

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

N° d'ordre : 653.

50376
1990
243

THÈSE

Nouveau Régime

présentée à

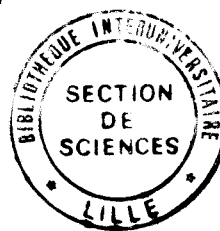
L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

François DENIS



CONTRIBUTION A L'ETUDE DES SEMANTIQUES AXIOMATIQUES DE PROLOG

Soutenue le 18 Décembre 1990 devant la Commission d'Examen composée de :

M. NIVAT
G. FERRAND
J.-P. JOUANNAUD
B. COURCELLE
M. DAUCHET
J.-P. DELAHAYE
P. DEVIENNE

.....
.....
.....
.....
.....
.....
.....

Président
Rapporteur
Rapporteur
Examinateur
Examinateur
Examinateur
Examinateur

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M.H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT
ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER,
DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF,
LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL,
PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PAREAU, J. LOMBARD, M. MIGEON, J. CORTOIS.

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES
DE LILLE FLANDRES ARTOIS

M. A. DUBRULLE.

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	Electronique
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie Appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du Solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean-Pierre	Chimie-Physique
M. BOSCOQ Denis	Probabilités
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie Végétale
M. BREZINSKI Claude	Analyse Numérique

M. BRIDOUX Michel
 M. CELET Paul
 M. CHAMLEY Hervé
 M. COEURE Gérard
 M. CORDONNIER Vincent
 M. DAUCHET Max
 M. DEBOURSE Jean-Pierre
 M. DHAINAUT André
 M. DOUKHAN Jean-Claude
 M. DYMENT Arthur
 M. ESCAIG Bertrand
 M. FAURE Robert
 M. FOCT Jacques
 M. FRONTIER Serge
 M. GRANELLE Jean-Jacques
 M. GRUSON Laurent
 M. GUILLAUME Jean
 M. HECTOR Joseph
 M. LABLACHE-COMBIER Alain
 M. LACOSTE Louis
 M. LAVEINE Jean-Pierre
 M. LEHMANN Daniel
 Mme LENOBLE Jacqueline
 M. LEROY Jean-Marie
 M. LHOMME Jean
 M. LOMBARD Jacques
 M. LOUCHEUX Claude
 M. LUCQUIN Michel
 M. MACKE Bruno
 M. MIGEON Michel
 M. PAQUET Jacques
 M. PETIT Francis
 M. POUZET Pierre
 M. PROUVOST Jean
 M. RACZY Ladislav
 M. SALMER Georges
 M. SCHAMPS Joel
 M. SEQUIER Guy
 M. SIMON Michel
 Melle SPIK Geneviève
 M. STANKIEWICZ François
 M. TILLIEU Jacques
 M. TOULOTTE Jean-Marc
 M. VIDAL Pierre
 M. ZEYTOUNIAN Radyadour

Chimie-Physique
 Géologie Générale
 Géotechnique
 Analyse
 Informatique
 Informatique
 Gestion des Entreprises
 Biologie Animale
 Physique du Solide
 Mécanique
 Physique du Solide
 Mécanique
 Métallurgie
 Ecologie Numérique
 Sciences Economiques
 Algèbre
 Microbiologie
 Géométrie
 Chimie Organique
 Biologie Végétale
 Paléontologie
 Géométrie
 Physique Atomique et Moléculaire
 Spectrochimie
 Chimie Organique Biologique
 Sociologie
 Chimie Physique
 Chimie Physique
 Physique Moléculaire et Rayonnements Atmosph.
 E.U.D.I.L.
 Géologie Générale
 Chimie Organique
 Modélisation - calcul Scientifique
 Minéralogie
 Electronique
 Electronique
 Spectroscopie Moléculaire
 Electrotechnique
 Sociologie
 Biochimie
 Sciences Economiques
 Physique Théorique
 Automatique
 Automatique
 Mécanique

PROFESSEURS - 2ème CLASSE

M. ALLAMANDO Etienne
 M. ANDRIES Jean-Claude
 M. ANTOINE Philippe
 M. BART André
 M. BASSERY Louis

Composants Electroniques
 Biologie des organismes
 Analyse
 Biologie animale
 Génie des Procédés et Réactions Chimiques

Mme BATTIAU Yvonne
 M. BEGUIN Paul
 M. BELLET Jean
 M. BERTRAND Hugues
 M. BERZIN Robert
 M.BKOUCHE Rudolphe
 M. BODARD Marcel
 M. BOIS Pierre
 M. BOISSIER Daniel
 M. BOIVIN Jean-Claude
 M. BOUQUELET Stéphane
 M. BOUQUIN Henri
 M. BRASSELET Jean-Paul
 M. BRUYELLE Pierre
 M. CAPURON Alfred
 M. CATTEAU Jean-Pierre
 M. CAYATTE Jean-Louis
 M. CHAPOTON Alain
 M. CHARET Pierre
 M. CHIVE Maurice
 M. COMYN Gérard
 M. COQUERY Jean-Marie
 M. CORIAT Benjamin
 Mme CORSIN Paule
 M. CORTOIS Jean
 M. COUTURIER Daniel
 M. CRAMPON Norbert
 M. CROSNIER Yves
 M. CURGY Jean-Jacques
 Melle DACHARRY Monique
 M. DEBRABANT Pierre
 M. DEGAUQUE Pierre
 M. DEJAEGER Roger
 M. DELAHAYE Jean-Paul
 M. DELORME Pierre
 M. DELORME Robert
 M. DEMUNTER Paul
 M. DENEL Jacques
 M. DE PARIS Jean Claude
 M. DEPREZ Gilbert
 M. DERIEUX Jean-Claude
 Melle DESSAUX Odile
 M. DEVRAINNE Pierre
 Mme DHAINAUT Nicole
 M. DHAMELIN COURT Paul
 M. DORMARD Serge
 M. DUBOIS Henri
 M. DUBRULLE Alain
 M. DUBUS Jean-Paul
 M. DUPONT Christophe
 Mme EVRARD Micheline
 M. FAKIR Sabah
 M. FAUQUAMBERGUE Renaud

Géographie
 Mécanique
 Physique Atomique et Moléculaire
 Sciences Economiques et Sociales
 Analyse
 Algèbre
 Biologie Végétale
 Mécanique
 Génie Civil
 Spectroscopie
 Biologie Appliquée aux enzymes
 Gestion
 Géométrie et Topologie
 Géographie
 Biologie Animale
 Chimie Organique
 Sciences Economiques
 Electronique
 Biochimie Structurale
 Composants Electroniques Optiques
 Informatique Théorique
 Psychophysiologie
 Sciences Economiques et Sociales
 Paléontologie
 Physique Nucléaire et Corpusculaire
 Chimie Organique
 Tectonique Géodynamique
 Electronique
 Biologie
 Géographie
 Géologie Appliquée
 Electronique
 Electrochimie et Cinétique
 Informatique
 Physiologie Animale
 Sciences Economiques
 Sociologie
 Informatique
 Analyse
 Physique du Solide - Cristallographie
 Microbiologie
 Spectroscopie de la réactivité Chimique
 Chimie Minérale
 Biologie Animale
 Chimie Physique
 Sciences Economiques
 Spectroscopie Hertziennne
 Spectroscopie Hertziennne
 Spectrométrie des Solides
 Vie de la firme (I.A.E.)
 Génie des procédés et réactions chimiques
 Algèbre
 Composants électroniques

