État de l'art de l'ASP possibiliste : définitions et calculs

Délivrable - Projet ANR ASPIQ
Rapport de recherche - mai 2014

Résumé

Ce rapport présente un état de l'art des travaux importants concernant le traitement de l'incertitude en ASP. L'introduction de l'incertitude dans un programme ASP se fait en indiquant pour chaque règle du programme ASP le niveau de certitude qui lui est affecté. Ce niveau de certitude est évalué dans le cadre de la théorie des possibilités qui est un formalisme de représentation qualitatif (ou ordinal) de l'incertitude. Notons que les travaux actuels traitent de l'introduction de l'incertitude dans l'ASP propositionnel. Dans ce rapport, nous expliquons tout d'abord l'intérêt de l'introduction du traitement de l'incertitude en ASP puis nous donnons les rappels de la théorie des possibilités nécessaires à la compréhension des modèles. Nous proposons ensuite la définiton des programmes logiques définis possibilistes puis celle des programmes logiques normaux possibilistes avec les différents calculs effectués à partir de ceux-ci. Nous finissons par une présentation d'un prototype implémentant les programmes logiques normaux possibilistes puis par une comparaison succincte avec des travaux connexes portant sur la non-monotonie et l'incertitude.

1 Introduction

Dans le projet ASPIQ, l'introduction de l'incertitude dans l'ASP est motivée par le fait que les informations ontologiques issues de la fusion de sources amènent à des informations qui nécessitent à la fois la prise en charge des exceptions et de l'incertitude. D'une part, les informations ontologiques peuvent être sujettes à exception et c'est le formalisme ASP, au cœur du projet, qui le prend en charge. D'autre part, l'incertitude peut représenter les niveaux de sûreté des informations élémentaires ou, plus simplement, des sources d'information. Les poids reflétant l'incertitude peuvent aussi être vus comme l'expression de priorités qui permettent de faire le choix entre les différentes solutions d'un problème.

L'introduction de l'incertitude dans l'Answer Set Programming est un centre d'intérêt de recherche relativement récent. Si le traitement des exceptions et le raisonnement non-monotone associé de même que le raisonnement sous incertitude sont des thématiques qui ont été étudiées et formalisées depuis de nombreuses années en intelligence artificielle, les travaux qui traitent simultanément des deux problèmes sont très peu nom-

breux et les premiers travaux sont assez récents [22, 25, 29]. Depuis, le traitement de l'incertitude dans l'ASP à base de théorie des possibilités, théorie ordinale de représentation de l'incertitude, a été plus précisément étudié [30, 2, 3, 9, 10] que les autres formalismes. Ce sont ces travaux qui font l'objet de ce rapport.

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. L'une d'elles, celle qui nous importe ici, est la sémantique des modèles stables [21] pour les programmes logiques avec négation par défaut. En ASP, 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. 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 [33] dont la sémantique des modèles stables pour les programmes logiques normaux constitue une réduction.

De son côté, la logique possibiliste [16] est issue de la théorie des possibilités de Zadeh [38] 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 [17] et 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 laquelle on adjoint un moyen de graduer la confiance que l'on a dans les différentes informations énoncées. En résumé, la logique possibiliste consiste en l'introduction du traitement de l'incertitude dans la logique classique.

Dans ce rapport, les travaux présentés s'inscrivent dans le formalisme ASP dans lequel les concepts de la théorie des possibilités ont été introduits comme décrit dans la figure 1. Même s'il existe de nombreux autres formalismes non-monotones, nous nous focalisons sur l'ASP qui est au cœur du projet ASPIQ.

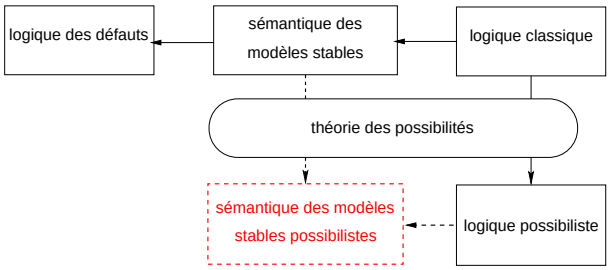

FIGURE 1 – Raisonnement non monotone incertain : l'ASP possibiliste

Pour illustrer cette problématique, prenons un exemple de programme logique qui ne représente pas l'incertitude : il s'agit 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. Il est alors nécessaire de rajouter au formalisme non monotone existant un moyen de représenter la certitude de chaque règle.

2 Rappels sur l'Answer Set Programming

Nous donnons ici des rappels de l'ASP propositionnel nécessaires à la compréhension de l'introduction de l'incertitude dans le formalisme ASP.

