Traitement de l'incertitude pour la programmation par ensembles réponses

Uncertainty handling for answer set programming

Pascal NICOLAS Laurent GARCIA Igor STÉPHAN Claire LEFÈVRE

LERIA, Université d'Angers
2, bd Lavoisier, 49045 ANGERS CEDEX 05
{pascal.nicolas,laurent.garcia,igor.stephan,claire.lefevre}@univ-angers.fr

Résumé

Nous présentons un formalisme apte à gérer de manière unifiée un raisonnement à la fois non monotone et incertain. Nous utilisons la théorie des possibilités pour étendre la sémantique non monotone des modèles stables pour les programmes logiques avec négation par défaut de manière à tenir compte d'un degré de certitude affecté à chaque règle du programme. Dans ce cadre-là, nous définissons un traitement sémantique basé sur une distribution de possibilité et un traitement syntaxique utilisant un opérateur de point fixe. Nous montrons que les deux approches sont équivalentes. Ensuite, nous décrivons notre implantation d'un système capable d'effectuer les calculs requis par l'automatisation d'un tel raisonnement et nous montrons comment notre formalisme peut contribuer à la restauration de la consistance de programmes sans modèles stables.

Mots Clef

Raisonnement non monotone et incertain, programmation par ensembles réponses, théorie des possibilités.

Abstract

We present a framework able to deal in a unified way with a non monotonic and uncertain reasoning. We use possibility theory to extend the non monotonic stable model semantics for logic programs with default negation in such a way that we can take into account a certainty degree for each rule in the program. In this framework, we define a semantic handling by means of a possibility distribution and a syntactic one by means of a fix-point operator. We show that the two approaches are equivalent. Furthermore, we describe our implementation of a system able to do the computation necessary for the automatization of such a reasoning and we show how our framework can help to restore the consistency of programs without stable models.

Keywords

Non monotonic and uncertain reasoning, answer set programming, possibility theory.

1 Introduction

La programmation par ensembles réponses (ASP pour Answer Set Programming en anglais) est un formalisme approprié pour représenter différents problèmes issus de l'Intelligence Artificielle et se manifestant quand l'information disponible est incomplète comme dans le raisonnement non monotone, la planification, le diagnostic,... D'un point de vue général, l'ASP est un paradigme couvrant différentes sémantiques déclaratives pour différentes sortes de programmes logiques. Dans tous les cas, l'information est codée par des règles logiques et les solutions correspondent à des modèles. Chaque modèle est un ensemble minimal d'atomes représentant des informations sûres (des faits) et des déductions obtenues à partir de règles par défaut. Ainsi, les conclusions sont basées sur de l'information présente et absente, elles forment un ensemble cohérent d'hypothèses et représentent un point de vue rationnel décrit par les règles. C'est pourquoi, il n'y a généralement pas un ensemble unique de conclusions mais plutôt plusieurs, les conclusions ne sont alors pas absolument sûres mais seulement plausibles ou plus ou moins certaines. On retrouve ici les traits majeurs du raisonnement à partir d'informations incomplètes comme on peut le modéliser à l'aide de la logique des défauts [19] dont la sémantique des modèles stables pour les programmes logiques normaux, utilisée ici, constitue une réduction.

La logique possibiliste [9] est issue de la théorie des possibilités de Zadeh [22] qui offre un cadre de représentation des états d'ignorance partielle, basé sur l'utilisation d'une paire de mesures duales de possibilité et de nécessité. La théorie des possibilités peut être quantitative ou qualitative [10] selon que l'intervalle de ces mesures est l'intervalle réel $[0, 1]$ ou une échelle finie ordonnée. Elle fournit une machinerie correcte et complète pour traiter l'incertitude de manière qualitative basée sur une sémantique exprimée en terme de distributions de possibilité qui ordonnent les interprétations. Notons que, dans la logique possibiliste, on évalue le niveau de certitude d'interprétations qui sont bivaluées. On reste donc dans le cadre de la logique classique (et non multivaluée ou floue) à la-

quelle on adjoint un moyen de graduer la confiance que l'on a dans les différentes informations énoncées.

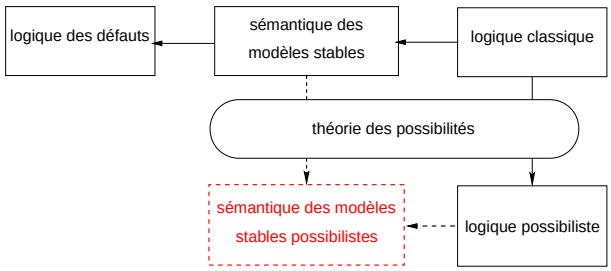

FIG. 1 – Raisonner avec des informations incomplètes et incertaines

Le but de notre travail est donc de proposer un moyen d'intégrer en un seul formalisme le traitement des informations incomplètes et incertaines. Pour cela nous proposons d'introduire les concepts de la théorie des possibilités au sein de la sémantique des modèles stables comme décrit dans la figure 1. Notre problématique peut être illustrée par l'exemple suivant d'un traitement médical dans lequel un patient souffre de deux maladies. Chaque maladie peut être soignée par un médicament mais les deux médicaments sont incompatibles. Le programme

$$P = \left\{ \begin{array}{l} med_1 \leftarrow mal_1, not\ med_2. \\ med_2 \leftarrow mal_2, not\ med_1. \\ g_1 \leftarrow med_1, mal_1. \\ g_2 \leftarrow med_2, mal_2. \\ mal_1 \leftarrow . \qquad mal_2 \leftarrow . \end{array} \right\}$$

signifie que le médicament med_1 (resp. med_2) est donné à un patient s'il est atteint de la maladie mal_1 (resp. mal_2) sauf s'il prend le médicament med_2 (resp. med_1) ; si un patient atteint de la maladie mal_1 (resp. mal_2) prend le médicament med_1 (resp. med_2) alors il est guéri g_1 (resp. g_2) de cette maladie ; le patient est atteint des maladies mal_1 et mal_2. De ce programme, on obtient deux modèles stables $\{mal_1, mal_2, med_1, g_1\}$ et $\{mal_1, mal_2, med_2, g_2\}$ indiquant que l'on peut guérir le patient de l'une de ses deux maladies mais pas des deux. Cependant, il semble intéressant pour un médecin de pouvoir évaluer quel choix faire parmi ces deux traitements incompatibles entre eux. L'un des critères de choix peut être l'efficacité de chaque traitement médical qui pourrait se représenter au moyen de degrés de certitude affectés à certaines règles. Ces degrés devraient être pris en compte lors du mécanisme inférentiel pour déterminer le niveau de certitude de chacune des conclusions et permettre ainsi au médecin de les comparer entre elles.

Cet article reprend l'essentiel des propositions théoriques formulées dans [18, 17] et décrit également le travail d'implémentation réalisé depuis. Pour cela, cette section se poursuit avec des rappels théoriques concernant la logique possibiliste et l'ASP. La section 2 décrit les programmes logiques définis possibilistes et leurs modèles possibilistes. La section 3 étend notre approche aux modèles stables possibilistes pour les programmes logiques contenant des négations par défaut. Précisons que la prise en compte de l'incertitude ne remet pas en cause le formalisme non monotone initial. Dans la section 4 nous décrivons brièvement l'implantation d'un système que nous avons développé pour calculer les modèles stables possibilistes. Nous nous en servons pour montrer une utilisation possible de notre formalisme dans le cadre de la résolution de problèmes combinatoires (domaine d'application possible de l'ASP) où les informations sont données avec des degrés de certitude. Puis nous montrons comment notre proposition peut servir à la restauration de la consistance de programmes inconsistants et nous terminons en plaçant notre travail en relation avec d'autres travaux concernant la même problématique.

