Henri Prade

L'IA symbolique et le dépassement de la logique classique

Volume 5, nº 2-3 (2024), p. 161-176.

https://doi.org/10.5802/roia.77

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 5, nº 2-3, 2024, 161-176

L'IA symbolique et le dépassement de la logique classique

Henri Prade[a]

[a] IRIT, CNRS, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex 9 (France)
E-mail : henri.prade@irit.fr.

Résumé. — La programmation logique et la représentation des connaissances ont constitué deux courants de recherche importants en intelligence artificielle qui se sont développés dans les 50 dernières années avec des préoccupations largement différentes, mais avec cependant des points de rencontre, en particulier sur le raisonnement non-monotone, ou sur des logiques multi-valuées. C'est ce que ce modeste article se propose de revisiter, principalement autour de liens et de complémentarités avec la logique floue et la logique possibiliste, dans une perspective plus historique que technique.

Mots-clés. — Programmation logique, ASP, représentation des connaissances, raisonnement non-monotone, conditionnelle, règle si - alors, règles à seuil, logiques tri-valuées, logique floue, logique possibiliste, contrainte flexible, histoire de l'IA.

1. Introduction[(1)]

L'IA, en y incluant le traitement du langage naturel, a été dominée jusqu'au début des années 1990 par la logique symbolique, puis a vu l'avènement des réseaux bayésiens et des méthodes probabilistes, qui au milieu des années 2010 ont été supplantées par le versant neuronal de l'IA numérique dans un nombre croissant de tâches.

L'IA dite « symbolique » a été marquée par deux courants de recherche, la programmation logique, avant tout soucieuse d'implémentations efficaces, et la représentation des connaissances plus tournée vers la richesse (voire la sophistication) des représentations permises. Rappelons aussi qu'une des motivations initiales du développement du langage PROLOG [23], premier langage de programmation logique (basé sur la logique du premier ordre, initialement restreinte aux clauses de Horn, où les implications ont une interprétation à la fois déclarative et procédurale [55, 56]), a été le traitement du langage naturel [19, 20]. Les rapports de PROLOG avec la linguistique font d'ailleurs l'objet d'un article dans ce numéro [31].

La période 1970-2000 en représentation des connaissances a vu notamment le développement de différents travaux sur les logiques non classiques, à commencer par

[(1)] Cet article reprend la matière d'une présentation lors de la Journée d'hommage scientifique à Alain Colmerauer, le 8 octobre 2021 à Marseille.

ceux touchant au raisonnement non-monotone [14]. En effet, l'expérience des systèmes experts dans les années 1970-1980 a mis en évidence assez tôt un certain nombre de limitations de la logique classique. La connaissance est souvent exprimée par des règles si - alors ; ces règles sont orientées à la différence des formules logiques. On distingue la base de faits de la base de connaissance. De plus la connaissance est souvent incertaine, et les règles peuvent avoir des exceptions connues. Indépendamment, la logique floue [88] montre l'intérêt des règles floues mettant en jeu des propriétés graduelles pour décrire implicitement des procédures d'interpolation, ce qui sera appliqué à la commande de systèmes dynamiques dans les années 1980-1990. Par ailleurs, à la fin des années 1990, émerge un nouveau paradigme de programmation logique, l'answer set programming (ASP)[(2)] [62, 64, 77], une forme de programmation déclarative où à côté de la négation logique, une négation par l'échec permet le raisonnement non-monotone.

L'article passe en revue quelques points de rencontre entre programmation logique et représentation des connaissances, à commencer par les logiques multi-valuées et le raisonnement non-monotone, avant d'en mentionner quelques autres. La présentation parcourt un vaste champ de recherche et fait référence à de nombreux travaux, auxquels les lecteurs pourront se reporter pour les détails techniques.

2. Présupposition et objet conditionnel

Comme on vient de le rappeler dans l'introduction le traitement du langage naturel, dans les années 1970, est basé sur la logique. Ainsi Alain Colmerauer et ses collègues travaillent sur la compréhension du langage naturel en utilisant la logique pour représenter la sémantique du langage, et en appliquant la résolution pour répondre à des questions [21].

Logique de la présupposition. Comme il l'écrira un peu plus tard (dans un article avec J.-F. Pique [22])

> « L'utilisation de la logique du premier ordre dans les bases de données pose des problèmes de représentation des relations qui ne sont pas définies partout. La solution à ce problème est d'utiliser une logique à 3 valeurs avec la valeur de vérité "indéfini" »[(3)].

L'idée semble provenir de la proposition en linguistique [53][(4)] d'utiliser une logique tri-valuée pour traiter les problèmes de présupposition.[(5)] Une proposition telle que « Le roi de France est chauve », pour reprendre un exemple classique, n'est ni vraie, ni fausse, s'il n'existe pas de roi de France. Dans ce dernier cas, il est commode

[(2)] Dans la suite on continuera à utiliser le sigle anglais ASP plutôt que « programmation par ensembles-réponses ».

[(3)] Texte original [22] : « The use of first-order logic in data bases raises problems about representing relations that are not defined everywhere. The solution to this problem is to use a 3-valued logic with the truth value « undefined ». »

[(4)] Référence citée par Dahl [26] et Colmerauer [20].

[(5)] Cette idée est déjà mentionnée dans [80], mais sans qu'une logique correspondante soit présentée.

de considérer que son degré de vérité est 'indéfini'. La vérité ou la fausseté de la proposition *présuppose* l'existence du roi de France.

Pour propager la valeur de vérité 'indéfini' dans des calculs, il est nécessaire de formaliser une logique. Verónica Dahl a été la première à le faire dans sa thèse[(6)] [25][(7)], où cette troisième valeur est exploitée dans une logique typée capable d'accommoder les présuppositions, de traiter certaines ambigüités du langage naturel, et aussi de contrôler l'application de la négation. Cette logique, ses justifications, sa sémantique, son application au traitement des langues représentées en logique du premier ordre, se trouvent présentées, avec plus ou moins de de détails, dans une série d'articles de Dahl [26, 27, 29, 30], ainsi que dans [20, 22].

Nous ne rappelons ici que la sémantique de la négation, de la conjonction et de l'implication. Soient e_1, e_2, des énoncés (formules fermées). Pour la négation $e = non(e_1)$, e est faux si e_1 est vrai, vrai si e_1 est faux, indéfini si e_1 est indéfini. La valeur de la conjonction $e = et(e_1, e_2)$ est donnée par

$$val(e) = \min(val(e_1), val(e_2)) \text{ où vrai} > \text{faux} > \text{indéfini}$$

(en accord avec la logique ternaire de Kleene).

La valeur de l'implication « si(·, ·) » est donnée par la Fig. 2.1 :

```
3) si e = si(e1,e2) alors
          si val(e1) = vrai alors val(e) = val(e2)
          si val(e1) ≠ vrai alors val(e) = indéfini
```

FIGURE 2.1. Verónica Dahl's implication in [25] p. 37

si(e1, e2) est donc indéfini dès que son antécédent e1 n'est pas vrai.