M. FONTAINE Hubert
 M. FOUQUART Yves
 M. FOURNET Bernard
 M. GAMBLIN André
 M. GLORIEUX Pierre
 M. GOBLOT Rémi
 M. GOSSELIN Gabriel
 M. GOUDMAND Pierre
 M. GOURIEROUX Christian
 M. GREGORY Pierre
 M. GREMY Jean-Paul
 M. GREVET Patrice
 M. GRIMBLOT Jean
 M. GUILBAULT Pierre
 M. HENRY Jean-Pierre
 M. HERMAN Maurice
 M. HOUDART René
 M. JACOB Gérard
 M. JACOB Pierre
 M. Jean Raymond
 M. JOFFRE Patrick
 M. JOURNAL Gérard
 M. KREMBEL Jean
 M. LANGRAND Claude
 M. LATTEUX Michel
 Mme LECLERCQ Ginette
 M. LEFEBVRE Jacques
 M. LEFEBVRE Christian
 Mlle LEGRAND Denise
 Mlle LEGRAND Solange
 M. LEGRAND Pierre
 Mme LEHMANN Josiane
 M. LEMAIRE Jean
 M. LE MAROIS Henri
 M. LEROY Yves
 M. LESENNE Jacques
 M. LHENAFF René
 M. LOCQUENEUX Robert
 M. LOSFELD Joseph
 M. LOUAGE Francis
 M. MAHIEU Jean-Marie
 M. MAIZIERES Christian
 M. MAURISSON Patrick
 M. MESMACQUE Gérard
 M. MESSELYN Jean
 M. MONTEL Marc
 M. MORCELLET Michel
 M. MORTREUX André
 Mme MOUNIER Yvonne
 Mme MOUYART-TASSIN Annie Françoise
 M. NICOLE Jacques
 M. NOTELET Francis
 M. PARSY Fernand

Dynamique des cristaux
 Optique atmosphérique
 Biochimie Structurale
 Géographie urbaine, industrielle et démog.
 Physique moléculaire et rayonnements Atmos.
 Algèbre
 Sociologie
 Chimie Physique
 Probabilités et Statistiques
 I.A.E.
 Sociologie
 Sciences Economiques
 Chimie Organique
 Physiologie animale
 Génie Mécanique
 Physique spatiale
 Physique atomique
 Informatique
 Probabilités et Statistiques
 Biologie des populations végétales
 Vie de la firme (I.A.E.)
 Spectroscopie hertzienne
 Biochimie
 Probabilités et statistiques
 Informatique
 Catalyse
 Physique
 Pétrologie
 Algèbre
 Algèbre
 Chimie
 Analyse
 Spectroscopie hertzienne
 Vie de la firme (I.A.E.)
 Composants électroniques
 Systèmes électroniques
 Géographie
 Physique théorique
 Informatique
 Electronique
 Optique-Physique atomique
 Automatique
 Sciences Economiques et Sociales
 Génie Mécanique
 Physique atomique et moléculaire
 Physique du solide
 Chimie Organique
 Chimie Organique
 Physiologie des structures contractiles
 Informatique
 Spectrochimie
 Systèmes électroniques
 Mécanique

M. PECQUE Marcel
M. PERROT Pierre
M. STEEN Jean-Pierre

Chimie organique
Chimie appliquée
Informatique

Je suis très reconnaissant à Maurice Nivat qui me fait l'honneur de présider le jury

Je tiens à remercier

- Gérard Ferrand et Jean-Pierre Jouannaud qui ont accepté d'être rapporteur de ma thèse, pour leurs nombreuses remarques et suggestions

- Bruno Courcelle d'avoir accepté de participer au jury

- Max Dauchet pour l'intérêt porté à mon travail mais aussi, pour m'avoir appris des choses importantes (et non formalisables) sur ce qu'est la recherche

- Philippe Devienne pour la relecture très minutieuse de cette thèse, assortie de nombreux conseils et suggestions

- Jean-Paul Delahaye, tout particulièrement, à qui je dois d'avoir mené à bien ce travail et qui a su me communiquer son enthousiasme pour la recherche. Les très nombreuses discussions qui ont rythmé l'élaboration de cette thèse (débordant souvent largement la programmation logique) ont constitué pour moi un apprentissage d'une valeur inestimable

- Henri Glanc qui a assuré la reprographie de cette thèse

- et bien sûr, l'équipe du LIFL qui, par la qualité de ses enseignements et la chaleur de son accueil a permis ma reconversion.

Table des matières

Introduction	1
I Préliminaires logiques	12
I.1 Généralités	12
I.2 Treillis et opérateurs	13
I.3 Modèles de Herbrand	14
I.4 Théorie Equationnelle de Clark	17
I.5 Programmes généraux	20
II Préliminaires à propos du langage Prolog	26
II.1 Programmes Prolog et langages du premier ordre	26
II.2 Unification	27
II.3 Résolution SLD	31
II.4 Sémantiques opérationnelles d'un programme Prolog	33
II.4.1 La sémantique opérationnelle d'un interpréteur Prolog idéal	34
II.4.2 La sémantique opérationnelle d'un interpréteur Prolog standard	34
II.4.3 La sémantique opérationnelle standard finie	35
II.4.4 La sémantique opérationnelle invariante finie	36
II.4.5 Remarques	37
II.5 Sémantiques axiomatiques d'un programme Prolog	39
II.5.1 La traduction clausale	41
II.5.2 Le programme complété de Clark	42
II.5.3 La traduction SSI dédoublée	43
II.5.4 La traduction P_θ	45
II.5.5 Les sémantiques axiomatiques de Prolog en logique trivaluée	47
II.5.6 Les sémantiques axiomatiques de Prolog en logique linéaire, en logique modale et en logique intuitionniste	48
III La sémantique standard	49
III.1 Programmes décrits par $T_\theta \uparrow \omega$	49
III.1.1 Conditions nécessaires pour que $SS_{st} \cup FF'_{st} = T_\theta \uparrow \omega$	50
III.1.2 Une classe de programmes pour laquelle $SS_{st} \cup FF'_{st} = T_\theta \uparrow \omega$	55
III.2 La traduction axiomatique standard triple	58
III.2.1 Définition de la traduction standard triple	58
III.2.2 $T_{st} \uparrow \omega$ -correction et P_{st}^{Her} -correction	62
III.2.3 $T_{st} \uparrow \omega$ -complétude et P_{st}^{Her} -complétude	66
III.3 Indécidabilité des hypothèses H_1 , H_2 et H_3	72

IV	Les sémantiques finies	75
IV.1	La sémantique standard finie	75
IV.1.1	Définition de la traduction axiomatique standard finie	75
IV.1.2	$T_{fst} \uparrow \omega$ -correction et P_{fst}^{Her} -correction	79
IV.1.3	$T_{fst} \uparrow \omega$ -complétude et P_{fst}^{Her} -complétude	83
IV.2	La sémantique invariante finie	87
IV.2.1	Définition de la traduction axiomatique invariante finie	87
IV.2.2	$T_f \uparrow \omega$ -correction et P_f^{Her} -correction.	90
IV.2.3	$T_f \uparrow \omega$ -complétude et P_f^{Her} -complétude.	94
IV.3.	Continuité des opérateurs T_θ , T_{st} , T_{fst} et T_f pour les programmes sans variable locale.	97
V	Un théorème de complétude	101
V.1	Introduction	101
V.2	Le théorème de complétude	103
V.2.1	Quelques éléments de la théorie des modèles	103
V.2.2	Dépliage d'une formule	105
V.2.3	Construction d'un modèle saturé d'un programme général positif	108
VI	Application aux sémantiques des programmes Prolog dans le cadre de la CET	116
VI.1	Complétude dans le cadre de la CET(L)	116
VI.1.1	La sémantique standard	117
VI.1.2	La sémantique standard finie	118
VI.1.3	La sémantique invariante finie	119
VI.2	Correction et complétude dans le cadre de la CET(L_k)	120
VI.2.1	La sémantique standard	121
VI.2.2	La sémantique standard finie	122
VI.2.3	La sémantique invariante finie	123
VI.3	Correction (et complétude) dans le cadre de la CET(L)	124
VI.3.1	Introduction	124
VI.3.2	La sémantique standard finie	124
VI.3.3	La sémantique invariante finie	131
	Tableau récapitulatif	138
VII	Logique trivaluée et dédoublement des symboles de prédicat	139
VII.1	Traduction dédoublée-trivaluée	140
VII.2	Programmes trivalués	145
VII.3	Théorèmes de complétude en logique trivaluée	153
VII.4	Complétude de la résolution SLDNF	158
	Perspectives	161
	Bibliographie	163
	Index	169