Parmi les différentes variantes de l'ASP nous nous intéressons ici aux *programmes logiques normaux*, interprétés par la *sémantique des modèles stables* [21]. 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. [21] 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 lie 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. Soit P un programme logique défini et A un ensemble d'atomes,

$$A \text{ est le modèle de Herbrand minimal de } P$$
$$\Longleftrightarrow$$
$$A = tête(App(P, A)) \text{ et } App(P, A) \text{ est enraciné}$$

3 La théorie des possibilités comme modèle qualitatif de l'incertitude

Dans cette section, nous donnons les principes de la théorie des possibilités et de la logique possibiliste qui vont nous permettre de représenter l'incertitude liée aux informations.

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. 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, habituellement, 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. Dans le cadre du projet ASPIQ, ces valeurs pourront provenir soit, comme toujours, de l'expert et donc chaque source évalue la confiance ou l'importance de ses informations les unes par rapport aux autres, soit de la fiabilité de chacune des sources, soit des deux évaluations à la fois. Notons que les valeurs pourront en outre représenter de l'incertitude ou des priorités entre les informations.

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 [38] ; $N(p) = 1 - \Pi(\neg p) = 1 - max\{\pi(\omega) \mid \omega \not\models p\}$ 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 [16] 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.

De même que, pour la logique possibiliste, on calcule, à partir de formules pondérées, les formules déduites avec leur degré de certitude, pour

l'ASP possibiliste, on calcule, à partir de règles pondérées, les atomes déduits avec leur degré de certitude. Les différents travaux portant sur l'incertitude en ASP proposent de traiter les programmes normaux de manière différentes mais ils concordent à la fois sur le traitement des programmes définis et dans le modèle de représentation des programmes normaux. C'est pourquoi dans la suite de ce rapport, nous présentons tout d'abord les programmes logiques définis possibilistes puis nous présentons le modèle des programmes logiques normaux possibilistes et nous expliquons les différents traitements appliqués à ces programmes. Nous continuons par la présentation d'un solveur en mettant en lumière le fait que le calcul effectif des modèles stables avec possibilités garde la même complexité que le calcul des modèles stables standard. Nous terminons par une comparaison succincte avec des travaux connexes de traitement d'incertitude et de raisonnement non-monotone. Notons enfin que les travaux sur l'ASP possibiliste ont été développés au niveau propositonnel et qu'une extension au premier ordre reste à étudier.

4 Programmes définis possibilistes

Nous présentons tout d'abord les 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.

4.1 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 : un atome a appartient à un ensemble d'atomes si et seulement si la variable correspondante a est vraie dans l'interprétation. 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 2). 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). 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. 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

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

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 3, 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 1. Soit P un p.l.d.p. $\pi_P : 2^{\mathcal{X}} \rightarrow [0, 1]$ la distribution de possibilité définie par

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

est la distribution de possibilité la moins spécifique.

La définition de π_P assure qu'elle est compatible avec P et le résultat suivant assure que seul le modèle de Herbrand minimal de P est totalement possible.

Proposition 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 3. 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 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 4. 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 4. Soit P un p.l.d.p. alors : $(\Pi\mathcal{M}(P))^*$ est le plus petit modèle de P^*.

Exemple 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.

4.2 Théorie du point fixe

Nous venons de définir un modèle au niveau sémantique, en terme de distribution de possibilités. Il est intéressant maintenant de s'intéresser à une approche syntaxique définie sur les règles. Les définitions données dans cette sous-section sont proches de celles données dans [15]. 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 5. Soit $\mathcal{A}$ 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 5. $\langle \mathcal{A}, \sqsubseteq \rangle$ est un treillis complet.

Définition 6. 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, $App(P, A, x) =$
$\{r \in P \mid tête(r^*) = x, r \text{ est } \nu\text{-applicable dans } A, \nu > 0\}$

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 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 [16].

Définition 7. 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

$$\Pi T_P^0 = \emptyset \qquad \text{et} \qquad \Pi T_P^{n+1} = \Pi T_P(\Pi T_P^n), \forall n \geq 0 \quad .$$

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 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. Soit P le p.l.d.p. de l'exemple 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 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 ce cadre tandis que, dans [15], seul un traitement syntaxique est proposé.

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 5.

5 Programmes normaux possibilistes

Nous présentons maintenant les programmes logiques normaux possibilistes c'est-à-dire l'ASP possibiliste proprement dit.

Les programmes logiques sont étendus au raisonnement non monotone en autorisant des négations par défaut dans les programmes. Un programme logique normal possibiliste 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 4).

Il faut alors formaliser la notion de *modèle stable possibiliste* (m.s.p.). Cette notion 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.

5.1 Modèles stables possibilistes selon Nicolas et col. [29, 30]

5.1.1 Définitions

Comme dans le cas classique sans valeurs de nécessité, on doit définir ce qu'est la réduction à la Gelfond-Lifschitz [21] d'un p.l.n.p. avant d'introduire la *sémantique des modèles stables possibilistes*.