Objets conditionnels. La table de vérité de l'implication tri-valuée de la logique de la présupposition de Dahl et Colmerauer a une longue histoire. Elle a été introduite en 1935 par Bruno De Finetti [33], pour préciser le sens du conditionnement en probabilités. Ainsi un évènement conditionnel 'b si a' (on parle aussi d'objet conditionnel, noté $b \mid a$), est une entité tri-valuée qui est vraie si $a \wedge b$ est vrai, faux si $a \wedge \neg b$ est vrai, et *inapplicable* si a est faux. La coïncidence entre les tables de vérité de l'objet conditionnel et de l'implication du calcul des présuppositions n'a rétrospectivement rien de fortuit. En effet, on peut considérer qu'une présupposition inclut une

[(6)] Cette thèse a été préparée à l'Université d'Aix-Marseille sous la supervision d'Alain Colmerauer. Verónica Dahl avait auparavant préparé son DEA, dans le cadre du Groupe d'Intelligence Artificielle de l'U.E.R. Scientifique de Luminy, sur la réalisation d'une banque de données en logique du premier ordre, consultable en français, conjointement avec Roland Sambuc [32]. Ce dernier avait soutenu l'année précédente une thèse d'exercice [82], où il introduisait pour l'aide au diagnostic médical [83], ce qu'il appelait des ensembles Φ-flous, un cas particulier important d'ensemble flou de type 2 [87], où l'incertitude sur le degré de vérité est représenté par un intervalle. On revient plus loin dans cet article sur les points de rencontre entre programmation logique et logique floue.

[(7)] Dans une section intitulée : « Système logique sous-jacent au sous-ensemble de langue naturelle choisi », pages 35 à 38.

conditionnelle implicite : l'énoncé « Le roi de France est chauve » est vrai (ou faux) conditionnellement à l'existence du roi.

L'objet conditionnel $b \mid a$ doit être pensé comme une règle $b \leftarrow a$. Une règle peut avoir des exceptions. C'est-à-dire, qu'on peut avoir en même temps une règle $\neg b \leftarrow (a \wedge c)$. Les deux objets conditionnels $b \mid a$ et $\neg b \mid a \wedge c$ ne conduisent pas à une contradiction en présence des faits a et c (à la différence d'une modélisation des règles par l'implication matérielle)[8], dans le cadre d'une logique tri-valuée différente de celle des présuppositions. La conjonction d'objets conditionnels a maintenant pour sémantique

$$val(e) = \min(val(e_1), val(e_2)) \text{ où inapplicable} > \text{vrai} > \text{faux}.^{(9)}$$

On montre [39] que cela correspond à une quasi-conjonction (associative) &, définie par

$$b \mid a \;\&\; d \mid c = (a \rightarrow b) \wedge (c \rightarrow d) \mid (a \vee c).$$

Cette conjonction s'interprète comme suit : L'ensemble constitué par les deux règles $b \leftarrow a$ et $d \leftarrow c$ est déclenchable si a ou c est vrai, et dans ce cas la règle déclenchée se comporte comme l'implication matérielle, c'est ce qu'exprime l'objet conditionnel résultat de la conjonction. Cette logique constitue la sémantique la plus simple [8] du système P d'inférence non monotone de Kraus, Lehmann, et Magidor [57].

Nous reviendrons dans la suite sur le traitement de la non-monotonie en programmation logique, mais nous allons auparavant examiner les points de rencontre entre Prolog et la logique floue, et plus généralement entre la programmation logique et la logique floue.

3. Prolog, programmation logique et logique floue

La logique floue [86] est une logique conçue pour prendre en compte des prédicats et des relations graduelles, c'est-à-dire dont la satisfaction est une question de degré. Ainsi un énoncé comme e= « Jean est grand » peut avoir un degré de vérité val entre 0 et 1 selon la taille de Jean (par exemple si Jean mesure 1,75 m, c'est-à-dire que val(Jean est grand) $= \mu_{grand}(taille(Jean)) = \mu_{grand}(1{,}75) < 1$, où μ_{grand} est une fonction caractéristique, à valeur sur $[0, 1]$, de l'ensemble flou 'grand' (dans un contexte donné).

La logique floue est une logique multivaluée qui est *compositionnelle* par rapport à *tous* les connecteurs logiques. En effet, si on considère un autre énoncé e' = « Paul est jeune », alors

$$val(e \wedge e') = \min(val(e), val(e')) = \min(\mu_{grand}(taille(Jean)), \mu_{jeune}(age(Paul)).$$

[8] Par contre, les probabilités $Prob(b \mid a)$ et $Prob(\neg b \mid a \wedge c)$ peuvent avoir toutes les deux des valeurs élevées ($> \frac{1}{2}$) sans contradiction (mais elles ne peuvent pas être simultanément égales à 1).

[9] On notera la forte différence avec la conjonction en logique tri-valuée de la présupposition. Par contre, la négation $\neg(b \mid a) = (\neg b \mid a)$ a une table de vérité analogue à sa table en logique de la présupposition, $\neg(b \mid a)$ est indéfini si et seulement si $b \mid a$ l'est.

La disjonction $val(e \vee e')$ est définie de manière similaire, avec max à la place de min. Quant à la négation, elle est définie par $val(\neg e) = 1 - val(e)$.

Prologs flous. Dans la deuxième moitié des années 1980, les succès de Prolog, puis un peu plus tard de la logique floue (tous les deux amplifiés par leur retentissement au Japon !(10)) ont sans doute encouragé des chercheurs de ce dernier domaine à développer des *Prolog flous* : On peut citer notamment, dans un ordre chronologique, ceux de Ishizuka et Kanai [51], de Martin, Baldwin, et Pilsworth [66], de Mukaidono, Shen, et Ding [68], ou de Li et Liu [61]. Le lecteur pourra se reporter à [36] pour le détail des spécificités de ces langages, ainsi que pour la présentation d'une demi-douzaine d'autres tentatives du même ordre, antérieures aux années 1990.

La plupart des Prolog flous utilisent les trois connecteurs de base rappelés au début de cette section pour calculer les degrés de vérité des formules non atomiques ; on montre alors que si deux clauses ont des degrés de vérité strictement plus grand que $\frac{1}{2}$, alors le degré de vérité du résolvant est compris entre le min et le max des degrés de vérité de ces deux clauses [59]. Certains Prolog flous permettent l'utilisation d'autres connecteurs que min et max pour la conjonction et la disjonction. Il existe quelques versions plus récentes de Prolog flous, comme [54, 85] ou [50](11).

Ces Prolog flous, développés par des chercheurs de la logique floue, ne semblent pas avoir suscité un grand intérêt dans le monde de Prolog(12). Ceci est sans doute en partie dû au fait que ces extensions de Prolog sont souvent présentées comme offrant un traitement de l'incertain, alors qu'elles utilisent un calcul de degrés de vérité qui fait sens pour des énoncés contenant des prédicats (ou des relations) graduels (mais pas pour des énoncés incertains qui ne peuvent qu'être vrai ou faux). Or ce n'est pas la même chose de disposer d'une bouteille *presque pleine* ou d'une bouteille *presque certainement* pleine (avec la seconde on peut se trouver, dans le pire cas, face à une bouteille vide !). L'application du calcul de la logique floue à des énoncés afin de prendre en compte leur incertitude est donc inappropriée.

Signalons par contre une série de travaux dans les dix dernières années en programmation logique floue (« Fuzzy ASP ») [12, 13, 69] qui ont été développés pour pouvoir modéliser des problèmes continus, tout en conservant les avantages du raisonnement déclaratif non monotone de Answer Set Programming (ASP). Un exemple d'application concerne la dynamique des réseaux de régulation génétique où il peut être intéressant d'autoriser des niveaux d'activation continus plutôt que booléens [71] ; voir aussi [70] pour une perspective applicative plus large.