Introduction

Les recherches sur la démonstration automatique de théorèmes ont débutées dès les années 30 avec les travaux de Herbrand [Her 30]. Elles se sont poursuivies à l'intérieur de la logique mathématique jusqu'au moment où, à la fin des années 60, les progrès de l'informatique (et plus particulièrement, l'augmentation de la vitesse de calcul et de la capacité mémoire des ordinateurs) ont permis d'implémenter efficacement les idées dégagées lors de ces recherches ([Dav 60, Gilm 60, Pra 60, Rob 65]).

Le problème initial de la **programmation logique** peut être posé en ces termes : concevoir un système informatique calculant automatiquement (et efficacement) les conséquences (où les conséquences ayant une forme donnée) d'un système d'axiomes; ce système doit également permettre de décider si une formule est ou non conséquence de ce système d'axiomes (au moins pour certains types de systèmes d'axiomes puisque l'on sait que le problème de la décision du calcul des prédicats est indécidable).

Un tel système informatique constituerait un outil de **programmation purement déclarative**. Kowalski [Kow 71] a remarqué qu'un algorithme est la réunion de deux composantes disjointes :

- la **partie logique** qui énonce en quoi consiste le problème à résoudre
- la **partie contrôle** qui indique comment le résoudre.

Un langage de programmation purement déclarative doit alors "permettre au programmeur de ne spécifier que le sens logique d'un algorithme, les aspects procéduraux étant pris en charge automatiquement par le système de programmation logique" [Llo 87].

Prenant appui sur les travaux de Robinson sur la **résolution**, Colmerauer a créé le langage Prolog ([Col 73]) dont le premier interpréteur a été implémenté en 1972 par Roussel ([Col 73, Rou 75]).

L'idée à la base de Prolog est que la logique¹ a une interprétation procédurale et peut être utilisée comme langage de programmation. Plus précisément :

- un programme Prolog est une conjonction de **clauses définies** (c'est-à-dire de formules de la forme $A \Leftarrow B_1 \wedge \dots \wedge B_n$ ($n \geq 0$)) qui peuvent être interprétées comme des **définitions de procédures**

- un **atome** A peut être considéré comme une **procédure** qui indique en sortie si A ou $\neg A$ est conséquence logique du programme.

Depuis, de nombreux systèmes n'utilisant qu'une partie des fonctionnalités de Prolog (il est fréquent de ne considérer pour le traitement des Bases de Données que des langages logiques sans symbole de fonction) ou au contraire, visant à étendre et à généraliser les formules à la base des programmes Prolog (en incorporant un module de calcul algébrique, en admettant des littéraux négatifs dans les corps des clauses, en permettant l'utilisation de "prédicats de contrôle", ...) ont été développés. Mais la plupart ont un noyau commun appelé Prolog pur.

La création de Prolog a été le point de départ d'un nombre important de travaux que l'on peut regrouper sous le thème : **Etude de la sémantique du langage Prolog** ou plus généralement, **Etude de la sémantique des langages à base de règles**.

¹Où du moins une partie : la logique des clauses de Horn.

La sémantique de Prolog traite des problèmes suivants :

- caractériser ce que calcule **effectivement** un interpréteur Prolog donné.
- caractériser **ce que l'on souhaite** qu'un interpréteur Prolog calcule à partir d'un programme donné (c'est-à-dire, caractériser le sens "intentionnel" (intended meaning) d'un programme)
- définir des classes de programmes les plus générales possibles pour lesquelles un interpréteur donné calcule effectivement ce que l'on en attend.

Cette définition est encore très générale et regroupe des travaux très divers suivant ce que l'on entend par "caractériser".

Citons en particulier, les travaux sur les **sémantiques opérationnelles et dénotationnelles de Prolog** [Arb 87, Bil 88, Deb 87, Jon 84, Ric 89] dans lesquels sont élaborées des structures mathématiques qui décrivent souvent de manière très précise ce que calcule un interpréteur Prolog. Ces travaux peuvent s'appliquer aux problèmes d'implémentation, de définition de normes, aux liens avec d'autres langages de programmation mais ces descriptions sont souvent très éloignées de ce qui a motivé initialement la création de Prolog : le lien conséquence logique - calcul effectif.

Nous avons choisi de placer notre travail dans le cadre plus restreint de la **sémantique logique** et plus précisément, de la **sémantique axiomatique** de Prolog. Nous pouvons alors préciser la définition ci-dessus.

La sémantique axiomatique de Prolog traite des problèmes suivants : déterminer des systèmes d'axiomes dont les conséquences logiques (d'une certaine forme) représentent :

- ce que calcule effectivement un interpréteur Prolog donné
- ce que l'on souhaite qu'un interpréteur Prolog calcule.

On peut noter un renversement par rapport au problème initial posé par la programmation logique : on cherchait à élaborer un système informatique devant calculer des conséquences logiques et l'on cherche maintenant à caractériser de manière logique ce que calcule un système informatique existant. Mais il s'agit toujours d'approfondir les rapports entre la logique et le calcul et ces études doivent permettre d'améliorer les techniques de calcul existantes mais aussi de prendre en compte, d'un point de vue logique, les difficultés réelles (capacité mémoire et vitesse de traitement limitées) liées à l'implémentation.

Un assez grand nombre de travaux ont été consacrés à la sémantique axiomatique des programmes définis (parmi les plus classiques, citons [Apt 82, Cla 78, Hil 74, Las 84, Llo 87]). Nous rappelons les principaux résultats déjà obtenus et présentons un exemple qui nous permettra d'exposer le sujet de notre travail¹.

Soit P un programme Prolog (défini) (Déf. II.1).

Si P_{SSI}^{CET2} est le complété de Clark de P (§ II.4) et si A est un atome clos. Alors :

¹Les notations que nous utilisons ici sont définies avec précision dans le chapitre II.

² CET pour Clark's Equational Theory.

- $P_{SSI}^{CET} \models A$ ssi un arbre SLD de racine :- A (§ II.3) contient la clause vide (et alors, tous la contiennent) [Apt 82]
- $P_{SSI}^{CET} \models \neg A$ ssi un arbre SLD **équitable** de racine :- A (§ II.3) est un arbre d'échecs fini (et alors, tous les arbres SLD équitables sont aussi des arbres d'échecs fini) ([Cla 78, Jaf 83])

On conclut généralement de ces résultats que le complété de Clark d'un programme défini est une **caractérisation axiomatique exacte** (correcte et complète) de ce que calcule un interpréteur Prolog.