Logique possibiliste. La logique possibiliste, évaluée par mesures de nécessité, traite de paires de la forme (p, α) où p est une formule de la logique classique et α est un élément d'un ensemble totalement ordonné. La paire (p, α) exprime le fait que la formule p est certaine au moins au degré α. Une base possibiliste est notée $\Sigma = \{(p_i, \alpha_i)\}_{i \in I}$. Les formules de degré 0 ne sont pas représentées explicitement dans la base, seules les informations un tant soit peu acceptées étant utiles et plus le degré α est élevé, plus la formule est certaine. Il est à noter que ce degré est évalué par une mesure de nécessité et n'est pas une probabilité. Ainsi, les valeurs numériques ne sont pas une évaluation absolue (comme dans la théorie des probabilités) mais définissent une échelle de certitude (ou de confiance). Notons que ces valeurs sont déterminées par un expert décrivant la base de connaissance ou pourraient être automatiquement obtenues si les règles et leur degré de confiance provenaient d'un processus d'apprentissage.

L'élément de base de la théorie des possibilités est une distribution π qui est une fonction de Ω, l'ensemble des interprétations vers l'intervalle $[0, 1]$. $\pi(\omega)$ représente le degré de compatibilité de l'interprétation ω avec les informations disponibles à propos du monde réel. Par convention, $\pi(\omega) = 0$ signifie que ω est impossible et $\pi(\omega) = 1$ signifie que rien n'empêche ω d'être le monde réel (ω est consistant avec toutes les informations, c'est un modèle de Σ). Quand $\pi(\omega) > \pi(\omega')$, ω est préférée à ω' pour être la description du monde réel. A partir d'une distribution de possibilité π, nous pouvons définir deux manières différentes d'ordonner les formules du langage et ceci selon deux mesures évaluant respectivement la possibilité et la certitude d'une formule p : $\Pi(p) = max\{\pi(\omega) \mid \omega \models p\}$ est le degré de possibilité (ou de consistance) qui évalue l'ampleur avec laquelle p est consistant avec les informations disponibles exprimées par π [22] ; $N(p) = 1 - \Pi(\neg p) = 1 - max\{\pi(\omega) \mid \omega \not\models p\}$ est le degré de nécessité (ou de certitude) qui évalue l'ampleur avec laquelle p est déduite des informations disponibles.

De plus, étant donné une base Σ, seules les distributions de possibilité respectant $\forall (p, \alpha) \in \Sigma$, $N(p) \geq \alpha$ ont un sens et sont dites *compatibles*. Une distribution de possibilité π est appelée distribution la moins spécifique parmi

toutes les distributions compatibles s'il n'y a pas de distribution de possibilité π', où $\pi' \neq \pi$, compatible avec Σ telle que $\forall \omega, \pi'(\omega) \geq \pi(\omega)$. La distribution de possibilité la moins spécifique π_Σ existe toujours [9] et est définie par :

$\pi_\Sigma(\omega) = 1$ si ω est un modèle de Σ

$\pi_\Sigma(\omega) = 1 - \max\{\alpha \mid \omega \not\models p, (p, \alpha) \in \Sigma\}$ sinon.

Sémantique des modèles stables. Parmi les différentes variantes de l'ASP nous nous intéressons ici aux *programmes logiques normaux*, interprétés par la *sémantique des modèles stables* [11]. Nous considérons donné un ensemble non vide d'atomes $\mathcal{X}$ qui détermine le langage des programmes. Un *programme logique normal* est un ensemble de règles de la forme :

$$c \leftarrow a_1, \ \ldots, \ a_n, \ not\ b_1, \ \ldots, \ not\ b_m.$$

où $n \geq 0$, $m \geq 0$, $\{a_1, \ldots, a_n, b_1, \ldots, b_m, c\} \subseteq \mathcal{X}$. Un terme comme $not\ b$ est appelé une *négation par défaut*. La signification intuitive d'une telle règle est : « si vous avez tous les a_i et aucun b_j, alors vous pouvez conclure c ».

Pour une règle r, nous utilisons les notations suivantes (extensibles à un ensemble de règles) : le prérequis positif de r, $corps^+(r) = \{a_1, \ \ldots, \ a_n\}$; le prérequis négatif de r, $corps^-(r) = \{b_1, \ \ldots, \ b_m\}$; la conclusion de r, $tête(r) = c$ et la projection positive de r, $r^+ = tête(r) \leftarrow corps^+(r)$. Si un programme P ne contient pas de négation par défaut (c'est-à-dire $corps^-(P) = \emptyset$), alors P est un *programme logique défini* et il possède un seul modèle de Herbrand minimal noté $Cn(P)$.

Le *réduit* P^X d'un programme P par rapport à un ensemble d'atomes X est le programme logique défini : $P^X = \{r^+ \mid r \in P, \ corps^-(r) \cap X = \emptyset\}$.

Définition 1.1 *[11] Soit P un programme logique normal et S un ensemble d'atomes. S est un modèle stable de P si et seulement si $S = Cn(P^S)$.*

Notons qu'un programme peut avoir aucun, un ou plusieurs modèles stables. Dans le premier cas, le programme est *inconsistant*, sinon il est *consistant*. Si un atome appartient à au moins un modèle stable de P, il est dit conséquence *crédule* de P et s'il appartient à tous les modèles stables de P, il est dit conséquence *sceptique* de P.

Soit A un ensemble d'atomes, r une règle et P un programme (défini ou normal). On dit que r est *applicable* dans A si $corps^+(r) \subseteq A$ et on note $App(P, A)$ le sous-ensemble de P de ses règles applicables dans A. A satisfait r (ou r est satisfaite par A), noté $A \models r$, si r est applicable dans $A \Rightarrow tête(r) \in A$. A est *clos* par rapport à P si $\forall r \in P, A \models r$. P est *enraciné* s'il peut être ordonné selon la suite $\langle r_1, \ldots, r_n \rangle$ telle que $\forall i, 1 \leq i \leq n, r_i \in App(P, tête(\{r_1, \ldots, r_{i-1}\}))$. $Cn(P)$, le modèle (de Herbrand) minimal d'un programme logique défini P, est le plus petit ensemble d'atomes clos sous P et il peut être calculé comme le plus petit point fixe de l'opérateur de conséquence $T_P : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{X}}$ tel que $T_P(A) = tête(App(P, A))$.

Ainsi, nous pouvons établir le résultat suivant qui clarifie les liens entre le modèle minimal A d'un programme P et les règles le produisant : A est un modèle *supporté* par l'ensemble de règles applicables $App(P, A)$ qui doit être enraciné.

Lemme 1.1 *Soit P un programme logique défini et A un ensemble d'atomes,*

A est le modèle de Herbrand minimal de P

$$\Longleftrightarrow$$

$A = tête(App(P, A))$ et $App(P, A)$ est enraciné

2 Programmes logiques définis possibilistes

On considère donné un ensemble fini $\mathcal{X}$ d'atomes et un ensemble fini et totalement ordonné $\mathcal{N} \subseteq]0, 1]$ de valeurs de nécessité. Un *atome possibiliste* est une paire $p = (x, \alpha) \in \mathcal{X} \times \mathcal{N}$. On note $p^* = x$ la projection classique de p et $n(p) = \alpha$ son degré de nécessité. Ces notations peuvent être étendues à un *ensemble d'atomes possibilistes* (e.a.p.) A qui est un ensemble dans lequel chaque atome x apparaît au plus une fois.

Un *programme logique défini possibiliste* (p.l.d.p.) est un ensemble de *règles possibilistes* de la forme :

$$(c \leftarrow a_1, \ \ldots, \ a_n. \ , \ \alpha)$$

où $n \geq 0, \{a_1, \ \ldots, \ a_n, \ c\} \subseteq \mathcal{X}, \alpha \in \mathcal{N}$.

La *projection classique* d'une règle possibiliste r est $r^* = c \leftarrow a_1, \ \ldots, \ a_n.$ $n(r) = \alpha$ est le degré de nécessité représentant le niveau de certitude de l'information décrite par r. Si R est un ensemble de règles possibilistes alors $R^* = \{r^* \mid r \in R\}$ est le programme défini obtenu à partir de P en omettant toutes les valeurs de nécessité.