(10) Qu'on se souvienne du choix de Prolog en 1982 pour le projet japonais d'ordinateurs de 5ème Génération, et en 1990 du projet LIFE autour de la logique floue du MITI (Ministry of International Trade and Industry au Japon).

(11) Cette dernière version est encore disponible aujourd'hui `https://dectau.uclm.es/bousi-prolog/`.

(12) A. Colmerauer n'était cependant pas un adversaire de la logique floue. Il a pu ainsi aider administrativement Elie Sanchez qui ne disposait pas du statut nécessaire à l'encadrement de thèses ; ils m'ont ainsi tous les deux convié au jury de la thèse de Mokhtar Beldjehem soutenue en 1993, sur « Un apport à la conception de systèmes hybrides neuro-flous par algorithmes de résolution d'équations de relations floues en min-max : le système Fennec » ; mais son sujet [6] était assez éloigné de Prolog.

Un cadre qualitatif logique approprié pour le traitement de l'incertain est celui de la logique possibiliste que nous rappelons brièvement maintenant, avant de voir comment elle rencontre la programmation logique.

4. Logique possibiliste, non monotonie et programmation logique

La logique possibiliste est une logique de l'*incertain*, elle s'applique à des énoncés qui ne sont susceptibles que d'être vrais ou faux, mais où le manque d'information disponible (ou sa fiabilité insuffisante) fait que la valeur de vérité des énoncés reste incertaine. Le niveau de certitude est évalué dans le cas de la logique possibiliste par la borne inférieure d'une mesure de nécessité [37]. Une formule possibiliste de base est une paire (e, α) où e est un énoncé (qui ne peut être que vrai ou faux) et $\alpha \in (0, 1]$[13], qui s'interprète sémantiquement comme $N(e) \geqslant \alpha$ où N est une mesure de nécessité qui n'est compositionnelle *que* pour la *conjonction* : $N(e_1 \wedge e_2) = \min(N(e_1), N(e_2))$. Cette propriété est caractéristique des mesures de nécessité. La formule (e, α) exprime que e est certain au moins au niveau α. Les mesures de nécessité ne sont compositionnelles ni pour la négation de e ($1 - N(\neg e)$ définit la mesure de possibilité duale $\Pi(e)$), ni pour la disjonction ($N(e \vee e') \geqslant \max(N(e), N(e'))$: en effet, on peut être certain de $e \vee e'$ sans être certain ni de e ni de e'). Etant donné deux formules possibilistes (e, α), (e', β), la logique possibiliste infère la formule $(\text{résolvant}(e, e'), \min(\alpha, \beta))$ en accord avec les mesures de nécessité [37].

La différence entre logique floue et logique possibiliste est claire. Même s'il y a des min utilisés en logique floue et en logique possibiliste, ces deux types de logique sont de nature très différentes et ne doivent pas être confondues : l'une est multi-valuée, l'autre est très proche de la logique classique (en stratifiant les formules selon leur niveau de certitude).

Non-monotonie. L'évaluation de conditions négatives dans les règles en programmation logique pose des problèmes de contrôle du calcul. Ainsi, évaluer la conjonction (bien sûr commutative !) $p(x) \wedge \neg p'(x)$ dans un contexte d'unification où on a $p(a), p'(b)$, repose sur la possibilité de tenir pour faux $p'(a)$, ce qui rend la conjonction vraie pour $x = a$ (il s'agit aussi ici de faire la différence entre $\exists x \neg p'(x)$ et $\neg \exists x p'(x)$). Ceci correspond à l'idée de la négation par l'échec [18] : ce qui ne peut être prouvé vrai est tenu pour faux, problème abordé très tôt dans Prolog [28] et qui a conduit à l'introduction de logiques tri-valuées pour rendre compte de la sémantique des programmes logiques[14]. Cela permet aussi au programme d'avoir des capacités de raisonnement non monotone [16]. La rencontre entre programmation logique et raisonnement non-monotone est naturelle, bien connue et ancienne [67]. Dans la

[13] Plus généralement, on peut utiliser des échelles bornées totalement ordonnées, voire des treillis.

[14] Il s'agit en général de la logique de Kleene où la troisième valeur est interprétée comme 'indéfini' et la conjonction est définie par $val(e) = \min(val(e_1), val(e_2))$ où vrai > faux > indéfini. Les motivations sont liées au contrôle de la procédure de calcul : conjonctions de fonctions booléennes dont l'évaluation peut ne pas se terminer [48], ou évaluation de négations [58] ; des extensions quadri-valuées, basées sur la logique de Belnap et les bi-treillis, ont été proposées [11, 47] pour des programmes logiques distribués, capables de surmonter l'incohérence.

suite, nous envisageons cette question dans le cadre de la logique possibiliste qui peut présenter un comportement non monotone et qui offre une sémantique simple des conditionnelles.

En logique possibiliste, les formules sont stratifiées selon leur niveau de certitude, des plus certaines aux moins certaines, ce qui permet de déterminer à quel niveau l'ensemble des formules de certitude égale à ce niveau ou plus élevée devient incohérent. L'ensemble des formules de certitude strictement plus élevée que ce niveau est donc cohérent et permet de faire des déductions saines. L'ajout de nouvelles formules possibilistes ne peut faire que monter le niveau où on rencontre l'incohérence, ce qui peut rendre impossible des déductions qui étaient au préalable saines avant l'ajout. Ceci confère une capacité de raisonnement non monotone à la logique possibiliste [9, 63].

En présence d'un ensemble $\mathcal{D}$ de n règles par défaut de la forme « si a_i alors b_i », il est possible de le stratifier en représentant chaque règle par la contrainte $\Pi(a_i \wedge b_i) > \Pi(a_i \wedge \neg b_i)$ qui exprime que b_i vrai est davantage possible que b_i faux quand a_i est vrai. On peut montrer que sauf incohérence entre les inégalités, il existe une unique mesure de possibilité qui satisfait toutes les contraintes et qui est la plus grande (c'est-à-dire qu'elle laisse un niveau de possibilité maximal à chaque interprétation, et est donc la plus prudente, puisque la moins contraignante). Le calcul de cette mesure de possibilité maximale permet ensuite d'associer à l'implication matérielle associée à chaque règle un niveau de certitude en utilisant la dualité possibilité-nécessité. A partir de $\mathcal{D}$ on peut chercher à déduire une nouvelle règle « si a alors b », qui soit applicable à la situation courante décrite par a. Deux formes d'inférence peuvent alors être définies selon qu'on considère que l'ensemble des contraintes représentant $\mathcal{D}$ implique la contrainte $\Pi(a \wedge b) > \Pi(a \wedge \neg b)$ *pour toute* mesure de possibilité Π, ou *seulement pour* la mesure de possibilité maximale. Dans le premier cas, on obtient un système d'inférence strictement équivalent à celui basé sur les objets conditionnels[(15)] [8] dont il a été question en fin de Section 2. Dans le second cas on obtient une inférence plus hardie qui satisfait une propriété dite de « monotonie rationnelle » [60]. Voir [8] pour un panorama d'ensemble et des références.

