RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

G. BLANC
K. LIOGIER

Négation constructive et axiomatique interne

RAIRO. Informatique théorique et applications, tome 31, n° 5 (1997), p. 411-428

<http://www.numdam.org/item?id=ITA_1997__31_5_411_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 5, 1997, pp. 411-428)

NÉGATION CONSTRUCTIVE ET AXIOMATIQUE INTERNE (*)

par G. BLANC (1) et K. LIOGIER (2)

Abstract. – *We propose in this work a mathematical aspect of Constructive Negation for constraint general logic programs. We can then obtain a completion theorem (soundness and completeness) with internal axiomatization of general program, in the classical logic deduction frame: the bivalued models.*

Keywords: *constraint programming, constructive negation, bivalved semantics.*

Résumé. – *Nous proposons dans ce travail une version théorique de la Négation Constructive pour les programmes généraux avec contraintes. Nous pouvons alors obtenir un théorème d'adéquation (correction et complétude) avec l'axiomatique interne d'un programme général, dans le cadre de la déduction en logique classique, c'est-à-dire celle des modèles bivalués.*

Mots clés : *programmation par contraintes, négation constructive, sémantique bivaluée.*

1. INTRODUCTION

Le processus de résolution connu maintenant sous le nom de Négation Constructive a été introduit par D. Chan [5], [6] en 1988, il a rapidement trouvé un cadre naturel dans la Programmation par Contraintes avec P. J. Stuckey [20] et P. Bruscoli (and all) [4]. Il s'est aussi clairement avancé pour une approche nouvelle de la concurrence avec F. Fages [11], [12]. Nous pensons que l'idée est suffisamment essentielle au principe même de la programmation déclarative pour mériter un traitement conceptuel, indépendant de son approche strictement procédurale.

La recherche d'efficacité dans la mise en œuvre de procédures de résolution dans telle ou telle structure particulière (algèbre d'arbres finis, infinis, arithmétique de l'addition et inégalité, etc.) conduit souvent à négliger ce qu'il

(*) Reçu Février 1996.

Département de Mathématique-Informatique, Faculté des Sciences de Luminy, 163, avenue de Luminy, Case 901, 13288 Marseille Cedex 9.

Institut de Mathématiques de Luminy, CNRS-UPR 9016

(1) gblanc@lumimath.univ-mrs.fr

(2) liogier@iml.univ-mrs.fr

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-5004/97/05/

peut y avoir d'important dans une méthodologie nouvelle ; c'est pour situer clairement cette méthodologie que nous proposons une approche strictement mathématique (non opérationnelle *a priori*) des réponses obtenues par négation constructive. On est en effet étonné que cet aspect de la résolution ne soit pas apparu plus tôt dans la programmation déclarative. Ceci tient sans doute à l'évolution théorique elle-même de la programmation logique qui, en permanence, s'est développée au fur et à mesure de l'apparition de problèmes posés par la pratique. Il en a été ainsi par exemple de la complétion de Clark pour la négation par échec fini [8], de la résolution dans l'algèbre des arbres infinis pour l'absence de contrôle d'occurrences, de la programmation par contraintes pour traiter les arbres infinis ou les nombres, des programmes généraux pour satisfaire le besoin, depuis longtemps présent, de l'usage de la négation dans les programmes.

Un cadre naturel pour la sémantique logique des programmes généraux avait beaucoup de mal à trouver sa justification théorique (K. Apt [1]). La programmation par négation constructive est tout simplement la version opérationnelle adéquate de ce que l'on attend *a priori* d'un programme général. Il suffisait pour cela d'exploiter au maximum toutes les informations qu'un but atomique, devant s'unifier aux têtes de clauses d'un programme P, pouvait offrir. Il s'agit essentiellement d'enregistrer et de répercuter le maximum de réponses possible, qu'elles soient de succès ou d'échec. Ainsi, si relativement au prédicat R, le programme P ne contient que les clauses suivantes :

$$R(b) \leftarrow$$

$$R(x) \leftarrow \lambda(x, y) \,\square$$

$$R(x) \leftarrow \varphi(x, y) \,\square\, A_1,\ A_2$$

$$R(x) \leftarrow \psi(x) \,\square\, B_1, B_2$$

le but $\leftarrow \square\, R(x)$ enregistre au moins les réponses de succès $x = b$, $\exists y\, \lambda(x, y)$ et au moins la réponse d'échec $x \neq b \wedge \forall y \neg \lambda(x, y) \wedge \forall y \neg \varphi(x, y) \wedge \neg \psi(x)$, tout en renvoyant évidemment à d'autres explorations.

Ce qui, jusqu'au travail de D. Chan, n'est pas apparu aussi clair, c'est évidemment la difficulté qu'il peut y avoir de décider de la satisfaisabilité ou non de ces réponses ; mais ceci est propre à chaque langage, à chaque structure, à chaque théorie sous-jacente aux contraintes : axiomatique des algèbres d'arbres, axiomatique de Presburger, axiomatique de Peano, axiomatique des corps réels clos, etc.

Le caractère opérationnel de cette méthodologie de négation par contraintes peut avantageusement, dans un pemier temps, nous intéresser par sa signification vis-à-vis de la déclaration que constitue le programme, oubliant la faisabilité des décisions. Nous préférerions personnellement parler de « réponses exploitées par chaînage arrière » d'un programme contraint général plutôt que de négation constructive, mais on ne change pas une appellation qui s'est aussi facilement et rapidement imposée; nous le ferions d'autant plus volontiers que l'aspect « réponses exploitées par chaînage avant » d'un programme contraint général lui est tout à fait dual ! Il s'agit dans les deux cas d'explorer le même ensemble de formules (combinaisons de contraintes) par des processus différents. Dans la version traditionnelle de la programmation logique, les réponses exploitées par chaînage arrière correspondent évidemment au « Success Set » et, celles exploitées par chaînage avant, aux points fixes de l'opérateur de conséquence immédiate. L'équivalence des deux aspects étant une étape importante pour le lien avec une sémantique déductive (non diachronique celle-ci, à la différence des deux autres).

C'est dans cet esprit que nous avons souhaité aussi obtenir une version sémantique déductive, correcte et complète, proche de ce que l'on attend de l'aspect déclaratif d'un tel programme, c'est-à-dire en déduction classique ordinaire (*i.e.* : bivaluée), à la différence des sémantiques partielles (trivaluées) déjà développées dans [17], [13] et [20].

Cette déduction se fera depuis l'axiomatique interne du programme P. Celle-ci est constituée des axiomes caractérisant élémentairement la structure de contraintes, des axiomes relatifs à P, explicitant les inférences relatives aux clauses, ainsi que la négation limitée aux possibilités expressives de P. C'est cette négation faible, déjà utilisée dans [9], [19], [14], [15], qui, explicitée dans l'écriture des axiomes, permet alors de construire des déductions classiques.