Définition 8. 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 9. 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), un p.l.n.p. P est dit *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. 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 modèles stables possibilistes 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 [14] qui a pour but de supprimer certaines alternatives parmi les solutions potentielles.

Proposition 7. 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 a donc permis 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. Nous décrivons cette implémentation dans la section 6.

5.1.2 Distribution de possibilité

Dans la définition 9, la méthode proposée est syntaxique et calculer les modèles stables possibilistes d'un p.l.n.p. utilise un opérateur de point fixe $\Pi C n$ 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.

Définition 10. Soit P un p.l.n.p. alors $\tilde{\pi}_P$ est la distribution de possibilité définie par $\tilde{\pi}_P : 2^{\mathcal{X}} \to [0, 1]$ 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 8. Soit P un p.l.n.p. et $A \in 2^{\mathcal{X}}$, alors
$$\tilde{\pi}_P(A) = 1 \iff A \text{ est un modèle stable de } P^*$$

Exemple 4. 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 11. 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 9. 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 nécessairement 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 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.

Pour finir cette sous-section, notons que les travaux de Confalioneri et col. [9, 10, 32] proposent une extension aux programmes logiques disjonctifs mais leurs définitions se basent sur les travaux de Nicolas et col.

5.2 Modèles stables possibilistes selon Bauters et col. [2, 3, 4]

Les travaux proposés par Kim Bauters et ses collègues [2, 3, 4] utilisent la même représentation des informations sous forme de programmes logiques normaux possibilistes mais ils divergent par l'interprétation qui en est faite. Leur motivation vient du fait que dans les travaux de Nicolas et col., les modèles stables possibilistes sont équivalents aux modèles stables

standard et que le niveau d'incertitude n'est pas pris en compte pour bloquer une règle. En clair, dans les travaux de Nicolas et col., lorsqu'une information est présente même avec une très faible certitude, elle bloque les règles pour lesquelles elle apparaît dans le *not*. Ce choix est discutable, comme on peut le voir dans l'exemple suivant :

$$\left\{ \begin{array}{l} (reservation\_concert \leftarrow ., 1), \\ (long\_trajet \leftarrow reservation\_concert, not\ annulation., 1), \\ (annulation \leftarrow ., 0.2) \end{array} \right\}$$

Cet exemple signifie qu'une réservation a été effectuée pour un concert et la réservation entraîne un long trajet (pour rejoindre la salle de spectacle) sauf si le concert est annulé. Ces deux informations sont totalement sûres et on applique aux deux règles correspondantes un degré de nécessité de 1. Une troisième information indique qu'il y a un faible risque d'annulation. Pour exprimer cette incertitude, la règle est valuée avec 0.2.

Dans ce cas, si on considère la calcul effectué comme dans la section 5.1, on obtient le modèle stable $\{(reservation\_concert, 1), (annulation, 0.2)\}$. Dans les travaux de Bauters et col., l'idée est de prendre en compte l'incertitude de l'atome *annulation* pour obtenir une valeur non nulle pour $long\_trajet$ et calculer le modèle stable possibiliste suivant :

$\{(reservation\_concert, 1), (long\_trajet, 0, 8), (annulation, 0.2)\}$.

Pour ce faire, les règles logiques possibilistes sont traduites par des contraintes sur les nécessités des atomes impliqués dans les règles et un modèle stable est calculé à partir de cet ensemble de contraintes.

5.2.1 Principe et définitions de base

Dans un programme, chaque règle est assimilée à une contrainte. On donne ici les définitions des contraintes ainsi que des modèles stables tout d'abord avec le principe du calcul sur de l'ASP standard.

On commence par s'intéresser à la définition pour les programmes définis. Intuitivement, dans une règle définie, la certitude de la conclusion est au moins aussi importante que la certitude des atomes de son corps.

Définition 12. Soit P un programme défini et $\pi : \mathcal{X} \rightarrow [0, 1]$ une distribution de possibilités. Pour chaque $r \in P$, la contrainte $\gamma(r)$ imposée par la règle $r = (a_0 \leftarrow a_1, \ldots, a_m.)$ est donnée par :

$N(a_0) \geq min(N(a_1), \ldots, N(a_m))$.

Cette définition permet de déterminer le modèle d'un programme. Le modèle correspond à la distribution de possibilités la moins spécifique vérifiant l'ensemble des contraintes.

Notons qu'ici, la définition est donnée en terme de possibilité mais les règles définies traitées sont des règles standard. Nous verrons juste après le traitement de règles logiques possibilistes. Avant, il est important de définir les contraintes sur les règles normales.