2.1 Théorie des modèles

A partir d'un p.l.d.p. P, on peut déterminer, comme c'est le cas en logique possibiliste, plusieurs distributions de possibilité sur tous les ensembles de $2^{\mathcal{X}}$ et compatibles avec P. Notons qu'il y a une correspondance entre les interprétations en logique des propositions et les ensembles d'atomes en ASP. Comme en logique possibiliste, le degré de possibilité d'un ensemble d'atomes est déterminé par les degrés de nécessité des règles du programme qui ne sont pas satisfaites par cet ensemble. La satisfiabilité d'une règle classique r est basée sur son applicabilité par rapport à un ensemble d'atomes A, ainsi $A \not\models r$ ssi $corps^+(r) \subseteq A$ et $tête(r) \notin A$ (voir la section 1). Mais, nous devons noter que la contradiction d'une règle n'est pas suffisante pour déterminer le degré de possibilité d'un ensemble dans la mesure où, en ASP, il est important de prendre en compte les notions d'enracinement et de stabilité (voir lemme 1.1). Par exemple, l'ensemble $A = \{a, b\}$ satisfait chaque règle de $P = \{a \leftarrow b., \ b \leftarrow a.\}$, mais l'enracinement n'est pas vérifié. De même, l'ensemble $A' = \{a, b, d\}$ satisfait toutes les règles de $P' = \{a., b \leftarrow a., d \leftarrow c.\}$ mais ce n'est pas un modèle supporté car d ne peut pas être produit par une règle de P' applicable dans A'. Dans ces deux cas, la possibilité doit être 0 dans la mesure où ces ensembles ne peuvent pas du tout être le modèle minimal, même s'ils satisfont toutes les règles de leur programme correspondant.

Définition 2.1 *Soit P un p.l.d.p.et $\pi : 2^{\mathcal{X}} \rightarrow [0,1]$ une distribution de possibilité. π est compatible avec P si*

$$\forall A \in 2^{\mathcal{X}} \begin{cases} A \not\subseteq \text{tête}(App(P^*, A)) \Rightarrow \pi(A) = 0 \\ App(P^*, A) \text{ n'est pas enraciné} \Rightarrow \pi(A) = 0 \\ A \text{ est modèle minimal de } P^* \Rightarrow \pi(A) = 1 \\ \text{et sinon } \pi(A) \leq 1 - \max_{r \in P}\{n(r) \mid A \not\models r^*\} \end{cases}$$

Le degré de nécessité attaché à chaque règle définit seulement une borne inférieure (et non pas une valeur exacte) de la certitude de la règle. Ainsi, comme rappelé à la section 1, plusieurs distributions de possibilité sont compatibles avec ces degrés. Cependant, on s'intéresse seulement à la moins informative, c'est-à-dire à la *moins spécifique*, dont la caractérisation est donnée ci-dessous.

Proposition 2.1 *Soit P un p.l.d.p. alors $\pi_P : 2^{\mathcal{X}} \rightarrow [0,1]$ définie par : $\forall A \in 2^{\mathcal{X}}$,*

1. *si $A \not\subseteq \text{tête}(App(P^*, A))$ alors $\pi_P(A) = 0$*
2. *si $App(P^*, A)$ n'est pas enraciné alors $\pi_P(A) = 0$*
3. *sinon*
   - *si $\forall r \in P, A \models r^*$ alors $\pi_P(A) = 1$*
   - *sinon $0 \leq \pi_P(A) < 1$ et $\pi_P(A) = 1 - \max_{r \in P}\{n(r) \mid A \not\models r^*\}$*

est la distribution de possibilité la moins spécifique.

La définition de π_P assure qu'elle est compatible avec P et le troisième point ordonne les ensembles qui peuvent être solutions en fonction des poids des règles falsifiées. Le résultat suivant assure que seul le modèle de Herbrand minimal de P est totalement possible.

Proposition 2.2 *Soit P un p.l.d.p. et $A \subseteq \mathcal{X}$ un ensemble d'atomes, alors $\pi_P(A) = 1 \iff A = Cn(P^*)$*

Nous donnons maintenant la définition de l'inférence qui est, dans le cadre de l'ASP, l'évaluation du degré de nécessité de chaque atome de l'univers.

Définition 2.2 *Soit P un p.l.d.p. et π_P la distribution de possibilité la moins spécifique compatible avec P, les deux mesures duales de possibilité et de nécessité sont :*

$$\Pi_P(x) = \max_{A \in 2^{\mathcal{X}}}\{\pi_P(A) \mid x \in A\}$$
$$N_P(x) = 1 - \max_{A \in 2^{\mathcal{X}}}\{\pi_P(A) \mid x \notin A\}$$

$\Pi_P(x)$ donne le niveau de consistance de x par rapport au p.l.d.p. P. Ainsi, quand un atome x appartient au modèle minimal du programme classique sa possibilité est totale. $N_P(x)$ évalue le niveau auquel x est inféré à partir de P.

Proposition 2.3 *Soit P un p.l.d.p., $Cn(P^*)$ le modèle minimal de P^* et $x \in \mathcal{X}$ alors :*

1. *$x \in Cn(P^*) \Rightarrow \Pi_P(x) = 1$ et $x \notin Cn(P^*) \Rightarrow \Pi_P(x) = 0$*
2. *$x \notin Cn(P^*) \iff N_P(x) = 0$*

De plus, la mesure de nécessité nous permet d'introduire la définition de *modèle possibiliste* d'un p.l.d.p.

Définition 2.3 *Soit P un p.l.d.p., alors l'ensemble*
$\Pi\mathcal{M}(P) = \{(x, N_P(x)) \mid x \in \mathcal{X}, N_P(x) > 0\}$
est son modèle possibiliste.

Proposition 2.4 *Soit P un p.l.d.p. alors : $(\Pi\mathcal{M}(P))^*$ est le plus petit modèle de P^*.*

Exemple 2.1 *Soit le p.l.d.p. $P = \{(a \leftarrow ., 0.8),\ (b \leftarrow a., 0.6),\ (d \leftarrow a., 0.5),\ (d \leftarrow c., 0.9)\}$. La distribution de possibilité la moins spécifique induite par P est nulle pour chaque ensemble d'atomes inclus dans $\mathcal{X} = \{a, b, c, d\}$ excepté pour $\pi_P(\emptyset)$ = 0.2, $\pi_P(\{a\})$ = 0.4, $\pi_P(\{a, b\})$=0.5, $\pi_P(\{a, d\})$=0.4 et $\pi_P(\{a, b, d\})$=1 (le modèle minimal de P^*). Par conséquent, la possibilité de chaque atome est : $\Pi_P(a) = \Pi_P(b) = \Pi_P(d) = 1$ et $\Pi_P(c) = 0$ et sa certitude en terme de degré de nécessité est : $N_P(a)$ = 0.8, $N_P(b)$ = 0.6, $N_P(c)$ = 0, $N_P(d)$ = 0.5. Ainsi, $\Pi\mathcal{M}(P) = \{(a, 0.8), (b, 0.6), (d, 0.5)\}$ est le modèle possibiliste de P.*

2.2 Théorie du point fixe

Les définitions données dans cette sous-section sont proches de celles données dans [8]. Cependant, nous adoptons ici le point de vue de l'ASP et nous utilisons les ensembles d'atomes à la place des interprétations classiques dans la mesure où la distribution de possibilité sous-jacente est définie sur les ensembles d'atomes.