Dans la partie 2, nous présenterons le cadre dans lequel notre travail se situe et les différentes notations employées. La partie 3 sera consacrée à la présentation des classes de réponses sur un but fournies par un programme. Cette définition, peu opérationnelle, a pour vocation de bien s'adapter à une démonstration de complétude en logique classique bivaluée. C'est la partie 7 qui, en annexe, fournira une version plus opérationnelle de ces classes. Dans la partie 4, nous définirons la sémantique déclarative associée à un programme, et donnerons les résultats de correction correspondants. La partie 5 présente, dans le cadre choisi, les concepts correspondant aux

opérateurs habituels de conséquences immédiates. Enfin, la partie suivante détaille la complétude en logique bivaluée pour l'axiomatique interne.

2. CADRE DE TRAVAIL ET NOTATIONS

Dans toute la suite, $\mathbb{L}$ désignera le langage de la structure de contraintes.

$\mathcal{R}$ désignera une réalisation de $\mathbb{L}$.

$\mathcal{T}$ désignera une axiomatique complète de $\mathcal{R}$. Plus précisément, $\mathcal{T}$ désignera un ensemble récursivement présentable de formules de $\mathbb{L}$ (les axiomes) satisfaisant pour chaque formule close Φ de $\mathbb{L}$:

$$\mathcal{T} \models \Phi \textit{ si et seulement si } \mathcal{R} \models \Phi$$

Remarquons que supposer la complétude de $\mathcal{T}$ n'est qu'une commodité de présentation, mais l'idée même de résolution de contraintes de $\mathcal{R}$ n'est réaliste que si la décision de satisfaisabilité de ces formules, dans $\mathcal{R}$, est effective. En nous plaçant dans l'hypothèse générale que chaque formule du premier ordre de $\mathbb{L}$ peut être traitée comme une contrainte, l'étude même ne peut avoir de réalité que si la structure $\mathcal{R}$ est décidable, celle-ci possédera alors nécessairement une axiomatique complète récursivement présentable.

Nous désignerons par $\tilde{\exists}\,\phi$ (resp. $\tilde{\forall}\,\phi$) la clôture existentielle (resp. universelle) de ϕ. Nous utiliserons fréquemment aussi des clôtures relatives seulement à quelques variables. Pour cela, elles apparaîtront toujours dans le texte relativement aux variables différentes de celles de la suite fixée $\tilde{x} : x_1, \ldots, x_n$. Ces clôtures seront alors désignées dans toute la suite par $\exists\,\phi$ (resp. $\forall\,\phi$).

DÉFINITION 2.1 : *Un* **programme logique contraint (CLP-programme) normal** *P est un ensemble de* **clauses** *de la forme :*

$$R(\tilde{v}) \leftarrow \Psi(\tilde{z}) \,\square\, D_1(\tilde{y}_1), \ldots, D_p(\tilde{y}_p), \neg E_1(\tilde{u}_1), \ldots, \neg E_q(\tilde{u}_q)$$

où les suites $\tilde{v}$, $\tilde{y}_1, \ldots, \tilde{y}_p$ $\tilde{u}_1, \ldots, \tilde{u}_q$ sont toutes des suites de variables distinctes ; ces suites étant de plus disjointes deux à deux, et toutes contenues dans $\tilde{z}$.

- *$R(\tilde{v})$ est la tête de la clause*
- *$\Psi(\tilde{z}) \,\square\, D_1(\tilde{y}_1), \ldots, D_p(\tilde{y}_p), \neg E_1(\tilde{u}_1), \ldots, \neg E_q(\tilde{u}_q)$ est le corps*
- *$\Psi(\tilde{z})$ est la contrainte*
- *les D_i sont les littéraux positifs*
- *les $\neg E_j$ sont les littéraux négatifs.*

De plus, les programmes seront toujours supposés faire figurer au moins une fois en tête chaque prédicat figurant dans un corps.

Remarque: Toutes ces conventions ne restreignent évidemment en rien la généralité et facilitent l'approche théorique.

DÉFINITION 2.2 : *Un* **But** $G(\tilde{x})$ *n'est rien d'autre qu'un corps de clause et se présentera sous la forme :*

$$\leftarrow \varphi(\tilde{x}) \,\square\, A_1, \ldots, A_p, \neg B_1, \ldots, \neg B_q$$

Nous utiliserons parfois la notation φ_G pour désigner la contrainte du but G.

3. RÉPONSES CALCULÉES, CLASSE DE SUCCÈS, CLASSE D'ÉCHECS

Les **réponses calculées** $\{\rho_i(\tilde{x})\}_{i\in I}$, du programme P sur le but $G(\tilde{x})$ seront des caractérisations d'éléments de structures de contraintes par l'intermédiaire de formules du premier ordre du langage de ces structures : le langage de contraintes $\mathbb{L}$ (*voir* définition 3.3).

Nous appellerons alors **succès** ou **échecs** des classes plus larges de formules que nous noterons $SS_P(G)$ et $FF_P(G)$ (*voir* définition 3.4). Il s'agit de formules caractérisant des éléments parmi ceux caractérisés globalement par les formules de réponses calculées.

Ainsi, si $\mathcal{R}$ correspond à la structure additive des entiers ([1]) et $\mathcal{T}_0$ correspond à une axiomatique complète de la théorie de l'addition (l'axiomatique de Presburger, par exemple). En considérant le programme P suivant :

$$R(x) \leftarrow (y \leq x + 3 \wedge x \leq y) \,\square\, S(y)$$
$$S(y) \leftarrow (y = v + 6) \,\square\, S(v)$$
$$S(v) \leftarrow (v = 0) \,\square$$

et le but $R(G) :=\leftarrow \square\, R(x)$,
si les réponses calculées de succès de P sur G sont les formules :

$$\sigma_k(x) = ((6.k) \leq x + 3 \wedge x \leq (6.k)) \text{ pour chaque entier } k,$$

il est alors clair que les formules $x = 5$, ou $(x = 4 \vee x = 6)$, ou encore $\exists z\,(x = 5.z \wedge x \leq 18)$ caractérisent aussi des succès de P sur $G(x)$ dans

([1]) Rappelons que, dans ce cas, la relation d'ordre, ainsi que chaque entier, y est définissable.

chaque modèle $\mathcal{M}$ de $\mathcal{T}_0$, pour la simple raison que dans $\mathcal{T}_0$ se déduit la formule utilisant la disjonction infinie $\vee_{k\in\omega}\sigma_k(x)$ de toutes les réponses calculées de succès ([2]) :

$$\exists z\,(x = 5.z \wedge x \leq 18) \rightarrow \bigvee_{k\in\omega} \sigma_k(x)$$

Par le théorème de compacité, il est clair que $\mathcal{T}_0 \models \psi(\tilde{x}) \rightarrow \bigvee_{i\in I} \phi_i(\tilde{x})$ est équivalent à l'existence de $i_1, i_2, \ldots, i_m$ dans I pour lesquels on a :

$$\mathcal{T}_0 \models \psi(\tilde{x}) \rightarrow (\phi_{i_1}(\tilde{x}) \vee \ldots \vee \phi_{i_m}(\tilde{x}))$$

Pour des commodités d'écriture et un souci de meilleure expressivité, nous garderons, dans toute la suite, l'usage de formules du langage infinitaire, utilisant les disjonctions infinies $\vee\mathcal{D}$ pour désigner la disjonction de toutes les formules de la classe $\mathcal{D}$, et $\mathcal{T} \models \psi \rightarrow \vee\mathcal{D}$ pourra être considérée comme une simple abréviation de l'existence de $\varphi_1, \ldots, \varphi_k$ dans $\mathcal{D}$ telles que $\mathcal{T} \models \psi \rightarrow (\varphi_1 \vee \ldots \vee \varphi_k)$.

