

JEAN-PAUL DELAHAYE

Sémantique logique et dénotationnelle des interpréteurs PROLOG

Informatique théorique et applications, tome 22, n° 1 (1988), p. 3-42

http://www.numdam.org/item?id=ITA_1988__22_1_3_0

© AFCET, 1988, tous droits réservés.

L'accès aux archives de la revue « Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

SÉMANTIQUE LOGIQUE ET DÉNOTATIONNELLE DES INTERPRÉTEURS PROLOG (*)

par Jean-Paul DELAHAYE (1)

Communiqué par A. ARNOLD

Résumé. – Une méthode générale est introduite dans le but d'obtenir la sémantique logique (en termes de systèmes d'axiomes) et la sémantique dénotationnelle (en termes de points fixes) des programmes PROLOG. L'idée principale est analogue à celle de Van Emden, Kowalski et Apt, mais cette fois-ci permet la prise en compte des interpréteurs standards (non équitables et en profondeur d'abord). Notre conclusion est que pour un programme PROLOG il y a au moins cinq systèmes d'axiomes à envisager, chacun donnant un sens logique au programme. Le dernier de ces systèmes est celui correspondant à un interpréteur standard, il est un peu compliqué mais c'est lui qui en général donne le mieux le sens logique d'un programme.

Abstract. – A general method is introduced for obtaining the logical semantics (sets of axioms) and the denotational semantics (operator and fixed points) of PROLOG programs. The main idea is analogous to the idea of Van Emden, Kowalski and Apt, but now we consider standard interpreters (which are not fair and are depth first). Our conclusion is that for one PROLOG program they are at least five associated sets of axioms, each of them giving a logical meaning to the program. Our last set of axioms gives the meaning of the program when it is used with a standard interpreter. This set of axioms is rather complicated but in general it is better than the others for obtaining the logical semantic of a real program.

1. INTRODUCTION

Van Emden, Kowalski et Apt [Apt 82, VEK 76] ont proposé une sémantique dénotationnelle (par point fixe) et une sémantique logique (par modèle de

(*) Reçu février 1986, révisé janvier 1987.

(1) L.I.F.L., U.A.-C.N.R.S. n° 369, Université des Sciences et Techniques de Lille, 59655 Villeneuve-d'Ascq Cedex.

Ce travail a été réalisé dans le cadre du projet METHEOL du « Programme de Recherches Coordonnées de Programmation ».

système d'axiomes) des programmes de programmation logique. Ces sémantiques ont été utilisées en particulier pour établir la correction (soundness) et la complétude de la *SLD*-résolution vis-à-vis de l'ensemble des succès et de l'ensemble des échecs finis d'un interpréteur ne considérant que des arbres *SLD* équitables («fair» voir [Las 84]) et les explorant «en largeur d'abord».

Leur approche (que nous redétaillons aux paragraphes 4 et 5) ne s'applique pas vraiment aux interpréteurs PROLOG «standards», car ceux-ci, s'ils explorent bien des arbres *SLD*, n'envisagent pas seulement des arbres équitables et les explorent «en profondeur d'abord».

La situation est donc qu'on ne sait pas caractériser par un système d'axiomes du calcul des prédicats, l'ensemble des succès et l'ensemble des échecs finis d'un interpréteur PROLOG standard.

Lorsqu'un programme PROLOG est donné, il lui est associé plusieurs systèmes d'axiomes du calcul des prédicats qui sont :

- son interprétation clausale que nous noterons $A x_\alpha$
- sa version SI, notée $A x_\beta$ [Apt 82], équivalente à $A x_\alpha$.
- sa version SSI, notée $A x_\gamma$ [Apt 82, Cla 78].

L'ensemble des succès d'un interpréteur PROLOG «en largeur d'abord et équitable», que nous noterons SS est le plus petit modèle de $A x_\alpha$ (donc de $A x_\beta$) et de $A x_\gamma$ [Apt 82, VEK 76].

L'ensemble des échecs finis d'un interpréteur PROLOG «en largeur d'abord et équitable», que nous noterons FF est inclus dans le complémentaire du plus grand modèle de $A x_\gamma$ [Apt 82, Llo 84] et est égal à ce complémentaire lorsque l'opérateur T_β associé au programme vérifie une certaine condition [Apt 82, Bla 82 a]. Ces résultats, rappelés plus loin, élucident complètement les rapports entre le sens logique du programme et le comportement d'un interpréteur «en largeur d'abord et équitable».

Pour obtenir le sens logique du programme lorsque l'interpréteur est un interpréteur PROLOG standard (c'est-à-dire explorant «en profondeur d'abord» l'arbre *SLD* déterminé par la règle de choix du but le plus à gauche) nous introduisons un système d'axiomes notés $A x_\theta$ plus complexe que les précédents et dédoublant chaque symbole de prédicats du programme.

Ce système correspond à ce que nous appelons l'hypothèse «interpréteur standard». Il donne une sémantique logique du programme qui coïncide le plus souvent avec la sémantique opérationnelle d'un interpréteur standard. Malheureusement et même si cette mise en coïncidence se produit dans de nombreux cas où elle n'a pas lieu avec les systèmes d'axiomes habituels, nous

n'avons pas réussi à obtenir un système d'axiomes qui fournissent à tout coup le sens logique d'un programme manipulé par un interpréteur standard.

La méthode du dédoublement des symboles de prédicat (qui est à mettre en parallèle avec l'utilisation de logiques trivaluées [Fit 85 b]) permet à chaque fois d'obtenir le pendant dénotationnel des résultats. Dans les bons cas nous obtenons donc l'expression en termes de points fixes non seulement de *SS* et de *FF* (caractéristiques des interpréteurs en largeur d'abord et équitables), mais aussi de *FFP* et *SSP* (caractéristiques des interpréteurs standards).

L'idée de dédoubler chaque prédicat pour trouver l'interprétation logique des programmes de programmation logique nous semble pouvoir s'appliquer à toutes sortes d'interpréteurs et, en particulier, à des interpréteurs parallèles ou à des interpréteurs incluant des moyens de contrôle particulier comme par exemple le «cut» [Del 87].

La sémantique du langage PROLOG a été envisagée sous des aspects différents de ceux considérés ici dans les travaux [Col 82, Fit 85 a, Fit 85 b, Hil 74, Jon 84, Der 83, VEA 85].

2. PRÉLIMINAIRES LOGIQUES

Nous commençons par définir formellement les notions de programme PROLOG et de système d'axiomes du calcul des prédicats. Nous utilisons volontairement des notations différentes pour les deux notions car, comme nous le verrons plus loin, à un programme PROLOG correspondent naturellement plusieurs systèmes d'axiomes du calcul des prédicats.

DÉFINITIONS : Un *programme PROLOG* est une suite finie de règles PROLOG.

Une *règle PROLOG* est une suite finie de la forme :

$$A \leftarrow B_1 B_2 \dots B_n, \quad n \geq 0$$

où \leftarrow est un symbole réservé et A, B_1, B_2, \dots, B_n sont des formules atomiques (ou atomes).

Un *atome* est une formule de la forme :

$$P(t_1, t_2, \dots, t_m)$$

où P est un symbole de prédicat et où t_1, t_2, \dots, t_m sont des termes.

Un *terme* est

– soit une constante

- soit une variable
- soit une formule de la forme :

$$f(t_1, t_2, \dots, t_k)$$

où f est un symbole de fonction et où t_1, t_2, \dots, t_k sont des termes.

Les *symboles de prédicats*, les *symboles de fonctions*, les *symboles de constantes* (qu'on peut considérer comme des symboles de fonctions à 0 place) et les *symboles de variables* sont des ensembles fixés, infinis et disjoints deux à deux.

Dans toute la suite nous supposons donné un programme PROLOG P et nous ferons les hypothèses que (voir [Apt 82, Clo 78]) :

- tout symbole de prédicat de P apparaît au moins une fois en tête d'une règle (c'est-à-dire comme symbole de prédicat de l'atome situé devant le symbole \leftarrow).
- un symbole de constante au moins, est utilisé dans P .

La définition que nous donnons correspond à ce qui est généralement appelé Programme en « PROLOG pur ». En particulier, nous n'autorisons pas l'utilisation du cut, ou de prédicats évaluables, mais quelques complications permettraient leur prise en compte.

Exemple : Pour illustrer les définitions et résultats que nous allons donner, nous utiliserons le programme P suivant :

$$P \left\{ \begin{array}{l} A(x) \leftarrow B(x) C(x) \\ B(x) \leftarrow B(f(x)) \\ B(f(a)) \leftarrow \\ C(a) \leftarrow \end{array} \right.$$

DÉFINITION : Un *système d'axiomes du calcul des prédicats* est un ensemble de formules du calcul des prédicats.

Une *formule du calcul des prédicats* est soit un atome, soit une formule de l'une des formes suivantes :

$$\neg A, (A_1 \vee \dots \vee A_n), (A_1, \wedge \dots \wedge A_n), (A \leftarrow B), (A \Leftrightarrow B) \\ \forall x A, \quad \exists x A$$

où

- $\neg, \vee, \wedge, \leftarrow, \Leftrightarrow, \forall, \exists$, sont des symboles réservés;
- A, A_i, B, B_i des formules du calcul des prédicats;

– x une variable.

Nous utiliserons sans les redéfinir (*voir* [Del 84 a]) les notions de formules closes, variables libres, variables liées, plus grand unificateur.

Dans l'écriture des formules nous enlèverons certaines parenthèses. L'ordre de priorité que nous avons considéré est le suivant :

Symbole de prédicat (y compris $=$ et \neq)

\exists, \forall, \neg

\vee, \wedge

$\Leftarrow, \Leftrightarrow$.

Une *substitution* est une opération qui, pour tout atome, remplace toutes les occurrences de certaines variables par certains termes, le même terme étant utilisé par toutes les occurrences d'une variable donnée. Cette notion s'étend naturellement aux règles PROLOG et aux formules du calcul des prédicats.

On appelle *instanciation* d'une règle R ou d'une formule F du calcul des prédicats, la règle ou la formule obtenue en appliquant une substitution à R ou F .

On appelle *instanciation fermée* toute instanciation ne contenant plus de variables libres.

Nous noterons $l(P)$ ou l le langage associé au programme PROLOG P , c'est-à-dire l'ensemble de fonctions et symboles de prédicats apparaissant dans P .

Nous noterons $l'(P)$ ou l' le langage associé au programme PROLOG P , dans lequel on remplace chaque symbole de prédicat R par R' .

Nous noterons $L(P)$ ou L , le langage $l \cup l'$.

Nous noterons $h(P)$ ou h [respectivement $h'(P)$ ou h' , respectivement $H(R)$ ou H] la base de Herbrand associée à l (respectivement l' , respectivement L) c'est-à-dire l'ensemble des atomes sans variables construits avec l (respectivement l' , respectivement L).

Exemple (suite) :

$$l(P) = \{ A, B, C, a, f \}$$

$$l'(P) = \{ A', B', C', a, f \}$$

$$h(P) = \{ A(f^n(a)), B(f^n(a)), C(f^n(a)) \}$$

$$\begin{array}{ccc} n \geq 0 & n \geq 0 & n \geq 0 \end{array}$$

$$h'(P) = \{ A'(f^n(a)), B'(f^n(a)), C'(f^n(a)) \}$$

$$\begin{array}{ccc} n \geq 0 & n \geq 0 & n \geq 0 \end{array}$$

$$L(P) = l(P) \cup l'(P)$$

$$H(P) = h(P) \cup h'(P)$$

Le symbole $=$ est un symbole de prédicat à deux places que nous supposons être différent de tous les autres symboles de prédicats de P . Ce symbole joue un rôle particulier. Nous écrivons :

$(t_1 = t_2)$ à la place de $=(t_1, t_2)$

$(t_1 \neq t_2)$ à la place de $\neg = (t_1, t_2)$

Soit $i \subset h(P)$ [ou $i \subset h'(P)$, ou $i \subset H(P)$].