Définition 2.4 *Soit $\mathcal{A} = 2^{\mathcal{X} \times \mathcal{N}}$ l'ensemble fini de tous les e.a.p. induits par $\mathcal{X}$ et $\mathcal{N}$. $\forall A, B \in \mathcal{A}$, on définit :*

$$\begin{array}{rcl} A \sqcap B & = & \{(x, \min\{\alpha, \beta\}), (x, \alpha) \in A, (x, \beta) \in B\} \\ A \sqcup B & = & \{(x, \alpha) \mid (x, \alpha) \in A, x \notin B^*\} \\ & \cup & \{(x, \beta) \mid x \notin A^*, (x, \beta) \in B\} \\ & \cup & \{(x, \max\{\alpha, \beta\}) \mid (x, \alpha) \in A, (x, \beta) \in B\} \end{array}$$

$$A \sqsubseteq B \iff \begin{cases} A^* \subseteq B^* \\ \text{et } \forall a, \alpha, \beta, \\ (a, \alpha) \in A \wedge (a, \beta) \in B \Rightarrow \alpha \leq \beta \end{cases}$$

Proposition 2.5 *$\langle \mathcal{A}, \sqsubseteq \rangle$ est un treillis complet.*

Définition 2.5 *Soit $r = (c \leftarrow a_1,\ \ldots,\ a_n., \alpha)$ une règle possibiliste et A un e.a.p.,*
- *r est α-applicable dans A si $corps(r^*) = \emptyset$*
- *r est β-applicable dans A si $\beta = \min\{\alpha, \alpha_1, \ldots, \alpha_n\}$, $\{(a_1, \alpha_1), \ldots, (a_n, \alpha_n)\} \subseteq A$,* [1]
- *r est 0-applicable sinon.*

Et, pour un p.l.d.p. P donné et un atome x, $App(P, A, x) = \{r \in P \mid \text{tête}(r^*) = x, r \text{ est } \nu\text{-applicable dans } A, \nu > 0\}$

Le degré d'applicabilité d'une règle r capture la certitude du processus d'inférence réalisé en fonction d'un e.a.p. A lors de l'application de r. Si le corps de r est vide, alors r est applicable avec son degré de certitude propre. Si le

[1] Pour deux e.a.p. A et B, $A \subseteq B$ signifie l'inclusion ensembliste classique à ne pas confondre avec l'ordre $\sqsubseteq$ de la définition 2.4.

corps n'est pas vérifié (pas satisfait par A), alors r n'est pas du tout applicable. Sinon, le niveau d'applicabilité de r dépend du niveau de certitude des propositions permettant son enracinement et de son propre degré de nécessité. Le degré de nécessité d'une conjonction (le corps de la règle) est d'abord la valeur minimale des valeurs de nécessité des sous-formules (atomes) qui y sont impliquées. Ensuite, la certitude d'application d'une règle est la valeur minimale entre la certitude de la règle et la certitude de son corps. Cette approche est similaire au modus ponens gradué en logique possibiliste [9].

Définition 2.6 *Soit P un p.l.d.p. et A un e.a.p. L'*opérateur de conséquence immédiate possibiliste ΠT_P *est défini d'un e.a.p. vers un autre de la manière suivante :* $\Pi T_P(A) = \left\{ \begin{array}{l} (x, \delta) \mid x \in tête(P^*), App(P, A, x) \neq \emptyset, \\ \delta = \max\limits_{r \in App(P,A,x)} \{\nu \mid r \text{ est } \nu\text{-applicable dans } A\} \end{array} \right\}$
et l'opérateur itéré ΠT_P^k *est défini par*
$\Pi T_P^0 = \emptyset$ *et* $\Pi T_P^{n+1} = \Pi T_P(\Pi T_P^n), \forall n \geq 0$.

Nous pouvons remarquer ici que, si une conclusion est obtenue par différents enchaînements, sa certitude est égale à la plus grande certitude de chacun des cas qui permettent d'obtenir cette conclusion (opérateur $\max$). C'est de nouveau en accord avec les principes de la logique possibiliste.

Proposition 2.6 *Soit P un p.l.d.p., alors ΠT_P a un plus petit point fixe $\sqcup_{n \geq 0} \Pi T_P^n$ que nous appelons l'ensemble des* conséquences possibilistes *de P et notons $\Pi Cn(P)$.*

Exemple 2.2 *Soit P le p.l.d.p. de l'exemple 2.1, alors*
$\Pi T_P^0 = \emptyset$, $\Pi T_P^1 = \{(a, 0.8)\}$,
$\Pi T_P^2 = \{(a, 0.8), (b, 0.6), (d, 0.5)\}$,
$\Pi T_P^3 = \{(a, 0.8), (b, 0.6), (d, 0.5)\} = \Pi T_P^k, \forall k > 3$.
Donc $\Pi Cn(P) = \{(a, 0.8), (b, 0.6), (d, 0.5)\}$.

Théorème 2.1 *Soit P un p.l.d.p.,* $\Pi Cn(P) = \Pi \mathcal{M}(P)$.

Comme illustré et formalisé ci-dessus, notre opérateur ΠT_P peut donc être utilisé pour calculer le modèle possibiliste d'un p.l.d.p. Ce résultat montre l'équivalence entre les approches syntaxique et sémantique de notre cadre tandis que, dans [8], seul un traitement syntaxique est proposé.

3 Programmes logiques normaux possibilistes

3.1 Modèles stables possibilistes

Ici, nous étendons notre approche au raisonnement non monotone en autorisant des négations par défaut dans les programmes et nous formalisons la notion de *modèle stable possibiliste* (m.s.p.). Ceci étend la sémantique des modèles stables en prenant en compte le degré de nécessité des règles d'un *programme logique normal possibiliste* (p.l.n.p.) donné. Un tel programme est un ensemble fini de règles de la forme :

$$(c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m.\ , \alpha)$$

où $n \geq 0, m \geq 0$ pour lequel nous devons simplement préciser que $\forall i, b_i \in \mathcal{X}$, le reste de la définition restant identique au cas d'un p.l.d.p. (voir le début de la section 2). Comme dans le cas classique sans valeurs de nécessité, nous devons définir ce qu'est la réduction à la Gelfond-Lifschitz [11] d'un p.l.n.p. avant d'introduire la *sémantique des modèles stables possibilistes*.

Définition 3.1 *Soit A un ensemble d'atomes et P un p.l.n.p. Le* réduit possibiliste *de P en fonction de A est le p.l.d.p.* $P^A = \{(r^{*^+}, n(r)) \mid r \in P, corps^-(r) \cap A = \emptyset\}$.

Définition 3.2 *Soit P un p.l.n.p. et S un e.a.p. S est un modèle stable possibiliste de P ssi* $S = \Pi Cn(P^{(S^*)})$.

Par analogie avec les programmes logiques normaux classiques (sans valeurs de nécessité attachée aux règles), on dit qu'un p.l.n.p. P est *consistant* si P a au moins un modèle stable possibiliste. Sinon P est dit *inconsistant*.
De plus, quand un atome possibiliste appartient à au moins un modèle stable possibiliste, il est appelé conséquence possibiliste *crédule* de P et, quand il appartient à tous les modèles stables possibilistes de P, il est appelé conséquence possibiliste *sceptique* de P.

Exemple 3.1 *Nous pouvons maintenant représenter l'exemple de l'introduction en y ajoutant des niveaux de certitude. Nous obtenons le p.l.n.p. suivant*

$$P = \left\{ \begin{array}{l} (med_1 \leftarrow mal_1, not\ med_2., 1) \\ (med_2 \leftarrow mal_2, not\ med_1., 1) \\ (g_1 \leftarrow med_1, mal_1., 0.7) \\ (g_2 \leftarrow med_2, mal_2., 0.3) \\ (mal_1 \leftarrow ., 0.9)(mal_2 \leftarrow ., 0.7) \end{array} \right\}$$

Les deux premières règles expriment le fait que, pour chaque maladie, nous avons un médicament approprié, ces deux médicaments sont incompatibles, mais leur convenance est totalement certaine. La troisième (resp. quatrième) règle exprime que le médicament med_1 (resp. med_2) a une efficacité assez certaine (resp. peu certaine) pour guérir la maladie mal_1 (resp. mal_2). Les deux dernières règles (qui sont des observations) décrivent que le diagnostic de la maladie mal_1 (resp. mal_2) est quasi certain (resp. assez certain). Ainsi, les deux m.s.p. de P sont $\{(mal_1, 0.9), (mal_2, 0.7), (med_1, 0.9), (g_1, 0.7)\}$ et $\{(mal_1, 0.9), (mal_2, 0.7), (med_2, 0.7), (g_2, 0.3)\}$. Le médecin se retrouve toujours devant une alternative. D'un côté, il peut donner le médicament med_1 et il est assez certain que le patient sera guéri de la maladie mal_1. D'un autre côté, il peut donner le médicament med_2 et il est peu certain que le patient sera guéri de la maladie mal_2. Ainsi, le médecin se retrouve avec une information additionnelle lui permettant par exemple de privilégier le traitement le plus sûr. Cependant, si le médecin considère que la maladie mal_2 est très grave, il choisira peut-être le médicament med_2 même si sa certitude de soin est basse. C'est pourquoi il est intéressant d'obtenir, et de conserver, les deux modèles stables dans lesquels chacune des conclusions est pondérée avec un degré de certitude,

contrairement à une approche par préférence [7] qui a pour but de supprimer certaines alternatives parmi les solutions potentielles.