L'idée intuitive est d'évaluer la certitude qu'un atome soit prouvé. Intuitivement, *not* a est vrai lorsqu'il n'y a pas de preuves de l'atome a. De plus, la preuve de a peut dépendre d'autres négations par défaut. Lors de la recherche de modèles stables traditionnelle, on devine un ensemble d'atomes et on teste si cet ensemble d'atomes est une solution. Avec une interprétation en terme de possibilités, la détermination nécessite un oracle pour deviner la certitude de ne pas trouver une preuve de l'atome a (alors

que dans le cas standard, l'oracle choisit si un atome est prouvé ou non). On a ainsi besoin d'une fonction de *guess* g sur les atomes sachant que $g(a)$ reflète la certitude que « il est consistant de supposer que a n'est pas vrai ».

Définition 13. Soit P un programme normal et $g : \mathcal{X} \rightarrow [0,1]$. Pour chaque $r \in P$, la contrainte $\gamma(r)$ imposée par la règle
$r = (a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n.)$ est donnée par :
$N(a_0) \geq min(N(a_1), \ldots, N(a_m), g(a_{m+1}), \ldots, g(a_n))$.

Pour éclairer cette définition, nous donnons l'exemple suivant :

Exemple 5. Soit le programme P $= \{a \leftarrow ., b \leftarrow b., c \leftarrow a, not\ b.\}$.
L'ensemble des contraintes impliquées par le programme est :
$N(a) \geq N(\top) = 1$
$N(b) \geq N(b)$
$N(c) \geq min(N(a), g(b))$.
En utilisant les définitions des possibilités et de la dualité entre possibilités et nécessités, on obtient les contraintes suivantes sur les possibilités :
$\Pi(\neg a) = 0$
$\Pi(\neg c) \leq 1 - g(b)$.
On en déduit la distribution de possibilités suivante :

$\pi(\{a,b,c\}) = 1$	$\pi(\{b,c\}) = 0$
$\pi(\{a,b\}) = 1 - g(b)$	$\pi(\{b\}) = 0$
$\pi(\{a,c\}) = 1$	$\pi(\{c\}) = 0$
$\pi(\{a\}) = 1 - g(b)$	$\pi(\emptyset) = 0$

On remarque que la distribution de possibilités ne dépend ni de $g(a)$ ni de $g(c)$, ce qui est normal puisqu'il n'y a pas de présence de *not* a ou *not* c dans les règles. Dans la mesure où on évalue la possibilité que b ne soit pas déduit, on a $g(b) = \Pi(\neg b) = max\{\pi(I) | I \models \neg b\} = 1$ et donc on obtient $g(b) = 1$.

On a donc $N(a) = 1$, $N(b) = 0$ et $N(c) = 1$, ce qui correspond au modèle stable $\{a, c\}$.

Un résultat important est que, en posant que, pour tout atome a, $g(a) = \Pi(\neg a)$ et que les valeurs de nécessité sont égales à 0 ou 1, l'ensemble $M = \{a | N(a) = 1, \forall a \in \mathcal{X}\}$ est un modèle stable de P.

5.2.2 Définitions pour les programmes normaux possibilistes

Cette sémantique des programmes logiques normaux peut maintenant être appliquée aux programmes logiques normaux possibilistes pour lesquels il faut en outre prendre en compte le degré de nécessité associé à la règle.

Définition 14. Soit P un programme logique normal et $g : \mathcal{X} \rightarrow [0,1]$. Pour chaque $r \in P$, la contrainte $\gamma(r)$ imposée par la règle
$r = (a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n., n(r))$, où $n(r)$ est la nécessité de la règle, est donnée par :
$N(a_0) \geq min(N(a_1), \ldots, N(a_m), g(a_{m+1}), \ldots, g(a_n), n(r))$.

Définition 15. Soit P un programme logique normal possibiliste et $g : \mathcal{X} \rightarrow [0,1]$. Si π est la distribution de possibilités la moins spécifique définie

sur P et g telle que $\forall a \in \mathcal{X}, g(a) = \pi(\neg a)$ alors $V = \{(a, N(a)) | a \in \mathcal{X}\}$ est un modèle stable de P.

Exemple 6. Reprenons l'exemple que nous avions évoqué pour introduire les travaux de Bauters et col. :

$$\left\{ \begin{array}{l} (reservation\_concert \leftarrow ., 1), \\ (long\_trajet \leftarrow reservation\_concert, not\ annulation., 1), \\ (annulation \leftarrow ., 0.2) \end{array} \right\}$$

On obtient les contraintes sur les nécessités suivantes :

$N(reservation\_concert) \geq min(N(\top), 1)$

$N(long\_trajet) \geq min(N(reservation\_concert), g(annulation), 1)$

$N(annulation) \geq min(N(\top), 0.2)$

Dans la mesure où $N(\top) = 1$, on peut réécrire la première contrainte par $N(reservation\_concert) = 1$. Comme $N(reservation\_concert) = 1$, la deuxième contrainte se simplifie en $N(long\_trajet) \geq g(annulation)$ puis se réécrit en $\Pi(\neg long\_trajet) \leq 1 - g(annulation)$. Comme $N(\top) = 1$, la dernière contrainte peut être simplifiée en $N(annulation) \geq 0.2$ soit $\Pi(\neg annulation) \leq 0.2$. On obtient donc les contraintes suivantes sur les possibilités :

