leftarrow f, not\ bug., 1)\} \end{array}$$

possibilistic logic base		possibilistic normal logic program
inconsistent base	$\Longrightarrow$	inconsistent program
Σ		$P(\Sigma)$
$\Downarrow$		$\Downarrow$
consistent subbase	$\Longleftrightarrow$	consistent subprogram
$\Sigma_{>\alpha}$		$P(\Sigma)_{>\alpha}$
$\Downarrow$		$\Downarrow$
propositional model	$\Longleftrightarrow$	stable model
α is the inconsistency degree of Σ and $P(\Sigma)$		

Figure 1: Inconsistency handling

In the sequel we use $\mathcal{X} = \cup_{a\in\mathcal{A}}\{a, a'\} \cup \{f, bug\}$ in order to make the correspondence between the language of the propositional base and the language of its translation.

Definition 6 *$X \subseteq \mathcal{X}$ is a* pseudo interpretation *if*
$$\forall a \in \mathcal{A}, (a \in X \vee a' \in X) \wedge (a \notin X \vee a' \notin X) \wedge bug \notin X \wedge f \notin X$$

In the following, we will say that a pseudo interpretation X *corresponds* to a classical interpretation ω if by translating each atom $a' \in X$ in literal $\neg a$, we obtain the interpretation[2] ω. By this way, every stable model of $P(\Sigma)$ is a pseudo interpretation that corresponds to a classical model for Σ (and conversely).

Proposition 5 *Let Σ be a possibilistic base and $P = P(\Sigma)$ its encoding in a p.n.l.p. $\forall X \subseteq \mathcal{X}$ we have*
X is not a pseudo interpretation and $\pi_P(X) = 0$
or
X is a pseudo interpretation and $\pi_P(X) = \pi_\Sigma(\omega)$
where ω is the interpretation that corresponds to X

Proof 5 *We note:*
$$r_{bug} = bug \leftarrow f, not\ bug.$$

- *if X is not a pseudo interpretation, we have the following cases*
  - $bug \in X \Longrightarrow r_{bug} \notin GR(P^*, X) \Longrightarrow X \not\subseteq head(GR(P^*, X))$*, and then* $\pi_P(X) = 0$.
  - $bug \notin X \wedge f \in X \Longrightarrow X \not\models r_{bug}$ *and since* $n(r_{bug}) = 1$ *then* $\pi_P(X) = 0$.
  - $bug \notin X \wedge f \notin X \wedge \exists x \in \mathcal{A}, x \in X \wedge x' \in X \Longrightarrow x \leftarrow not\ x' \notin GR(P^*, X) \wedge x' \leftarrow not\ x \notin GR(P^*, X) \Longrightarrow X \not\subseteq head(GR(P^*, X))$ *and then* $\pi_P(X) = 0$.
  - $bug \notin X \wedge f \notin X \wedge \exists x \in \mathcal{A}, x \notin X \wedge x' \notin X \Longrightarrow X \not\models x \leftarrow not\ x'$*and with* $n(x \leftarrow not\ x') = 1$ *then* $\pi_P(X) = 0$.

  thus, in every case where x is not a pseudo-interpretation, $\pi_P(X) = 0$.
- *if X is a pseudo interpretation, then $\forall a \in \mathcal{A}$, one and only one of the 2 rules $a \leftarrow not\ a'$ and $a' \leftarrow not\ a$ is in $GR(P^*, X)$ and then $GR(P^*, X)$ is grounded and $X \subseteq head(GR(P^*, X))$.*
  Let ω be the interpretation for Σ that corresponds to X.
  $\forall cl = (\neg a_1 \vee \cdots \vee \neg a_n \vee b_1 \vee \cdots \vee b_m), \in \Sigma$,
  - $\omega \not\models cl \Longrightarrow \omega \models a_1 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \cdots \wedge \neg b_m \Longrightarrow \{a_1, \ldots, a_n, b'_1, \ldots, b'_m\} \subseteq X$*. But,* $f \notin X$*, so* $X \not\models P(cl)$.
  - $X \not\models P(cl) \Longrightarrow \{a_1, \ldots, a_n, b'_1, \ldots, b'_m\} \subseteq X \wedge f \notin X \Longrightarrow \omega \models a_1 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \cdots \wedge \neg b_m \Longrightarrow \omega \not\models cl$

  Thus, we have $\omega \not\models cl \iff X \not\models P(cl)$ *and since* $\forall (cl, \alpha) \in \Sigma, n(P(cl)) = \alpha$*, we have* $\pi_\Sigma(\omega) = \pi_P(X), \forall X \in \mathcal{X}$ *and its corresponding interpretation* ω. □