On définit la notion de *formule vraie dans i* , de la manière suivante :

Une formule F :

- (a) contenant des variables libres,
- (b) de la forme $\forall x G$,
- (c) de la forme $\exists x G$,
- (d) de la forme $(A \Leftarrow B)$,
- (e) de la forme $(A_1 \vee A_2 \dots \vee A_n)$,
- (f) de la forme $(A_1 \wedge A_2 \dots \wedge A_n)$.
- (g) de la forme $(A \Leftrightarrow B)$,
- (h) de la forme $\neg A$,
- (i) qui est un atome de la base de Herbrand,
- (j) de la forme $(t_1 = t_2)$ où t_1 et t_2 sont des termes sans variable

est vraie dans i si et seulement si :

- (a) toute instanciation fermée de F est vraie dans i ,
- (b) pour toute substitution θ de x par un terme sans variable θG est vraie dans i ,
- (c) il existe une substitution θ de x par un terme sans variable telle que θG est vraie dans i ,
- (d) lorsque B est vraie dans i , alors A l'est aussi,
- (e) l'une des formules A_j est vraie dans i ,
- (f) toutes les formules A_j sont vraies dans i ,
- (g) $(A \Leftarrow B)$ et $(B \Leftarrow A)$ sont vraies dans i ,
- (h) A n'est pas vraie dans i ,
- (i) $F \in i$,
- (j) les termes t_1 et t_2 sont identiques.

Remarque: La formule $(A_1 \vee \dots \vee A_n)$ lorsque $n=0$ se note \square , d'après la partie (e) de la définition \square est fausse dans tout i .

La formule $(A_1 \wedge \dots \wedge A_n)$ lorsque $n=0$ se note V , car, d'après la partie (f) de la définition, V est vraie dans tout i .

Exemple (suite): Si

$$i = \left\{ A(a), \underset{n \geq 0}{B(f^n(a))}, \underset{n \geq 1}{C(f^n(a))} \right\}$$

alors les formules :

$$\begin{aligned} & \forall x x \neq f(x), \\ & \exists x A(x), \\ & \exists x \exists y (x \neq y \wedge B(x) \wedge B(y)), \\ & \forall x (B(x) \Leftarrow C(x)), \end{aligned}$$

sont des formules vraies dans i , et les formules :

$$\begin{aligned} & \exists x (x = f(x)), \\ & \exists x (f(x) = f(y) \wedge x \neq y), \\ & \forall x C(x), \end{aligned}$$

sont des formules fausses dans i .

Bien remarquer qu'une formule n'utilisant que le symbole de prédicat $=$ est vraie dans tout i , ou fausse dans tout i .

On appelle *interprétations positives* de P (resp. *interprétations négatives* de P , resp. *interprétation complète* de P) tout $i \subset h(P)$ [resp. $i \subset h'(P)$, resp. $i \subset H(P)$].

Si Ax est un système d'axiomes utilisant le langage $l(P)$ [resp. $l'(P)$; resp. $L(P)$] et si i est une interprétation positive (resp. négative, resp. complète) de P telle que tout axiome de Ax est vrai dans i , nous disons que i est un *modèle* de Ax .

La classe des modèles ainsi définie est la classe des modèles de Herbrand de Ax dans lesquels chaque symbole de fonction est interprété librement, et dans lesquels le prédicat $=$ est interprété canoniquement.

La restriction à ces modèles de Herbrand particuliers, qui définit le cadre habituel dans lequel on étudie la sémantique logique de PROLOG, permet d'éviter d'introduire des axiomes particuliers pour l'égalité, et les axiomes qui forcent l'interprétation libre des symboles de fonctions ([Llo 84, She 84]).

Le dédoublement de chaque symbole de prédicat dans ce que nous appelons les interprétations complètes de P , est destiné à rendre compte de l'obtention des succès et de l'obtention des échecs lors de l'utilisation d'un interpréteur PROLOG.

3. ARBRES *SLD*, ENSEMBLE DE SUCCÈS, ENSEMBLE DES ÉCHECS FINIS

Dans ce paragraphe, nous présentons la notion d'arbre *SLD*.

Cette notion est fondée sur le principe de résolution de Robinson [Rob 65] dont une stratégie d'utilisation particulière est donnée par la *SL* résolution [Hil 74] qui pour les clauses définies (c'est-à-dire pour les règles de la forme de celle d'un programme PROLOG) donne la *SLD* résolution.

On dit que la liste d'atomes L' est obtenue à partir de la liste d'atomes $L = (A t_1, A t_2, \dots, A t_k)$ en dérivant le m -ième atome par la règle $r : B \leftarrow B_1, B_2, \dots, B_l$ si :

$$L' = (\theta A t_1, \dots, \theta A t_{m-1}, \theta B_1, \dots, \theta B_l, \theta A t_{m+1}, \dots, \theta A t_k)$$

où θ est le plus grand unificateur entre B et $A t_m$.

Lorsque L est la liste vide, le couple $(L, 0)$ est noté \square et appelé clause vide.

On appelle arbre de dérivation *SLD* de racine L tout arbre dont chaque nœud est un couple (L, m) avec :

L une liste d'atomes,

m un entier;

et vérifiant que :

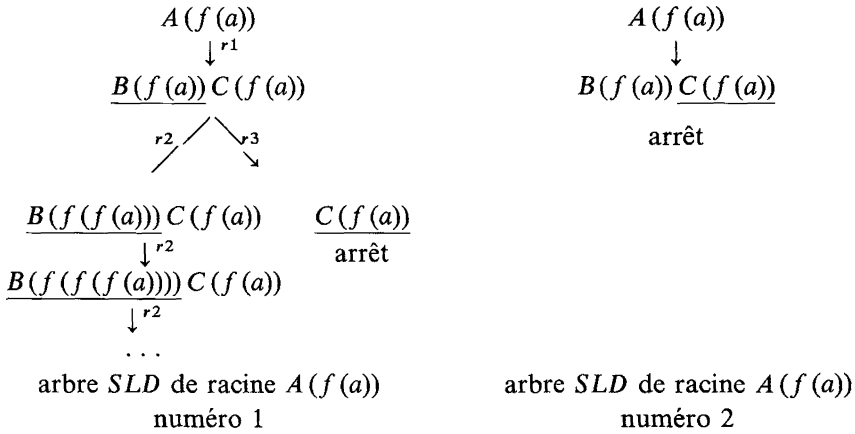
(1) la racine est (L, r) ,

(2) si (L', m') est le fils de (L, m) alors L' est obtenu à partir de L en dérivant le m -ième atome de L par une règle de P ,

(3) pour toute règle permettant de dériver le m -ième atome de L , le nœud (L, m) a un fils correspondant; les différents fils étant ordonnés en fonction du numéro de la règle utilisée pour les obtenir.

Exemple (suite) : Pour représenter un arbre de dérivation, à la place de noter m_i à chaque nœud, nous soulignons le m_i -ième élément de la liste L du nœud.

Voici deux arbres *SLD* de racine $A(f(a))$ (le programme que nous considérons est toujours celui introduit au paragraphe 2).



Ces deux arbres *SLD* correspondent au même programme et ont la même racine. Le premier est infini et le second est fini. Aucun des deux ne contient la clause vide.

Le premier correspond à l'arbre *SLD* considéré par PROLOG et qui consiste à choisir toujours $m_i = 1$; ce qui veut dire que c'est toujours le premier but de la liste qui est dérivé (éventuellement de plusieurs manières différentes si plusieurs règles du programme le permettent).

Déterminer un interpréteur PROLOG, c'est :

- définir une stratégie de choix des coefficients m_i , ce qui engendre certains arbres *SLD*,
- puis choisir une stratégie d'exploration des arbres *SLD* déterminés par le premier choix.

Les interpréteurs PROLOG standards sont fondés :

- sur la stratégie très simple $m_i = 1$ pour le choix de l'arbre *SLD*;
- sur la stratégie de parcours dite « en profondeur d'abord avec retours en arrière » pour le parcours de l'arbre.

Un résultat très important d'Apt et Van Emden [Apt 82] indique que :

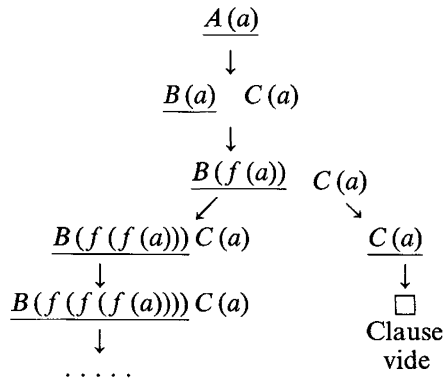
si l'un des arbres SLD de racine L contient la clause vide, alors tous la contiennent.

Ceci permet de donner la définition suivante : l'ensemble des succès d'un programme P est l'ensemble des atomes At sans variable dont un (ou tout) arbre *SLD* ayant At comme racine, contient la clause vide.

Nous noterons cet ensemble SS (ce qui coïncide avec la notation de [Apt 82]) car cet ensemble correspond à l'ensemble des atomes sans variable

pour lesquels un interpréteur PROLOG explorant son arbre *SLD* avec une stratégie « en largeur d'abord » donne un succès.

Il est essentiel de noter ici que si on considère un interpréteur PROLOG standard ($m_i = 1$ et « en profondeur d'abord ») alors il est possible que $At \in SS$ et que l'interpréteur n'indique pas un succès pour At , car il ne rencontre jamais la clause vide. Voici un exemple d'une telle situation :



Dans un tel arbre infini, la branche infinie la plus à gauche ne contient pas la clause vide, et en conséquence l'interpréteur standard poursuit indéfiniment sa recherche dans cette branche, sans jamais donner de succès (car il ne trouve jamais la clause vide) et sans jamais donner d'échec (car son exploration n'est jamais terminée).

L'ensemble des succès d'un interpréteur PROLOG standard, que nous notons *SSP*, est l'ensemble des atomes sans variable pour lesquels l'arbre *SLD* correspondant contient une clause vide sans qu'il y ait de branche infinie à gauche de cette clause vide.

Le premier résultat cité peut être complété par le résultat suivant de Lassez et Maher [Las 84].

Si l'un des arbres SLD de racine L est fini, alors tout arbre SLD « équitable » (« fair » en anglais) de racine L est aussi fini.

(Un arbre *SLD* équitable est un arbre *SLD* tel qu'aucun atome n'est laissé non sélectionné indéfiniment dans une branche; le premier arbre *SLD* que nous avons dessiné n'est pas équitable car $C(f(a))$ n'est jamais sélectionné dans la branche infinie de gauche, le second est équitable.)

Ceci nous permet de donner la définition suivante :

L'ensemble des échecs finis de P , noté *FF* (ce qui correspond à la notation de [Apt 82]) est l'ensemble des atomes At sans variable pour lesquels il existe

un arbre *SLD* de racine *At* fini et ne contenant pas la clause vide ou, ce qui revient au même, c'est l'ensemble des atomes tels que n'importe quel arbre *SLD* équitable de racine *At* est fini et sans clause vide.

L'ensemble *FF* sera parfois appelé ensemble des échecs finis d'un interpréteur équitable, car c'est effectivement l'ensemble des atomes sans variable pour lesquels un interpréteur ne considérant que des arbres équitables donne un échec.

On trouvera dans [Del 86 *b*] des schémas généraux donnant des interpréteurs équitables.

Ainsi que le montre l'exemple indiqué, un interpréteur PROLOG standard n'est pas équitable et son ensemble d'échecs finis, que nous noterons *FFP* est l'ensemble des atomes sans variable pour lesquels l'arbre *SLD* PROLOG ($m_i = 1$) est fini et ne contient pas la clause vide.

Il résulte immédiatement des définitions données et des résultats cités que :

$$SSP \subset SS \subset \mathcal{C}FF \subset \mathcal{C}FFP.$$

Des résultats complémentaires sur ces notions peuvent être trouvés dans [Bla 82 *a*] [Bla 82 *b*] [Del 87] [Fla 86] [JLL 83] [Las 84] [Llo 84] [VEA 85].