Programmation possibiliste. Mise à part quelques considérations théoriques [7, 36], la première description complète d'un langage de programmation possibiliste, propositionnel, mais autorisant aussi des prédicats graduels est l'œuvre de Alsinet et Godo [1]. En programmation logique possibiliste, les règles telles que $b_1, \ldots, b_m \leftarrow a_1, \ldots, b_n, not\ c_1, \ldots, not\ c_k$ (où '$not\ c_j$' est vrai si on ne peut pas prouver 'c_j') sont associées avec des niveaux de certitude. Pascal Nicolas et ses collègues [74, 75, 76] ont été les premiers à proposer et à implémenter un langage de programmation logique possibiliste de type ASP. Cette approche a été étendue à des programmes avec des expressions emboîtées [78], et à des programmes disjonctifs [79]. Schockaert et col. [5] ont proposé une approche en programmation possibiliste avec une sémantique différente pour la négation par l'échec : $not\ c$ y est compris comme « le degré auquel $\neg c$ est possible », ou, de manière équivalente, « le degré de certitude avec lequel on ne peut pas dériver c », ce qui contraste avec le traitement non gradualisé dans [75]

[(15)] à la règle $a \leftarrow a$ près, fausse si $a = \bot$.

de *not c*, compris comme « on ne peut pas dériver c avec un niveau de certitude strictement positif ». Schockaert et col. [5] proposent aussi deux formes d'ASP disjonctif possibiliste. En effet à côté de la compréhension d'une disjonction $b \vee b'$ comme on est certain que b est vrai ou on est certain que b' est vrai (ce qui correspond au traitement habituel de la disjonction en ASP), une autre interprétation, plus faible, de $b \vee b'$ se réduit à la certitude que la disjonction est vraie, ce qui a des avantages computationnels.

Il existe quelques autres travaux en programmation possibiliste. Citons celui de Cárdenas Viedma et de Galindo-Navarro [17] qui développent un environnement Prolog capable de traitement de contraintes temporelles appliquées à des dates mal connues représentées par des distributions de possibilité restreignant les valeurs plus ou moins possibles d'une date, sujet déjà abordé dans [45].

De manière générale, le cadre de la théorie des possibilités permet de représenter différentes formes de contraintes flexibles. Les contraintes avec niveau de *priorité* expriment à quel point il est nécessaire de satisfaire la contrainte, ce qui laisse la possibilité d'ignorer les contraintes les moins impératives si on ne peut pas satisfaire toutes les contraintes simultanément. Les contraintes *floues* ou élastiques (comme par exemple « ne pas arriver trop tard ») sont représentées par des distributions de possibilité qui restreignent l'ensemble des valeurs plus ou moins compatibles avec la contrainte (qui, dans l'exemple, correspondent aux dépassements plus moins admissibles par rapport à une date où on serait juste à l'heure). Ce cadre permet une approche *possibiliste* des problèmes de satisfaction de contraintes y compris en situation d'*incertitude* [84], [34]. Il ne semble cependant pas qu'il existe des tentatives d'extension de Prolog IV (qui intègre le traitement de contraintes) [10] à des contraintes flexibles.

Confalonieri et Prade [24] ont adapté le traitement en logique possibiliste de la prise de décision qualitative sous incertitude dans le cadre de l'ASP. Ils ont montré comment les croyances pondérées et les préférences avec priorité appartenant à deux bases de connaissances distinctes peuvent être traitées en ASP en modélisant la prise de décision qualitative en termes de programmation logique abductive où la connaissance (incertaine) du monde et les préférences avec priorité sont codées respectivement comme un programme logique défini possibiliste et par des littéraux possibilistes.

5. Logique possibiliste généralisée et ASP

Nous allons maintenant brièvement indiquer comment la logique possibiliste généralisée (LPG) peut être utilisée pour caractériser la sémantique de ASP. En LPG [44], on ne manipule pas seulement des formules de la forme (e, α) et leur conjonction, mais on peut aussi appliquer la négation ou la disjonction à de telles formules : ainsi $\neg(e, \alpha)$ signifie que $N(e) \geqslant \alpha$ est faux, c'est-à-dire qu'on a $N(e) < \alpha$ ou encore de façon équivalente que $\Pi(\neg e) > 1 - \alpha$. De même $(e, \alpha) \vee (e', \alpha')$ signifie « $N(e) \geqslant \alpha$ ou $N(e') \geqslant \alpha'$ ». Il s'agit donc d'une logique où on manipule des formules de la logique possibiliste de base au moyen de la logique classique.

LPG est saine et complète pour la sémantique possibiliste par rapport au système suivant d'axiomes (où (a, λ) est maintenant noté $\mathbf{N}_\lambda(a)$) :

$$\text{Les axiomes de la logique classique} \tag{5.1}$$

$$\mathbf{N}_\lambda(a) \rightarrow \mathbf{N}_\lambda(b), \text{ si } a \rightarrow b \text{ est une tautologie (classique)} \tag{5.2}$$

$$\mathbf{N}_\lambda(a \wedge b) \rightarrow \mathbf{N}_\lambda(a) \wedge \mathbf{N}_\lambda(b) \tag{5.3}$$

$$\mathbf{N}_\lambda(a) \wedge \mathbf{N}_\lambda(b) \rightarrow \mathbf{N}_\lambda(a \wedge b) \tag{5.4}$$

$$\mathbf{N}_1(\top) \tag{5.5}$$

$$\mathbf{N}_\lambda(a) \rightarrow \mathbf{\Pi}_1(a) \tag{5.6}$$

$$\mathbf{N}_{\lambda_1}(a) \rightarrow \mathbf{N}_{\lambda_2}(a), \text{ si } \lambda_1 \geqslant \lambda_2 \tag{5.7}$$

muni de la règle du modus ponens.

Le niveau de certitude λ est supposé ici appartenir à une échelle finie $\Lambda_k = \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}, 1\}$, et $\mathbf{\Pi}_\lambda$(a) est ici une abréviation pour $\neg\mathbf{N}_{n(\lambda)}(\neg a)$ avec $n(\lambda) = 1 - \lambda + \frac{1}{k}$. $\mathbf{\Pi}_1$(a) équivaut donc à $\neg\mathbf{N}_{\frac{1}{k}}(\neg a)$, ce qui signifie $\neg(N(\neg a) \geqslant \frac{1}{k})$, c'est-à-dire $\neg(N(\neg a) > 0) \Leftrightarrow N(\neg a) = 0$ (puisqu'on est sur une échelle discrète)

Pour représenter une règle avec une négation par échec telle que $b \leftarrow \mathit{not}\ a$, on a besoin d'au moins 3 niveaux, c'est-à-dire qu'on prend $\Lambda_2 = \{0, 1/2, 1\}$. En effet, dans ce cas, $b \leftarrow \mathit{not}\ a$ peut se transposer en $\mathbf{\Pi}_1(\neg a) \rightarrow \mathbf{N}_1(b)$ qui exprime que s'il est cohérent de supposer $\neg a$, on peut dériver b avec (complète) certitude ; comme, on a $\mathbf{\Pi}_1(\neg a) \rightarrow \mathbf{N}_1(b) \equiv \mathbf{N}_{1/2}(a) \vee \mathbf{N}_1(b)$, cela se lit aussi « si on n'est pas quelque peu certain que a est vrai, alors on est certain que b est vrai ».[(16)]

La logique possibiliste généralisée (LPG) a une remarquable puissance de représentation, elle permet, entre autres, de coder la « logique d'équilibre » de Pearce [81], et en particulier ASP, ce qui permet de mettre clairement à jour la sémantique des règles ; pour plus de détails le lecteur pourra se référer à [42, 43, 44].