Proposition 3.1 *Soit P un p.l.n.p.*

1. *Soit A un m.s.p. de P et $\alpha \in \mathcal{N}, \alpha > 0$, alors $(x, \alpha) \in A \iff \alpha = N_{P^{(A^*)}}(x)$.*
2. *Soit S un modèle stable de P^*, alors $\{(x, N_{P^S}(x)) \mid x \in \mathcal{X}, N_{P^S}(x) > 0\}$ est un modèle stable possibiliste de P.*
3. *Soit A un modèle stable possibiliste de P, alors A^* est un modèle stable de P^*.*

Ce résultat montre qu'il y a une correspondance un à un entre les modèles stables possibilistes d'un p.l.n.p. P et les modèles stables de sa contrepartie classique P^*. Il montre que le problème de décision de l'existence d'un m.s.p. pour un p.l.n.p. reste dans la même classe de complexité que le problème de l'existence d'un modèle stable pour un programme logique normal, c'est-à-dire la classe des problèmes NP-complets. Cela permet donc d'envisager l'implantation d'un système de calcul de modèles stables possibilistes à un coût algorithmique équivalent à celui d'un système de calcul de modèles stables classiques. C'est ce que nous décrivons dans la sous-section 4.1.

3.2 Distribution de possibilité

Dans la définition 3.2, nous avons proposé une méthode syntaxique pour calculer les modèles stables possibilistes d'un p.l.n.p. utilisant un opérateur de point fixe ΠCn défini pour un p.l.d.p. Maintenant, nous examinons la sémantique qui peut être donnée à ce formalisme via une distribution de possibilité induite par les valeurs de nécessité associées à chaque règle normale. Cette distribution $\tilde{\pi}_P$ reflète la capacité de chaque ensemble d'atomes à être un modèle stable de P^* conformément à la définition 1.1.

Définition 3.3 *Soit P un p.l.n.p. alors $\tilde{\pi}_P$ est la distribution de possibilité définie par $\tilde{\pi}_P : 2^{\mathcal{X}} \rightarrow [0, 1]$ et respectant : $\forall A \in 2^{\mathcal{X}}, \tilde{\pi}_P(A) = \pi_{P^A}(A)$.*

On peut observer que cette définition de $\tilde{\pi}_P$ peut être paraphrasée par : « la possibilité pour un ensemble d'atomes A d'être un modèle stable de P est sa possibilité d'être le modèle minimal du programme P réduit par A ».

Proposition 3.2 *Soit P un p.l.n.p. et $A \in 2^{\mathcal{X}}$, alors*
$\tilde{\pi}_P(A) = 1 \iff$ *A est un modèle stable de P^**

Exemple 3.2 *La distribution de possibilité induite par $P = \{(a \leftarrow ., 1), (b \leftarrow ., 1), (c \leftarrow a, not\ d., 0.4), (d \leftarrow b, not\ c., 0.8), (e \leftarrow c., 0.6), (e \leftarrow d., 0.5)\}$ est nulle pour chaque ensemble d'atomes inclus dans $\mathcal{X} = \{a, b, c, d, e\}$ excepté pour $\tilde{\pi}_P(\{a, b\}) = 0.2$, $\tilde{\pi}_P(\{a, b, c\}) = 0.4$, $\tilde{\pi}_P(\{a, b, d\}) = 0.5$ et $\tilde{\pi}_P(\{a, b, c, e\}) = \tilde{\pi}_P(\{a, b, d, e\}) = 1$ ($\{a, b, c, e\}$ et $\{a, b, d, e\}$ sont les deux modèles stables de P^*).*

Définition 3.4 *Soit P un p.l.n.p. et $\tilde{\pi}_P$ sa distribution de possibilité associée, on définit, comme pour les p.l.d.p., les deux mesures duales de possibilité et de nécessité :*

$$\tilde{\Pi}_P(x) = \max_{A \in 2^{\mathcal{X}}} \{\tilde{\pi}_P(A) \mid x \in A\}$$
$$\tilde{N}_P(x) = 1 - \max_{A \in 2^{\mathcal{X}}} \{\tilde{\pi}_P(A) \mid x \notin A\}$$

Proposition 3.3 *Soit P un p.l.n.p. consistant et $x \in \mathcal{X}$,*

1. *x est une conséquence crédule de $P^* \Rightarrow \tilde{\Pi}_P(x) = 1$*
2. *x n'est pas une conséquence sceptique de $P^* \iff \tilde{N}_P(x) = 0$*

Remarquons que, si un atome x n'est pas une conséquence crédule de P alors ceci n'implique pas forcément que $\tilde{\Pi}_P(x) = 0$ comme c'est le cas pour un atome qui n'est pas une conséquence d'un p.l.d.p. (voir la proposition 2.3 item 1). Par exemple, $P = \{(a. \leftarrow, 0.6), (b \leftarrow not\ a., 0.7)\}$ a un seul m.s.p. $\{(a, 0.6)\}$ et donc b n'est pas une conséquence possibiliste crédule de P. Mais, la distribution de possibilité est $\tilde{\pi}_P(\emptyset) = 0.3, \tilde{\pi}_P(\{a\}) = 1, \tilde{\pi}_P(\{b\}) = 0.4$ et $\tilde{\pi}_P(\{a, b\}) = 0$ donc $\tilde{\Pi}_P(b) = 0.4$. Ceci vient du fait que, pour ce programme, b n'est pas complètement impossible. En fait, b n'est pas une conséquence crédule simplement parce que a bloque l'applicabilité de la règle concluant b. Ainsi, en d'autres termes, si on a a alors on ne peut pas avoir b, mais si a est absent alors on peut avoir b. Dans la mesure où la certitude d'avoir a est seulement de 0.6, la possibilité d'avoir b est de manière naturelle 0.4.

4 Autour de la sémantique des modèles stables possibilistes

4.1 Calcul de modèles stables possibilistes

Comme nous l'avons énoncé à la fin de la sous-section 3.1 il y a une correspondance un à un entre les m.s.p. d'un p.l.n.p. P et les modèles stables classiques de P^*. En fait, étant donné S un modèle stable de P^*, $\Pi Cn(P^S)$ est un modèle stable possibiliste de P. Un moyen simple pour implémenter un système calculant les m.s.p. d'un p.l.n.p. est donc de s'appuyer sur un système existant pour calculer les modèles stables de P^* ; puis, pour chaque modèle S trouvé, d'appliquer l'opérateur ΠT_{P^S} pour calculer, en temps polynômial, le m.s.p. de P lui correspondant $\Pi Cn(P^S)$.

Divers logiciels capables de calculer les modèles stables d'un programme sont disponibles, par exemple : Smodels (`www.tcs.hut.fi/Software/smodels` [20]) ou DLV (`www.dbai.tuwien.ac.at/proj/dlv` [13]). Pour des raisons de disponibilités de code source et de familiarité avec les logiciels, nous avons choisi de baser notre système sur Smodels. Smodels n'admettant que des programmes sans variable, il est utilisé avec le système frontal Lparse qui permet d'étendre le langage avec des variables (et de nombreux autres éléments). Lparse se charge donc d'instancier efficacement les règles d'un programme afin

de le rendre utilisable par Smodels. Partant de là, nous avons développé en C++ le système posSmodels[2] dont le fonctionnement global est synthétisé dans la figure 2.