De manière schématique on peut commenter ces inclusions en disant :

Les interpréteurs PROLOG standards ne trouvent pas assez de succès car ils parcourent les arbres *SLD* qu'ils engendrent en utilisant une stratégie « en profondeur d'abord ».

Les interpréteurs PROLOG standards ne trouvent pas assez d'échecs car ils engendrent des arbres *SLD* qui ne sont pas toujours équitables.

4. LES HYPOTHÈSES D'UTILISATION D'UN PROGRAMME PROLOG ET LES SYSTÈMES D'AXIOMES ASSOCIÉS

La traduction sous forme clausale d'un programme PROLOG *P* (ou la traduction *SI* qui est équivalente) est la traduction la plus simple et la plus naturelle de *P*. Nous la noterons Ax_x .

Faire l'*hypothèse du monde ouvert* (OWA) [Rei 78] à propos de *P* consiste à considérer comme vrai ce qui résulte de Ax_x et comme indéterminé tout le reste. Cette hypothèse correspond à des situations où on n'est pas certain d'avoir exprimé dans le programme tout ce qui est vrai à propos des objets (les termes sans variables) déterminés par le programme.

Faire l'*hypothèse du monde fermé* (CWA) ([Apt 82, Rei 78]) à propos de *P* consiste à considérer comme vrai ce qui résulte de Ax_x et comme faux tout

le reste. Cette hypothèse correspond à des situations où on est certain d'avoir exprimé dans le programme toutes les connaissances positives concernant les objets. Cette hypothèse peut se formuler en disant : le « monde de P » c'est le plus petit modèle de Ax_α .

En fait, lorsqu'on écrit des règles comme par exemple :

$$\text{humain}(x) \leftarrow \text{adulte}(x)$$

$$\text{humain}(x) \leftarrow \text{enfant}(x)$$

on ne veut pas seulement exprimer que :

$$\text{humain}(x) \Leftarrow (\text{adulte}(x) \vee \text{enfant}(x))$$

ce que donne la traduction clausale Ax_α . On souhaite aussi exprimer que tout humain est un adulte ou un enfant. Ni l'hypothèse du monde ouvert, ni l'hypothèse du monde fermé n'expriment cela.

L'hypothèse qu'effectivement on a en tête en écrivant des règles comme celles au-dessus s'appelle l'hypothèse *SSI* (ou *IFF*) (voir [Rei 78]). Elle se fonde sur ce que nous appelons la traduction *SSI* de P et que nous notons Ax_γ . De manière précise, cette hypothèse revient à considérer comme vrai, tout ce qui résulte de Ax_γ (i. e. ce qui est dans le plus petit modèle de Ax_γ) et comme faux tout ce qui est inconsistant avec Ax_γ (i. e. ce qui n'est pas dans le plus grand modèle de Ax_γ).

Une autre façon de rendre compte de l'hypothèse *SSI* consiste à utiliser ce que nous appelons des interprétations complètes de P (c'est-à-dire avec dédoublement de chaque prédicat R en R et R'), et à réécrire Ax_γ avec cette notation (ce qui donne ce que nous notons Ax_δ). L'avantage de cette méthode sur celle d'Apt et Van Emden est qu'elle permet l'expression de l'hypothèse *SSI* uniquement par la prise en considération du plus petit modèle de Ax_δ (et du plus petit point fixe de T_δ , voir §6). Elle permet aussi de caractériser le calcul d'un interpréteur « en largeur d'abord » et « équitable » comme le calcul d'une limite : $SS \cup FF' = \lim_{n \rightarrow \infty} T_\delta^n(\emptyset)$ (voir théorème 1).

Malheureusement, même l'hypothèse *SSI* ne rend pas réellement compte de ce qu'on a en tête en écrivant un programme PROLOG dont on sait qu'il va être utilisé par un interpréteur standard. Par exemple, pour définir la clôture transitive de la relation sans cycle rel, on peut écrire :

$$\text{clorel}(x, y) \leftarrow \text{rel}(x, y)$$

$$\text{clorel}(x, y) \leftarrow \text{rel}(x, z) \text{ clorel}(z, y)$$

définition qui fonctionnera bien pour toute question (ouverte ou fermée) concernant clorel.

On n'écrira pas :

$$\begin{aligned} \text{clorel}(x, y) &\leftarrow \text{rel}(x, y) \\ \text{clorel}(x, y) &\leftarrow \text{clorel}(z, y) \text{ rel}(x, y) \end{aligned}$$

ni d'ailleurs :

$$\begin{aligned} \text{clorel}(x, y) &\leftarrow \text{clorel}(x, z) \text{ rel}(z, y) \\ \text{clorel}(x, y) &\leftarrow \text{rel}(x, y) \end{aligned}$$

qui pourtant donnent exactement les mêmes traductions Ax_α , Ax_γ que la première version, mais qui « font boucler » les interpréteurs standards.

La raison en est qu'on utilise pour écrire des programmes PROLOG ce qu'on sait des interpréteurs standards. L'ensemble des succès de l'interpréteur ne correspond pas du tout au plus petit modèle de Ax_α ni de Ax_γ , et l'ensemble des échecs finis de ces interpréteurs ne correspond pas non plus au complémentaire du plus grand modèle de Ax_γ .

Dans l'espoir d'obtenir une interprétation logique (sémantique logique) des programmes PROLOG plus proche de ce que calculent les interpréteurs standards, nous introduisons un nouveau système d'axiomes Ax_θ . Nous verrons que Ax_θ donne le plus souvent satisfaction (le plus petit modèle de Ax_θ est $SSP \cup FFP'$) mais que malgré tout il ne constitue pas une solution définitive du problème.

4.1. La traduction clausale : Ax_α

La traduction clausale d'un programme P est le système d'axiomes Ax_α obtenu de la manière suivante :

- Pour chaque règle de P de la forme $At \leftarrow$ on écrit l'axiome :

$$At$$

- Pour chaque règle de P de la forme :

$$At \leftarrow At_1 At_2 \dots At_n, \quad n \geq 1$$

on écrit l'axiome :

$$At \leftarrow (At_1 \wedge At_2 \wedge \dots \wedge At_n).$$

Si certains axiomes contiennent des variables, le sens des axiomes obtenus est le même que celui de ces mêmes axiomes précédés de $\forall x_i$ pour chaque variable x_i apparaissant dans les atomes. (Ceci résulte de la définition des formules vraies dans une interprétation i , et correspond aux conventions usuelles du calcul des prédicats.)

Exemple (suite) : La traduction clausale de P est :

$$Ax_\alpha \left\{ \begin{array}{l} A(x) \Leftarrow (B(x) \wedge C(x)) \\ B(x) \Leftarrow B(f(x)) \\ B(f(a)) \\ C(a) \end{array} \right.$$

Tout modèle de Ax_α contient nécessairement $C(a)$, $B(f(a))$, donc d'après l'axiome 2, contient aussi $B(a)$, donc d'après l'axiome 1, contient aussi $A(a)$.

On vérifie immédiatement que :

$$\{ A(a), B(a), B(f(a)), C(a) \}$$

est un modèle de Ax_α .

D'après ce qu'on vient de dire, c'est le plus petit.

Un autre modèle, par exemple, est :

$$\{ A(a), B(f^n(a)), C(a) \}.$$

$n \geq 0$

Il faut remarquer que $h(P)$ est une interprétation qui rend vrai chaque axiome de Ax_α . Ax_α admet donc $h(P)$ comme plus grand modèle et donc la notion de plus grand modèle est sans intérêt ici contrairement à la notion de plus petit modèle.

4.2. La traduction $SI : Ax_\beta$

La traduction SI d'un programme P (introduite dans [Apt 82]) est équivalente à la traduction clausale. Elle sert en fait d'intermédiaire pour arriver à la traduction SSI et permet un système de notation bien homogène.

La traduction SI d'un programme P est le système d'axiomes Ax_β obtenu à partir de P de la manière suivante :

(a) Pour toute règle de P de la forme :

$$P(t_1, t_2, \dots, t_n) \Leftarrow A t_1 A t_2 \dots A t_k$$

écrire l'axiome :

$$P(x_1, x_2, \dots, x_n) \Leftarrow (\exists y_1 \exists y_2 \dots \exists y_m \\ x_1 = t_1 \wedge x_2 = t_2 \wedge \dots \wedge x_n = t_n \wedge A t_1 \wedge A t_2 \wedge \dots \wedge A t_k)$$

où y_1, y_2, \dots, y_m sont les variables apparaissant soit dans les termes t_i , soit dans les atomes $A t_j$. Les variables x_1, x_2, \dots, x_n introduites pour écrire l'axiome sont, bien sûr, supposées différentes de toutes celles intervenant dans P .

(b) Regrouper ensemble tous les axiomes obtenus par (a) ayant le même prédicat de tête :

$$\begin{aligned} P(x_1, x_2, \dots, x_n) &\Leftarrow Q_1 \\ P(x_1, x_2, \dots, x_n) &\Leftarrow Q_2 \\ &\dots \\ P(x_1, x_2, \dots, x_n) &\Leftarrow Q_r \end{aligned}$$

et les remplacer par :

$$P(x_1, x_2, \dots, x_n) \Leftarrow Q_1 \vee Q_2 \vee \dots \vee Q_r$$

Exemple (suite) : On obtient d'abord :

$$\left\{ \begin{array}{l} A(x_1) \Leftarrow \exists x (x_1 = x \wedge B(x) \wedge C(x)) \\ B(x_1) \Leftarrow \exists x (x_1 = x \wedge B(f(x))) \\ B(x_1) \Leftarrow x_1 = f(a) \\ C(x_1) \Leftarrow x_1 = a \end{array} \right.$$

puis :

$$A x_B \left\{ \begin{array}{l} A(x_1) \Leftarrow \exists x (x_1 = x \wedge B(x) \wedge C(x)) \\ B(x_1) \Leftarrow \exists x (x_1 = x \wedge B(f(x))) \vee x_1 = f(a) \\ C(x_1) \Leftarrow x_1 = a \end{array} \right.$$

Remarque : Si on doit traduire une règle de la forme :

$$P \Leftarrow$$

où P ne comporte aucune variable, on doit écrire :

$$P \Leftarrow V$$

car V est la conjonction vide qui est vraie dans toute interprétation (voir §2).

PROPOSITION 1 : *Les traductions $A x_\alpha$ et $A x_\beta$ sont équivalentes, c'est-à-dire possèdent exactement les mêmes modèles.*

Démonstration : Immédiate à partir de la définition des formules vraies dans une interprétation (voir aussi [Apt 82, Llo 84]).

4.3. La traduction SSI : $A x_\gamma$

La traduction SSI (si et seulement si) de P permet de donner un sens à la notion d'échec en PROLOG : les atomes pour lesquels on obtient un échec sont des atomes dont la négation résulte de la traduction SSI (voir plus loin les rappels des résultats d'Apt et Van Emden).

La traduction SSI de P est le système d'axiomes $A x_\gamma$ obtenu en remplaçant chaque symbole « \Leftarrow » de $A x_\beta$ par « \Leftrightarrow ».

Exemple (suite) :

$$A x_\gamma \left\{ \begin{array}{l} A(x_1) \Leftrightarrow \exists x (x_1 = x \wedge B(x) \wedge C(x)) \\ B(x_1) \Leftrightarrow \exists x (x_1 = x \wedge B(f(x))) \vee x_1 = f(a) \\ C(x_1) \Leftrightarrow x_1 = a. \end{array} \right.$$

4.4. La traduction SSI dédoublée : $A x_\delta$

Introduite dans le but de faire la sémantique de l'hypothèse SSI (comme $A x_\gamma$) uniquement avec la notion de plus petit modèle, et la notion de plus petit point fixe, le système d'axiomes $A x_\delta$ s'obtient en ajoutant à chaque axiome de $A x_\beta$ de la forme :

$$P(x_1, x_2, \dots, x_n) \Leftarrow C$$

l'axiome « réciproque » :

$$P'(x_1, x_2, \dots, x_n) \Leftarrow C'$$

où C' est la formule C dans laquelle :

- chaque prédicat R a été remplacé par R' ,
- chaque symbole $=$ a été remplacé par \neq ,
- chaque symbole \neq a été remplacé par $=$,
- chaque symbole \vee a été remplacé par \wedge ,
- chaque symbole \wedge a été remplacé par \vee ,

chaque symbole \exists a été remplacé par \forall ,
chaque symbole \forall a été remplacé par \exists .