En désignant par $G(\tilde{x})$ le but $\leftarrow \varphi \square A_1, \ldots, A_p, \neg B_1, \ldots, \neg B_q$, on appelle **descendant immédiat de** $G(\tilde{x})$ tout but G′ qui est le corps d'une clause de P, ayant pour tête l'une des formules atomiques A_i ou B_j (après un éventuel renommage des variables, bien évidemment).

Nous désignerons ainsi systématiquement dans toute la suite par G^i_r (resp. G^j_s) le corps de la r^ième (resp. s^ième) clause de P ayant pour tête A_i (resp. B_j).

Remarquons qu'une formalisation plus rigoureuse devrait nous conduire à distinguer les notations G^i_r et G'^j_s suivant que l'on traite un descendant immédiat de G, issu d'un littéral positif ou négatif. Nous pensons toutefois que la distinction par les noms d'indices reste suffisante pour ne pas nuire à la clarté des constructions, tout en maintenant une lecture plus aérée.

DÉFINITION 3.1 : *A chaque entier n et à chaque but $G(\tilde{x})$, nous associons deux classes $\mathcal{S}^n_G$ et $\mathcal{E}^n_G$ de formules de $\mathbb{L}$ dont les variables libres sont parmi celles de $\tilde{x}$, par induction sur n et par la relation de descendance immédiate, de la façon suivante :*

1. $\mathcal{S}^0_G = \mathcal{E}^0_G = \{\bot\}$
2. $\mathcal{S}^{n+1}_G + 1$ *est constitué des formules de la forme :*

$$\varphi \wedge \bigwedge_i \Big(\bigvee_r \exists \psi^i_r\Big) \wedge \bigwedge_j \Big(\bigwedge_s \forall \phi^j_s\Big)$$

([2]) Formules du langage infinitaire $\mathbb{L}_{\omega_1,\omega}$ [16].

avec chaque $\psi_r^i \in \bigcup_{k\leq n} \mathcal{S}^k_{G_r^i}$ *et chaque* $\phi_s^j \in \bigcup_{k\leq n} \mathcal{E}^k_{G_s^j}$.

3. $\mathcal{E}_G^{n+1}$ *est constitué des formules de la forme :*

$$\neg\varphi \vee \bigvee_i \left(\bigwedge_r \forall \phi_r^i\right) \vee \bigvee_j \left(\bigvee_s \exists \psi_s^j\right)$$

avec chaque $\phi_r^i \in \bigcup_{k\leq n} \mathcal{E}^k_{G_r^i}$ *et chaque* $\psi_s^j \in \bigcup_{k\leq n} \mathcal{S}^k_{G_s^j}$.

REMARQUE 3.2 : *Pour tout but* G, $\neg\varphi_G$ *appartient à* $\mathcal{E}_G^1$.

De plus, si G *est un but ne contenant aucun littéral dans son corps,* φ_G *appartient à* $\mathcal{S}_G^1$.

DÉFINITION 3.3 : *On appelle* **classe des réponses exploratives de succès (resp. classe des réponses exploratives d'échecs)** *la réunion :*

$$\mathcal{S}_G = \bigcup_{n\in\omega} \mathcal{S}_G^n \,(\text{resp. } \mathcal{E}_G = \bigcup_{n\in\omega} \mathcal{E}_G^n)$$

On peut alors définir les formules de succès (resp. d'échecs) par le fait qu'elles caractérisent, dans chaque modèle de $\mathcal{T}$, des éléments acceptés par l'une des réponses exploratives de succès (resp. d'échecs).

Plus précisément :

DÉFINITION 3.4 : *On dit que la formule* $\psi(\tilde{x})$ *de* $\mathbb{L}$ **caractérise des succès (resp. des échecs)** *de* P *sur* G *si* $\mathcal{T} \models \psi \rightarrow \vee\mathcal{S}_G$ *(resp.* $\mathcal{T} \models \psi \rightarrow \vee\mathcal{E}_G$*).*

Nous désignerons par $SS_P(G)$ *(resp.* $FF_P(G)$*) la classe des formules caractérisant des succès (resp. des échecs) de* P *sur* G.

Remarques :

• Il est important de noter, dans la définition précédente, que le caractère d'agent d'exploration du programme apparaît dans la construction des classes $\mathcal{S}_G$ et $\mathcal{E}_G$, et donc à droite du signe d'inférence $\models$; alors que le caractère déclaratif de P, qui apparaîtra dans la définition 4.4, placera le programme à gauche du signe d'inférence.

• La définition, retenue ici, des réponses de succès ou d'échecs reste très générale et formelle, elle ne se préoccupe en aucun cas d'une exploration procédurale, par quelque dérivation que ce soit. D'autres classes de réponses exploratives plus procédurales sont données dans la partie 7 ; toutefois elles définissent évidemment les mêmes classes $SS_P(G)$ et $FF_P(G)$.

4. AXIOMATIQUE INTERNE D'UN PROGRAMME

Considérons le langage $\mathbb{L}_P$ étendu de $\mathbb{L}$ par les symboles relationnels $\Pi_P = \{R, S, \ldots\}$ propres au programme P, ainsi que par une copie $\Pi'_P = \{R', S', \ldots\}$ de ces prédicats programme.

Si D désigne une formule atomique construite sur le prédicat R de P, on notera D' la formule atomique de $\mathbb{L}_P$ obtenue par substitution de R' à R dans D.

DÉFINITION 4.1 : *A chaque but* $G(\tilde{x})$ *sont associées deux formules* $d(G)$ *et* $d'(G)$ *de* $\mathbb{L}_P$, *appelées les* **formules déclaratives de G.**

Pour le but : $G(\tilde{x}) :=\leftarrow \varphi \square A_1, \ldots, A_p, \neg B_1, \ldots, \neg B_q$

$d(G) := \varphi \wedge A_1 \wedge \ldots \wedge A_p \wedge B'_1 \wedge \ldots \wedge B'_q$

$d'(G) := \neg\varphi \vee A'_1 \vee \ldots \vee A'_p \vee B_1 \vee \ldots \vee B_q$

DÉFINITION 4.2 : *On appellera alors* **Axiomatique déclarative complète de P**, *qu'on notera* $\mathcal{D}(P)$, *la collection des formules,* $\mathcal{D}(R)$ *et* $\mathcal{D}'(R)$ *ci-dessous, associées à chaque prédicat programme* R *de* P :

$\mathcal{D}(R) := \tilde{\forall}((\vee_r \exists(d(G_r))) \rightarrow R(\tilde{x}))$

$\mathcal{D}'(R) := \tilde{\forall}((\wedge_r \forall(d'(G_r))) \rightarrow R'(\tilde{x}))$

dans lesquelles r *indexe les clauses de* P *de tête* $R(\tilde{x})$ et G_r *le corps correspondant.*

DÉFINITION 4.3 : *On appelle* **Axiomatique interne de P** *la classe notée* $\mathcal{I}(P)$ *des formules de* $\mathcal{T}$ *réunie à celle de l'axiomatique déclarative complète de* P, *on notera* $\mathcal{I}(P) = \mathcal{T} + \mathcal{D}(P)$.