6. Règles possibilistes, réseaux de neurones, règles à seuils

Les règles, qui constituent un format naturel pour l'expression des connaissances, ont un caractère orienté, que la programmation logique met en valeur. Ce qui contraste avec la logique classique où une règle « si a alors b » est représentée par une implication matérielle contraposable.

Calcul matriciel possibiliste. Le caractère orienté des règles peut être préservé dans un cadre possibiliste au moyen de distributions de possibilités conditionnelles. La règle « si a alors b » est ainsi associée avec $\pi(b \mid a)$ et $\pi(\neg b \mid a)$, respectivement possibilité que b soit vrai et que b soit faux, quand a est vrai.

$\pi(b \mid a)$ est compatible avec la notion d'objet conditionnel $b \mid a$ (si $b \mid a \vDash d \mid c$ then $\pi(b \mid a) \leqslant \pi(d \mid c)$ [38].

[(16)] On a bien alors que $\mathbf{\Pi}_1(\neg a) \rightarrow \mathbf{N}_1(b) \neq \mathbf{\Pi}_1(\neg b) \rightarrow \mathbf{N}_1(a)$, ce qui n'est pas le cas avec deux niveaux : si on prend $k = 1$, on a $\Lambda_1 = \{0, 1\}$ et $\mathbf{\Pi}_1(\neg a) \rightarrow \mathbf{N}_1(b)$ est alors équivalent à $\mathbf{N}_1(a) \vee \mathbf{N}_1(b)$, ce qui est équivalent à $\mathbf{\Pi}_1(\neg b) \rightarrow \mathbf{N}_1(a)$, la règle devient contraposable, ce qui n'est pas acceptable.

Une condition de normalisation impose que $\max(\pi(b \mid a), \pi(\neg b \mid a)) = 1$. La propagation de l'incertitude se fait par le produit matriciel suivant :

$$\begin{bmatrix} \pi(b) \\ \pi(\neg b) \end{bmatrix} = \begin{bmatrix} \pi(b \mid a) & \pi(b \mid \neg a) \\ \pi(\neg b \mid a) & \pi(\neg b \mid \neg a) \end{bmatrix} \otimes \begin{bmatrix} \pi(a) \\ \pi(\neg a) \end{bmatrix} = \begin{bmatrix} 1 & s \\ r & 1 \end{bmatrix} \otimes \begin{bmatrix} \lambda \\ \rho \end{bmatrix}$$

où $\otimes$ est un produit de matrices max-min [40], en conformité avec la théorie des possibilités. Ce produit préserve la normalisation. $1 - r$ (resp. $1 - s$) évalue à quel point la règle « si a alors b » (resp. « si $\neg a$ alors $\neg b$ ») est certaine.

Plusieurs règles en parallèle « si a_i alors b_i », avec a_i = « $p(x) \in P_i$ », b_i = « $q(x) \in Q_i$ » peuvent se réécrire comme un produit de matrices *min-max* (en termes des s_i, r_i, qui codent les règles, et des λ_i, ρ_i qui correspondent aux entrées). Voir [40] pour le principe, et [3] pour la théorie générale.

Règles et réseau neuronal. L'idée de règle, à l'honneur à l'époque des systèmes experts, est souvent opposée aux réseaux neuronaux de l'apprentissage « moderne » (voir par exemple [89]). Le cadre de représentation ci-dessus offre un exemple où cette opposition n'est pas conceptuellement fondée. En effet, une cascade de tels produits min-max de matrices peut être représenté par un *réseau neuronal min-max multi-couches* comme établi formellement dans [3]. De plus, il permet de calculer des *explications* [4, 46]. Bien sûr, la taille de tels réseaux neuronaux est beaucoup plus réduites que celle des réseaux d'apprentissage profond qui travaillent sur des représentations de niveau bas. Il est cependant notable qu'on puisse passer d'ensembles de blocs de règles parallèles à des réseaux neuronaux et vice versa, au moins dans le principe.

Règles à seuils En dehors des règles par défaut et des règles incertaines, un autre type de règles important pour l'expression de connaissances, est constitué par des règles à seuils. Il peut s'agir de

- règles de *sélection* de la forme si $x_1 \geqslant \alpha_1$ et $\cdots$ $x_j \geqslant \alpha_m$ alors $y \geqslant \gamma$;
- règles de *élimination* de la forme si $x_1 \leqslant \alpha_1$ et $\cdots$ $x_j \leqslant \alpha_n$ alors $y \leqslant \gamma$.

Quand tous les seuils d'une même règle sont égaux, un tel ensemble de règles est équivalent à une intégrale de Sugeno (un type de fonctions d'agrégation qualitatives), qui peut être aussi codé en *logique possibiliste* [41]. Voir [15] pour le cas où les seuils varient à l'intérieur d'une même règle.

Il est notable que ces règles à seuils soient proches des neurones impulsionnels (neurones à « spikes ») [49], physiologiquement plus plausibles que les neurones artificiels, et qui sont déclenchés quand toutes les entrées cumulées sont confrontées à un seuil unique (alors que les dépassements des seuils dans les conditions des règles sont jugés séparément [2]).

Au plan de la programmation logique, ces règles à seuils sont des contraintes conditionnelles fermées pour le minimum [72, 73], ce qui fait qu'elles sont traitables en temps polynomial.

7. Pour conclure

Ce petit article a mis en évidence quelques points de rencontre entre d'une part des travaux en programmation logique tournée vers le contrôle de la mise en oeuvre de la logique classique posant ainsi des problèmes de sémantique, et d'autre part des études visant d'abord à augmenter l'expressivité de la logique classique, les deux courants de recherche opérant un dépassement de la logique classique.

Ceci nous a permis de faire une place importante à la notion de règle, moyen privilégié d'expression des connaissances, ouvrant éventuellement la possibilité d'un enrichissement mutuel sur ce sujet entre la programmation logique et la représentation des connaissances.

Dans ce parcours, on a croisé un certain nombre de logiques non classiques, principalement tri-valuées, motivées par différentes problématiques de contrôle et d'expressivité. Une vision plus unifiée de ces logiques offrant à la programmation logique une sémantique plus articulée avec la représentation des connaissances est un sujet d'étude pour l'avenir.

Il semble de nos jours qu'il y ait une sorte de divorce entre l'IA dite « symbolique » et une IA « numérique », avec un antagonisme entre fonctions apprenables et ensemble de règles fournies par un expert. Le tout accompagné d'un changement de paradigme dans le traitement du langage naturel avec des progrès spectaculaires en traduction automatique et en génération de textes, avec des modèles de langue tels que le système conversationnel *chat-GPT* d'OpenAI, capables de produire des synthèses et des argumentaires sur des sujets variés, mais incapables de raisonner et de contrôler un raisonnement. Cette situation suggère de dépasser cet antagonisme et de développer des coopérations entre « IA symbolique » et « IA numérique ».