Remarque : Si dans :

$$P'(x_1, x_2, \dots, x_n) \Leftarrow C'$$

on remplace chaque symbole de prédicat R' par $\neg R$ on obtient une formule équivalente à :

$$\neg P(x_1, x_2, \dots, x_n) \Leftarrow \neg C.$$

Exemple (suite) :

$$A x_0 \left\{ \begin{array}{l} A(x_1) \Leftarrow \exists x(x_1 = x \wedge B(x) \wedge C(x)) \\ A'(x_1) \Leftarrow \forall x(x_1 \neq x \vee B'(x) \vee C'(x)) \\ B(x_1) \Leftarrow \exists x(x_1 = x \wedge B(f(x))) \vee x_1 = f(a) \\ B'(x_1) \Leftarrow \forall x(x_1 \neq x \vee B'(f(x))) \wedge x_1 \neq f(a) \\ C(x_1) \Leftarrow x_1 = a \\ C'(x_1) \Leftarrow x_1 \neq a. \end{array} \right.$$

4.5. La traduction interpréteur standard : $A x_0$

Le but de ce cinquième système d'axiomes associé à P est d'exprimer le plus exactement possible le sens logique des règles de P sachant que ces règles sont utilisées par un interpréteur standard.

Avant de donner la définition formelle de $A x_0$ considérons un petit exemple.

Les règles :

$$\left\{ \begin{array}{l} A(x) \leftarrow B(x) C(x) \\ A(x) \leftarrow D(x) \end{array} \right.$$

si ce sont les seules règles de tête A , signifient, pour tout « programmeur PROLOG » que :

– $A(x)$ donne un succès si [$B(x)$ et $C(x)$ en donnent], ou si [$B(x)$ donne un échec et $D(x)$ donne un succès], ou si [$B(x)$ donne un succès, $C(x)$ donne un échec et $D(x)$ donne un succès]. Aucun autre cas ne donne de succès et par exemple si [$C(x)$ donne un échec, $D(x)$ donne un succès et $B(x)$ boucle], alors $A(x)$ boucle.

– $A(x)$ donne un échec si [$B(x)$ et $D(x)$ donnent des échecs], ou si [$B(x)$ donne un succès, et $C(x)$ et $D(x)$ donnent des échecs].

Traduit sous forme d'axiomes en considérant que $A(x)$ signifie maintenant « $A(x)$ donne un succès » et que $A'(x)$ signifie « $A(x)$ donne un échec » on obtient :

$$\left\{ \begin{array}{l} A(x) \Leftarrow (B(x) \wedge C(x)) \vee (B'(x) \wedge D(x)) \vee (B(x) \wedge C'(x) \wedge D(x)) \\ A'(x) \Leftarrow (B'(x) \wedge D'(x)) \vee (B(x) \wedge C'(x) \wedge D'(x)). \end{array} \right.$$

De manière formelle, le système d'axiomes Ax_0 est obtenu à partir de P de la manière suivante :

Pour chaque symbole de prédicat R apparaissant dans P , on regroupe toutes les règles ayant R en tête (le plus souvent les interpréteurs exigent déjà que les règles soient ainsi regroupées), ce qui donne :

$$\begin{aligned} R(t_1^1, t_2^1, \dots, t_n^1) &\leftarrow A t_1^1 \dots A t_{m_1}^1 \\ R(t_1^2, t_2^2, \dots, t_n^2) &\leftarrow A t_1^2 \dots A t_{m_2}^2 \\ &\dots \\ R(t_1^p, t_2^p, \dots, t_n^p) &\leftarrow A t_1^p \dots A t_{m_p}^p \end{aligned}$$

et on écrit :

$$\begin{aligned} R(x_1, x_2, \dots, x_n) &\Leftarrow B_1 \vee (B'_1 \wedge B_2) \vee \dots \vee (B'_1 \wedge B'_2 \wedge \dots \wedge B'_{p-1} \wedge B_p) \\ R'(x_1, x_2, \dots, x_n) &\Leftarrow B'_1 \wedge B'_2 \wedge \dots \wedge B'_p \end{aligned}$$

— où B_i est la formule :

$$\exists y_1^i \exists y_2^i \dots \exists y_{k_i}^i (x_1 = t_1^i \wedge x_2 = t_2^i \wedge \dots \wedge x_n = t_n^i \wedge A t_1^i \wedge A t_2^i \wedge \dots \wedge A t_{m_i}^i)$$

les variables $y_1^i, \dots, y_{k_i}^i$ étant les variables apparaissant dans les termes t_1^i, \dots, t_n^i et les atomes $A t_1^i, \dots, A t_{m_i}^i$

— et où B'_i est la formule :

$$\begin{aligned} \forall y_1^i \forall y_2^i \dots \forall y_{k_i}^i, \quad x_1 \neq t_1^i \vee x_2 \neq t_2^i \vee \dots \vee x_n \neq t_n^i \\ \vee A t_1^{i'} \vee (A t_1^i \wedge A t_2^{i'}) \vee \dots \vee (A t_1^i \wedge A t_2^i \wedge \dots \wedge A t_{m_i-1}^i \wedge A t_{m_i}^{i'}) \end{aligned}$$

le symbole $A t_j^{i'}$ désignant l'atome $A t_j^i$ dans lequel le symbole de prédicat R a été remplacé par R' .

Exemple (suite) :

$$\left. \begin{array}{l}
 A(x_1) \Leftarrow \exists x(x_1 = x \wedge B(x) \wedge C(x)) \\
 A'(x_1) \Leftarrow \forall x(x_1 \neq x \vee B'(x) \vee (B(x) \wedge C'(x))) \\
 B(x_1) \Leftarrow \exists x(x_1 = x \wedge B(f(x))) \\
 \quad \vee (\forall x(x_1 \neq x \vee B'(f(x))) \wedge x_1 = f(a)) \\
 B'(x_1) \Leftarrow \forall x(x_1 \neq x \vee B'(f(x))) \wedge x_1 \neq f(a) \\
 C(x_1) \Leftarrow x_1 = a \\
 C'(x_1) \Leftarrow x_1 \neq a.
 \end{array} \right\} Ax_0$$

Le plus souvent, on peut simplifier un peu Ax_0 par exemple ici Ax_0 est équivalent à :

$$\left. \begin{array}{l}
 A(x_1) \Leftarrow B(x_1) \wedge C(x_1) \\
 A'(x_1) \Leftarrow B'(x_1) \vee (B(x_1) \wedge C'(x_1)) \\
 B(x_1) \Leftarrow B(f(x_1)) \\
 \quad \vee (B'(f(x_1)) \wedge x_1 = f(a)) \\
 B'(x_1) \Leftarrow B'(f(x_1)) \wedge x_1 \neq f(a) \\
 C(x_1) \Leftarrow x_1 = a \\
 C'(x_1) \Leftarrow x_1 \neq a.
 \end{array} \right\} Ax_0$$

5. OPÉRATEURS ASSOCIÉS AUX TRADUCTIONS β , δ ET θ

L'ensemble des interprétations positives de P (resp. l'ensemble des interprétations complètes de P) qui, par définition, n'est autre que l'ensemble $\mathcal{P}(h(P))$ [resp. $\mathcal{P}(H(P))$], des parties de $h(P)$ [resp. de $H(P)$], est naturellement muni d'une structure de treillis complet pour les opérations \cup et \cap , et pour la relation \subset .

Cette structure a été utilisée dans [Apt 82, Las 84, Llo 84] pour associer à chaque programme un opérateur dont le plus petit point fixe et le plus grand point fixe fournissent sur P des informations intéressantes et permettent d'interpréter le calcul effectué par les interpréteurs « en largeur d'abord » et « équitables ».

Nous reprenons cette idée, mais au lieu de considérer un opérateur associé à chaque programme, nous en considérons trois. Le premier T_β est celui défini par Apt et Van Emden et on peut, de manière schématique, dire qu'il correspond à l'hypothèse du monde ouvert. Le second T_δ correspond, lui, à

l'hypothèse *SSI* et le troisième T_θ correspond à l'hypothèse « Interpréteur PROLOG standard ».

Les traductions $A x_\beta$, $A x_\delta$ et $A x_\theta$ sont toutes les trois des conjonctions de formules de la forme

$$R(x_1, x_2, \dots, x_n) \Leftarrow F(x_1, x_2, \dots, x_n)$$

où R est un symbole de prédicat de $L(P)$ et F une formule du calcul des prédicats ne comportant comme variables libres que les variables x_1, x_2, \dots, x_n .

De plus, pour chaque symbole de prédicats de $l(P)$ [resp. de $L(P)$], il y a exactement une formule de $A x_\beta$ (resp. de $A x_\delta$, $A x_\theta$) ayant ce prédicat comme premier symbole.

DÉFINITION : T_β (resp. T_δ ; resp. T_θ) est un opérateur (i. e. une application)

de $\mathcal{P}(h(P))$ dans $\mathcal{P}(h(P))$

[resp. de $\mathcal{P}(H(P))$ dans $\mathcal{P}(H(P))$];

[resp. de $\mathcal{P}(H(P))$ dans $\mathcal{P}(H(P))$].

Pour tout $i \subset h(P)$ [resp. $i \subset H(P)$; resp. $i \subset H(P)$], $T_\beta(i)$ [resp. $T_\delta(i)$; resp. $T_\theta(i)$] est l'ensemble des atomes θA tels que :

- il existe un axiome de la forme $A \Leftarrow B$ dans $A x_\beta$ (resp. $A x_\delta$, resp. $A x_\theta$),
- $\theta(A \Leftarrow B)$ ne contient pas de variable libre,
- θB est vrai dans i .

Remarque : On peut exprimer la définition de manière imagée en disant que $T_\beta(i)$ est tout ce qu'une étape de « modus ponens » à partir de i en utilisant les axiomes de $A x_\beta$, permet d'obtenir.

Exemple (suite) :

$$T_\beta(\emptyset) = \{B(f(a)), C(a)\}$$

$$T_\beta^2(\emptyset) = \{B(a), B(f(a)), C(a)\}$$

$$T_\beta^3(\emptyset) = \{A(a), B(a), B(f(a)), C(a)\}$$

$$T_\beta^n(\emptyset) = T_\beta^3(\emptyset) \text{ pour tout } n \geq 3.$$

$$T_\beta(h(P)) = \left\{ A \left(\underset{n \geq 0}{f^n(a)} \right), B \left(\underset{n \geq 0}{f^n(a)} \right), C(a) \right\}$$

$$T_\beta^2(h(P)) = \left\{ A(a), B \left(\underset{n \geq 0}{f^n(a)} \right), C(a) \right\}$$

$$T_{\beta}^n(\emptyset) = T_{\beta}^2(\emptyset) \text{ pour tout } n > 2.$$

PROPOSITION 1 : Les opérateurs T_{β} , T_{δ} et T_{θ} sont des opérateurs monotones.

Démonstration : Soient i, j , $i \subset j$. On doit montrer que :

$$T_{\beta}(i) \subset T_{\beta}(j) \quad [\text{resp. } T_{\delta}(i) \subset T_{\delta}(j); \text{ resp. } T_{\theta}(i) \subset T_{\theta}(j)].$$

Pour cela, d'après la définition, il suffit d'établir que, pour toute formule $F(x_1, x_2, \dots, x_n)$ pouvant être la partie arrière d'un axiome, si $F(t_1, t_2, \dots, t_n)$ est vraie dans i , alors $F(t_1, t_2, \dots, t_n)$ est vraie dans j . Or, cela résulte de ce que toute formule F pouvant être la partie arrière d'un axiome est construite à partir :

- d'atomes utilisant des prédicats de P ,
- d'atomes et de négations d'atomes utilisant le prédicat $=$,
- des connecteurs \vee et \wedge ,
- de quantificateurs.