$\Pi(\neg reservation\_concert) = 0$

$\Pi(\neg long\_trajet) \leq 1 - g(annulation)$

$\Pi(\neg annulation) \leq 0.8$

La distribution de possibilités la moins spécifique associée à cet ensemble de contraintes est :

$\pi(\{reservation\_concert, long\_trajet, annulation\}) = 1$

$\pi(\{reservation\_concert, long\_trajet\}) = 0.8$

$\pi(\{reservation\_concert, annulation\}) = 1 - g(annulation)$

$\pi(\{reservation\_concert\}) = min(0.8, 1 - g(annulation))$

$\pi(\{long\_trajet, annulation\}) = 0$

$\pi(\{long\_trajet\}) = 0$

$\pi(\{annulation\}) = 0$

$\pi(\emptyset) = 0$

Si on exprime maintenant $g(annulation)$ tel que $g(annulation) = \Pi(\neg annulation)$, on se retrouve avec $g(annulation) = 0.8$ car

$\Pi(\neg annulation) = max(\pi(A) | annulation \notin A)$

$= \pi(\{reservation\_concert, long\_trajet\}) = 0.8.$

On a alors $\pi(\{reservation\_concert, annulation\}) = 0.2$ et

$\pi(\{reservation\_concert\}) = 0.2.$

On en déduit $N(reservation\_concert) = 1$, $N(long\_trajet) = 0.8$ et $N(annulation) = 0.2$. Le seul modèle stable du programme est donc $\{(reservation\_concert, 1), (long\_trajet, 0.8), (annulation, 0.2)\}$.

Les définitions précédentes décrivent les modèles stables possibilistes au niveau sémantique. Comme d'habitude, il est intéressant de définir aussi les modèles stables au niveau syntaxique c'est-à-dire en terme de réduit du programme.

Définition 16. Soit E un ensemble d'atomes possibilistes. Le réduit P^E d'un programme normal possibiliste P est défini par :

$$P^V = \{(a_0 \leftarrow a_1, \ldots, a_m., min(c_1, c_2))| \\ min(c_1, c_2) > 0, \\ (a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n., c_1) \in P, \\ c_2 = max(c|\{a_{m+1}, \ldots, a_n\} \cap E^{1-c} = \emptyset, c \in [0, 1])\}$$

Exemple 7. À partir de l'exemple du programme

$$\left\{ \begin{array}{l} (reservation\_concert \leftarrow ., 1), \\ (long\_trajet \leftarrow reservation\_concert, not\ annulation., 1), \\ (annulation \leftarrow ., 0.2) \end{array} \right\},$$

si on considère l'ensemble d'atomes

$E = \{(reservation\_concert, 1), (long\_trajet, 0.8), (annulation, 0.2)\}$, on obtient le réduit du programme par E suivant :

$$\left\{ \begin{array}{l} (reservation\_concert \leftarrow ., 1), \\ (long\_trajet \leftarrow reservation\_concert., 0.8), \\ (annulation \leftarrow ., 0.2) \end{array} \right\}.$$

Un ensemble d'atomes possibilistes E est un modèle stable d'un programme normal possibilistes P si et seulement si $E = \Pi C_n(P^E)$.

Exemple 8. En reprenant le même programme, on montre que l'ensemble d'atomes possibilistes

$E = \{(reservation\_concert, 1), (long\_trajet, 0.8), (annulation, 0.2)\}$ est un modèle stable car, étant donné le réduit du programme par E que l'on vient de voir, on vérifie facilement que $E = \Pi C_n(P^E)$.

D'autre part, si on prend l'ensemble d'atomes possibilistes

$E' = \{(reservation\_concert, 1), (annulation, 0.2)\}$, on vérifie que ce n'est pas un modèle stable possibiliste car on obtient le même réduit $P^E = P^{E'}$ mais on a $E' \neq \Pi C_n(P^{E'})$.

Notons enfin que, dans le cas où les poids affectés aux règles d'un programme sont tous égaux à 1 c'est-à-dire qu'on se place dans un cadre équivalent à l'ASP standard, les mêmes résultats sont obtenus quelle que soit la méthode utilisée pour le calcul des modèles stables possibilistes et on retrouve les résultats désirés pour l'ASP standard.

6 Implémentation

Cette section présente le solveur posSmodels permettant de mettre en œuvre le calcul des modèles stables des programmes logiques normaux possibilistes proposé par Nicolas et col.