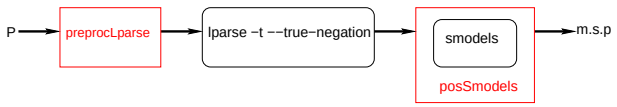

FIG. 2 – Chaîne de traitement

Les règles acceptées par notre système sont de la forme

$$\alpha \quad c \text{ :- } a_1, \ldots, a_m, \ not\ b_1, \ldots, \ not\ b_n.$$

où α est un entier et où les atomes peuvent contenir des variables.

Exemple 4.1 *Supposons que l'on dispose d'un graphe orienté à 4 sommets $s(1), \ldots, s(4)$ comportant des arcs orientés (X, Y) totalement certains (ou sûrs) si $X < Y$ et peu certains (ou peu fiables) si $Y < X$. On cherche à déterminer quel est le chemin hamiltonien (passant une et une seule fois par chaque sommet), partant du sommet 1 et ayant la plus grande certitude, sécurité. Selon le principe admis que la sûreté d'une chaîne est égale à la sûreté de son plus faible maillon, on peut utiliser le programme ci-dessous donné dans la syntaxe admise par notre système* posSmodels *pour représenter ce problème*[3]*. Ainsi, chaque m.s.p. de ce programme correspond à une solution représentée par les atomes $in(i, j)$ et le degré de l'atome $fin(k)$ représente la sûreté du chemin grâce à la propagation le long de différentes règles du niveau de certitude de chaque arc constituant le chemin solution.*

% les sommets
100 $s(1..4).$
% les arcs (X,Y) sont plus sûrs quand $X < Y$
100 $a(X, Y)$:- $s(X), s(Y), X < Y.$
20 $a(X, Y)$:- $s(X), s(Y), X > Y.$
% le sommet de départ
100 $dep(1).$
% mettre ou ne pas mettre un arc dans le chemin
100 $in(X, Y)$:- $a(X, Y), dep(X), not\ out(X, Y).$
100 $out(X, Y)$:- $a(X, Y), dep(X), not\ in(X, Y).$
100 $in(X, Y)$:- $a(X, Y), vu(X), not\ out(X, Y).$
100 $out(X, Y)$:- $a(X, Y), vu(X), not\ in(X, Y).$
% les sommets parcourus
100 $vu(Y)$:- $s(X), s(Y), in(X, Y).$
% X a un successeur dans le chemin
100 $a\_succ(X)$:- $s(X), s(Y), in(X, Y).$
% Y est le sommet où se termine le chemin
100 $fin(Y)$:- $s(X), s(Y), in(X, Y), not\ a\_succ(Y).$
% chaque sommet, sauf celui de départ, doit être vu
100 :- $s(X), not\ dep(X), not\ vu(X).$
% le sommet de départ ne doit pas être vu
100 :- $dep(X), vu(X).$
% un seul arc au plus part de chaque sommet
100 :- $s(X), s(Y1), s(Y2), Y1! = Y2,$
$in(X, Y1), in(X, Y2).$
% un seul arc au plus arrive en chaque sommet
100 :- $s(Y), s(X1), s(X2), X1! = X2,$
$in(X1, Y), in(X2, Y).$

La première partie du système, preprocLparse, effectue un pré-traitement syntaxique sur le programme possibiliste de manière à le rendre exploitable par Lparse (qui va instancier et simplifier les règles). Il faut donc supprimer les degrés de nécessité α des règles pour que Lparse puisse les traiter, puis restituer ces degrés sur les règles instanciées et simplifiées. Pour cela, chaque règle possibiliste va être transformée en une règle classique dans le corps de laquelle un nouvel atome $nu\_\_(\alpha)$, mémorisant le degré α de la règle, est ajouté. On a donc $preproc(r) = r'$ *t.q.*

$$\begin{cases} tête(r') & = & tête(r) \\ corps^+(r') & = & corps^+(r) \cup \{nu\_\_(n(r))\} \\ corps^-(r') & = & corps^-(r) \end{cases}$$

Le pré-traitement d'un p.n.l.p. P est donc le suivant :

$$\begin{array}{rcl} Preproc(P) & = & \{preproc(r) \mid r \in P\} \\ & \cup & \{\sharp external\ nu\_\_(X).\} \\ & \cup & \{nu\_\_(\alpha). \mid \alpha \in \mathcal{A}\} \end{array}$$

La directive $external\ nu\_\_(X)$ assure que les atomes de la forme $nu\_\_(\alpha)$ sont laissés tels quels par Lparse. Par exemple, la règle de l'exemple 4.1

$$20 \quad a(X, Y) \text{ :- } s(X), s(Y), X > Y.$$

sera transformée par preprocLparse en

$$a(X, Y) \text{ :- } s(X), s(Y), X > Y, nu\_\_(20).$$

Après ce pré-traitement, Lparse instancie classiquement chaque règle et fait les simplifications usuelles, ce qui, pour la règle citée ici, donne

$a(2,1)$:- $s(2), s(1), nu\_\_(20).$ $\quad a(3,1)$:- $s(3), s(1), nu\_\_(20).$
$a(4,1)$:- $s(4), s(1), nu\_\_(20).$ $\quad a(3,2)$:- $s(3), s(2), nu\_\_(20).$
$a(4,2)$:- $s(4), s(2), nu\_\_(20).$ $\quad a(4,3)$:- $s(4), s(3), nu\_\_(20).$

Une fois obtenu le programme normal instancié par Lparse, on reconstruit facilement le programme possibiliste correspondant en supprimant l'atome $nu\_\_(\alpha)$ du corps de chaque règle et en lui affectant le degré α. Ainsi, nous avons pu fabriquer un p.n.l.p. totalement instancié à partir d'un programme avec variables en conservant les degrés adéquats et surtout en se reposant sur un logiciel efficace pour cette difficile tâche d'instanciation des variables.

La troisième partie, posSmodels, utilise l'API de Smodels pour calculer les modèles stables de P^* et applique, selon l'algorithme de la figure 3, l'opérateur de conséquence possibiliste associé au réduit possibiliste de P pour chaque modèle stable calculé.

Le programme de l'exemple 4.1 admet six m.s.p. correspondant aux six chemins hamiltoniens possibles à partir du sommet 1. Le degré de chaque atome $in(i, j)$ correspond au degré du maillon le plus faible dans le chemin depuis le sommet de départ jusqu'au sommet j (la sûreté de cette chaîne), et le degré de l'atome $fin(j)$ correspond à la sûreté du chemin total. Ainsi, seul le modèle stable correspondant au chemin $(1, 2, 3, 4)$ comprend l'atome fin avec

[2] Téléchargeable à partir de `www.info.univ-angers.fr/pub/pn/Softwares/PosSmodels`

[3] On utilise parfois des règles sans tête (ex : $\leftarrow a, not\ b.$) agissant sur les modèles comme des contraintes et correspondant à la simplification de règles dont le corps négatif contient la tête (ex : $c \leftarrow a, not\ b, not\ c.$).

calculTousMSP(données Solv : solveur ASP, P : p.l.n.p)
début
tant que Solv(P^*) retourne un m.s. S faire
$PP \leftarrow \{(r^{*+}, n(r)) \mid r \in P, corps^+(r^*) \subseteq S$
$et\ corps^-(r^*) \cap S = \emptyset\}$
pour chaque règle $R \in PP$
calculer $L(R)$, la cardinalité de $corps^+(R)$
finpour
$Res \leftarrow \emptyset$ /* le m.s.p. à calculer */
répéter
$PointFixe \leftarrow vrai$
pour chaque règle $R \in PP$ faire
si $L(R) = 0$ alors
$deg \leftarrow$ degré d'applicabilité de R dans Res
si $tête(R) \notin Res^*$ alors /* nouvel atome */
pour chaque règle $R' \mid tête(R) \in corps^+(R')$
$L(R') \leftarrow L(R') - 1$
finpour
finsi
$Res \leftarrow Res \sqcup \{(tête(R), deg)\}$
siRes a été modifié alors
$PointFixe \leftarrow faux$
finsi
si $n(R) = deg$ alors retirer R de PP finsi
finsi
finpour
jusqu'à $PointFixe$
écrire Res
fintq
fin

FIG. 3 – Calcul des modèles stables possibilistes

le degré maximal[4].
Notons que les résultats expérimentaux avec ce système confirment nos résultats théoriques quant à la complexité du problème : dès lors qu'un problème difficile est soumis au logiciel, le temps de calcul des conséquences possibilistes devient négligeable par rapport à celui du calcul des modèles stables par Smodels. Le traitement de l'incertitude intégré au raisonnement non monotone est donc en quelque sorte gratuit du point de vue du coût algorithmique.