DÉFINITION 4.4 : *On dit qu'une* **formule** λ **de** $\mathbb{L}$ **prouve G (resp.** $\neg G$**)** *par l'axiomatique interne de* P *si* $\mathcal{I}(P) \models \lambda \rightarrow d(G)$ $(resp.\ \mathcal{I}(P) \models \lambda \rightarrow d'(G))$.

Nous pouvons maintenant vérifier la correction des réponses de succès ou d'échecs pour l'axiomatique interne : si λ caractéristise des succès (resp. des échecs) (*voir* définition 3.4) de P sur G, alors λ prouve G (resp. $\neg G$) par l'axiomatique interne de P. Cela résultera évidemment de la proposition suivante :

PROPOSITION 4.5

a) Si $\lambda \in \mathcal{S}_G$ *alors* $\mathcal{D}(P) \models \lambda \rightarrow d(G)$

B) Si $\lambda \in \mathcal{E}_G$ *alors* $\mathcal{D}(P) \models \lambda \rightarrow d'(G)$.

∇

La démonstration se fait simultanément pour a) et b) par induction sur la construction des formules. On montre en fait que pour tout entier n, et quel que soit le but G, si $\lambda \in \mathcal{S}_G^n$ alors $\mathcal{D}(P) \models \lambda \rightarrow d(G)$ et si $\mu \in \mathcal{E}_G^n$ alors $\mathcal{D}(P) \models \mu \rightarrow d'(G)$.

C'est évident pour $n = 0$.

Le pas de récurrence nous conduit à supposer que λ est de la forme $\varphi \wedge \bigwedge_i (\bigvee_r \exists \psi_r^i) \wedge \bigwedge_j (\bigwedge_s \forall \phi_s^j)$.

Avec l'hypothèse de récurrence : $\mathcal{D}(P) \models \psi_r^i \rightarrow d(G_r^i)$ pour chaque i et pour chaque r, et $\mathcal{D}(P) \models \phi_s^j \rightarrow d'(G_s^j)$ pour chaque j et pous chaque s. Compte tenu que les implications $(\bigvee_r \exists d(G_r^i) \rightarrow A_i)$ et $(\bigwedge_s \forall d'(G_s^j) \rightarrow B_j')$ figurent dans $\mathcal{D}(P)$, il est aisé de conclure.

Δ

On peut se demander, à la vue de ce résultat, pourquoi alors choisir la définition 4.3 pour la sémantique déclarative, et ne pas se contenter de $\mathcal{D}(P)$ comme axiomatique déclarative. Ce serait oublier que la définition retenue ici pour les réponses d'exploration (*voir* définition 3.3) sont tout à fait formelles et ne se préoccupent pas d'optimisation dans la construction de ces réponses. Une classe de réponses révélant déjà plus un souci procédural, comme par exemple les classes $\mathcal{R}^+$ et $\mathcal{R}^-$ (de la partie 7), tiennent évidemment compte de la théorie $\mathcal{T}$. L'équivalence dans la caractérisation des succès ou des échecs, par l'une ou l'autre classe, s'obtient d'ailleurs modulo la théorie $\mathcal{T}$. La classe $\mathcal{R}$, par exemple (*voir* l'annexe), économise les explorations lorsqu'elles ne consistent qu'à rajouter des conditions à une contrainte déjà insatisfaisable dans $\mathcal{T}$.

C'est un des objectifs de ce travail que de distinguer ce qui est tributaire du cadre particulier des contraintes, de ce qui est général au principe même d'exploration par chaînage arrière des programmes généraux contraints : la Négation Constructive.

5. CLASSES DE RÉPONSES COMPLÈTES PAR CHAÎNAGE AVANT

Comme toujours, la complétude de la sémantique déclarative (déduction logique depuis l'axiomatique interne) pour la sémantique opérationnelle (exploration des classes $\mathcal{S}_G$ et $\mathcal{E}_G$) passera par la production de réponses générales, de succès ou d'échecs, par la notion de conséquences immédiates.

Il s'agira ici de la constitution pour chaque but $G(\tilde{x})$ des classes $\mathcal{C}_G$ et $\mathcal{C}_G'$ de formules de $\mathbb{L}$ sur les variables libres $\tilde{x}$. Fournie par le concept

habituel de conséquences immédiates, la construction de ces classes ne sera pas inductivement arborescente comme les classes $\mathcal{S}$ et $\mathcal{E}$ ou $\mathcal{R}^+$ et $\mathcal{R}^-$, mais séquentielles : $\mathcal{C}_{G(\tilde{x})}$ (resp. $\mathcal{C}'_{G(\tilde{x})}$) est en fait une suite $(\theta_n^G(\tilde{x}))_{n\in\omega}$ (resp. $\theta_n'^G(\tilde{x})$) de formules de $\mathbb{L}$.

On définit simultanément les formules θ_n^G, $\theta_n'^G$, θ_n^R, $\theta_n'^R$ pour chaque but G et pour chaque prédicat programme R.

Si $G(\tilde{x}) :=\leftarrow \varphi \,\square\, A_1, \ldots, A_p, \neg B_1, \ldots, \neg B_q$

1. $\theta_n^G := \varphi \wedge \theta_n^{A_1} \wedge \ldots \wedge \theta_n^{A_p} \wedge \theta_n'^{B_1} \wedge \ldots \wedge \theta_n'^{B_q}$
   $\theta_n'^G := \neg\varphi \vee \theta_n'^{A_1} \vee \ldots \vee \theta_n'^{A_p} \vee \theta_n^{B_1} \vee \ldots \vee \theta_n^{B_q}$
2. $\theta_{n+1}^R := \bigvee_r \exists\, \theta_n^{G_r}(\tilde{x})$
   $\theta_{n+1}'^R := \bigwedge_r \forall\, \theta_n'^{G_r}(\tilde{x})$

   dans lesquelles r indexe chaque clause de tête $R(\tilde{x})$ et G_r désigne le corps correspondant.
3. $\theta_0^R := \bigvee_k \exists\, \varphi_k$
   $\theta_0'^R := \bigwedge_l \forall \neg \varphi_l$

   dans lesquelles

   – k indexe chaque clause $R(\tilde{x}) \leftarrow \varphi_k \,\square$ de tête $R(\tilde{x})$ et sans aucun littéral dans le corps,

   – l indexe chaque clause de tête $R(\tilde{x})$ et φ_l désigne la contrainte correspondante.