Pourtant, considérons le programme propositionnel P suivant :

$$\begin{cases} A :- A \\ A \\ B :- A, C \end{cases}$$

Le complété de Clark P_{SSI}^{CET} de ce programme est le système d'axiomes :

$$\begin{cases} A \\ B \Leftrightarrow A \wedge C \\ \neg C \end{cases}$$

Les littéraux A et $\neg B$ sont des conséquences logiques de P_{SSI}^{CET} et pourtant un interpréteur standard ne retournera jamais un succès pour le but :- A (car il bouclera sur la première règle de P). Il ne retournera pas non plus d'échec fini pour le but :- B pour la même raison.

En fait, les résultats énoncés plus haut ne s'appliquent qu'à des **interpréteurs idéaux**, c'est-à-dire :

- qui explorent tous les noeuds des arbres SLD qu'ils engendrent. C'est le cas par exemple s'ils sont dotés d'une stratégie d'exploration "en largeur d'abord"
- qui n'engendrent que des arbres SLD équitables.

Les interpréteurs standard sont dotés eux d'une stratégie d'exploration "en profondeur d'abord" et ils engendrent des arbres SLD construits à l'aide de la règle qui consiste à sélectionner systématiquement le premier atome d'un but. L'exemple précédent montre que ces arbres ne sont pas toujours équitables.

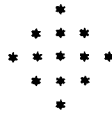
Il peut paraître un peu curieux que l'on se soit contenté de résultats théoriques ne concernant que les interpréteurs idéaux. La quantité de mémoire nécessaire varie pourtant de manière exponentielle suivant qu'un arbre est exploré en "profondeur d'abord" ou en "largeur d'abord", et il n'est pas envisageable pratiquement d'utiliser des interpréteurs idéaux. Il y a là une difficulté réelle, liée au calcul, et qui doit être prise en compte par la programmation logique.

La non-équité des arbres SLD engendrés est par contre un inconvénient qui semble plus lié à l'implémentation actuelle de Prolog que correspondre à un problème insurmontable. Les travaux de [Bek 86, Del 86, Jan 90] montrent qu'il est possible de concevoir des "interpréteurs équitables" moyennant une perte d'efficacité de l'ordre de 20%, ce qui est

assez raisonnable. Mais quoi qu'il en soit, ces interpréteurs ne sont pas couramment répandus et il semble naturel d'essayer de décrire d'un point de vue logique les stratégies de sélection d'un atome dans un but. Il est à noter d'ailleurs que ces stratégies de sélection sont prises en compte dans bon nombre d'études sur les programmes normaux (c.-à-d. avec des littéraux négatifs dans les corps des clauses); il est vrai que pour ce type de programmes, la question est encore plus cruciale puisque la résolution SLDNF n'est même pas correcte (en général) lorsqu'un littéral négatif non clos est sélectionné.

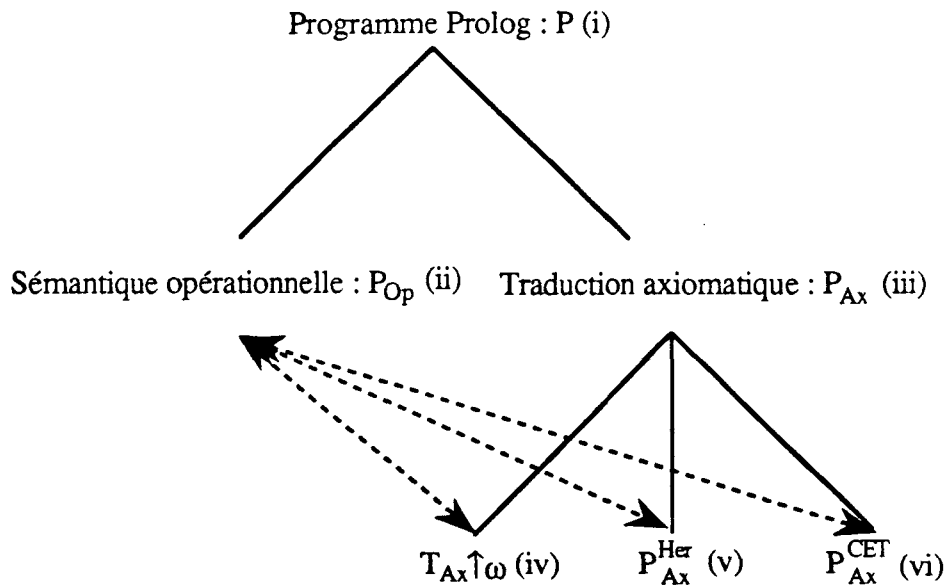
Ce sont ces constatations qui nous ont amené à nous poser la question :

" les interpréteurs standards admettent-ils une sémantique axiomatique ? "



Une grande partie de notre travail peut être commentée à partir du schéma suivant :

Schéma 1 :



(i) Les définitions permettant de décrire le fonctionnement des interpréteurs Prolog sont données au chapitre II. Précisons que nous nous intéressons, au moins dans les six premiers chapitres qu'à des **programmes Prolog purs**, c'est-à-dire dont les corps des clauses ne contiennent que des atomes (de prédicat différent de '=') du langage du premier ordre sous-jacent. En particulier, nous n'autorisons pas l'utilisation de la négation ni de "prédicats" non logiques tels que : CUT, VAR, ASSERT, etc ...

(ii) La notion de **sémantique opérationnelle** correspond à la notion de propriétés observables [Fal 89b], c'est-à-dire à la sélection de certaines propriétés du comportement opérationnels des programmes lorsqu'ils sont soumis à certains types d'interpréteurs. Nous montrons dans le chapitre II que l'on peut définir au moins quatre sémantiques opérationnelles pour un programme Prolog si l'on tient compte du mode d'implémentation des interpréteurs. Toutes les sémantiques que nous envisageons dans ce travail peuvent être représentées par des **ensembles d'atomes clos**. D'autres sémantiques, représentées par des classes d'équivalence d'ensembles d'atomes pouvant contenir des variables, ont été étudiées par [Der 87, Fal 89a, 89b, Fer 86, Lev 90] dans le cas d'interpréteurs idéaux. Nous prévoyons d'étendre nos travaux à ce type de sémantique ultérieurement.

Les quatre sémantiques que nous considérons sont les suivantes :

a) La **sémantique opérationnelle d'un interpréteur idéal** est représentée par les ensembles SS (Set of Success) et FF (Finite Failure set). Ce sont respectivement les ensembles d'atomes clos pour lesquels un interpréteur idéal retourne un succès (resp. un échec). L'exemple de programme donné ci-dessus montre qu'un atome de SS (resp. de FF) peut ne pas retourner un succès (resp. un échec fini) pour un interpréteur standard. Les résultats classiques ne portent que sur cette sémantique.

b) La **sémantique opérationnelle d'un interpréteur standard** est représentée par les ensembles SS_{st} (succès standards) et FF_{st} (échecs standards). Ce sont respectivement les ensembles d'atomes clos pour lesquels un interpréteur **standard** retourne un succès (resp. un échec).