Comme nous l'avons énoncé à la fin de la sous-section 5.1.1 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. Parmi, les plus efficaces, on peut citer :

Smodels(`www.tcs.hut.fi/Software/smodels`[36]),
DLV(`www.dbai.tuwien.ac.at/proj/dlv`[23]) ou
clasp(`potassco.sourceforge.net`[20]).

L'implémentation des travaux de Nicolas et col. se base 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 [2]. Partant de ces considérations, le système posSmodels [3] a été développé en C++ [31, 30]. Son fonctionnement global est synthétisé dans la figure 2.

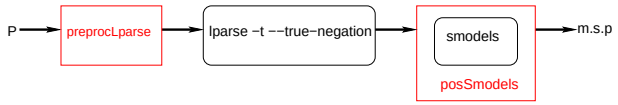

FIGURE 2 – Chaîne de traitement du calcul de modèles stables possibilistes

Les règles acceptées par le système sont de la forme
$\alpha \quad c$:- $a_1, \ldots, a_m, \ not\ b_1, \ldots, \ not\ b_n$. où α est un entier et où les atomes peuvent contenir des variables.

Exemple 9. 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 ou 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 le système posSmodels pour représenter ce problème [4]. 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.

2. Nous ne revenons pas ici sur les problèmes liés à la phase d'instanciation qui ont été expliqués dans un autre rapport.