Hommage. Cet article est un hommage à la mémoire d'Alain Colmerauer, qui a développé toute sa vie avec constance et cohérence un axe de recherche important qu'il avait contribué à créer. J'ai eu la chance de le rencontrer dès le début de ma carrière (en décembre 1978) alors que je cherchais à connaître les quelques acteurs français de l'époque en intelligence artificielle au moment de devenir attaché de recherche au CNRS. Ensuite j'ai eu bien des occasions d'apprécier ses qualités humaines et son ouverture d'esprit à l'occasion de multiples rencontres en tant que chercheur en IA ou que membre du Comité National de la Recherche Scientifique. J'ai été très heureux qu'Alain Colmerauer accepte de préfacer le volume 2 sur l'algorithmique de l'IA des versions française, puis anglaise, du vaste panorama des recherches en intelligence artificielle que Pierre Marquis, Odile Papini et moi-même avons élaboré et assemblé [65].

Remerciements

L'auteur remercie Verónica Dahl pour les riches échanges qu'il a pu avoir avec elle, à propos de ses premiers travaux en programmation logique, et Guy-Alain Narboni pour les références qu'il lui a communiquées.

Des remerciements sont dûs à Didier Dubois pour une relecture attentive d'une première version du texte et pour ses commentaires.

Bibliographie

[1] T. Alsinet & L. Godo, « A complete calculus for possibilistic logic programming with fuzzy propositional variables », in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* (San Francisco, CA, USA), UAI'00, Morgan Kaufmann Publishers Inc., 2000, p. 1-10.

[2] I. Baaj, D. Dubois, F. Faux, H. Prade, A. Rico & O. Strauss, « Réseau de neurones et logique : un cadre qualitatif », in *31e Rencontres francophones sur la Logique Floue et ses Applications (LFA 2022)* (Toulouse), Cépaduès, 2022, p. 127-134.

[3] I. Baaj, J.-P. Poli, W. Ouerdane & N. Maudet, « Min-max inference for possibilistic rule-based system », in *Proc. IEEE 30th Int. Conf. on Fuzzy Systems (Fuzz-IEEE'21), Luxembourg, July 11-14*, 2021.

[4] ———, « Representation of Explanations of Possibilistic Inference Decisions », in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (J. Vejnarová & N. Wilson, éds.), Lecture Notes in Computer Science, vol. 12897, Springer International Publishing, 2021, p. 513-527.

[5] K. Bauters, S. Schockaert, M. De Cock & D. Vermeir, « Characterizing and extending answer set semantics using possibility theory », *Theory and Practice of Logic Programming* **15** (2015), no 1, p. 79–116.

[6] M. Beldjehem, « The fennec system », in *Proceedings of the 1994 ACM Symposium on Applied Computing* (H. Berghel, T. Hlengl & J. E. Urban, éds.), SAC '94, Association for Computing Machinery, 1994, p. 126–130.

[7] S. Benferhat, D. Dubois & H. Prade, « Possibilistic logic: From nonmonotonicity to logic programming », in *Proc. 2nd Europ. Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'93), Granada, Nov. 8-10* (M. Clarke, R. Kruse & S. Moral, éds.), Lecture Notes in Computer Science, vol. 747, Springer Berlin Heidelberg, 1993, p. 17-24.

[8] S. Benferhat, D. Dubois & H. Prade, « Nonmonotonic reasoning, conditional objects and possibility theory », *Artif. Intell.* **92** (1997), no 1, p. 259-276.

[9] ———, « Practical handling of exception-tainted rules and independence information in possibilistic logic », *Appl. Intell.* **9** (1998), no 2, p. 101-127.

[10] F. Benhamou, P. Bouvier, A. Colmerauer, H. Garetta, B. Giletta, J. L. Massat, G. A. Narboni, S. N'Dong, R. Pasero, J. F. Pique, T. Touraïvane, M. Van Caneghem & E. Vétillard, *Le manuel de Prolog IV*, PrologIA, Marseille, 1996.

[11] H. A. Blair & V. S. Subrahmanian, « Paraconsistent logic programming », *Theor. Comput. Sci.* **68** (1989), no 2, p. 135-154.

[12] M. Blondeel, S. Schockaert, D. Vermeir & M. De Cock, « Fuzzy Answer Set Programming: An Introduction », in *Soft Computing: State of the Art Theory and Novel Applications* (R. R. Yager, A. M. Abbasov, M. Z. Reformat & S. N. Shahbazova, éds.), Studies in Fuzziness and Soft Computing, vol. 291, Springer, Berlin, Heidelberg, 2013, p. 209-222.

[13] M. Blondeel, S. Schockaert & M. Vermeir, D. De Cock, « Complexity of fuzzy answer set programming under Łukasiewicz semantics », *Int. J. Approx. Reasoning* **55** (2014), no 9 (English), p. 1971-2003.

[14] D. G. Bobrow (ed.), « Special Issue on Non-Monotonic Reasoning Artificial Intelligence », *Artif. Intell.* (1980), no (1,2), p. 1-172.

[15] Q. Brabant, M. Couceiro, D. Dubois, H. Prade & A. Rico, « Learning rule sets and Sugeno integrals for monotonic classification problems », *Fuzzy Sets and Systems* **401** (2020), p. 4-37.

[16] G. Brewka, V. W. Marek & M. Truszczynski (éds.), *Nonmonotonic Reasoning. Essays Celebrating Its 30th Anniversary College Publications*, Mathematical Logic and Foundations, vol. 31, College Publications, 2011.

[17] M. A. Cárdenas-Viedma & F. M. Galindo-Navarro, « PROlogic: A fuzzy temporal constraint PROLOG », *Int. J. of Applied Mathematics* **32** (2019), no 4, p. 677-719.

[18] K. L. Clark, « Negation as Failure », in *Proc. Symp. on Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977* (New York) (H. Gallaire & J. Minker, éds.), Advances in Data Base Theory, Plemum Press, 1977, p. 293-322.

[19] A. Colmerauer, « Metamorphosis grammars », in *Natural Language Communication with Computers* (L. Bolc, éd.), Lecture Notes in Computer Science, vol. 63, Springer, Berlin, Heidelberg, 1978, p. 133-188.

[20] ———, « Un sous-ensemble intéressant du français », *RAIRO. Informatique théorique* **13** (1979), nº 4, p. 309-336.

[21] A. Colmerauer, H. Kanoui, Ph. Roussel & R. Pasero, « Un système de communication homme-machine en français », Rapport groupe d'intelligence artificielle, Université d'Aix-Marseille, juin 1973.

[22] A. Colmerauer & J. F. Pique, « About Natural Logic », in *Advances in Data Base Theory: Volume 1* (H. Gallaire, J. Minker & J.-M. Nicolas, éds.), Springer US, Boston, MA, 1981, based on Proc. Workshop on Formal Bases for Data Bases, Dec. 12-14, 1979, CERT, Toulouse, p. 343-365.

[23] A. Colmerauer & P. Roussel, « The birth of Prolog », in *The Second ACM SIGPLAN Conference on History of Programming Languages, Cambridge, Massachusetts, USA*, HOPL-II, Association for Computing Machinery, 1993, p. 37–52.

[24] R. Confalonieri & H. Prade, « Using possibilistic logic for modeling qualitative decision: answer set programming algorithms », *Int. J. Approx. Reasoning* **55** (2014), nº 2, p. 711-738.

[25] V. Dahl, « Un système déductif d'interrogation de banques de données en espagnol », Thèse de 3e cycle, groupe d'intelligence artificielle, Université d'Aix-Marseille, 14 nov. 1977.

[26] ———, « Quantification in a three-valued logic for natural language question-answering systems », in *Proceedings of the 6th International Joint Conference on Artificial Intelligence – Volume 1*, IJCAI'79, Morgan Kaufmann Publishers Inc., 1979, p. 182–187.