Cette sémantique n'est pas entièrement satisfaisante d'un point de vue pratique dans la mesure où elle retient comme succès un atome pour lequel un interpréteur standard retourne un succès puis boucle indéfiniment. Un tel atome n'est pas réutilisable (sans précaution) pour définir d'autres prédicats, comme le montre l'exemple suivant :

$\begin{cases} A \\ A :- A \\ B :- A, C \end{cases}$	Un interpréteur standard boucle sur le but :-A et sur le but :-B
--	--

Ceci nous conduit à définir la notion de succès standard fini.

c) La **sémantique opérationnelle standard finie** est représentée par les ensembles SS_{fst} (succès standards finis) et FF_{st} . Un succès standard fini est un atome clos pour lequel un interpréteur standard retourne un succès puis s'arrête (au bout d'un certain temps). Nous pensons que cette sémantique est la mieux adaptée, d'un point de vue pratique, aux interpréteurs standards¹.

d) La **sémantique opérationnelle invariante finie** est représentée par les ensembles SS_f (succès invariants finis) et FF_f (échecs invariants finis). Ce sont respectivement les ensembles d'atomes clos pour lesquels **tout** interpréteur retourne un succès fini (resp. un échec fini). Cette sémantique est invariante par rapport à l'ordre des règles dans le programme et par rapport à l'ordre des atomes dans chaque clause. Elle paraît bien adaptée aux calculs dans les bases de données dans la

¹Sauf peut-être, pour le traitement des processus infinis comme les systèmes d'exploitations ...

mesure où la réponse à une requête ne doit pas dépendre de la manière dont sont gérés les faits et les règles dans ces bases.

Les résultats classiques de [Apt 82, Cla 78, Hil 74, Las 84, Jaf 83] montrent que la sémantique opérationnelle idéale a une représentation axiomatique exacte. Nous nous proposons de trouver des résultats analogues pour les trois autres sémantiques opérationnelles.

Comme nous l'avons déjà mentionné, de nombreux travaux ont été publiés sur la sémantique standard mais selon une **approche dénotationnelle** [Arb 87, Bil 88, Deb 87, Jon 84, Ric 89]. Cette sémantique a également été étudiée du point de vue de la **logique linéaire** [Cer 89, 90, Reg 88] mais, à notre connaissance, uniquement pour des programmes Prolog **propositionnels**¹. La **logique modale** a aussi été utilisée pour décrire cette sémantique [Bal 90].

Le point de départ de notre travail est [Del 88a]. Dans cet article, J.-P. Delahaye utilise une technique de dédoublement des symboles de prédicats (déjà utilisée dans des travaux sur les paradoxes mathématiques et sémantiques [Fef 82, Gilm 74]) qui lui permet de définir une traduction axiomatique des programmes Prolog qui constitue une approximation de la sémantique opérationnelle standard. Nous donnons au chapitre III une caractérisation d'une classe de programmes pour lesquels cette traduction axiomatique représente exactement leur sémantique standard.

Le problème qui consiste à déterminer une sémantique axiomatique équivalente à une sémantique opérationnelle donnée peut être décomposé en quatre étapes (voir schéma 1) :

(iii) La première étape consiste à définir des **traductions axiomatiques** P_{Ax} pour chacune des trois sémantiques envisagées (au chapitre III pour la sémantique standard et au chapitre IV pour les sémantiques standard finie et invariante finie). Ces traductions se présentent sous la forme de **programmes généraux positifs** (définis au chapitre I). Leurs axiomes sont de la forme :

$$\forall(A \Leftarrow F)$$

où A est un atome et F une formule **positive** d'un langage du premier ordre avec égalité (c'est-à-dire dans laquelle le connecteur $\neg$ ne peut s'appliquer qu'au prédicat '=').

A chacun de ces systèmes d'axiomes, il est possible d'associer un **opérateur de conséquence immédiate** monotone T_{Ax} défini sur le treillis des interprétations de Herbrand du langage associé au programme (c.-à-d. les ensembles d'atomes clos).

(iv) La seconde étape consiste à établir, pour chaque sémantique P_{Op} , l'égalité :

$$P_{Op} = T_{Ax} \uparrow \omega.$$

où $T_{Ax} \uparrow \omega$ est l'ensemble obtenu par itération successive de l'opérateur T_{Ax} à partir de l'ensemble vide. Ceci est fait au chapitre III pour la sémantique standard (pour une certaine classe de programmes) et au chapitre IV pour les deux sémantiques finies (pour tout programme). Cette égalité prouve d'une certaine manière que la traduction P_{Ax} est "correcte et complète", mais pas encore au sens logique du terme. Nous désignerons

¹Mais pouvant contenir des littéraux négatifs dans les corps des clauses.

l'inclusion $P_{Op} \subset T_{Ax} \uparrow \omega$ sous le terme de $T_{Ax} \uparrow \omega$ -**correction** et l'inclusion inverse $T_{Ax} \uparrow \omega \subset P_{Op}$ sous celui de $T_{Ax} \uparrow \omega$ -**complétude**.

(v) La troisième étape consiste à essayer de donner un sens logique à cette égalité **dans le cadre des modèles de Herbrand**. Si le programme P utilise les symboles fonctionnels et de prédicats d'un langage du premier ordre L , il semble très naturel, au moins du point de vue de la programmation logique, de chercher à limiter les interprétations de P_{Ax} à celles dont le domaine est constitué par les atomes clos de L . Cette restriction peut être axiomatisée et nous notons P_{Ax}^{Her} la réunion des axiomes de P_{Ax} et de ceux restreignant les interprétations de P_{Ax} aux interprétations de Herbrand de L (voir chapitre I).

Nous n'obtenons malheureusement dans ce cadre logique que des résultats assez décevants (voir chapitre III et IV). Plus précisément, P_{Ax}^{Her} n'est une caractérisation axiomatique correcte et complète de la sémantique opérationnelle correspondante que lorsque $T_{Ax} \uparrow \omega$ est un plus petit point fixe de T_{Ax} (ce qui se produit quand T_{Ax} est continu). C'est le cas pour les **programmes propositionnels**, pour les **programmes sans variable locale**, pour les **programmes sans occurrence d'un symbole fonctionnel** mais c'est faux dans le cas général (voir fin du chapitre IV).

Nous retrouvons là en fait une situation comparable à celle que l'on rencontre dans le cas idéal. Si T_P est l'opérateur de conséquence immédiate usuel associé à un programme défini P , l'ensemble $T_P \downarrow \omega$ n'est en général pas un point fixe de T_P (voir par exemple [Llo 87]). Le théorème de complétude de la règle de négation par l'échec pour les programmes définis [Jaf 83] n'a pas été obtenu dans le cadre des modèles de Herbrand, mais dans celui de la Théorie Equationnelle de Clark.

Pour axiomatiser la restriction des interprétations de Ax aux modèles de la CET, il suffit d'ajouter à P_{Ax} les axiomes restreignant les interprétations aux modèles de Herbrand de L **sauf un** : l'**axiome de clôture** qui assure que les domaines des interprétations ne contiennent pas d'autres éléments que les termes clos du langage L . Nous notons P_{Ax}^{CET} la réunion de ces axiomes. Les modèles de la CET sont nettement plus complexes que les modèles de Herbrand. En particulier, leur domaine peut contenir des éléments qui ne sont la dénotation d'aucun terme du langage L (voir chapitre I).

(vi) La quatrième étape consiste donc à essayer de donner un sens logique à l'égalité $P_{Op} = T_{Ax} \uparrow \omega$ **dans le cadre de la Théorie Equationnelle de Clark**.

Avant de poursuivre dans cette voie, il nous faut faire un détour par la logique trivaluée, et en particulier mentionner un théorème établi par Kunen dans [Kun 87]. Dans ce résultat (qu'il nous est impossible de détailler complètement ici : c'est fait au chapitre VII), Kunen montre que pour certains types de programmes P (les programmes normaux), il est possible de donner un sens logique fort à l'itéré $F_P \uparrow \omega$ où F est un opérateur de conséquence immédiate trivalué (l'opérateur de Fitting) si l'on se place à la fois dans le cadre de la **Théorie Equationnelle de Clark** et dans le cadre de la **logique trivaluée**.