3. Téléchargeable à partir de `www.info.univ-angers.fr/pub/garcia/possmodels.html`

4. Des règles sans tête sont parfois utilisées (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 : $c \leftarrow a, not\ b, not\ c.$).

```
% les sommets et le sommet de départ
100   s(1..4).
100   dep(1).
% 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.
% 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 est 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 $preproc(r) = r'$ tel que :

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

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 9

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

est 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

$$\begin{array}{ll} a(2,1) \text{ :- } s(2), s(1), nu\_\_(20). & a(3,1) \text{ :- } s(3), s(1), nu\_\_(20). \\ a(4,1) \text{ :- } s(4), s(1), nu\_\_(20). & a(3,2) \text{ :- } s(3), s(2), nu\_\_(20). \\ a(4,2) \text{ :- } s(4), s(2), nu\_\_(20). & a(4,3) \text{ :- } s(4), s(3), nu\_\_(20). \end{array}$$

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, un p.n.l.p. totalement instancié est fabriqué à partir d'un programme avec variables en conservant les degrés adéquats.

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.

calculTousMSP(données Solv : solveur ASP, P : p.l.n.p)
début
  tant que Solv(P^*) retourne un m.s. S
    $PP \leftarrow \{({r^*}^+, n(r)) \mid r \in P, corps^+(r^*) \subseteq S$
                $et\ corps^-(r^*) \cap S = \emptyset\}$
    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 $R \in PP$
        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
          $Res \leftarrow Res \sqcup \{(tête(R), deg)\}$
          siRes a été modifié alors $PointFixe \leftarrow faux$
          si $n(R) = deg$ alors retirer R de PP
      finpour
    jusqu'à $PointFixe$
    écrire Res
  fintq
fin

FIGURE 3 – Calcul des modèles stables possibilistes

Le programme de l'exemple 9 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 le degré maximal.

Notons que les résultats expérimentaux avec ce système confirment les

résultats théoriques (cf. fin de la sous-section 5.1.1) 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.

Notons enfin que les travaux de Confalonieri et col. [9, 10, 32] ont aussi donné lieu à une implémentation [5].

7 Autres travaux

D'autres travaux ne traitent pas directement l'ASP mais étendent également des modèles de raisonnement non-monotone à l'aide de degrés qualitatifs ou quantitatifs. Sans faire une comparaison détaillée, inutile pour le travail dans le projet ASPIQ, nous donnons quelques points de comparaisons.

Notons tout d'abord que [5] propose d'utiliser la logique possibiliste pour représenter le raisonnement par défaut. Dans l'ASP possibiliste, il y une gestion simultanée d'un raisonnement par défaut et incertain alors que dans [5] les degrés de certitude sont une manière d'encoder le raisonnement par défaut. Dans [13, 18], seule la théorie des possibilités est utilisée pour représenter à la fois l'incertitude et la non monotonie alors que l'ASP possibiliste intègre deux formalismes différents.

Dans [27], 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 l'ASP possibiliste dans la mesure où les poids sont mis sur les règles et non pas sur chaque atome des règles. Une autre divergence est la signification des poids : dans l'ASP possibiliste, on traite d'incertitude alors que [27] développe un formalisme dans lequel la signification des poids n'est pas établie a priori. Dans [12, 11, 15, 6, 24, 1, 34, 35], 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 et donnant lieu à une mise en œuvre par le biais d'une implémentation. Notons néanmoins qu'une implémentation de programmes logiques utilisant des probabilités a été proposée [34] mais elle est restreinte aux programmes définis sans négation par défaut.

Un travail très proche de l'ASP possibiliste peut être trouvé dans [37]. 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

5. Téléchargeable à partir de https ://github.com/rconfalonieri/posPsmodels.

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 [26] 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, l'ASP possibiliste peut-être considéré comme un cas particulier de cette approche. 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 l'ASP possibiliste reste dans le cadre des ensembles réponses (bivalués). Enfin, dans [8] 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 l'ASP possibiliste. Mais là encore, l'approche via une distribution de possibilité n'est pas considérée. De plus, dans toutes ces approches, l'aspect implémentation a été négligé.

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 [14], il est intéressant d'analyser l'ASP possibiliste 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 »[7]. D'autre part, certains travaux utilisent les poids entre règles contradictoires pour restaurer la cohérence du programme en supprimant les règles les moins prioritaires [28]. En d'autres termes, dans ces travaux, 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 l'ASP possibiliste dans la mesure où, quand plusieurs modèles stables existent, la certitude des propositions est calculée en fonction de chaque modèle stable. Mais il n'y a pas d'élimination de modèles stables car ils sont tous considérés comme des solutions possibles.

8 Conclusion

Dans ce rapport, nous avons présenté l'introduction de l'incertitude dans le formalisme ASP. Nous avons introduit les définitions, basées sur la théorie des possibilités qui est une théorie ordinale de traitement de l'incertitude, permettant de présenter le cadre des programmes logiques avec incertitude. Nous avons décrit les principaux travaux permettant le calcul des modèles stables pour les programmes logiques normaux possibilistes ainsi que l'implémentation de la mise en œuvre d'un de ces calculs. Nous

avons terminé avec un comparaison succincte avec des travaux connexes concernant le traitement de la non-monotonie et de l'incertitude hors ASP.

Dans le cadre du projet ASPIQ, l'ASP possibiliste est intéressant à divers niveaux. D'une part, lors du traitement de bases exprimées en DL-Lite possibiliste, la traduction obtenue en ASP sera naturellement de l'ASP possibiliste pour lequel il faudra étudier l'introduction de requêtes d'interrogation. D'autre part, lors du processus de fusion, l'incertitude liée aux informations ou aux sources impliquées dans la fusion amènera à utiliser l'ASP possibiliste pour représenter cette incertitude.

Notons enfin que , sachant qu'il n'existe pas pour l'instant de travaux sur l'ASP possibiliste au premier ordre, il sera intéressant de définir un cadre de programmes logiques possibilistes au premier ordre et de proposer son implémentation sur le solveur `ASPeRiX`.

Références