REMARQUE 5.1 : *Il est aisé de constater que :*

$\theta_{n+1}^G := \varphi \wedge \bigwedge_i (\bigvee_r \exists\, \theta_n^{G_r^i}) \wedge \bigwedge_j (\bigwedge_s \forall\, \theta_n'^{G_s^j})$

$\theta_{n+1}'^G := \neg\varphi \vee \bigvee_i (\bigwedge_r \forall\, \theta_n'^{G_r^i}) \vee \bigvee_j (\bigvee_s \exists\, \theta_n^{G_s^j})$

*avec les indexations et quantifications habituelles (*voir *définition 3.1)*

PROPOSITION 5.2 : *Pour chaque but G et pour chaque entier n, les formules ci-dessous sont des tautologies du calcul des prédicats :*

$$\theta_n^G \rightarrow \theta_{n+1}^G \qquad et \qquad \theta_n'^G \rightarrow \theta_{n+1}'^G$$

$\triangledown$

D'après la remarque 5.1 ci-dessus, c'est évident pour le pas d'induction n, et c'est une simple vérification pour $n = 0$.

$\triangle$

PROPOSITION 5.3 : *Pour chaque but $G(\tilde{x})$ et pour chaque entier n,*

a) θ_n^G est équivalente à une formule de $\mathcal{S}_G$

b) $\theta_n'^G$ *est équivalente à une formule de* $\mathcal{E}_G$

∇

La démonstration simultanée de a) et b) se fait par induction sur n.

Le passage de n à $n+1$ est évident par la remarque 5.1.

Pour $n = 0$, en définissant ψ_r^i par $\varphi_{G_r^i}$ si G_r^i ne contient aucun littéral et par $\bot$ dans le cas contraire, et en définissant ϕ_s^j par $\neg\,\varphi_{G_s^j}$, la formule $\varphi \wedge \bigwedge_i (\bigvee_r \exists\, \psi_r^i) \wedge \bigwedge_j (\bigwedge_s \forall\, \phi_s^j)$ appartient à $\mathcal{S}_G^2$ d'après la remarque 3.2 et elle est évidemment équivalente à θ_0^G.

De la même façon, en définissant ϕ_r^i par $\neg\,\varphi_{G_r^i}$ et ψ_s^j par $\varphi_{G_s^j}$ si G_s^j ne contient aucun littéral et par $\bot$ dans le cas contraire, la formule $\neg\,\varphi \vee \bigvee_i (\bigwedge_r \forall\, \phi_r^i) \vee \bigvee_j (\bigvee_s \exists\, \psi_s^j)$ appartient à $\mathcal{E}_G^2$ d'après la remarque 3.2 et elle est évidemment équivalente à $\theta_0'^G$.

$\triangle$

6. COMPLÉTUDE EN LOGIQUE BIVALUÉE

Nous nous proposons dans cette partie de montrer qu'une formule λ, du langage de contraintes $\mathbb{L}$, qui prouve un but G (resp. $\neg\, G$) par l'axiomatique interne de P, caractérise des succès (resp. des échecs) de P sur G.

Plus précisément, cela résultera de la proposition 5.3 et de la proposition suivante :

PROPOSITION 6.1

a) Si $\mathcal{I}(P) \models \lambda(\tilde{x}) \rightarrow d(G(\tilde{x}))$ *alors* $\mathcal{T} \models \lambda(\tilde{x}) \rightarrow \bigvee(\mathcal{C}_G)$

b) Si $\mathcal{I}(P) \models \lambda(\tilde{x}) \rightarrow d'(G(\tilde{x}))$ *alors* $\mathcal{T} \models \lambda(\tilde{x}) \rightarrow \bigvee(\mathcal{C}'_G)$

Les démonstrations de a) et b) sont identiques et tiennent essentiellement à la proposition 6.5 ci-dessous. Nous montrerons a), laissant l'adaptation de b) au lecteur.

Nous nous proposons donc, disposant d'une structure $\mathcal{M}$ modèle de $\mathcal{T}$, et d'une suite $\tilde{\alpha}$ d'éléments de la base $|\mathcal{M}|$ de $\mathcal{M}$ satisfaisant λ (*i.e.* : $\mathcal{M} \models \lambda(\tilde{\alpha})$), de montrer que $\tilde{\alpha}$ satisfait l'une des formules θ de $\mathcal{C}_G$ (*i.e.* : $\mathcal{M} \models \theta(\tilde{\alpha})$) sous l'hypothèse que $\mathcal{I}(P) \models \lambda(\tilde{x}) \rightarrow d(G(\tilde{x}))$.

Dans le langage étendu $\mathbb{L} + (\tilde{\alpha})$, extension de $\mathbb{L}$ par les constantes $\tilde{\alpha}$, nous disposons donc d'un modèle $\widehat{\mathcal{M}} = \mathcal{M} + (\tilde{\alpha})$ de $\mathcal{T} + \lambda(\tilde{\alpha})$, et nous souhaitons conclure que $\widehat{\mathcal{M}} \models \theta(\tilde{\alpha})$ pour une formule θ de $\mathcal{C}_G$.

C'est l'extension de $\widehat{\mathcal{M}}$ à un modèle $\mathcal{N}$ de $\mathcal{I}(P) + \lambda(\tilde{\alpha})$ qui fournira la réponse. Ce modèle $\mathcal{N}$ sera une réunion d'ultrapuissances (Chang-Keisler [7]) emboîtées $\mathcal{N}_n$ de réalisations $\mathcal{M}_n$ de $\mathbb{L}_P + (\tilde{\alpha})$.

Définissons tout d'abord cette famille $\mathcal{M}_n$: $\mathcal{M}_n$ est définie par $\widehat{\mathcal{M}}$ en tant que réalisation de $\mathbb{L} + (\tilde{\alpha})$, et par $\mathcal{M}_n \models R(\tilde{a})$ si et seulement si $\widehat{\mathcal{M}} \models \theta_n^R(\tilde{a})$ et $\mathcal{M}_n \models R'(\tilde{a})$ si et seulement si $\widehat{\mathcal{M}} \models \theta_n'^R(\tilde{a})$ pour les interprétations de R et R' pour chaque prédicat programme R de P.

Il résulte alors de cette définition et de la définition des formules θ et θ', la proposition suivante :

PROPOSITION 6.2 : *Pour chaque but $G(\tilde{x})$, pour chaque entier n et pour chaque suite $\tilde{b}$ de $|\mathcal{M}|$, on a $\mathcal{M}_n \models d(G)(\tilde{b})$ si et seulement si $\mathcal{M} \models \theta_n^G(\tilde{b})$ et $\mathcal{M}_n \models d'(G)(\tilde{b})$ si et seulement si $\mathcal{M} \models \theta_n'^G(\tilde{b})$.*

En appelant **formules $\mathbb{L}$-positives** les formules de $\mathbb{L}_P + (\tilde{\alpha})$ définies inductivement par :

1) Chaque formule de $\mathbb{L} + (\tilde{\alpha})$ est $\mathbb{L}$-positive.

2) Chaque formule atomique de $\mathbb{L}_P + (\tilde{\alpha})$ est $\mathbb{L}$-positive.

3) Si ϕ_1 et ϕ_2 sont des formules $\mathbb{L}$-positives, alors $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\exists z\, \phi_1$ et $\forall z\, \phi_2$ le sont aussi.