4.2 Gestion de l'inconsistance

L'une des caractéristiques de la logique possibiliste est son aptitude à gérer une base de connaissances inconsistante. En effet, en l'absence de modèle classique, la logique possibiliste permet d'une part la définition de « *modèles préférés* »et d'autre part propose une méthode pour restaurer la consistance de cette base. L'idée fondamentale est de considérer que les degrés de certitude partitionnent la base en strates et de supprimer toutes les formules des strates les plus basses jusqu'à ce que l'on obtienne un ensemble de formules consistant. C'est cette approche du problème que nous reprenons ici pour les p.l.n.p. en faisant remarquer par avance que le caractère non monotone de la sémantique des modèles stables rend l'approche moins directe.

Définition 4.1 *Soit P un p.l.n.p., l'*α-coupe stricte *de P est le sous-programme $P_{>\alpha} = \{r \in P \mid n(r) > \alpha\}$. Le* degré de coupe consistante *de P est*

$$ConsCutDeg(P) = \begin{cases} 0 \textit{ si } P \textit{ est consistant} \\ 1 \textit{ si } P_{>\alpha} \textit{est inconsistant} \forall \alpha \in \mathcal{N} \\ \min\limits_{\alpha \in \mathcal{N}} \{P_{>\alpha} \textit{ est consistant}\} \textit{ sinon} \end{cases}$$

Le degré de coupe consistante d'un programme P définit donc le plus bas niveau de certitude au-dessus duquel une stricte α-coupe de P est consistante.

Définition 4.2 *Soit P un p.l.n.p., son degré d'inconsistance est $InconsDeg(P) = 1 - \max\limits_{A \in 2^{\mathcal{X}}} \{\tilde{\pi}_P(A)\}$.*

Définition 4.3 *Soit cut la fonction définie sur les p.l.n.p. par* $\begin{cases} cut(P) = P \textit{ si } InconsDeg(P) = 0 \\ cut(P) = cut(P_{>InconsDeg(P)}) \textit{ sinon} \end{cases}$

Proposition 4.1 *Soit P un p.l.n.p. alors $cut(P) = P_{>ConsCutDeg(P)}$.*

Ce résultat nous assure qu'en présence d'un programme inconsistant P, la fonction cut nous permet de calculer le plus grand sous-programme consistant de P. Pour expliciter la notion de « plus grand », faisons remarquer que en fixant $P_\alpha = \{r \in P \mid n(r) = \alpha\}$, alors $P = P_{\alpha_1} \cup \cdots \cup P_{\alpha_n}$ avec $\alpha_1 > \cdots > \alpha_n$ et que la fonction cut retourne le sous-ensemble consistant $P_{\alpha_1} \cup \cdots \cup P_{\alpha_k}$ tel que $\forall l > k, P_{\alpha_1} \cup \cdots \cup P_{\alpha_l}$ est inconsistant.
Dans [17] nous montrons que lorsque le p.l.n.p. P est la traduction d'une base logique possibiliste Σ inconsistante, notre restauration de la consistance appliquée à P nécessite un seul appel à la fonction cut et qu'elle donne le même résultat que celui trouvé en logique possibiliste.
Nous terminons la description de notre restauration de consistance par son illustration sur un exemple de problème de satisfaction de contraintes valuées.

Exemple 4.2 *Soit A le problème de 2-coloration (rouge : r ou vert : v) du graphe non orienté $G = (\{s1, s2, s3\}, \{(s1, s2), (s2, s3), (s3, s1)\})$. Sa représentation par un programme logique normal est*

$$P(A) = \left\{ \begin{array}{l} s(1) \leftarrow .\ \ s(2) \leftarrow .\ \ s(3) \leftarrow . \\ a(1,2) \leftarrow .\ \ a(2,3) \leftarrow .\ \ a(3,1) \leftarrow . \\ r(X) \leftarrow s(X), not\ v(X). \\ v(X) \leftarrow s(X), not\ r(X). \\ \leftarrow a(X,Y), r(X), r(Y). \\ \leftarrow a(X,Y), v(X), v(Y). \end{array} \right\}$$

[4] Le m.s.p. complet correspondant retourné par posSmodels est
$\left\{ \begin{array}{l} \text{(s(2), 100), (vu(2), 100), (s(3), 100), (vu(3), 100), (s(4), 100),} \\ \text{(vu(4), 100), (dep(1), 100), (s(1), 100), (in(1,2), 100), (in(2,3),} \\ \text{100), (in(3,4), 100), (a(1,2), 100), (a(1,4), 100), (out(1,4),} \\ \text{100), (a(1,3), 100), (out(1,3), 100), (a(2,3), 100), (a(2,4),} \\ \text{100), (out(2,4), 100), (a(3,4), 100), (a(2,1), 20), (out(2,1), 20),} \\ \text{(a(3,1), 20), (out(3,1), 20), (a(4,1), 20), (out(4,1), 20), (a(3,2),} \\ \text{20), (out(3,2), 20), (a(4,2), 20), (out(4,2), 20), (a(4,3), 20),} \\ \text{(out(4,3), 20), (a\_succ(1), 100), (a\_succ(2), 100), (a\_succ(3),} \\ \text{100), (fin(4), 100)} \end{array} \right\}$

Mais, $P(A)$ est inconsistant puisqu'il est évident qu'il est impossible de colorer G avec seulement 2 couleurs, de telle sorte que 2 sommets adjacents n'aient pas la même couleur. Dans un tel problème, les arêtes du graphe représentent des contraintes. Supposons que ces contraintes puissent être ordonnées selon leur importance par un degré fixé sur chaque arête comme c'est illustré dans le graphe de gauche de la figure 4.

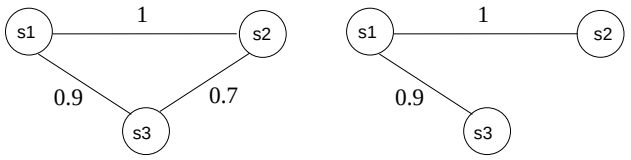

FIG. 4 – Relaxation de contraintes

Le p.l.n.p. correspondant à ce problème où les contraintes sont valuées est alors le suivant :

$$P'(A) = \left\{ \begin{array}{l} (s(1) \leftarrow ., 1), (s(2) \leftarrow ., 1), (s(3) \leftarrow ., 1), \\ (a(1,2) \leftarrow .1), (a(2,3) \leftarrow ., 0.7), \\ (a(3,1) \leftarrow ., 0.9), \\ (r(X) \leftarrow s(X), not\ v(X)., 1), \\ (v(X) \leftarrow s(X), not\ r(X)., 1), \\ (\leftarrow a(X,Y), r(X), r(Y)., 1), \\ (\leftarrow a(X,Y), v(X), v(Y)., 1) \end{array} \right\}$$

Pour ce programme, nous trouvons $InconsDeg(P'(A)) = 0.7$. $cut(P'(A)) = P'(A)_{>0.7}$ est un programme consistant qui représente une relaxation du problème initial dans lequel la contrainte la moins importante a été supprimée comme illustré dans le graphe de droite de la figure 4. Finalement, les modèles stables $\{\ldots, r(1), v(2), v(3), \ldots\}$ et $\{\ldots, v(1), r(2), r(3), \ldots\}$ de $cut(P'(A))^*$ représentent des solutions approchées du problème initial A.