Remarque : Si les parties arrières d'axiomes pouvaient contenir des négations de prédicats de P , ou si le connecteur \Leftrightarrow était utilisé, le raisonnement ne fonctionnerait plus.

Si h est un treillis complet et T un opérateur monotone, sur h , on pose :

$$T \uparrow 0 = \emptyset, \quad T \downarrow 0 = h$$

$$\left. \begin{array}{l} T \uparrow (n+1) = T(T \uparrow n) \\ T \downarrow (n+1) = T(T \downarrow n) \end{array} \right\} \text{ pour tout ordinal successeur } (n+1)$$

$$\left. \begin{array}{l} T \uparrow \alpha = \bigcup_{\beta < \alpha} T \uparrow \beta \\ T \downarrow \alpha = \bigcap_{\beta < \alpha} T \downarrow \beta \end{array} \right\} \text{ pour tout ordinal limite } \alpha$$

La nécessité d'utiliser des ordinaux limites apparaît dans [Apt 82].

On sait que, pour tout opérateur croissant T , il existe un plus petit ordinal η_T , et un plus petit ordinal μ_T tels que :

$$T \uparrow \eta_T = T \uparrow (\eta_T + 1)$$

$$T \downarrow \mu_T = T \downarrow (\mu_T + 1)$$

et que, de plus $T \uparrow \eta_T$ et $T \downarrow \mu_T$ sont respectivement le plus petit point fixe de T et le plus grand point fixe de T . (Voir [Apt 82, Llo 84].)

Bien sûr, on a :

$$T \uparrow \omega \subset T \uparrow \eta_T \subset T \downarrow \mu_T \subset T \downarrow \omega$$

Un opérateur T est dit continu si, pour toute chaîne croissante

$$x_1 \subset x_2 \subset \dots \subset x_n \dots,$$

on a :

$$T(\cup \{x_i \mid i < \omega\}) = \cup \{T(x_i) \mid i < \omega\}.$$

Si T est un opérateur continu, on sait que : $T \uparrow \omega$ est le plus petit point fixe de T (voir [Apt 82, VEK 76]).

PROPOSITION 2 : T_β est continu.

Démonstration : Voir [Apt 82, Llo 84].

PROPOSITION 3 : T_δ et T_θ ne sont pas continus en général.

Démonstration : Soit le programme :

$$\begin{cases} P(s(x)) \leftarrow P(x) \\ Q(a) \leftarrow P(x) \end{cases}$$

Ici, $A x_\delta$ coïncide avec $A x_\theta$ et T_δ avec T_θ . On obtient :

$$A x_\delta \begin{cases} P(x_1) \leftarrow \exists x (x_1 = s(x) \wedge P(x)) \\ P'(x_1) \leftarrow \forall x (x_1 \neq s(x) \vee P'(x)) \end{cases}$$

$$A x_\theta \begin{cases} Q(x_1) \leftarrow \exists x (x_1 = a \wedge P(x)) \\ Q'(x_1) \leftarrow \forall x (x_1 \neq a \vee P'(x)) \end{cases}$$

$$h(P) = \{P(s^n(a)), Q(s^n(a))\}_{n \geq 0}$$

$$T_\delta(\emptyset) = T_\theta(\emptyset) = \{P'(s^m(a)), Q'(s^m(a))\}_{m \geq 1}$$

$$T_\delta^2(\emptyset) = T_\theta^2(\emptyset) = \{P'(s^m(a)), Q'(s^m(a))\}_{m < 2, m \geq 1}$$

$$T_\delta^n(\emptyset) = T_\theta^n(\emptyset) = \{P'(s^m(a)), Q'(s^m(a))\}_{m < n, m \geq 1}$$

$$T_\delta \uparrow \omega = T_\theta \uparrow \omega = \{P'(s^m(a)), Q'(s^m(a))\}_{m \geq 0, m \geq 1}$$

$$T_\delta \uparrow (\omega + 1) = T_\theta \uparrow (\omega + 1) = \{P'(s^m(a)), Q'(s^m(a))\}_{m \geq 0, m \geq 0}$$

PROPOSITION 4 : On a les équivalences suivantes :

$T_\beta(i) \subset i \Leftrightarrow i$ est un modèle de $A x_\alpha$ et $A x_\beta$,

$T_\delta(i) \subset i \Leftrightarrow i$ est un modèle de $A x_\delta$,

$T_\theta(i) \subset i \Leftrightarrow i$ est un modèle de Ax_θ .

Démonstration : Soit i tel que $T_\beta(i) \subset i$.

Pour tout axiome ax de Ax_β et toute instanciation fermée de ax , si la queue de l'axiome est vraie dans i alors la tête de l'axiome qui, par définition, est dans $T_\beta(i)$ sera dans i , d'après l'hypothèse, donc ax sera vrai dans i .

Réciproquement, si i est un modèle de Ax_β , alors chaque instanciation fermée de chaque axiome ax est vraie dans i et donc $T_\beta(i) \subset i$.

La démonstration est analogue pour T_δ et T_θ .

PROPOSITION 5 : L'ensemble $i \subset h(P)$ est un modèle de Ax_γ si et seulement si $T_\beta(i) = i$.

Démonstration : Immédiate à partir de la définition des formules vraies [partie (g) de la définition].

PROPOSITION 6 :

(a) le plus petit modèle de Ax_α et de Ax_β est le plus petit point fixe de T_β (c'est-à-dire $T_\beta \uparrow \omega$);

(b) le plus petit modèle de Ax_δ est le plus petit point fixe de T_δ (c'est-à-dire $T_\delta \uparrow \eta_{T_\delta}$);

(c) le plus petit modèle de Ax_θ est le plus petit point fixe de T_θ (c'est-à-dire $T_\theta \uparrow \eta_{T_\theta}$).

Démonstration : Immédiate à partir des propositions 2 et 4 et des rappels faits à propos des opérateurs continus.

PROPOSITION 7 : Le plus petit modèle de Ax_α (qui coïncide avec le plus petit modèle de Ax_β et avec le plus petit modèle de Ax_γ , avec $T_\beta \uparrow \omega$) est égal à SS .

Démonstration : Voir [Apt 82, Llo 84].

PROPOSITION 8 : Le plus grand point fixe de T_β est le plus grand modèle de Ax_γ .

Démonstration : Immédiate (avec la proposition 5).

PROPOSITION 9 : L'ensemble FF est le complémentaire dans $h(P)$ de $T_\beta \downarrow \omega$: $FF = \complement T_\beta \downarrow \omega$.

Démonstration : Voir [Apt 82, Llo 84].

Remarque : Les propositions 7 et 9 caractérisent logiquement et dénotationnellement les interpréteurs PROLOG « en largeur d'abord » et « équitables ».

6. SÉMANTIQUE LOGIQUE ET DÉNOTATIONNELLE DE L'HYPOTHÈSE SSI PAR L'UTILISATION DE $A x_\delta$ ET T_δ

Les résultats rappelés au paragraphe 5 fournissent la sémantique logique et dénotationnelle de l'hypothèse SSI, mais ils utilisent à la fois les notions de plus petit point fixe et de plus grand point fixe de T_β , et les notions de plus petit modèle et de plus grand modèle de $A x_\gamma$.

La méthode des interprétations complètes (dédoublément de chaque prédicat) appliquée pour l'hypothèse SSI a donné la traduction $A x_\delta$ et l'opérateur T_δ .

Nous établissons maintenant qu'elle permet, grâce aux seules notions de plus petit modèle et de plus petit point fixe, de refaire les sémantiques logiques et dénotationnelles de l'hypothèse SSI.

Si $i \subset h(P)$ on notera i' le sous-ensemble de $h'(P)$ obtenu en ajoutant des « ' » à chaque prédicat d'atomes de i .

Si $i \subset H(P)$ on écrira $i = \text{pos}(i) \cup \text{neg}(i)'$.

Par exemple avec :

$$i = \{ A(a), A'(b), A(c), B'(b) \}$$

on obtient :

$$\text{pos}(i) = \{ A(a), A(c) \} \quad \text{neg}(i) = \{ A(b), B(b) \}.$$

THÉORÈME 1 :

$$(a) \quad T_\delta \uparrow \omega = T_\beta \uparrow \omega \cup (\mathbb{C}(T_\beta \downarrow \omega))'.$$

$$(b) \quad T_\delta \uparrow \eta_{T_\delta} = T_\beta \uparrow \omega \cup (\mathbb{C}(T_\beta \downarrow \mu_{T_\beta}))'.$$

COROLLAIRE :

$$(a) \quad T_\delta \uparrow \omega = SS \cup FF'.$$

$$(b) \quad \text{ppm}(A x_\delta) = SS \cup (\mathbb{C} \text{pgm}(A x_\gamma))'.$$

Ces résultats signifient que ce qu'un interpréteur PROLOG « en largeur d'abord » et « équitable » calcule, c'est $T_\delta \uparrow \omega$, c'est-à-dire une limite.

Le plus petit point fixe de T_δ (qui est aussi le plus petit modèle de $A x_\delta$) fournit donc :

- l'ensemble de succès SS (atomes sans « ' ») et
- l'ensemble des atomes dont la négation résulte de la traduction SSI de P (atomes marqués d'un « ' »).

Le fait qu'on n'ait pas toujours $T_\delta \uparrow \omega = T_\delta \uparrow \eta_{T_\delta}$ correspond exactement au fait que, pour T_β on n'a pas toujours $T_\beta \downarrow \omega = T_\beta \downarrow \mu_{T_\beta}$.

LEMME 1 : *Soit*

$$\begin{aligned} i &\subset H(P) \quad (i = \text{pos}(i) \cup \text{neg}'(i)). \\ T_\delta(i) &= \text{pos}(T_\delta(\text{pos}(i))) \cup \text{neg}'(T_\delta(\text{neg}'(i))) \\ \text{neg}(T_\delta(i)) &= \text{neg}(T_\delta(\text{neg}'(i))) \\ \text{pos}(T_\delta(i)) &= \text{pos}(T_\delta(\text{pos}(i))). \end{aligned}$$

Démonstration : Les axiomes de Ax_δ sont de deux types, ceux qui concluent sur des prédicats « sans prime » et qui n'ont aucun prédicat « avec prime » dans leurs prémisses, et ceux qui concluent sur des prédicats « avec prime » et qui n'ont aucun prédicat « sans prime » dans leurs prémisses. Il en résulte que si $At \in h(P)$ alors le fait que $At \in T_\delta(i)$ ne dépend que de $\text{pos}(i)$ et que si $At \in h'(P)$, le fait que $At \in T_\delta(i)$ ne dépend que de $\text{neg}(i)$.

Remarque : Cette démonstration n'a pas d'équivalent pour T_θ car les prédicats « avec prime » et « sans prime » sont mélangés dans les axiomes de Ax_θ .

LEMME 2 : *Pour tout $i \subset h(P)$: $\text{pos}(T_\delta(i)) = T_\beta(i)$.*

Démonstration : Résulte du lemme 1 et de ce que l'ensemble des axiomes « sans prime » de Ax_δ est exactement Ax_β .

LEMME 3 : *Pour tout $i \subset h(P)$:*

$$\begin{cases} \text{neg}(T_\delta(i')) = \mathbb{C} T_\beta(\mathbb{C} i) \\ \text{neg}(T_\delta(\mathbb{C} i')) = \mathbb{C} T_\beta(i). \end{cases}$$

Démonstration du lemme 3 : Les axiomes de Ax_δ sont de deux sortes :

- ceux identiques aux axiomes de Ax_β
- ceux obtenus en prenant un axiome $A \Leftarrow B$ de Ax_β et en écrivant :

$$\text{prime}(A) \Leftarrow \text{prime}(\text{neat}(\text{non}(B)))$$

où $\text{prime}(\)$, $\text{neat}(\)$ et $\text{non}(\)$ sont les trois applications de $L(P)$ dans lui-même définies par :

$\text{non}(F)$ = la formule obtenue en écrivant $\neg F$ la négation de F , et en faisant descendre le symbole \neg devant les atomes par utilisation répétée des formules :

$$\begin{aligned} \neg(A \vee B) &= \neg A \wedge \neg B; & \neg(A \wedge B) &= \neg A \vee \neg B; \\ \neg \exists x A &= \forall x \neg A; & \neg \forall x A &= \exists x \neg A. \end{aligned}$$

$\text{neat}(F)$ = la formule obtenue en remplaçant chaque atome At par $\neg At$ et en remplaçant $\neg \neg At$ par At .

prime(F) = la formule obtenue en mettant un prime à tous les prédicats, sauf =.