[1] Teresa Alsinet and Lluis Godo. A complete calcultis for possibilistic logic programming with fuzzy propositional variables. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI*, pages 1–10. Morgan Kaufmann, 2000.

[2] Kim Bauters, Steven Schockaert, Martine De Cock, and Dirk Vermeir. Possibilistic answer set programming revisited. In Peter Grünwald and Peter Spirtes, editors, *UAI*, pages 48–55. AUAI Press, 2010.

[3] Kim Bauters, Steven Schockaert, Martine De Cock, and Dirk Vermeir. Possible and necessary answer sets of possibilistic answer set programs. In *ICTAI*, pages 836–843. IEEE, 2012.

[4] Kim Bauters, Steven Schockaert, Martine De Cock, and Dirk Vermeir. Characterizing and extending answer set semantics using possibility theory. *CoRR*, abs/1312.0127, 2013.

[5] Salem Benferhat, Didier Dubois, and Henri Prade. Representing default rules in possibilistic logic. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *KR*, pages 673–684. Morgan Kaufmann, 1992.

[6] Salem Benferhat, Didier Dubois, and Henri Prade. Possibilistic logic : From nonmonotonicity to logic programming. In Michael Clarke, Rudolf Kruse, and Serafín Moral, editors, *ECSQARU*, volume 747 of *Lecture Notes in Computer Science*, pages 17–24. Springer, 1993.

[7] Gerhard Brewka. Complex preferences for answer set optimization. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR*, pages 213–223. AAAI Press, 2004.

[8] Philippe Chatalic, Christine Froidevaux, and Camilla Schwind. Graded hypothesis theories. *Theor. Comput. Sci.*, 171(1-2) :247–280, 1997.

[9] Roberto Confalonieri, Juan Carlos Nieves, Mauricio Osorio, and Javier Vázquez-Salceda. Possibilistic semantics for logic programs with ordered disjunction. In Sebastian Link and Henri Prade, editors, *FoIKS*, volume 5956 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2010.

[10] Roberto Confalonieri, Juan Carlos Nieves, Mauricio Osorio, and Javier Vázquez-Salceda. Dealing with explicit preferences and uncertainty in answer set programming. *Ann. Math. Artif. Intell.*, 65(2-3) :159–198, 2012.

[11] Carlos Viegas Damásio and Luís Moniz Pereira. Antitonic logic programs. In Eiter et al. [19], pages 379–392.

[12] Carlos Viegas Damásio and Luís Moniz Pereira. Monotonic and residuated logic programs. In Salem Benferhat and Philippe Besnard, editors, *ECSQARU*, volume 2143 of *Lecture Notes in Computer Science*, pages 748–759. Springer, 2001.

[13] Florence Dupin de Saint-Cyr and Henri Prade. Possibilistic handling of uncertain default rules with applications to persistence modeling and fuzzy default reasoning. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR*, pages 440–451. AAAI Press, 2006.

[14] James P. Delgrande, Torsten Schaub, Hans Tompits, and Kewen Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2) :308–334, 2004.

[15] Didier Dubois, Jérôme Lang, and Henri Prade. Towards possibilistic logic programming. In Koichi Furukawa, editor, *ICLP*, pages 581–595. MIT Press, 1991.

[16] Didier Dubois, Jérôme Lang, and Henri Prade. Possibilistic logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming-Nonmonotonic Reasoning and Uncertain Reasoning(Volume 3)*, pages 439–513. Clarendon Press, Oxford, 1994.

[17] Didier Dubois and Henri Prade. Possibility theory :qualitative and quantitative aspects. In Dov M. Gabbay and Philippe Smets, editors, *Quantified Representation of Uncertainty and Imprecision*, volume 1 of *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 169–226. Kluwer Academic Publishers, http ://www.wkap.nl/, 1998. BB.

[18] Florence Dupin de Saint-Cyr and Henri Prade. Handling uncertainty and defeasibility in a possibilistic logic setting. *Int. J. Approx. Reasoning*, 49(1) :67–82, 2008.

[19] Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczynski, editors. *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, September 17-19, 2001, Proceedings*, volume 2173 of *Lecture Notes in Computer Science*. Springer, 2001.

[20] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving : From theory to practice. *Artificial Intelligence*, 187-188 :52–89, 2012.

[21] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.

[22] Moisés Goldszmidt and Judea Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artif. Intell.*, 84(1-2) :57–112, 1996.

[23] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3) :499–562, 2006.

[24] Yann Loyer and Umberto Straccia. Default knowledge in logic programs with uncertainty. In Catuscia Palamidessi, editor, *ICLP*, volume 2916 of *Lecture Notes in Computer Science*, pages 466–480. Springer, 2003.

[25] Thomas Lukasiewicz. Weak nonmonotonic probabilistic logics. *Artif. Intell.*, 168(1-2) :119–161, 2005.

[26] Cristinel Mateis. Extending disjunctive logic programming by t-norms. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *LPNMR*, volume 1730 of *Lecture Notes in Computer Science*, pages 290–304. Springer, 1999.

[27] Anil Nerode, Jeffrey B. Remmel, and V. S. Subrahmanian. Annotated nonmonotonic rule systems. *Theor. Comput. Sci.*, 171(1-2) :111–146, 1997.

[28] Pascal Nicolas, Laurent Garcia, and Igor Stéphan. A possibilistic inconsistency handling in answer set programming. In Lluis Godo, editor, *ECSQARU*, volume 3571 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 2005.

[29] Pascal Nicolas, Laurent Garcia, and Igor Stéphan. Possibilistic stable models. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 248–253. Professional Book Center, 2005.

[30] Pascal Nicolas, Laurent Garcia, Igor Stéphan, and Claire Lefèvre. Possibilistic uncertainty handling for answer set programming. *Ann. Math. Artif. Intell.*, 47(1-2) :139–181, 2006.

[31] Pascal Nicolas and Claire Lefèvre. Possibilistic stable model computing. In Marina De Vos and Alessandro Provetti, editors, *Answer Set Programming*, volume 142 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[32] Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés. Semantics for possibilistic disjunctive programs. *TPLP*, 13(1) :33–70, 2013.

[33] Raymond Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2) :81–132, 1980.

[34] Fabrizio Riguzzi and Terrance Swift. The pita system : Tabling and answer subsumption for reasoning under uncertainty. *TPLP*, 11(4-5) :433–449, 2011.

[35] Fabrizio Riguzzi and Terrance Swift. Well-definedness and efficient inference for probabilistic logic programming under the distribution semantics. *TPLP*, 13(2) :279–302, 2013.

[36] Tommi Syrjänen and Ilkka Niemelä. The smodels system. In Eiter et al. [19], pages 434–438.

[37] Gerd Wagner. A logical reconstruction of fuzzy inference in databases and logic programs. In *In Proc. of 7th Int. Fuzzy Systems Association World Congress (IFSA'97)*, 1997.

[38] Lotfi A. Zadeh. Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1 :3–28, 1978.