On vérifie la proposition suivante :

PROPOSITION 6.3 : *$(\mathcal{M}_n)_{n\in\omega}$ est une famille emboîtée pour les formules $\mathbb{L}$-positives. Plus précisément : si $\phi(\tilde{x})$ est une formule $\mathbb{L}$-positive et si $\mathcal{M}_n \models \phi(\tilde{a})$ alors $\mathcal{M}_{n+1} \models \phi(\tilde{a})$*

∇

C'est une vérification aisée compte tenu de la proposition 5.2 et du fait que $\mathcal{M}_n$ et $\mathcal{M}_{n+1}$ sont sur la même base $|\mathcal{M}|$.

Δ

Considérons alors un ultrafiltre de Fréchet D (*i.e.* : un ultrafiltre sur ω contenant les segments cofinaux $[n, +\infty[$) et définissons les réalisations $\mathcal{N}_n$ de $\mathbb{L}_P + (\tilde{\alpha})$ comme les D-ultrapuissances de $\mathcal{M}_n$, en particulier la base de $\mathcal{N}_n$ est le quotient de $|\mathcal{M}|^\omega$ par la relation d'équivalence $((b_i) \equiv (c_i)$ si et seulement si $\{i | b_i = c_i\} \in \mathrm{D})$.

PROPOSITION 6.4 : *$(\mathcal{N}_n)_{n\in\omega}$ est une famille emboîtée pour les formules $\mathbb{L}$-positives.*

∇

C'est un résultat général puisque les $\mathcal{N}_n$ sont des puissances réduites des $\mathcal{M}_n$ qui sont emboîtés eux-mêmes pour les formules $\mathbb{L}$-positives.

Si, en effet, $\mathcal{N}_n \models \phi(\widetilde{(a_i)})$ c'est que $\{i|\mathcal{M}_n \models \phi(\tilde{a}_i)\} \in \mathrm{D}$, or par la proposition 6.3, $\{i|\mathcal{M}_n \models \phi(\tilde{a}_i)\} \subset \{i|\mathcal{M}_{n+1} \models \phi(\tilde{a}_i)\}$ donc $\mathcal{N}_{n+1} \models \phi(\widetilde{(a_i)})$

△

Cette proposition nous permet alors de définir la réalisation $\mathcal{N}$ comme la réunion des réalisations $\mathcal{N}_n$. Une telle réunion va bénéficier de la propriété de cohérence ci-dessous (proposition 6.5), qui en fera un modèle de $\mathcal{I}(P) + \lambda(\tilde{\alpha})$; ce qui ne serait pas le cas, en général, de la réunion des $\mathcal{M}_n$ comme le montre l'exemple ci-dessous :

Exemple : Dans la structure $\mathcal{M}$ des entiers positifs ou nuls avec l'addition, avec le programme suivant :

$$P = \begin{cases} p(x) \leftarrow x = y + 1 \,\square\, p(y) \\ q(x) \leftarrow x = 0 \,\square\, p(y) \end{cases}$$

PROPOSITION 6.5 (Cohérence de la famille $(\mathcal{N}_n)_{n\in\omega}$) : *Pour chaque formule $\mathbb{L}$-positive $\phi(\tilde{x})$, on a $\mathcal{N} \models \phi(\tilde{a})$ si et seulement s'il existe un entier n_0 pour lequel $\mathcal{N}_{n_0} \models \phi(\tilde{a})$*

▽

La démonstration de cette équivalence se fait évidemment par induction sur la définition des formules $\mathbb{L}$-positives. Le seul point délicat est celui du pas d'induction par quantification universelle pour la condition nécessaire. Nous nous limiterons à ce cas et montrerons la contraposée.

Supposons donc que pour chaque entier n, il existe $b^n = (b_i^n)$ pour lequel $\mathcal{N}_n \models \neg\phi(\tilde{a}, b^n)$, ce qui revient à dire que pour chaque entier n, $B_n = \{i|\, \mathcal{M}_n \models \neg\phi(\tilde{a}_i, b_i^n)\}$ appartient à l'ultrafiltre D.

Définissons alors, pour chaque i, l'entier $\nu(i)$ de la manière suivante :

– s'il existe un entier $k \leq i$ tel que $\mathcal{M}_k \models \neg\phi(\tilde{a}_i, b_i^k)$ alors $\nu(i)$ est le plus grand de ces entiers k.

– sinon $\nu(i) = 0$.

On remarque alors que, par ce choix, si $i \in [n, +\infty[\cap B_n$ alors $\mathcal{M}_{\nu(i)} \not\models \phi(\tilde{a}_i, b_i^{\nu(i)})$, et de plus, comme $\nu(i) \geq n$, on a $\mathcal{M}_n \not\models \phi(\tilde{a}_i, b_i^{\nu(i)})$ d'après la proposition 6.3 (puisque ϕ est une formule $\mathbb{L}$-positive).

Ainsi $\{i|\, \mathcal{M}_n \models \neg\phi(\tilde{a}_i, b_i^{\nu(i)})\}$ appartient à son tour à l'ultrafiltre de Fréchet D, d'où pour chaque n, on a $\mathcal{N}_n \models \neg\phi(\tilde{a}, b)$ en posant $b = (b_i^{\nu(i)})$.

D'où finalement : $\mathcal{N} \not\models \phi(\tilde{a}, b)$, par hypothèse d'induction sur la construction des formules $\mathbb{L}$-positives.

△

COROLLAIRE 6.6 : *$\mathcal{N}$ est un modèle de $\mathcal{I}(P) + \lambda(\tilde{\alpha})$*

∇

Il est clair que $\mathcal{N}$ satisfait $\mathcal{T} + \lambda(\tilde{\alpha})$ puisque chaque $\mathcal{N}_n$ est élémentairement équivalent à $\mathcal{M}_n$ qui, pour $\mathbb{L} + (\tilde{\alpha})$, est la même interprétation que $\widehat{\mathcal{M}}$.

Considérons donc dans $\mathcal{I}(P)$ les axiomes relatifs aux prédicats programme R et R'. Il s'agit de $\tilde{\forall}(\bigvee_r (\exists d(G_r)) \rightarrow R(\tilde{x}))$ et $\tilde{\forall}(\bigwedge_r (\forall d'(G_r)) \rightarrow R'(\tilde{x}))$.

La démonstration va être identique pour l'un et pour l'autre.

Soit $\tilde{a} = (\widetilde{(a_i)})$ fixé dans $|\mathcal{N}|$ et supposons que $\mathcal{N} \models \bigvee_r (\exists d(G_r))(\tilde{a})$, d'après la proposition 6.5 précédente il existe un entier n_0 pour lequel $\{i|\mathcal{M}_{n_0} \models \bigvee_r (\exists d(G_r))(\tilde{a}_i)\} \in D$.

Ceci ne signifie pas autre chose, d'après la proposition 6.2, que l'appartenance à D de $\{i|\mathcal{M} \models \bigvee_r (\exists \theta^{G_r}_{n_0})(\tilde{a}_i)\}$.