[27] ———, « A three-valued logic for natural language computer applications », in *Proc. 10th IEEE Int. Symp. Multiple-Valued Logic, Evanston, IL, 3-5 June*, 1980.

[28] ———, « Two solutions for the negation problem », in *Proc. Logic Programming Workshop, Debrecen, Hungary* (S.-A. Tarnlund, éd.), 1980, p. 61-72.

[29] ———, « Translating Spanish into logic through logic », *American J. of Computational Linguistics* (1981), nº 3, p. 149-164.

[30] ———, « On database systems development through logic », *ACM Trans. Database Syst.* **7** (1982), nº 1, p. 102-123.

[31] ———, « Dimensions linguistiques de Prolog: le passé, le futur », 2024, *Revue Ouverte d'Intelligence Artificielle*, ce numéro.

[32] V. Dahl & R. Sambuc, « Un système de banque de données en logique du premier ordre, en vue de sa consultation en langue naturelle », Rapport de dea en intelligence artificielle, Université d'Aix-Marseille, 1976.

[33] B. De Finetti, « La logique de la probabilité », in *Actes Congrès Int. de Philos. Scient., Paris 1935*, Hermann et Cie Editions, Paris, 1936, p. IV1- IV9.

[34] D. Dubois, H. Fargier & H. Prade, « Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty », *Appl. Intell.* **6** (1996), nº 4, p. 287-309.

[35] D. Dubois, J. Lang & P. H., « Fuzzy sets in approximate reasoning, Part 2: Logical approaches », *Fuzzy Sets and Systems* **40** (1991), nº 1, p. 203-244.

[36] D. Dubois, J. Lang & H. Prade, « Towards possibilistic logic programming », in *Proc. 8th Int. Conf. on Logic Programming (ICLP'91), Paris, June 24-28* (K. Furukawa, éd.), MIT Press, 1991, p. 581-595.

[37] ———, « Possibilistic logic », in *Handbook of Logic in Artificial Intelligence and Logic Programming, (Vol. 3): Nonmonotonic Reasoning and Uncertain Reasoning* (D. M. Gabbay, C. J. Hogger, J. A. Robinson & D. Nute, éds.), Oxford University Press, 1994, p. 439-513.

[38] D. Dubois & H. Prade, « Conditioning, non-monotonic logic and non-standard uncertainty models », in *Conditional Logic in Expert Systems* (I. R. Goodman, M. M. Gupta, H. T. Nguyen & G. S. Rogers, éds.), North-Holland, 1991, p. 115-158.

[39] ———, « Conditional objects as nonmonotonic consequence relationships », *IEEE Trans. on Syst., Man & Cybern.* **24** (1994), nº 12, p. 1724-1740.

[40] ———, « From possibilistic rule-based systems to machine learning », in *Proc. 14th Int. Conf. Scalable Uncert. Management (SUM'2020)* (Cham), Lecture Notes in Computer Science, vol. 12322, Springer, 2020, p. 35-51.

[41] D. Dubois, H. Prade & A. Rico, « The logical encoding of Sugeno integrals », *Fuzzy Sets and Systems* **241** (2014), p. 61-75.

[42] D. Dubois, H. Prade & S. Schockaert, « Règles et métarègles en théorie des possibilités. De la logique possibiliste à la programmation par ensembles-réponses », *Revue d'Intelligence Artificielle* **26** (2012), nº 1-2, p. 63-83.

[43] ———, « Extending Answer Set Programming using Generalized Possibilistic Logic », in *Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25-27, 2015* (O. Papini, S. Benferhat, L. Garcia, M.-L. Mugnier, E. L. Fermé, T. Meyer, R. Wassermann, T. Hahmann, K. Baclawski, A. Krisnadhi, P. Klinov, B. S., O. Kutz & P. D., éds.), CEUR Workshop Proceedings, vol. 1517, CEUR-WS.org, 2015.

[44] ———, « Generalized possibilistic logic: Foundations and applications to qualitative reasoning about uncertainty », *Artif. Intell.* **252** (2017), p. 139-174.

[45] S. Dutta, « A temporal logic for uncertain events and an outline of a possible implementation in an extension of PROLOG », in *Proc. 4th Conf. on Uncertainty in Artificial Intelligence (UAI'88) Minneapolis, July 10-12* (L. Kanal, J. Lemmer, T. Levitt & R. Shachter, éds.), 1988, p. 90-97.

[46] H. Farreny & H. Prade, « Explications de raisonnements dans l'incertain », *Revue d'Intelligence Artificielle* **4** (1990), nº 2, p. 43-75.

[47] M. Fitting, « Bilattices and the semantics of logic programming », *J. of Logic Programming* **11** (1991), p. 91-116.

[48] M. Fitting & M. Ben-Jacob, « Stratified and three-valued logic programming semantics », in *Proc. 5th Int. Conf. and Symp. on Logic Programming (ICLP/SLP'88), Seattle, Aug. 15-19 1988* (R. A. Kowalski & K. A. Bowen, éds.), MIT Press, 1988, p. 1054-1069.

[49] S. Ghosh-Dastidar & H. Adeli, « Spiking neural networks », *Int. J. of Neural Syst.* **19** (2009), p. 295-308.

[50] P. J. Iranzo, C. Rubio-Manzano & G.-C. J., « Bousi~Prolog: A Prolog extension language for flexible query answering », *Electron. Notes Theor. Comput. Sci.* **248** (2009), p. 131-147.

[51] M. Ishizuka & N. Kanai, « Prolog-ELF incorporating fuzzy logic », in *Proc. 9th Int. Joint Conf. on Artificial Intelligence (IJCAI'85), Los Angeles, Aug. 18-23* (A. K. Joshi, éd.), Morgan Kaufmann, 1985, p. 701-703.

[52] ———, « Prolog-ELF incorporating fuzzy logic », *New Gener. Comput.* **3** (1985), nº 4, p. 479-486.

[53] E. L. Keenan, « On semantically based grammar », *Linguistic Inquiry* **3** (1972), nº 4, p. 413-461.

[54] F. Klawonn & R. Kruse, « A Łukasiewicz logic based Prolog », *Mathware & Soft Computing* **1** (1994), p. 5-29.

[55] R. A. Kowalski, « Algorithm = Logic + control », *Commun. ACM* **22** (1979), nº 7, p. 424-436.

[56] R. A. Kowalski & M. Van Emden, « The semantics of predicate logic as a programming language », *J. Assoc. Comput Mach.* **23** (1976), nº 4, p. 733-743.

[57] S. Kraus, D. Lehmann & M. Magidor, « Nonmonotonic reasoning, preferential models and cumulative logics », *Artificial Intellence* **44** (1990), nº 1-2, p. 167-207.

[58] K. Kunen, « Negation in logic programming », *J. Log. Program.* **4** (1987), nº 4, p. 289-308.

[59] R. C. T. Lee, « Fuzzy logic and the resolution principle », *J. Assoc. Comput. Mach.* **19** (1972), p. 109-119.

[60] D. Lehmann & M. Magidor, « What does a conditional knowledge base entail? », *Artif. Intell.* **55** (1992), nº 1, p. 1-60.

[61] D. Li & D. Liu, *A fuzzy PROLOG database system*, Research Studies Press, 1990, 426 pages.

[62] V. Lifschitz, « What is Answer Set Programming? », in *Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI'08), Chicago, July 13-17* (D. Fox & C. P. Gomes, éds.), 2008, p. 1594-1597.