Proposition 6 *Let Σ be a possibilistic base, then*

- $Inc(\Sigma) = InconsDeg(P(\Sigma))$.
- *if* $Inc(\Sigma) = \alpha$, $P(\Sigma_{>\alpha}) = (P(\Sigma))_{>\alpha}$
- $InconsDeg(P(\Sigma)) = 0 \Rightarrow (P(\Sigma))^*$ *has at least one stable model S that corresponds to a propositional model of Σ*
- $InconsDeg(P(\Sigma)) = \alpha > 0 \Rightarrow (P(\Sigma)_{>\alpha})^*$ *has at least one stable model S that corresponds to a propositional model of $\Sigma_{>\alpha}$.*

Proof 6

- $InconsDeg(P(\Sigma)) = 1 - \max_{X\in 2^{\mathcal{X}}}\{\pi_{P(\Sigma)}(X)\}$. *By proposition 5, we have* $\pi_{P(\Sigma)}(X) = 0$ *if X is not a pseudo-interpretation, so we can restrict the application of* max *to the set of pseudo-interpretations, and then (again by proposition 5) we have:*
  $$InconsDeg(P(\Sigma)) = 1 - \max_{\omega\in\Omega}\{\pi_\Sigma(\omega)\} = Inc(\Sigma)$$
- *Since the translation is based on a given atom set $\mathcal{A}$,* $P(\Sigma_{>\alpha}) = (P(\Sigma))_{>\alpha}$ *is obvious.*
- *If* $InconsDeg(P(\Sigma)) = 0$*, then the proposition 3 ensures that $(P(\Sigma))^*$ has a stable model. This stable model corresponds to a model of Σ because the translation exposed in the beginning of this subsection establishes a one to one correspondence between stable models of $P(\Sigma)$ and propositional models of Σ.*
- $InconsDeg(P(\Sigma)) = \alpha \Longrightarrow Inc(\Sigma) = \alpha$ *by the first item of this proposition. So, $\Sigma_{>\alpha}$ is consistent by a possibilistic logic result so* $Inc(\Sigma_{>\alpha}) = 0$ *and then* $InconsDeg(P(\Sigma_{>\alpha})) = 0$ *by the first item of this proposition. Thus, $(P(\Sigma)_{>\alpha})^*$ has a stable model that corresponds to a propositional model of $\Sigma_{>\alpha}$ by the previous item.*

□

[2] A pseudo interpretation leads necessary to an interpretation since it contains one occurrence of each atom (ie a or its negation) and no occurrence of f nor bug.

These results establish that our methodology illustrated in the next example and exposed in figure 1 is valid.

Example 4 *Let Σ be a possibilistic base*

$$\Sigma = \left\{ \begin{array}{l} (a, 0.5), (\neg d, 0.6), (\neg e, 0.9), \\ (\neg a \vee b, 0.7), (b \vee c, 0.8), \\ (\neg b \vee d, 0.3), (\neg b \vee e, 0.7) \end{array} \right\}$$

its encoding as a p.n.l.p. is

$$P(\Sigma) = \left\{ \begin{array}{l} (f \leftarrow a'., 0.5), (f \leftarrow d., 0.6), \\ (f \leftarrow e., 0.9), \\ (f \leftarrow a, b'., 0.7), (f \leftarrow b', c'., 0.8), \\ (f \leftarrow b, d'., 0.3), (f \leftarrow b, e'., 0.7) \end{array} \right\} \cup \left\{ \begin{array}{l} (x \leftarrow not\ x'., 1), (x' \leftarrow not\ x., 1) \\ \mid x \in \{a, b, c, d, e\} \end{array} \right\} \cup \{(bug \leftarrow f, not\ bug., 1)\}$$

Then, we have $InconsDeg(P(\Sigma)) = 0.5$ that corresponds to $Inc(\Sigma) = 0.5$ and the preferred consistent subprogram of $P(\Sigma)$ is

$$P(\Sigma)_{>0.5} = \left\{ \begin{array}{l} (f \leftarrow d., 0.6), (f \leftarrow e., 0.9), \\ (f \leftarrow a, b'., 0.7), (f \leftarrow b', c'., 0.8), \\ (f \leftarrow b, e'., 0.7), \end{array} \right\} \cup \left\{ \begin{array}{l} (x \leftarrow not\ x'., 1), (x' \leftarrow not\ x., 1) \\ \mid x \in \{a, b, c, d, e\} \end{array} \right\} \cup \{(bug \leftarrow f, not\ bug., 1)\}$$

So, we obtain $P(\Sigma)_{>0.5} = P(\Sigma_{>0.5})$ and $P(\Sigma)_{>0.5}$ has one stable model $S = \{(a', 1), (b', 1), (c, 1), (d', 1), (e', 1)\}$ such that $\{\neg a, \neg b, c, \neg d, \neg e\}$ is a propositional model of $\Sigma_{>0.5}$ the consistent subbase obtained in possibilistic logic.