Ceci implique, par la définition des formules θ, que $\{i|\mathcal{M} \models \theta^R_{n_0+1}(\tilde{a}_i)\}$ appartient à D,

qui implique, par définition des réalisations $\mathcal{M}_n$, que $\{i|\mathcal{M}_{n_0+1} \models R(\tilde{a}_i)\} \in D$, qui n'est rien d'autre que $\mathcal{N}_{n_0+1} \models R(\widetilde{(a_i)})$.

Donc, par définition de $\mathcal{N}$, $\mathcal{N} \models R(\tilde{a})$.

$\triangle$

COROLLAIRE 6.7 : *Pour chaque but $G(x_1, \ldots, x_k)$ et chaque suite $\tilde{a} = (a_1, \ldots, a_k)$ d'éléments de la base $|\mathcal{M}|$, en notant $\widetilde{(a_i)} = ((a^i_1), (a^i_2), \ldots, (a^i_k))$ l'élément correspondant de la base $|\mathcal{N}|$* (i.e. *: défini par $a^i_j = a_j$), on a :*

$\mathcal{N} \models d(G)(\widetilde{(a_i)})$ si et seulement s'il existe un entier n_0 pour lequel $\mathcal{M} \models \theta^G_{n_0}(\tilde{a})$.

∇

On sait, en effet, que $\mathcal{N}$ satisfait $d(G)(\widetilde{(a_i)})$ si et seulement s'il existe un entier n_0 pour lequel $\mathcal{N}_{n_0} \models d(G)(\widetilde{(a_i)})$ par la proposition 6.5 (puisque les formules $d(G)$ sont $\mathbb{L}$-positives).

Donc $\mathcal{N} \models d(G)(\widetilde{(a_i)})$ si et seulement s'il existe un entier n_0 tel que $\{i|\mathcal{M}_{n_0} \models d(G)(\widetilde{(a_i)})\} \in D$, qui dans ce cas et équivalent à $\mathcal{M} \models \theta^G_{n_0}(\tilde{a})$, par la proposition 6.2.

$\triangle$

Ces corollaires nous permettent alors de conclure la proposition 6.1, puisque de l'hypothèse $\mathcal{I}(P) \models \lambda(\tilde{x}) \rightarrow d(G(\tilde{x}))$, il résulte du corollaire 6.6 que $\mathcal{N} \models d(G)(\tilde{\alpha})$ et, du corollaire 6.7, qu'il existe une formule θ de $\mathcal{C}_G$ telle que $\mathcal{M} \models \theta(\tilde{\alpha})$.

Ce qui démontre donc la complétude.

7. ANNEXE

Nous proposons dans cette partie une version plus opérationnelle des classes de réponses, évitant par exemple des explorations inutiles. Notamment lorsqu'il s'agit d'attendre la réponse d'une exploration pour la rajouter par conjonction à une formule que l'on sait déjà insatisfaisable.

Pour cela, considérons l'arbre $\mathcal{A}(G_0)$ des descendants itérés du but G_0.

On appelle **hérédité** d'un but G, nœud de cet arbre, la conjonction Γ_G des contraintes de ses ancêtres.

On appelle alors **arbre d'exploration** de G_0, que l'on note $\mathcal{A}_E(G_0)$, le sous-arbre de $\mathcal{A}(G_0)$ des nœuds d'hérédité satisfaisable dans $\mathcal{T}$ (*i.e.* : $\mathcal{T} \models \tilde{\exists}\Gamma_G$).

Nous allons pouvoir définir, pour chaque nœud de l'arbre $\mathcal{A}_E(G_0)$, ses **réponses de succès** et ses **réponses d'échecs**, à l'aide des réponses de ses descendants immédiats dans cet arbre :

Soit le but $G :=\leftarrow \varphi \square L_1, \ldots, L_p, L_{p+1}, \ldots, L_{p+q}$

où $L_i = A_i$ pour $i \leq p$ et $L_{p+j} = \neg B_j$ pour $j \leq q$

1. Si G est une feuille, sa seule réponse de succès est φ et sa seule réponse d'échec est $\neg\varphi$.

2. Sinon, nous allons introduire et utiliser les notions de pré-succès et de pré-échecs d'un littéral :

(a) (i) Les **réponses de succès de G** sont les formules de la forme :

$$\varphi_G \wedge \pi_1 \wedge \ldots \wedge \pi_{p+q}$$

où chaque π_k est un pré-succès de L_k.

(ii) Les **réponses d'échecs de G** sont les formules de la forme :

$$\neg\varphi_G \vee \pi_{k_1} \vee \ldots \vee \pi_{k_l}$$

où chaque π_{k_i} est un pré-échec de L_{k_i} et où $1 \leq k_1 < k_2 < \ldots < k_l \leq p+q$.

(b) Les pré-succès et pré-échecs de L ($L = A$ ou $L = \neg B$).

(i) Les **pré-échecs de A** (resp. **pré-succès de $\neg$ B**) sont les formules :

$$\pi(\tilde{x}) = \forall\pi_{t_1} \wedge \forall\pi_{t_2} \wedge \ldots \wedge \forall\pi_{t_m}$$

où les indices $t_1, \ldots, t_m$ dénotent *toutes* les clauses $A \leftarrow G_{t_i}$ (resp. $B \leftarrow G_{t_i}$) et π_{t_i} une réponse d'échec de G_{t_i}.

(ii) Les **pré-succès de A** (resp. **pré-échecs de** $\neg$ **B**) sont les formules :

$$\pi(\tilde{x}) = \exists \pi_{r_1} \vee \exists \pi_{r_2} \vee \ldots \vee \exists \pi_{r_n}$$

où les indices $r_1, \ldots, r_n$ dénotent *des* clauses distinctes $A \leftarrow G_{r_j}$ (resp. $B \leftarrow G_{r_j}$) et π_{r_j} une réponse de succès de G_{r_j}.

En désignant par $\mathcal{R}^+_{G_0}$ (resp. $\mathcal{R}^-_{G_0}$) les réponses de succès (resp. d'échecs) de G_0, on peut vérifier que cette classe est équivalente modulo $\mathcal{T}$ à $\mathcal{S}_{G_0}$ (resp. $\mathcal{E}_{G_0}$).

8. CONCLUSION

C'est P. J. Stuckey [20] qui situa combien le principe de négation constructive prenait naturellement sa place dans la programmation par contraintes. C'est dans ce cadre-là que nous avons voulu décrire une caractérisation de la classe des contraintes qui particularisent des éléments de réponses de succès ou d'échecs (*cf.* Définition 3.4). Les réponses elles-mêmes dépendent alors de la procédure choisie pour explorer cette classe. Ainsi les SLD$^\forall$-dérivations de P. Bruscoli *et al.* [4], les SLD-CNF-dérivations de D. Chan [6] et les SLD-dérivations concurrentes par élagage de F. Fages [11] ne se correspondent pas biunivoquement, mais les réponses de chacune des procédures finissent par se retrouver par leur sémantique logique trivaluée de Kunen-Fitting. Les réponses de ces diverses sémantiques opérationnelles finissent aussi par recouvrir la classe des réponses que nous définissons, même si notre travail utilise la logique bivaluée. En effet, notre choix pour une sémantique logique a été de privilégier la logique bivaluée classique. L'axiomatique que nous choisissons d'associer à un programme ne s'appuie pas sur la traditionnelle complétion de Clark, mais sur l'usage, aussi traditionnel ([19], [9], [14], [15], [4], [10]), du dédoublement des prédicats. Nous utilisons cette transformation par dédoublement uniquement pour l'axiomatique déclarative $\mathcal{D}(P)$ d'un programme général P (*cf.* Définition 4.2) et non pour modifier le programme lui-même, comme le font par exemple P. Bruscoli *et al.* [4] pour la sémantique opérationnelle de la négation intensionnelle. Par ce biais, la construction des réponses reste donc proche de la SLD-résolution, tout comme le fait aussi F. Fages [11].