[63] Léa Sombé, Group, P. Besnard, M. Cordier, D. Dubois, L. Farinas del Cerro, C. Froidevaux, Y. Moinard, H. Prade, C. Schwind & P. Siegel, *Reasoning under Incomplete Information in Artificial Intelligence: A Comparison of Formalisms Using a Single Example*, Wiley, 1990, Aussi, *Int. J. of Intelligent Systems*, **5**, n° 4, 323–471.

[64] V. Marek & M. Truszczyński, « Stable models and an alternative logic programming paradigm », in *The Logic Programming Paradigm: a 25-Year Perspective* (K. R. Apt, V. W. Marek, M. Truszczyński & D. S. Warren, éds.), Springer Verlag, Berlin, Heidelberg, 1999, p. 375-398.

[65] P. Marquis, O. Papini & H. Prade (éds.), *Panorama de l'Intelligence Artificielle – 3 volumes*, Cépaduès, 2014, [Version anglaise mise à jour et augmentée : *A Guided Tour of Artificial Intelligence Research - 3 volumes*, Springer, 2020].

[66] T. P. Martin, J. F. Baldwin & B. W. Pilsworth, « The implementation of Fprolog - A fuzzy Prolog interpreter », *Fuzzy Sets and Systems* **23** (1987), nº 1, p. 119-129.

[67] J. Minker, « An overview of nonmonotonic reasoning and logic programming », *The J. of Logic Programming* **17** (1993), nº 2–4, p. 95-126.

[68] M. Mukaidono, Z. Shen & L. Ding, « Fundamentals of fuzzy Prolog », *Int. J. Approx. Reason.* **3** (1989), nº 2, p. 179-193.

[69] M. Mushthofa, S. Schockaert & M. De Cock, « Solving disjunctive fuzzy answer set programs », in *Proc. 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), Lexington, KY, Sept. 27-30, 2015* (F. Calimeri, G. Ianni & M. Truszczynski, éds.), Lecture Notes in Computer Science, vol. 9345, Springer, 2015, p. 453-466.

[70] ———, « Fuzzy Answer Set Programming: From theory to practice », in *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications* (O. Kosheleva, S. P. Shary, Gang. Xiang & R. Zapatrin, éds.), Studies in Computational Intelligence, Springer, Cham, 2020, p. 213–228.

[71] M. Mushthofa, S. Schockaert, L.-H. Hung, K. Marchal & M. De Cock, « Modeling multi-valued biological interaction networks using fuzzy answer set programming », *Fuzzy Sets and Syst.* **345** (2018), p. 63-82.

[72] G. A. Narboni, « On rule systems whose consistency can be locally maintained », *AI Commun.* **26** (2013), nº 1, p. 67-77.

[73] ———, « Propagation properties of min-closed CSPs », in *Proc. 30th Int. Conf. on Logic Programming (ICLP'14), Vienna, July 19-22 2014, Technical Communication, Supplementary materials*, 2014, https://static.cambridge.org/content/id/urn:cambridge.org:id:article:S1471068414000581/resource/name/S1471068414000581sup001.pdf, p. 164-174.

[74] P. Nicolas, L. Garcia & I. Stéphan, « Possibilistic stable models », in *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), Edinburgh, July 30 - Aug. 5 2005* (L. Pack Kaelbling & A. Saffiotti, éds.), 2005, p. 248-253.

[75] P. Nicolas, L. Garcia, I. Stéphan & C. Lefèvre, « Possibilistic uncertainty handling for answer set programming », *Ann. Math. Artif. Intell.* **47** (2006), nº 1–2, p. 139–181.

[76] P. Nicolas & C. Lefèvre, « Possibilistic stable model computing », in *Proc. 3rd Int. Answer Set Programming Workshop (ASP'05), Bath, Sept. 27-29, 2005* (M. De Vos & A. Provetti, éds.), CEUR Workshop Proceedings, vol. 142, 2005, p. 203-215.

[77] I. Niemelä, « Logic programs with stable model semantics as a constraint programming paradigm », *Ann. Math. Artif. Intell.* **25** (1999), nº 3-4, p. 241–273.

[78] J. C. Nieves & H. Lindgren, « Possibilistic nested logic programs and strong equivalence », *Int. J. Approx. Reason.* **59** (2015), p. 1-19.

[79] J. C. Nieves, M. Osorio & U. Cortés, « Semantics for possibilistic disjunctive programs », *Theory Pract. Log. Program.* **13** (2013), nº 1, p. 33-70.

[80] R. Pasero, « Un essai de communication sensée en langue naturelle », Rapport interne, Université d'Aix-Marseille, sept 1976.

[81] D. Pearce, « Equilibrium logic », *Ann. Math. Artif. Intell.* **47** (2006), nº 1-2, p. 3-41.

[82] Sambuc, « Fonctions Φ-floues : Applications à l'aide au diagnostic en pathologie thyroïdienne », Thèse d'exercice, Université d'Aix-Marseille 2, séc. 1975.

[83] E. Sanchez & R. Sambuc, « Relations floues. Fonctions Φ-floues. Application à l'aide au diagnostic en pathologie thyroïdienne », in *Proc. Medical Data Processing Symposium, Toulouse, March 2-5 1976* (M. Laudet, J. Anderson & F. Begon, éds.), Taylor and Francis, 1976.

[84] T. Schiex, H. Fargier & G. Verfaillie, « Valued constraint satisfaction problems: Hard and easy problems », in *Proc. 14th Int. Joint Conf. on Artificial Intelligence – Volume 1, (IJCAI'95), Montréal, Aug. 20-25 1995*, Morgan Kaufmann, 1995, p. 631-639.

[85] C. Vaucheret, S. Guadarrama & S. Muñoz, « Fuzzy Prolog: A simple general implementation using CLP ($\mathcal{R}$) », in *Logic for programming, artificial intelligence, and reasoning. 9th international*

conference, LPAR 2002, Tbilisi, Georgia, October 14–18, 2002. Proceedings, Springer, Berlin, 2002, p. 450-463.

[86] L. A. Zadeh, « Fuzzy sets », *Inf. Control* **8** (1965), p. 338-353.

[87] ———, « The concept of a linguistic variable and its application to approximate reasoning. I », *Inf. Sci.* **8** (1975), p. 199-249.

[88] L. A. Zadeh, « Outline of a new approach to the analysis of complex systems and decision processes », *IEEE Trans. Syst. Man Cybern.* **3** (1973), p. 28-44.

[89] C. Zeitoun, Y. Pigenet & H. Leroux, « Quels neurones pour l'intelligence artificielle ? », *CNRS Le Journal* **291** (2018), p. 30–38.

Abstract. — Logic programming and knowledge representation have been two important streams of research in artificial intelligence that have developed in the last 50 years with largely different concerns, but with some points of convergence, in particular on non-monotonic reasoning, or on multi-valued logics. This is what this modest article proposes to revisit, mainly around links and complementarities with fuzzy logic and possibilistic logic, in a more historical than technical perspective.

Keywords. — Logic programming, answer set programming, knowledge representation, non-monotonic reasoning, conditional statement, if-then rule, threshold rule, trivalued logics, fuzzy logic, possibilistic logic, flexible constraint, history of AI.

Manuscrit reçu le 27 mai 2024, accepté le 12 juillet 2024.