Les équivalences suivantes sont évidentes :

$$\begin{aligned} F \text{ vraie dans } i &\Leftrightarrow \text{non } (F) \text{ fausse dans } i \\ F \text{ vraie dans } i &\Leftrightarrow \text{neat } (F) \text{ vraie dans } \ulcorner i \\ F \text{ vraie dans } i &\Leftrightarrow \text{prime } (F) \text{ vraie dans } i'. \end{aligned}$$

Soit at un atome et $i \subset h(P)$ on a la suite d'équivalences :

$$\begin{aligned} &[at \in \ulcorner T_\beta(i)] \\ \Leftrightarrow &[at \notin T_\beta(i)] \\ \Leftrightarrow &[\text{la règle unique } A \Leftarrow B \text{ de } A x_\beta \text{ pouvant donner } at \text{ ne donne pas } at \\ \text{dans } i] \\ \Leftrightarrow &[\text{la substitution } \theta \text{ telle que } \theta A = at, \text{ est telle que } \theta B \text{ est fausse dans } i] \\ \Leftrightarrow &[\text{non } (\theta B) \text{ est vraie dans } i] \\ \Leftrightarrow &[\text{neat } (\text{non } (\theta B)) \text{ est vraie dans } \ulcorner i] \\ \Leftrightarrow &[\text{prime } (\text{neat } (\text{non } (\theta B))) \text{ est vraie dans } (\ulcorner i)'] \\ \Leftrightarrow &[\text{la règle unique } A' \Leftarrow B' \text{ de } A x_\beta \text{ pouvant donner } at' \text{ donne } at' \text{ dans } (\ulcorner i)'] \\ \Leftrightarrow &[at' \in T_\beta(\ulcorner i)'] \\ \Leftrightarrow &[at \in \text{neg } (T_\beta(\ulcorner i)')]. \end{aligned}$$

LEMME 4 : Pour tout ordinal α :

$$\text{pos}(T_\delta \uparrow \alpha) = T_\beta \uparrow \alpha.$$

Démonstration : Si $\alpha = 0$ il faut montrer :

$$\text{pos}(T_\delta \uparrow 0) = T_\beta \uparrow 0$$

ce qui est évident.

Supposons le lemme vrai pour tout ordinal $< \lambda$ et montrons-le pour λ (induction transfinie). Si λ est un ordinal limite par définition :

$$\begin{aligned} \text{pos}(T_\delta \uparrow \lambda) &= \text{pos}\left(\bigcup_{\eta < \lambda} T_\delta \uparrow \eta\right) \\ &= \bigcup_{\eta < \lambda} (\text{pos } T_\delta \uparrow \eta) \\ &= \bigcup_{\eta < \lambda} T_\beta \uparrow \eta \quad (\text{hypothèse d'induction}) \\ &= T_\beta \uparrow \lambda. \end{aligned}$$

Si λ est un ordinal successeur ($\lambda = \eta + 1$) on a :

$$\begin{aligned} \text{pos}(T_\delta \uparrow \lambda) &= \text{pos}(T_\delta(T_\delta \uparrow \eta)) \\ &= \text{pos}(T_\delta \text{pos}(T_\delta \uparrow \eta)) \quad (\text{lemme 1}) \\ &= \text{pos}(T_\delta(T_\beta \uparrow \eta)) \quad (\text{hypothèse d'induction}) \\ &= T_\beta(T_\beta \uparrow \eta) \quad (\text{lemme 2}) \\ &= T_\beta \uparrow \lambda. \end{aligned}$$

LEMME 5 : Pour tout ordinal α : $\text{neg}(T_\delta \uparrow \alpha) = \mathfrak{C} T_\beta \downarrow \alpha$.

Démonstration :

– Si $\alpha = 0$ il faut montrer

$$\text{neg}(T_\delta \uparrow 0) = \mathfrak{C} T_\beta \downarrow 0$$

c'est-à-dire

$$\text{neg}(\emptyset) = \mathfrak{C} h(P)$$

ce qui est vrai.

– Supposons le lemme vrai pour tout ordinal $< \lambda$ et montrons-le pour λ (induction transfinie).

– Si λ est un ordinal limite par définition :

$$\begin{aligned} T_\delta \uparrow \lambda &= \bigcup_{\eta < \lambda} T_\delta \uparrow \eta \\ T_\beta \downarrow \lambda &= \bigcap_{\eta < \lambda} T_\beta \downarrow \eta \end{aligned}$$

D'après l'hypothèse d'induction on a :

$$\begin{aligned} \bigcap_{\eta < \lambda} T_\beta \downarrow \eta &= \bigcap_{\eta < \lambda} \mathfrak{C} \text{neg } T_\delta \uparrow \eta \\ &= \mathfrak{C} \bigcup_{\eta < \lambda} \text{neg } T_\delta \uparrow \eta \\ &= \mathfrak{C} \text{neg} \bigcup_{\eta < \lambda} T_\delta \uparrow \eta \\ &= \mathfrak{C} \text{neg } T_\delta \uparrow \lambda. \end{aligned}$$

Le lemme est donc vrai pour λ .

Si λ est un ordinal successeur ($\lambda = \eta + 1$) on a :

$$\text{neg}(T_\delta \uparrow \eta) = \mathfrak{C} T_\beta \downarrow \eta$$

par hypothèse d'induction et donc :

$$\begin{aligned}
 \text{neg}(T_\delta \uparrow \lambda) &= \text{neg}(T_\delta(T_\delta \uparrow \eta)) \\
 &= \text{neg}(T_\delta \text{neg}'(T_\delta \uparrow \eta)) \\
 &= \text{neg}(T_\delta(\mathbb{C} T_\beta \downarrow \eta)') \\
 &= \mathbb{C} T_\beta(T_\beta \downarrow \eta) \\
 &= \mathbb{C} T_\beta \downarrow \lambda.
 \end{aligned}$$

Démonstration du théorème : Pour tout ordinal α on a :

$$T_\delta \uparrow \alpha = \text{pos}(T_\delta \uparrow \alpha) \cup \text{neg}'(T_\delta \uparrow \alpha) = T_\beta \uparrow \alpha \cup (\mathbb{C} T_\beta \downarrow \alpha)'$$

d'après les lemmes 4 et 5.

En prenant $\alpha = \omega$ on obtient (a).

La propriété 2 du paragraphe 5 donne (b). Le corollaire résulte lui des propositions 6, 7, 8 et 9.

7. SÉMANTIQUE LOGIQUE ET DÉNOTATIONNELLE DE L'HYPOTHÈSE « INTERPRÉTEUR PROLOG STANDARD »

La caractérisation de *SS* et de *FF* par des systèmes d'axiomes du calcul des prédicats donnerait le sens logique des programmes PROLOG si toutefois les interpréteurs utilisés étaient « en largeur d'abord » et « équitables ». Comme ce n'est pas le cas, le sens logique des programmes PROLOG ne peut pas s'établir par les systèmes d'axiomes classiquement associés à un programme *P* (c'est-à-dire Ax_β et Ax_γ).

C'est pourquoi nous avons introduit un nouveau système d'axiomes qui, plus souvent que les autres, permet de caractériser le sens logique d'un programme sous l'hypothèse qu'il est manipulé par un interpréteur standard.

Deux obstacles empêchent de comprendre les calculs d'un interpréteur standard comme des calculs de modèles. Le premier, qui se présentait déjà dans le cas de *SS* et *FF*, est la non continuité en général des opérateurs *T*. Le second est que, bien que plus proche du « sens opérationnel » des programmes, le système Ax_θ est encore insuffisant dans certaines situations pour le donner complètement.

THÉORÈME 2 :

(a) Pour tout $i \in H(P)$:

$$T_\theta(i) \subset T_\delta(i)$$

(b) Pour tout ordinal α :

$$T_0 \uparrow \alpha \subset T_8 \uparrow \alpha$$

(c) $\text{ppm } A x_0 \subset \text{ppm } A x_8$.

Démonstration : (a) A chaque axiome de $A x_8$ correspond un axiome de $A x_0$ ayant le même prédicat de tête. La formule de l'axiome de $A x_0$ derrière le symbole \Leftarrow est plus restrictive que la formule correspondante de l'axiome de $A x_8$ et donc d'après les définitions de T_0 et T_8 pour tout $i \in H(P)$ on a : $T_0(i) \subset T_8(i)$.

(b) Immédiat par induction transfinie à partir de (a).

(c) Immédiat à partir de (b) et de la proposition 6 du paragraphe 5.

THÉORÈME 3 :

$$SSP \cup FFP' \subset T_0 \uparrow \omega \subset SS \cup FF'$$

Les exemples complémentaires donnés plus loin montrent que les inclusions peuvent être strictes.

Démonstration : La seconde inclusion est une conséquence immédiate du théorème 2(b) et du corollaire du théorème 1.

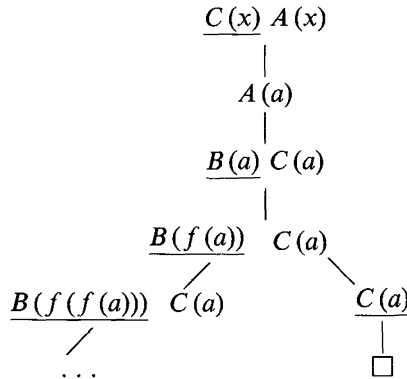
Pour établir la première inclusion nous allons donner une définition « par suite croissante d'ensembles » de SSP et FFP .

Comme nous ne pouvons plus éviter de prendre en compte les variables, nous allons nous placer dans l'univers de Herbrand généralisé H_G associé au programme P et qui est défini comme l'ensemble des listes finies d'atomes utilisant le vocabulaire du programme et des variables.

La notion d'arbre SLD s'étend sans problème à ce cadre (la seule chose à changer est de ne pas imposer à la racine d'être un atome instancié).

De même la notion d'arbre SLD PROLOG.

Exemple (suite) : L'arbre SLD PROLOG de racine $C(x) A(x)$ est :



Soit L une liste de buts, nous notons $L_1, L_2, \dots, L_r, r \geq 0$, les listes de buts que donne une étape de dérivation avec $m=1$ (dérivation PROLOG standard). Les listes L_1, L_2, \dots, L_r sont simplement les nœuds de niveau 1 de l'arbre SLD PROLOG de racine L . On suppose bien sûr qu'ils sont classés en fonction du numéro de la règle de P qui a permis la dérivation.