Conclusion

In this work, we have described a methodology to restore the consistency of a logic program by discarding the less preferred rules. For instance, this can be useful when a program P encodes a combinatorial problem. In this case, there are two kinds of rules in P, the "rules of guess" that describe the search space of the problem and the "rules of check" that represent its constraints. By setting a necessity value of 1 to the first ones and a gradual necessity (between 0 and 1) to the second ones we are able to specify that some constraints are more important than others. By this way, if P has no model we can compute $P_{InconsDeg(P)}$ that corresponds to a relaxation of the original problem which is not containing the less important constraints. If this new program is consistent, we obtain an approximate solution of the original problem.

Our methodology is based on the computation of an inconsistency degree of the given program. For this, we have developed a first implementation in Prolog. But, it is clear that this calculus has an exponential time complexity, since it has to compute the possibility π of every subset of the atom set $\mathcal{X}$. So a future work could be to study which heuristics can be used to improve the computation of this inconsistency degree. Last, as we have mentioned in example 3, some rules are discarded even if they have no influence on the inconsistency. Thus, it would be interesting to study how to keep all rules not directly involved in the consistency.

Appendix :Possibilistic Answer Set Programming

The aim of this present work is a part of a more general research about an extension of the ASP paradigm that we have called *Possibilistic Answer Set Programming* (PASP)(Nicolas, Garcia, & Stéphan 2004a; Nicolas, Garcia, & Stéphan 2004b). In this new framework, we propose to integrate the concept of necessity and possibility coming from possibilistic logic in order to represent the uncertainty that is inherent to the conclusions obtained by a formalism as ASP that deals with incomplete information.

To summarize this new concept, we describe now major points of PASP. Let us note that PASP deals with programs that are not limited to be a translation of possibilistic bases.

A *possibilistic atom* is a pair $(x,\ \alpha) \in \mathcal{X} \times \mathcal{N}$. A *possibilistic atom set* (p.a.s.) A is a set of possibilistic atoms where $\forall x \in \mathcal{X}, |\{(x,\ \alpha) \in A\}| \leq 1$ (ie: every x occurs at most one time in A).

Let A be a p.a.s., and r a possibilistic rule $r = (c \leftarrow a_1,\ \ldots,\ a_n., \alpha)$, we say

- r is 1-applicable in A if $body^+(r^*) = \emptyset$
- r is δ-applicable in A if $\{(a_1, \alpha_1), \ldots, (a_n, \alpha_n)\} \subseteq A$ $\delta = min\{\alpha_1, \ldots, \alpha_n\}$ and $0 < \delta \leq 1$.
- r is 0-applicable otherwise

For a given p.n.l.p. P and an atom x, we define $H(P, x) = \{r \in P \mid head(r^*) = x\}$ the set that collects all rules in P having the same head x.

For all atom x we note:
$App(P, A, x) = \{r \in H(P, x), r$ is δ-applicable in $A, \delta > 0\}$

Thus, the applicability degree of a rule captures the certainty of its body as in possibilistic logic the certainty of a conjunction is the minimal value of the necessity values of subformulae involved in it.

We say that a p.n.l.p. is a *possibilistic definite logic program* (p.d.l.p.) when the classical projection of each of its rule is definite.

Such a p.d.l.p. P is *closed* under a p.a.s. A if and only if

$$\begin{array}{l} \forall r \in P, r \text{ is } \delta\text{-applicable in } A, \delta > 0 \\ \Rightarrow (head(r^*), \beta) \in A\ with \\ \beta = \max_{r' \in App(P, A, head(r^*))} \left\{ \begin{array}{l} \min\{n(r'), \delta'\}, \\ r' \text{ is } \delta'\text{-applicable in } A \end{array} \right\} \end{array}$$

We note $PossCn(P)$ the *possibilistic deductive closure* of P defined as the smallest possibilistic atom set that is closed under P.

The *possibilistic reduct* of a p.n.l.p. P wrt. an atom set S is the p.d.l.p.

$$P^S = \{(r^{*+}, n(r)) \mid r \in P, body^-(r^*) \cap S = \emptyset\}$$

An atom set S is a *possibilistic stable model* (p.s.m.) of P if and only if $S = PossCn(P^{S^*})$.

As in the case where there is no certainty value attached to the rules, a p.n.l.p. may have no, one or many p.s.m.