Ce dédoublement des prédicats du programme permet dans une large mesure, ainsi que l'a remarqué W. Drabent [10], de ramener la logique

trivaluée à la logique bivaluée; dans une large mesure seulement, car le dédoublement des prédicats pour une classe $\mathcal{E}$ de formules transformée en $\mathcal{D}(\mathcal{E})$ (lorsque la syntaxe le permet, c'est-à-dire lorsque les formules sont construites sans équivalence et avec éventuellement des négations ne figurant que dans les littéraux) ne satisfait pas, quelle que soit la formule φ, l'équivalence entre $\mathcal{E} \models_3 \varphi$ et $\mathcal{D}(\mathcal{E}) \models \mathcal{D}(\varphi)$. Comme l'a montré A. Brieu [3], on a par contre l'équivalence dans le cas particulier où $\mathcal{E}$ est le complété de Clark d'un programme, grâce à l'existence d'un plus petit modèle partiel.

Il n'existe pas une seule manière de dédoubler un programme, mais plusieurs. Par exemple, W. Drabent propose dans [10] un dédoublement $comp(split(P))$ qui n'est pas équivalent à $\mathcal{D}(P)$, comme on peut le vérifier sur les programmes $P_1 = \{p \leftarrow \neg q,\ q \leftarrow q\}$ et $P_2 = P_1 \cup \{p \leftarrow q\}$; malheureusement, on n'a pas non plus l'équivalence entre $comp(split(P)) \models split(\varphi)$ et $\mathcal{D}(P) \models \mathcal{D}(\varphi)$. Le sens donné aux symboles de prédicat dédoublés n'est en effet pas le même dans le travail de W. Drabent et dans le notre. C'est leur adéquation sur les littéraux qui fait l'équivalence avec la sémantique logique trivaluée, par l'équivalence de $comp(split(P)) \models p$ (resp. $\neg p'$) et $\mathcal{D}(P) \models p$ (resp. p').

Outre un travail d'analyse des diverses formes de dédoublement que l'on pourrait choisir, il serait intéressant d'étudier comment la vision unifiée des classes de réponses que nous introduisons (traitement symétrique des réponses de succès et d'échecs) peut donner naissance à d'autres sémantiques opérationnelles, qui puissent se comparer à celles existantes dans le domaine de la programmation par contraintes. Le paragraphe 7 en est une approche que nous espérons développer.

REMERCIEMENTS

Les auteurs souhaitent remercier les rapporteurs anonymes qui, par leurs commentaires et suggestions, nous ont permis de produire une version finale qui nous paraît plus satisfaisante.

BIBLIOGRAPHIE

1. K. R. Apt, Introduction to Logic Programming, Handbook of Theoretical Computer Science (J. Van Leeuwen), North Holland, 1990.
2. N. Bleuzen-Guernalec et B. Gil, Internal Negation in Logic Programming, Rapport Technique, U.A.225, Université d'Aix-Marseille-II, 1989.

3. A. Brieu, Négation en Programmation par Contraintes, *Thèse de Doctorat* (en préparation), Institut de Mathématiques de Luminy, Université de la Méditerranée.
4. P. Bruscoli, F. Levi, G. Levi et M. C. Meo, Compilative Constructive Negation in Constraint Logic Programs, *CAAP'94*, 19th International Colloquium, Edinburg, April 11-14, in Proceedings Stison, Springer-Verlag, 1994, p. 52-57.
5. D. Chan, Constructive Negation Based on the Completed Database, *Technical Report*, ECRC, 1988.
6. D. Chan, Constructive Negation based on the Completed Database, in Proceedings of the fifth International Conference on Logic Programming, R. A. Kowalski and K. A. Bowen eds., *MIT Press*, Cambridge, 1988, p. 111-125.
7. C. C. Chang and H. G. Keisler, Model Theory, 3th Edition, Studies in Logic, North-Holland, Amsterdam, 1990.
8. K. L. Clark, Negation as Failure, in Logic and Data Bases, H. Gallaire et J. Minkers eds., Plenum Press, New York, 1978, p. 293-322.
9. J.-P. Delahaye, Sémantiques Logiques et Dénotationnelles des Interpréteurs PROLOG, in *Theoretical Informatics and Applications*, 1988, vol. *22*, p. 3-42.
10. W. Drabent, Completeness of SLDNF-resolution for non-floundering queries, in *Journal of Logic Programming*, 1996, vol. *27*, n° 2, p. 89-106.
11. F. Fages, Constructive Negation by Pruning, in *Journal of Logic Programming*, 1997, vol. *32*, n° *2*, p. 85-118.
12. F. Fages, Constructive negation by pruning and optimization higher-order predicates, in *CLP and CC languages*, In M. Nivat et A. Podelski eds., *Constraints: Basics and Trends*, Lecture Notes in Computer Science, n° 910, Springer-Verlag, 1995, p. 68-89.
13. M. Fitting, A KRIPKE-KLEENE Semantics for Logic Programs, in *Journal of Logic Programming*, 1985, vol. *2*, n° 4, p. 295-312.
14. B. Gil, Intériorisation de la Négation en Programmation Logique, *Thèse*, Université d'Aix-Marseille-II, 1991.
15. B. Gil, Complete Extension of General Logic Programs, *Theoretical Computer Science 94*, Elsevier, 1992, p. 281-294.
16. H. J. Keisler, Model Theory for Infinitary Logic (Logic with Countable Conjonctions and Finite Quantifiers), Studies in Logic and The Foundations of Mathematics, vol. *62*, North-Holland Publishing Company, Amsterdam-London, 1971.
17. K. Kunen, Negation in Logic Programming, in *Journal of Logic Programming*, 1987, n° 4, p. 289-308.
18. K. Kunen, Signed Data Dependencies in Logic Programs, in *Journal of Logic Programming*, 1989, n° 7, p. 231-245.
19. B. Meltzer, Theorem-proving for Computers: Some Results on Resolution and Renaming, in *Automation of Reasoning*, J. Siekmann et G. Wrightson eds., Springer-Verlag, Berlin, 1983, vol. *1*, p. 493-495. (Première parution de l'article in *Computing Journal 8*, 1966, p. 341-343.)
20. P. J. Stuckey, Constructive Negation for Constraint Logic Programming, in *Proceedings LICS*, Amsterdam, 1991, p. 328-335.