SSP_i et $FFP_i, i \geq 0$ sont définis récursivement par les relations suivantes :

$$R1 \quad SSP_0 = \emptyset.$$

$$R2 \quad FFP_0 = \emptyset.$$

$$R3 \quad [(L = \square) \text{ ou} \\ (\exists j \in (1, 2, \dots, r) L_1 \in FFP_i \text{ et } \dots L_{j-1} \in FFP_i \text{ et } L_j \in SSP_i \\ \text{et } \max(\text{long}(L_k) \mid 1 \leq k \leq j) = \text{long}(L) - 1)] \\ \Rightarrow L \in SSP_i \quad \text{pour } i \geq 1$$

$$R4 \quad [(L = \square) \text{ ou} \\ (\exists j \in (1, 2, \dots, r) L_1 \in FFP_i \text{ et } \dots L_{j-1} \in FFP_i \text{ et } L_j \in SSP_i \\ \text{et } \max(\text{long}(L_k) \mid 1 \leq k \leq j) > \text{long}(L) - 1)] \\ \Rightarrow L \in SSP_{i+1} \quad \text{pour } i \geq 1$$

$$R5 \quad [(r=0) \text{ ou} \\ (L_1 \in FFP_i \text{ et } \dots \text{ et } L_r \in FFP_i \\ \text{et } \max(\text{long}(L_k) \mid 1 \leq k \leq r) = \text{long}(L) - 1)] \\ \Rightarrow L \in FFP_i \quad \text{pour } i \geq 1$$

$$R6 \quad [(r=0) \text{ ou} \\ (L_1 \in FFP_i \text{ et } \dots L_r \in FFP_i \\ \text{et } \max(\text{long}(L_k) \mid 1 \leq k \leq r) > \text{long}(L) - 1)] \\ \Rightarrow L \in FFP_{i+1} \quad \text{pour } i \geq 1.$$

On pose alors

$$SSP_G = \bigcup_{i \in \mathbf{N}} SSP_i$$

$$FFP_G = \bigcup_{i \in \mathbf{N}} FFP_i$$

SSP_G et FFP_G sont respectivement les ensembles de succès généralisés (en acceptant les variables) et les ensembles d'échecs finis généralisés d'un interpréteur PROLOG standard. Pour le voir, il suffit d'écrire la définition

réursive directe de SSP_G et FFP_G :

$$L \in SSP_G \quad \text{si} \quad \begin{cases} L = \square \text{ ou} \\ \exists j \in (1, 2, \dots, r) L_j \in FFP_G \text{ et} \dots \end{cases}$$

et $L_{j-1} \in FFP_G$ et $L_j \in SSP_G$

$$L \in FFP_G \quad \text{si} \quad \begin{cases} r = 0 \text{ ou} \\ L_1 \in FFP_G \text{ et} \dots \text{ et} L_r \in FFP_G \end{cases}$$

La définition par suite croissante d'ensembles que nous avons formulée, va nous permettre de mettre en correspondance directe $T_0^i(\emptyset)$ et $SSP_i FFP'_i$. Il peut paraître un peu compliqué d'y faire intervenir la longueur des listes L_i . Une autre définition ne faisant pas intervenir ces longueurs est possible mais elle ne permet pas la mise en correspondance énoncée dans le lemme.

Soit L une liste de buts :

$$L = A t_1 A t_2 \dots A t_i$$

Nous associons à L deux formules du calcul des prédicats notées $\llbracket L \rrbracket$ et $\llbracket L \rrbracket'$ qui sont définies par :

$$\llbracket L \rrbracket = \exists x_1 \dots \exists x_n A t_1 \wedge A t_2 \dots \wedge A t_i$$

où x_1, \dots, x_n sont les variables apparaissant dans les $A t_i$.

$$\begin{aligned} \llbracket L \rrbracket' = \forall x_1 \dots \forall x_n A t'_1 \vee (A t_1 \wedge A t'_2) \vee \dots \\ (A t_1 \wedge \dots \wedge A t_{n-1} \wedge A t'_n) \end{aligned}$$

où x_1, \dots, x_n sont les variables apparaissant dans les $A t_i$ et où $A t'_i$ désigne l'atome $A t$ dans lequel on a remplacé tout symbole de prédicat R par R' .

Remarquons que si L est une liste d'un seul atome instancié $A t$ alors :

$$\llbracket L \rrbracket = A t,$$

$$\llbracket L \rrbracket' = A t'.$$

Comme on a, bien sûr,

$$SSP = SSP_G \cap h$$

$$FFP = FFP_G \cap h$$

le résultat du théorème est donc une conséquence immédiate du lemme suivant.

1° $C(a) \in FFP_1$ (partie 5 de la définition car $r=0$).

2° $D(x) C(x) \in FFP_1$ (formule 1° et partie 5 de la définition).

3° $B(f(x)) C(x) \in FFP_2$ (formule 2° et partie 6 de la définition).

4° $\square \in SSP_1$ (partie 3 de la définition car $L = \square$).

5° $D(y) \in SSP_1$ (formule 4° et partie 3 de la définition).

6° $A(x) \in SSP_3$ (formules 5° et 3° et partie 4 de la définition).

$\llbracket C(a) \rrbracket' = C'(a)$ est vrai dans $T_0^1(\emptyset)$ d'après a_8 .

$\llbracket D(x) C(x) \rrbracket' = \forall x (D'(x) \vee (D(x) \wedge C'(x)))$ est vrai dans $T_0^1(\emptyset)$ car :

– pour $x=a$ on a :

$$D(a) \quad (a_5)$$

et :

$$C'(a) \quad (a_8)$$

– pour $x \neq a$ on a :

$$D'(x) \quad (a_6)$$

$\llbracket B(f(x)) C(x) \rrbracket' : \forall x (B'(f(x)) \vee (B(f(x)) \wedge C'(x)))$ est vrai dans $T_0^2(\emptyset)$ car :

– pour $x=a$ $D(a)$ est vrai dans $T_0^1(\emptyset)$ donc $B(f(a))$ est vrai dans $T_0^2(\emptyset)$ (a_3);

Avec $C'(a)$ vrai dans $T^1(\emptyset)$ donc dans $T_0^2(\emptyset)$ on a bien $B(f(a)) \wedge C'(a)$ vrai dans $T_0^2(\emptyset)$.

– pour $x \neq a$ on a $D'(x)$ vrai dans $T_0^1(\emptyset)$ donc, en utilisant (a_4) on a $B'(f(x))$.

$\llbracket D(y) \rrbracket : \exists y D(y)$ est vrai dans $T_0^1(\emptyset)$ d'après (a_5).

$\llbracket A(x) \rrbracket = \exists x A(x)$ est vrai dans $T_0^3(\emptyset)$ d'après (a_1).

Démonstration du lemme : On raisonne par récurrence sur i .

Le lemme est vrai pour $i=0$ car tout est vide.

Supposons le lemme vrai pour tout $i < i_0$ et montrons le pour i_0 .

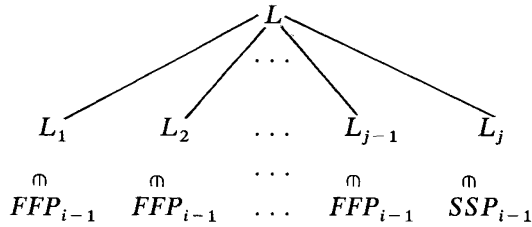
– Soit $L \in SSP_{i_0}$.

Montrons que $\llbracket L \rrbracket$ est vrai dans $T_0^i(\emptyset)$.

Si $L \in SSP_{i-1}$, c'est évident d'après l'hypothèse de récurrence. Nous supposons donc que $L \notin SSP_{i-1}$.

1^{er} cas : $L \in SSP_{i_0}$ a été obtenu par R_4 .

On a le schéma suivant :



Par hypothèse :

$$[[L_1]]', [[L_2]]', \dots, [[L_{j-1}]]', [[L_j]]'$$

sont vraies dans $T_\theta^{i-1}(\emptyset)$.

Les conditions sur les longueurs des L_r signifient que les règles utilisées pour passer de L à L_1 , de L à L_2 , \dots , de L à L_j ne sont pas toutes des règles sans queue.

Dans l'axiome de Ax_θ correspondant au premier but de la liste L , il existe une condition (celle notée $B'_1 \wedge B'_2 \wedge \dots \wedge B'_{j-1} \wedge B_j$ dans la définition du paragraphe 4) qui se trouve satisfaite dans $T_\theta^{i-1}(\emptyset)$ et donc L est vrai dans $T_\theta^i(\emptyset)$.

Les autres cas se traitent de la même manière, la forme des axiomes de Ax_θ ayant été choisie pour que, justement, le passage des conditions sur les L_r et L'_r aux conditions sur les B_r et B'_r des règles puisse se faire sans difficulté.

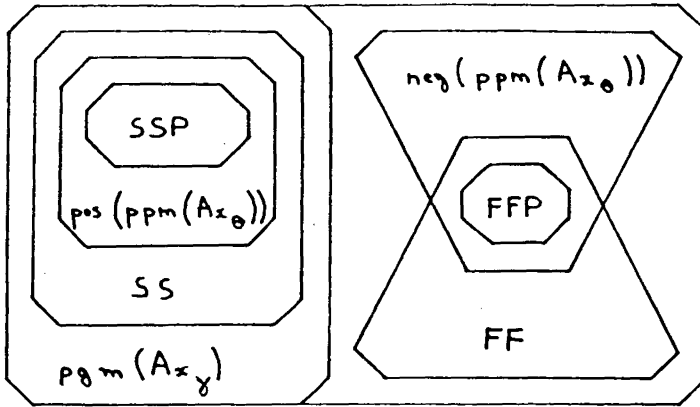
Fin de la démonstration du lemme.

Fin de la démonstration du théorème 3.

8. SCHEMA DE DECOPAGE DE LA BASE DE HERBRAND ET EXEMPLES COMPLEMENTAIRES

La base de Herbrand h (ensemble des atomes sans variables construit à partir du vocabulaire du programme P) contient diverses parties dont l'étude précédente n'est en fait que la mise en rapport des unes vis à vis des autres.

Le schéma des diverses inclusions que nous avons rencontrées est le suivant :



Exemple complémentaire 1 :

$$\begin{array}{l}
 P \left\{ \begin{array}{l}
 A(x) \leftarrow B(x) C(x) \\
 B(x) \leftarrow B(x) \\
 B(a) \leftarrow \\
 B(c) \leftarrow \\
 C(a) \leftarrow \\
 C(b) \leftarrow
 \end{array} \right. \\
 Ax_\beta \left\{ \begin{array}{l}
 A(x_1) \leftarrow \exists x (x_1 = x \wedge B(x) \wedge C(x)) \\
 B(x_1) \leftarrow \exists x (x = x_1 \wedge B(x)) \vee x_1 = a \vee x_1 = c \\
 C(x_1) \leftarrow x_1 = a \vee x_1 = b \\
 [Ax_\beta]^+ \\
 Ax_\delta \left\{ \begin{array}{l}
 A'(x_1) \leftarrow \forall x (x_1 \neq x \vee B'(x) \vee C'(x)) \\
 B'(x_1) \leftarrow \forall x (x_1 \neq x \vee B'(x)) \wedge x_1 \neq a \wedge x_1 \neq c \\
 C'(x_1) \leftarrow x_1 \neq a \wedge x_1 \neq b
 \end{array} \right. \\
 Ax_0 \left\{ \begin{array}{l}
 A(x_1) \leftarrow \exists x (x_1 = x \wedge B(x) \wedge C(x)) \\
 A'(x_1) \leftarrow \forall x (x_1 \neq x \vee B'(x) \vee (B(x) \wedge C'(x))) \\
 B(x_1) \leftarrow \exists x (x_1 = x \wedge B(x)) \\
 \quad \vee (\forall x (x_1 \neq x \vee B'(x)) \wedge x_1 = a) \\
 \quad \vee (\forall x (x_1 \neq x \vee B'(x)) \wedge x \neq a \wedge x_1 = c) \\
 B'(x_1) \leftarrow \forall x (x_1 \neq x \vee B'(x)) \wedge x_1 \neq a \wedge x_1 \neq c \\
 C(x_1) \leftarrow x_1 = a \vee x_1 = b \\
 C'(x_1) \leftarrow x_1 \neq a \wedge x_1 \neq b
 \end{array} \right.
 \end{array}$$

$$SS = \{ A(a), B(a), B(c), C(a), C(b) \}$$

$$FF = \{ A(c), C(c) \}$$

$$SSP = \{ C(a), C(b) \}$$

$$FFP = \{ C(c) \}$$

$$T_8^1(\emptyset) = \{ B(a), B(c), C(a), C(b), C'(c) \}$$

$$T_8^\alpha(\emptyset) = \underbrace{\{ A(a), B(a), B(c), C(a), C(b) \}}_{SS} \underbrace{\{ A'(c), C'(c) \}}_{FF'}$$

pour tout $\alpha \geq 2$.

$$T_0^\alpha(\emptyset) = \underbrace{\{ C(a), C(b) \}}_{SSP} \underbrace{\{ C'(c) \}}_{FFP'}$$

pour tout $\alpha \geq 1$.