Ce résultat nous a donné l'idée d'essayer de démontrer un résultat analogue mais s'appliquant dans le cadre de la logique classique (bivaluée). Le théorème que nous avons obtenu est exposé au chapitre V. Il énonce que :

Si P est un programme général positif alors, pour toute formule positive close F,

$$\text{si } P^{CET} \models F \text{ alors il existe un entier } n \text{ tel que } T_P \uparrow n \models F$$

avec pour corollaire,

$$\text{si } A \text{ est un atome clos et si } P^{CET} \models A \text{ alors } A \in T_P \uparrow \omega$$

De plus, si le langage $\mathcal{L}$ contient "suffisamment" de symboles de fonctions¹, un résultat de [Kun 87] permet d'obtenir les équivalences :

$$P^{CET} \models F \text{ ssi il existe un entier } n \text{ tel que } T_P \uparrow n \models F$$

et

$$P^{CET} \models A \text{ ssi } A \in T_P \uparrow \omega.$$

Ce théorème peut alors être appliqué à toutes les traductions axiomatiques que nous avons définies (chapitre VI). Nous montrons que **les traductions axiomatiques standard, standard finie et invariante finie sont, dans le cadre de la CET, des caractérisations correctes et complètes des sémantiques opérationnelles correspondantes** (avec une restriction importante en ce qui concerne la sémantique standard).

Enonçons, par exemple, un corollaire de ces résultats pour les sémantiques finies.

Soit P un programme Prolog et A un atome clos du langage sous-jacent.

- a) $P_{fst}^{CET} \models A \vee A'^2$ ssi l'arbre SLD standard de $P \cup \{:- A\}$ est fini.
- b) $P_{fst}^{CET} \models A'$ ssi $A \in FF_{st}$.
- c) $P_{fst}^{CET} \models A$ ssi $A \in SS_{fst}$.

Soit P un programme Prolog et A un atome clos du langage sous-jacent.

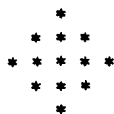
- a) $P_f^{CET} \models A \vee A'$ ssi tous les arbres SLD de $P \cup \{:- A\}$ sont finis.
- b) $P_f^{CET} \models A'$ ssi $A \in FF_f$.
- c) $P_f^{CET} \models A$ ssi $A \in SS_f$.

¹Plus précisément : si pour tout entier n, $\mathcal{L}$ contient une infinité de symboles de fonctions d'arité n. Nous appelons langage de Kunen et nous notons $\mathcal{L}_k$ un tel langage

²Nous rappelons que nous avons utilisé une technique de dédoublement des symboles de prédicats pour laquelle A' a pour sens intentionnel $\neg A$.

Ces résultats sont des corollaires immédiats du théorème démontré au chapitre V si le langage sous-jacent est un langage de Kunen. Mentionnons au passage qu'il est très naturel de considérer de tels langages en programmation logique dans la mesure où l'utilisateur d'un programme Prolog ou d'une base de données déductives doit pouvoir utiliser tous les symboles fonctionnels qu'il désire pour formuler ses requêtes.

Nous avons néanmoins également prouvés ces résultats sans aucune hypothèse supplémentaire sur le langage sous-jacent. Ces démonstrations, assez techniques, occupent la majeure partie du chapitre VI.



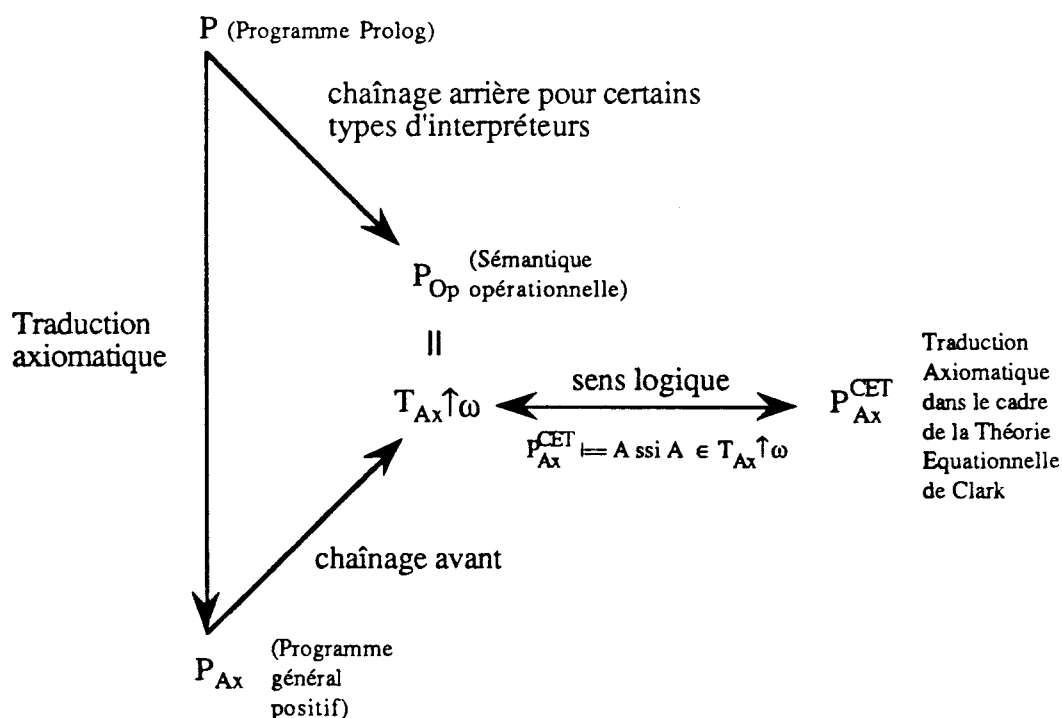
Nous pouvons donc répondre positivement à la question posée plus haut :

"La sémantique opérationnelle standard finie d'un programme Prolog P est équivalente à la sémantique axiomatique P_{fst}^{CET} "

"La sémantique opérationnelle invariante finie d'un programme Prolog P est équivalente à la sémantique axiomatique P_f^{CET} "

Le schéma 1 décrit la méthode que nous avons suivie au cours de ce travail. Il peut être redessiné différemment en tenant compte des résultats obtenus :

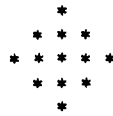
Schéma 2 :



- Les sémantiques opérationnelles P_{Op} considérées sont définies en prenant en compte un interpréteur (ou une classe d'interpréteurs Prolog) fonctionnant en **chaînage arrière**.
- La recherche d'une traduction axiomatique peut être vue comme la recherche d'un programme (général positif) P_{Ax} qui calcule par "**chaînage avant**" un ensemble $T_{Ax} \uparrow \omega$ égal à P_{Op} .
- Le théorème du chapitre V venant alors donner (de manière un peu miraculeuse) un sens logique à $T_{Ax} \uparrow \omega$. C'est un théorème de **complétude** en ce sens qu'il établit la complétude d'un "chaînage avant" pour une classe générale de programmes et de formules conséquences par rapport à l'axiomatique P_{Ax}^{CET} .