4.3 Travaux liés et perspectives

Il existe d'autres formalismes pour étendre les modèles de raisonnement non monotone avec des degrés qualitatifs ou quantitatifs. Notons tout d'abord que [2] propose d'utiliser la logique possibiliste pour représenter le raisonnement par défaut. Dans notre travail nous gérons simultanément un raisonnement par défaut et incertain, alors que dans [2] les degrés de certitude servent à encoder le raisonnement par défaut. Dans [16], des systèmes non monotones de règles annotées sont introduits afin d'affecter à chaque information de base un poids représentant une probabilité, une mesure d'incertitude, un temps,... Par exemple, pour un programme logique, il mène à des règles de la forme $(c, 0.4) \leftarrow (a, 0.5), (not\ b, 0.7)$. Cette approche est différente de la nôtre dans la mesure où nous mettons des poids sur les règles et non pas sur chaque atome des règles. Une autre divergence est la signification des poids : dans notre travail, nous traitons d'incertitude alors qu'ils développent un formalisme dans lequel la signification des poids n'est pas établie a priori. Dans [5, 6, 8, 3, 14, 1], le lecteur peut trouver différentes propositions à propos de programmes logiques multivalués ou probabilistes, de programmes logiques définis possibilistes, de niveaux de certitude ordonnant les atomes (mais pas les règles), utilisés pour du raisonnement non monotone. Mais aucun de ces travaux ne décrit un formalisme traitant de l'incertitude dans un programme logique avec des négations par défaut dans le cadre de la théorie des possibilités, à la fois des points de vue sémantique et syntaxique.

Un travail très proche du nôtre peut être trouvé dans [21]. Mais il consiste à reconstruire une logique possibiliste en définissant une interprétation multivaluée et une relation de satisfaction correspondante en oubliant la notion de distribution de possibilité. Ainsi, il n'est pas capable de fournir un résultat à propos de la possibilité d'un ensemble d'atomes d'être un modèle stable. De plus, la nécessité (certitude) d'une conclusion n'est pas en relation avec la possibilité de l'ensemble d'atomes. Ceci fait qu'il nous semble que cette approche s'éloigne de l'esprit de la logique possibiliste. De même [15] propose un cadre général étendant les programmes disjonctifs et la sémantique des modèles stables par introduction d'un intervalle de fiabilité pour chaque règle. D'une certaine manière notre approche peut-être considérée comme un cas particulier de celle-ci. Mais le formalisme décrit ne fait pas référence à la théorie des possibilités pour traiter l'incertitude et se base sur des interprétations multivaluées alors que nous restons dans le cadre des ensembles réponses (bivalués). Enfin, dans [4] qui introduit un formalisme logique général traitant de l'incertitude à l'aide de modalités (chaque niveau d'incertitude est représenté par une modalité) on trouve la description d'une logique des défauts graduée. Étant donné que la sémantique des modèles stables est une réduction de la logique des défauts (voir figure 1), cette logique des défauts graduée peut être considérée comme une généralisation de notre formalisme. Mais là encore, l'approche via une distribution de possibilité n'est pas considérée.

En logique possibiliste, les degrés de nécessité sont communément interprétés comme étant des préférences entre les formules : plus une formule est certaine, plus on la préfère. Dans la mesure où il y a de nombreux travaux traitant des préférences entre les règles dans le cadre du raisonnement non monotone [7], il est intéressant d'analyser notre travail si on considère que les degrés de nécessité sur les règles déterminent un ordre de préférence. Si on regarde uniquement dans le secteur de l'ASP, la plupart de ces travaux utilise les préférences exprimées entre les règles pour faire un choix entre les différents modèles stables pour garder seulement ceux qui sont « préférés ». En d'autres termes, les priorités entre les règles n'évaluent pas le degré de certitude des règles mais sont un outil pour choisir entre les règles contradictoires. Ceci diffère de notre travail dans le mesure où, quand plusieurs modèles stables existent, nous calculons la certitude des propositions en fonction de chaque modèle stable. Mais nous n'essayons pas d'éliminer de modèles stables car nous les considérons tous comme des solutions possibles.

Néanmoins, nous envisageons dans le futur une analyse

en profondeur des liens entre notre formalisme et le traitement des préférences en ASP. Par ailleurs, ce travail pourra se poursuivre par son extension aux programmes logiques disjonctifs [12]. Enfin, l'implantation de la fonction cut de restauration de consistance est aussi à l'étude pour compléter notre système lorsque le programme donné est inconsistant.

Références

[1] T. Alsinet and L. Godo. A complete calculus for possibilistic logic programming with fuzzy propositional variables. In C. Boutilier and M. Goldszmidt, editors, *Conference in Uncertainty in Artificial Intelligence*, pages 1–10, Stanford University, Stanford, California, USA, 2000. Morgan Kaufmann.

[2] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *International Conference on the Principles of Knowledge Representation and Reasoning*, pages 673–684, 1992.

[3] S. Benferhat, D. Dubois, and H. Prade. Possibilistic logic : From nonmonotonicity to logic programming. In M. Clarke, R. Kruse, and S. Moral, editors, *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 747 of *LNCS*, pages 17–24. Springer, 1993.

[4] P. Chatalic, C. Froidevaux, and C. Schwind. Graded hypothesis theories. *Theoretical Computer Science*, 171(1-2) :247–280, 1997.

[5] C. V. Damasio and L. M. Pereira. Antitonic logic programs. In T. Eiter, W. Faber, and M. Truszczynski, editors, *International Conference on Logic Programming and NonMonotonic Reasoning*, volume 2173 of *LNCS*, pages 379–392. Springer-Verlag, 2001.

[6] C. V. Damasio and L. M. Pereira. Monotonic and residuated logic programs. In S. Benferhat and P. Besnard, editors, *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 2143 of *LNCS*, pages 748–759. Springer-Verlag, 2001.

[7] J. Delgrande, T. Schaub, H. Tompits, and K. Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2) :308–334, 2004.

[8] D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In K. Furukawa, editor, *International Conference on Logic Programming*, pages 581–595, Cambridge, MA, 1991. MIT Press.

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

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

[11] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *International Conference on Logic Programming*, pages 1070–1080. The MIT Press, 1988.

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

[13] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, to appear in 2005.

[14] Y. Loyer and U. Straccia. Default knowledge in logic programs with uncertainty. In *International Conference on Logic Programming*, volume 2916 of *LNCS*, pages 466–480, Mumbai, India, 2003. Springer-Verlag.

[15] C. Mateis. Extending disjunctive logic programming by t-norms. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *LPNMR*, volume 1730 of *LNCS*, pages 290–304. Springer-Verlag, 1999.

[16] A. Nerode, J. B. Remmel, and V. S. Subrahmanian. Annotated nonmonotonic rule systems. *Theoretical Computer Science*, 171(1-2) :111–146, 1997.

[17] P. Nicolas, L. Garcia, and I. Stéphan. A possibilistic inconsistency handling in answer set programming. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 3571 of *LNCS*, pages 402–414, Barcelona, Spain, 2005. Springer.

[18] P. Nicolas, L. Garcia, and I. Stéphan. Possibilistic stable models. In *International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, August 2-5 2005.

[19] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2) :81–132, 1980.

[20] T. Syrjänen and I. Niemelä. The Smodels systems. In *International Conference on Logic Programming and NonMonotonic Reasoning*, pages 434–438, Vienna, Austria, September 2001. Springer-Verlag.

[21] G. Wagner. A logical reconstruction of fuzzy inference in databases and logic programs. In *Intelligent Fuzzy Set Association World Congress*, Prague, Czech, 1997.

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