$T_8 \uparrow \omega$ donne $SS \cup FF'$.

$T_0 \uparrow \omega$ donne $SSP \cup FFP'$.

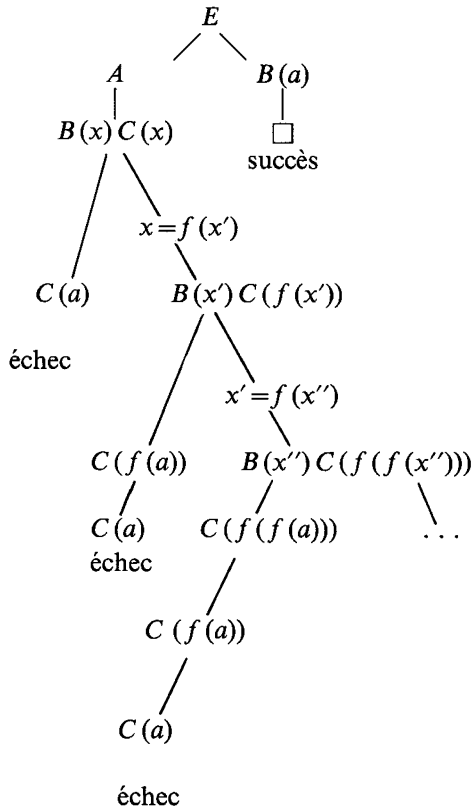
Dans cet exemple, l'opérateur T_0 est le seul des trois opérateurs introduits qui rend vraiment compte des calculs d'un interpréteur PROLOG standard. De même le système d'axiomes Ax_0 est le seul (des cinq que nous avons considérés) qui dans cet exemple indique l'ensemble de succès et l'ensemble d'échec d'un interpréteur standard. Les opérateurs T_β et T_γ et les systèmes d'axiomes Ax_β et Ax_γ donnent des points fixes et des modèles trop gros qui ne correspondent pas réellement au comportement de PROLOG.

Exemple complémentaire 2 : Exemple d'une situation où :

$$T_0 \uparrow (\omega + 1) \neq T_0 \uparrow \omega$$

$$P \left\{ \begin{array}{l} E \leftarrow A \\ E \leftarrow B(a) \\ A \leftarrow B(x) C(x) \\ B(a) \leftarrow \\ B(f(x)) \leftarrow B(x) \\ C(f(x)) \leftarrow C(x) \end{array} \right.$$

$$\left. \begin{aligned}
 & E \Leftarrow A \vee (A' \wedge B(a)) \\
 & E' \Leftarrow A' \wedge B'(a) \\
 & A \Leftarrow \exists x (B(x) \wedge C(x)) \\
 & A' \Leftarrow \forall x (B'(x) \vee (B(x) \wedge C'(x))) \\
 & B(x_1) \Leftarrow x_1 = a \vee \exists x (x_1 = f(x) \wedge B(x)) \\
 & B'(x_1) \Leftarrow x_1 \neq a \wedge \forall x (x_1 \neq f(x) \vee B'(x)) \\
 & C(x_1) \Leftarrow \exists x (x_1 = f(x) \wedge C(x)) \\
 & C'(x_1) \Leftarrow \forall x (x_1 \neq f(x) \vee C'(x))
 \end{aligned} \right\} Ax_0$$



Voici maintenant le calcul détaillé des $T_0^n(\emptyset)$:

$$T_0^1(\emptyset) = \{ C'(a), B(a) \}$$

$$T_0^2(\emptyset) = \{ C'(a), C'(f(a)), B(a), B(f(a)) \}$$

$$T_0^n(\emptyset) = \{ C'(a), \dots, C'(f^{n-1}(a)), B(a), \dots, B(f^{n-1}(a)) \}$$

$$\begin{aligned}
 T_0^\omega(\emptyset) &= \{ C'(f^i(a)), B(f^i(a)) \mid i \in \mathbf{N} \} \\
 T_0^{\omega+1}(\emptyset) &= T_0^\omega(\emptyset) \cup \{ A' \} \\
 T_0^{\omega+2}(\emptyset) &= T_0^{\omega+1}(\emptyset) \cup \{ E \} \\
 T_0^\alpha(\emptyset) &= T_0^{\omega+2}(\emptyset), \alpha \geq \omega + 2.
 \end{aligned}$$

Les interpréteurs standards comme les interpréteurs en largeur et équitables calculent le plus souvent une limite d'itérés ($T_0 \uparrow \omega$ pour les interpréteurs en largeur et équitables, $T_0 \uparrow \omega$ pour les interpréteurs standards).

Malheureusement cette limite n'est pas forcément un plus petit point fixe. Dans le cas des interpréteurs en largeur et équitables il y a des exemples bien connus ([Apt 82], ou propriété 3 du paragraphe 5). Dans le cas des interpréteurs standards le programme donné ici est un exemple : A' et E appartiennent au plus petit modèle de Ax_0 mais A ne donne pas un échec et E ne donne pas un succès (avec un interpréteur standard) car A' et E ne sont pas dans $T_0 \uparrow \omega$.

Exemple complémentaire 3 : Exemple d'une situation où :

$$\begin{aligned}
 & T_0 \uparrow \omega \stackrel{\neq}{\supseteq} SSP \cup FFP' \\
 & \left. P \right\} \begin{aligned} & A \leftarrow B(x)C(x) \\ & B(a) \leftarrow \\ & B(b) \leftarrow \\ & C(b) \leftarrow \\ & C(a) \leftarrow C(a) \end{aligned} \\
 & \left. Ax_0 \right\} \begin{aligned} & A \leftarrow \exists x (B(x) \wedge C(x)) \\ & A' \leftarrow \forall x (B'(x) \vee (B(x) \wedge C'(x))) \\ & B(x_1) \leftarrow x_1 = a \vee (x_1 \neq a \wedge x_1 = b) \\ & B'(x_1) \leftarrow x_1 \neq a \wedge x_1 \neq b \\ & C(x_1) \leftarrow x_1 = b \vee (x_1 \neq b \wedge (x_1 = a \wedge C(a))) \\ & C'(x_1) \leftarrow x_1 \neq b \wedge (x_1 \neq a \vee C'(a)) \end{aligned} \\
 & SSP = \{ B(a), B(b), C(b) \} \\
 & FFP = \emptyset \\
 & T_0 \uparrow 0 = \emptyset \\
 & T_0 \uparrow 1 = \{ B(a), B(b), C(b) \}
 \end{aligned}$$

$$T_0 \uparrow 2 = \{ A, B(a), B(b), C(b) \}$$

$$T_0 \uparrow \alpha = T_0 \uparrow 2 \text{ pour tout ordinal } \alpha \geq 2.$$

Ici nous sommes dans une situation où ce que calcule un interpréteur standard est strictement inclus dans $T_0 \uparrow \omega$ (on a $A \in T_0 \uparrow \omega$, bien que $A \notin SSP$).

Ax_0 donne une meilleure approximation du sens logique d'un programme confié à un interpréteur standard que Ax_α , Ax_β , Ax_γ et Ax_δ (voir théorèmes 2, 3 et exemple complémentaire 1).

Cependant deux difficultés demeurent :

– celle due à la non continuité de T_0 (illustrée dans l'exemple complémentaire 2);

– celle due à certains bouclages de l'interpréteur non pris en compte dans Ax_0 (illustrée dans l'exemple complémentaire 3).

L'utilisation du cut introduit encore d'autres difficultés (voir [Del 87]).

BIBLIOGRAPHIE

- [Apt 82] K. APT et M. VAN EMDEN, *Contribution to the Theory of Logic Programming*, Journal of the Association for Computing Machinery, 29.3, 1982, p. 841-842.
- [Bla 82 a] H. A. BLAIR, *The Undecidability of Two Completeness Notions for the "Negation as Failure Rule" in Logic Programming*, Proceeding of the first Logic programming Conference, Van Caneghem éd., Marseille, 1982, p. 163-168.
- [Bla 82 b] H. A. BLAIR, *The Recursion Theoretic Complexity of the Semantics of Predicate Logic as a Programming Language*, Information and Control, vol. 54, 1982, p. 25-47).
- [Cla 78] K. L. CLARK, *Negation as Failure. Logic and Data Bases*, H. GALLAIRE et J. MINKER éd., Plenum Press, New York, 1978, p. 293-324.
- [Cla 79] K. L. CLARK, *Predicate Logic as a Computational Formalism*, Res. Rep., Dep of Computing, Imperial College, London, 1979.
- [Col 82] A. COLMERAUER, *Prolog II, Manuel de Référence et Modèle Théorique*, Groupe d'Intelligence Artificielle, Univ. d'Aix-Marseille, 1982.
- [Del 86 a] J. P. DELAHAYE, *Outils logiques pour l'intelligence artificielle*, Eyrolles, Paris, 1986.
- [Del 86 b] J. P. DELAHAYE et P. PARADINAS, *Définitions de stratégies équitables en programmation logique*, Actes du Séminaire de Programmation en Logique de Tregastel, 1986, p. 7-24.
- [Del 86 c] J. P. DELAHAYE, *Sémantique logique et dénotationnelle des interpréteurs Prolog*, Note IT 1986, Université des Sciences et Techniques de Lille.
- [Del 87] J. P. DELAHAYE, *Effet de l'utilisation du coupe choix sur la sémantique déclarative de Prolog*, Note IT 1987, Université des Sciences et Techniques de Lille.
- [Der 83] P. DERANSART, *An Operational Algebraic Semantics of Prolog Programs*, Actes du Séminaire de « Programmation en Logique », Mehmet Dincbas éd., C.N.E.T., Lannion, 1983.

- [Fit 85 a] M. FITTING, *A Deterministic Prolog Fixpoint Semantics* J. of Logic Programming, vol. 2, 1985, p. 111-118.
- [Fit 85 b] M. FITTING, *A Kripke-Kleene Semantics for Logic Programs*, J. of Logic Programming, vol. 4, 1985, p. 295-312.
- [Fla 86] T. FLANNAGAM, *The Consistency of Negation as Failure*, J. Logic Programming, vol. 2, 1986, p. 93-114.
- [Hil 74] R. HILL, *Lush-Resolution and its Completeness*, Dep. of Artificial Intelligence, Univ. of Edinburgh, 1974.
- [JLL 83] J. JAFFAR, J. L. LASSEZ et L. LLOYD, *Completeness of the Negation as Failure Rule*, IJCAI-83 Karlsruhe, 1983, p. 500-506.
- [Jon 84] N. D. JONES et A. MYCROFT, *Stepwise Development of Operational Semantics for PROLOG*, I.E.E.E., 1984, p. 182-288.
- [Las 84] J. L. LASSEZ et M. J. MAHER, *Closures and Fairness in Semantics of Programming Logic*, T.C.S., 29, 1984, p. 167-184).
- [Llo 84] J. LLOYD, *Foundation of Logic Programming*, Springer-Verlag, New York, 1984.
- [Rei 78] R. REITER, *On Closed world Data Bases. Logic and Data Bases*, H. GALLAIRE et J. MINKER, éd., Plenum Press, New York, 1978, p. 55-76.
- [Rob 65] J. A. ROBINSON, *A Machine Oriented Logic Based on the Resolution Principle*, J. of A.C.M., vol. 12, 1965.
- [She 84] J. C. SHEPHERDSON, *Negation as Failure I and II*, J. Logic Programming, vol. 1, 1984, p. 51-79, vol. 3, 1985, p. 185-202.
- [VEK 76] M. H. VAN EMDEN et R. A. KOWALSKI, *The Semantics of Predicate Logic as a Programming Language*, J.A.C.M., vol. 23-4, 1976, p. 733-742
- [VEA 85] M. H. VAN EMDEN et M. A. NAIT ABDALLAH, *Top-Down Semantics of Fair Computations of Logic Programs*, J. of Logic Programming, vol. 1, 1985, p. 67-75.