Deux constatations peuvent être tirées des résultats précédents :

- Le théorème de complétude confirme que le cadre de la Théorie Equationnelle de Clark, qui peut sembler *a priori* moins naturel que celui des modèles de Herbrand, est le mieux adapté à ces problèmes de sémantiques axiomatiques.
- La technique de dédoublement des symboles de prédicats s'est révélée très performante. Tout en restant une technique exprimable dans la logique classique (ce qui est un avantage important), elle est évidemment très liée à la logique trivaluée puisqu'elle permet de dissocier la valeur de vérité d'un atome de celle de sa négation.



Le chapitre VII ne se situe pas dans la lignée de la question posée plus haut. C'est plutôt une annexe consacrée à l'étude des rapports entre la technique de dédoublement des symboles de prédicats et la logique trivaluée. Nous établissons un "lexique" qui permet de traduire des notions et résultats exprimés selon une des deux formes dans l'autre. Cela nous permet, en particulier, d'énoncer un **théorème de complétude en logique trivaluée**.

Un **programme trivalué** est un ensemble d'axiomes de la forme $\forall(L \leftarrow F)$ où L est un littéral (positif ou négatif) et F une formule quelconque bivaluée (c.-à-d. ne contenant que des connecteurs bivalués de la logique classique) et où le connecteur trivalué ' $\leftarrow$ ' a été défini dans [Del 87c] par :

$$vv(A \leftarrow B) = \begin{cases} \text{Faux si } B \text{ est Vrai et } A \text{ n'est pas Vrai} \\ \text{Vrai sinon} \end{cases}$$

Si P est un programme trivalué consistant et si F est une formule close bivaluée alors :

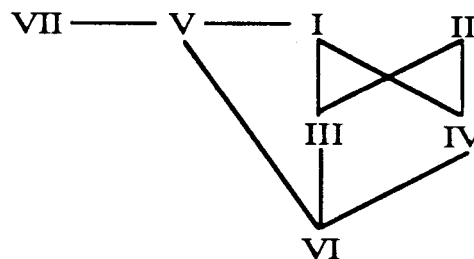
$$P^{CET} \models_1 F \text{ ssi il existe un entier } n \text{ tel que } T_P \uparrow_n \models_1 F$$

où $\models_1$ signifie : "a pour conséquence au sens de la logique trivaluée".

Ce théorème de complétude est très général puisque la formule F est une formule quelconque de la logique bivaluée classique. Il confirme donc que la logique trivaluée constitue un cadre logique bien adapté à la programmation logique (comme l'avaient déjà montré les travaux de [Del 87c, Fit 85, Kun 87 et 89, Thi 90] entre autres).

Nous concluons le chapitre VII sur une traduction dédoublée du théorème de complétude de la résolution SLDNF pour les programmes (et les requêtes) autorisées de [Kun 89]. Un théorème équivalent a par ailleurs été démontré dans [Cer 90] en logique linéaire. Cette traduction prolonge les résultats de [Ble 90, Gil 89] sur ce sujet.

Dépendance des chapitres



Remarques concernant la lecture de cette thèse :

- Nous avons défini un nombre important de notations. Nous donnons en fin de mémoire un index qui permet de retrouver leurs définitions.
- Les démonstrations concernant les trois sémantiques (standard, standard finie, invariante finie) ont été rédigées de façon à pouvoir être lues indépendamment les unes des autres. D'où certaines redondances lorsqu'on lit les trois à la suite.
- Le chapitre III décrit le lien entre notre travail et l'article [Del 88a] ainsi que l'étude de la sémantique standard (pour laquelle les résultats sont assez incomplets). Sa lecture n'est pas indispensable à la compréhension de la suite.
- Les chapitres I, V et VII forment un ensemble indépendant.
- Un tableau récapitulatif (p 137) donne une vue d'ensemble sur les principales propriétés des différentes sémantiques considérées ici.

Chapitre I :

Préliminaires logiques

Ce premier chapitre est destiné à définir les notions de logique dont nous nous servirons par la suite et à en fixer les notations. Nous énonçons également des résultats classiques (sans démonstration) et d'autres plus spécifiques à notre travail (que nous démontrons). On suppose connues les notions usuelles de logique du premier ordre (Cf par exemple [Cha 73, Del 86, Pab 76, Yas 71]).

I.1 Généralités

Nous rappelons qu'un **langage du premier ordre** est défini par la donnée d'un ensemble dénombrable Var de symboles de **variables**, d'un ensemble fini ou dénombrable de symboles de **fonctions** (les constantes pourront être considérées comme des fonctions d'arité 0) et d'un ensemble fini de symboles de **prédicats**. Un **langage du premier ordre avec égalité** possède '=' comme symbole de prédicat.

On suppose par la suite que les langages considérés contiennent au moins une constante. Si $\mathcal{L}$ est un langage du premier ordre avec égalité contenant au moins une constante a , on désigne par Vrai (resp. Faux) la formule $a = a$ (resp. $a \neq a$).

On utilise le mot **modèle** d'un langage comme synonyme d'**interprétation** (dans le sens usuel) de ce langage, le mot interprétation étant redéfini par la suite. Ceci peut se justifier par le fait qu'une interprétation d'un langage est un modèle d'un ensemble (vide) d'axiome de ce langage.

Notations :

Les langages du premier ordre seront dénotés par la lettre $\mathcal{L}$ éventuellement indicée.

Soit $\mathcal{L}$ un langage du premier ordre. On note $\mathcal{L}_d$ le langage possédant les mêmes symboles de fonctions (et donc de constantes) que $\mathcal{L}$ et qui, pour tout symbole de prédicat p de $\mathcal{L}$, possède deux symboles de prédicats : p et p' . $\mathcal{L}_d$ est donc un sur-langage de $\mathcal{L}$.

Si F est une formule de $\mathcal{L}$ dont les variables libres sont parmi $\{x_1, \dots, x_n\}$ (ce que l'on représente par la notation $F(x_1, \dots, x_n)$), et si $\mathcal{U}$ est un modèle de $\mathcal{L}$, la notation $\mathcal{U} \models F[a_1/x_1, \dots, a_n/x_n]$ (ou $\mathcal{U} \models F[a_1, \dots, a_n]$ s'il n'y a pas d'ambiguïté) signifie

que la substitution $(a_1/x_1, \dots, a_n/x_n)$ des variables $x_1, \dots, x_n$ par des éléments $a_1, \dots, a_n$ du domaine de $\mathcal{U}$ satisfait F dans $\mathcal{U}$.

Soit Σ un ensemble de formules de $\mathcal{L}$ dont les variables libres sont parmi $\{x_1, \dots, x_n\}$ (ce que l'on pourra représenter par $\Sigma(x_1, \dots, x_n)$). La notation $\mathcal{U} \models \Sigma[a_1, \dots, a_n]$ signifie que pour tout $F \in \Sigma$, $\mathcal{U} \models F[a_1, \dots, a_n]$.

Si $\Sigma \cup \{F\}$ est un ensemble de formules closes de $\mathcal{L}$, la notation $\Sigma \models F$ signifie que F est conséquence logique de Σ , c'est-à-dire que tout modèle de Σ est un modèle de F .

Si F est une formule de $\mathcal{L}$, la clôture universelle (resp. existentielle) est notée $\forall F$ (resp. $\exists F$).

Définition I.1: Soit Σ un ensemble de formules de $\mathcal{L}$ dont les variables libres sont parmi $\{x_1, \dots, x_n\}$ et soit $\mathcal{U}$ un modèle de $\mathcal{L}$, on dit que $\mathcal{U}$ réalise Σ (ou que Σ est satisfiable dans $\mathcal{U}$) ssi il existe $a_1, \dots, a_n$ dans le domaine de $\mathcal{U}$ tq $\mathcal{U} \models \Sigma[a_1, \dots, a_n]$.