To illustrate our new framework, let us take the well known Nixon's paradox[3] that is classically encoded by

$$Pnixon = \{p \leftarrow q, not\ \neg p.,\ \ \neg p \leftarrow r, not\ p.,\ \ q.,\ \ r.\}$$

[3] p:pacifist, q:quaker, r:republican

With our new proposal of PASP we are now able to represent it in a richer formalism with the following p.n.l.p.

$$PossPnixon = \left\{ \begin{array}{l} (p \leftarrow q, not\ p'., 0.6) \\ (p' \leftarrow r, not\ p., 0.9) \\ (q., 1) \\ (r., 1) \\ (f \leftarrow p, p', notf., 1) \\ (f \leftarrow q, q', notf., 1) \\ (f \leftarrow r, r', notf., 1) \end{array} \right\}$$

where atoms p', q' and r' stand for the strong negation of p, q and r respectively. In this program, we have given a greater necessity value (0.9) to the second rule in order to represent the greater confidence of the expert in this rule. Remember that the exact necessity value has no particular meaning but only the order between the values has an importance. All the other rules have a necessity value of 1, because we are sure that Nixon is a Republican and a Quaker, and because the other rules result from strong negation encoding. Then, $PossPnixon$ has two p.s.m.

$$A_1 = \{(q, 1), (r, 1), (p', 0.9)\}$$
$$A_2 = \{(q, 1), (r, 1), (p, 0.6)\}$$

So, we have now an additional information saying that "*it is certain at a level 0.9 that Nixon is not pacifist*" and *"it is certain at a level 0.6 that Nixon is pacifist*". The two possible, and incompatible, points of view on the real world incompletely described can now be compared. Thus, this can be used to infer a kind of meta conclusion by preferring the more certain conclusion. So we see here that our possibilistic framework for ASP has something to do with the preference handling in ASP for which many works have been done (see (Schaub & Wang 2003)). But, we have to mention that in all these works on preference handling, the goal is to eliminate some less preferred models, whereas in our framework we keep all the stable models but we have new complementary informations that can be used to rank between them the conclusions.

References

[Baral & Gelfond 1994] Baral, C., and Gelfond, M. 1994. Logic programming and knowledge representation. *Journal of Logic Programming* 19/20:73–148.

[Brewka & Dix 1998] Brewka, G., and Dix, J. 1998. Knowledge Representation with Logic Programs. In Dix, J.; Pereira, L.; and Przymusinski, T., eds., *Logic Programming and Knowledge Representation*, volume 1471 of *Lecture Notes in Artificial Intelligence*, 1–55. Springer Verlag.

[Dubois & Prade 1988] Dubois, D., and Prade, H. 1988. *Possibility Theory - An Approach to Computerized Processing of Uncertainty*. New-York: Plenum Press.

[Dubois & Prade 1998] Dubois, D., and Prade, H. 1998. Possibility theory: qualitative and quantitative aspects. In Smets, P., ed., *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 1. Kluwer Academic Press. 169–226.

[Dubois, Lang, & Prade 1995] Dubois, D.; Lang, J.; and Prade, H. 1995. Possibilistic logic. In Gabbay, D.; Hogger, C.; and Robinson, J., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Oxford University Press. 439–513.

[Gelfond & Lifschitz 1988] Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R. A., and Bowen, K., eds., *Proceedings of the International Conference on Logic Programming*, 1070–1080. The MIT Press.

[Gelfond & Lifschitz 1991] Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3-4):363–385.

[Linke & Schaub 2000] Linke, T., and Schaub, T. 2000. Alternative foundations for Reiter's default logic. *Artificial Intelligence* 124:31–86.

[Linke 2001] Linke, T. 2001. Graph theoretical characterization and computation of answer sets. In Nebel, B., ed., *Proceedings of the IJCAI*, 641–645. Morgan Kaufmann Publishers.

[Nicolas, Garcia, & Stéphan 2004a] Nicolas, P.; Garcia, L.; and Stéphan, I. 2004a. Programmation par ensembles réponses possibilistes. In *Actes des Journées Francophones de Programmation en Logique et Programmation par Contraintes*. Angers: Hermès.

[Nicolas, Garcia, & Stéphan 2004b] Nicolas, P.; Garcia, L.; and Stéphan, I. 2004b. Towards possibilistic answer set programming. submitted.

[Niemelä 1999] Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25((3-4)):241–273.

[Schaub & Wang 2003] Schaub, T., and Wang, K. 2003. A semantic framework for preference handling in answer set programming. *Theory and Practice of Logic Programming* 3(4-5):569–607.

[Simons 2000] Simons, P. 2000. Extending and implementing the stable model semantics. Research Report A58, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory for Theoretical Computer Science, Espoo, Finland. Doctoral dissertation.

[Zadeh 1978] Zadeh, L. 1978. Fuzzy sets as a basis for a theory of possibility. In *Fuzzy Sets and Systems*, volume 1. 3–28.