I.2 Treillis et opérateur

La structure algébrique de treillis a été fréquemment utilisée en programmation logique ([Apt 82], [Fit 86], [Gue 88], [Llo 87]). Nous rappelons quelques définitions et résultats classiques.

Définition I.2 : Un treillis complet est un ensemble ordonné $(E, \leq)$ dont toute partie X admet une borne inférieure (notée $\text{Inf}(X)$) et une borne supérieure (notée $\text{Sup}(X)$). On note $\perp$ (resp. $\top$) la borne inférieure (resp. supérieure) de E .

Exemple : Soit A un ensemble et $E = 2^A$ l'ensemble des parties de A . Alors, $(E, \subset)$ est un treillis complet. Si X est une partie de E , alors $\text{Sup}(X) = \bigcup_{x \in X} x$ et $\text{Inf}(X) = \bigcap_{x \in X} x$. On a donc $\perp = \emptyset$ et $\top = A$.

Définition I.3 : Soit $(E, \leq)$ un treillis complet et $T : E \rightarrow E$, une application. On dit que T est monotone si pour tous $x, y \in E$, $x \leq y$ implique $T(x) \leq T(y)$. On peut définir alors les puissances ordinales de T par :

- $T \uparrow 0 = \perp$
- $T \uparrow (\alpha + 1) = T(T \uparrow \alpha)$ pour tout ordinal α
- $T \uparrow \alpha = \text{Sup}\{T \uparrow \beta ; \beta < \alpha\}$ pour tout ordinal limite α .
- $T \downarrow 0 = \top$
- $T \downarrow (\alpha + 1) = T(T \downarrow \alpha)$ pour tout ordinal α
- $T \downarrow \alpha = \text{Inf}\{T \downarrow \beta ; \beta < \alpha\}$ pour tout ordinal limite α .

Définition I.4 : Soit E un treillis complet. Une partie X de E est dirigée (directed) si pour tous éléments a et b de X , leur borne supérieure $\text{sup}(\{a, b\})$ appartient à X .

Définition I.5 : Soit $(E, \leq)$ un treillis complet et $T : E \rightarrow E$, une application. On dit que T est continue si pour toute partie dirigée X de E , on a : $T(\sup(X)) = \sup(T(X))$.

Remarque : Toute application continue est monotone. La réciproque est fausse.

Définition I.6 : Soit $(E, \leq)$ un treillis complet et $T : E \rightarrow E$, une application. On appelle **point fixe** de T tout élément x de E tq $T(x) = x$. On note $\text{pppf}(T)$ et $\text{pgpf}(T)$ respectivement les **plus petit point fixe** et **plus grand point fixe** (s'ils existent) de T .

On utilisera fréquemment les deux résultats suivants dus à Knaster [Kna 28] et Tarski [Tar 55]. On peut également trouver leur démonstration dans ([Fit 86], [Gue 88], [Llo 87]).

Proposition I.1 : Soit $(E, \leq)$ un treillis complet et $T : E \rightarrow E$, une application monotone. Alors T a un plus petit point fixe et un plus grand point fixe. De plus, il existe un plus petit ordinal α et un plus petit ordinal β tels que :

$$\text{pppf}(T) = T \uparrow \alpha \quad \text{et} \quad \text{pgpf}(T) = T \downarrow \beta.$$

Proposition I.2 : Soit $(E, \leq)$ un treillis complet et $T : E \rightarrow E$, une application continue. Alors :

$$\text{pppf}(T) = T \uparrow \omega.$$

I.3 Modèles de Herbrand

Il est souvent implicitement admis en programmation logique que l'on restreint les modèles du langage à une classe particulière de modèles : les **modèles de Herbrand**.

Définitions I.7 : Soit L un langage du premier ordre.

L'univers de Herbrand U_L (ou U s'il n'y a pas d'ambiguïté) de L est l'ensemble de tous les termes clos que l'on peut former avec les constantes (on rappelle que L est supposé contenir au moins une constante) et les symboles de fonctions de L .

La base de Herbrand H_L (ou H s'il n'y a pas d'ambiguïté) de L est l'ensemble de tous les atomes clos qui peuvent être formés avec les symboles de prédicats (sauf '=') de L .

Définition I.8 : On appelle **interprétation de Herbrand de L** toute partie de H . Ce sont les interprétations de L au sens classique du terme auxquelles on impose que :

- le domaine de l'interprétation soit U .
- les constantes de L soient interprétées par elles mêmes
- les symboles de fonctions n -aire f soient interprétés par les fonctions de U^n à valeurs dans U définies par :

$$(t_1, \dots, t_n) \mapsto f(t_1, \dots, t_n).$$

- les symboles de prédicats n -aire p soient interprétés par les relations définies sur U^n par :

$p(t_1, \dots, t_n)$ est vrai dans l'interprétation si $p(t_1, \dots, t_n)$ lui appartient.

Si I est une interprétation de Herbrand de L , il est donc équivalent d'écrire :

$$p(t_1, \dots, t_n) \in I \text{ et } I \models p(t_1, \dots, t_n).$$

Définition 1.9 : Soit S un ensemble de formules de L . Un **modèle de Herbrand de S** est une interprétation de Herbrand de L qui est un modèle (au sens usuel) de S .

La restriction aux modèles de Herbrand peut être axiomatisée de la manière suivante : soit L' le sur-langage de L obtenu en lui adjoignant le symbole de prédicat '=' (s'il n'appartenait pas déjà à L). Les restrictions à L des modèles de L' des 9 axiomes suivants sont les modèles de Herbrand de L (voir par exemple [She 87]).

Axiomes de la théorie de l'égalité :

$$1- x = x$$

$$2- x = y \Rightarrow y = x$$

$$3- x = y \wedge y = z \Rightarrow x = z$$

4- pour tout symbole de fonction f d'arité n ,

$$x_1 = y_1 \wedge \dots \wedge x_n = y_n \Rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$$

5- pour tout symbole de prédicat p d'arité n ,

$$x_1 = y_1 \wedge \dots \wedge x_n = y_n \Rightarrow [p(x_1, \dots, x_n) \Rightarrow p(y_1, \dots, y_n)].$$

Interprétation libre des symboles de fonctions :

6- pour tous symboles de fonctions f et g *distincts*,

$$f(x_1, \dots, x_n) \neq g(y_1, \dots, y_m)$$

7- pour tout symbole de fonction f ,

$$f(x_1, \dots, x_n) = f(y_1, \dots, y_n) \Rightarrow x_1 = y_1 \wedge \dots \wedge x_n = y_n$$

8- pour tout terme t contenant la variable x et distinct de x ,

$$t \neq x. \text{ (Axiome du test d'occurrence)}$$

Axiome de clôture :

9- $\forall x (x = t_1 \vee \dots \vee x = t_n \dots)$ où les t_i parcourent l'univers de Herbrand U .

Si le langage L contient des symboles fonctionnels d'arité ≥ 1 ou une infinité de constantes, c'est une disjonction infinie. Ce n'est donc pas vraiment un axiome, au sens strict du terme.

Remarques sur la théorie de l'égalité : Soit $\mathcal{M}$ un modèle de la théorie de l'égalité de domaine M et soit $=_{\mathcal{M}}$ l'interprétation dans $\mathcal{M}$ du prédicat '='. La relation $=_{\mathcal{M}}$ est une relation d'équivalence sur M (par les axiomes 1, 2 et 3). Soit M' l'ensemble quotient $M / =_{\mathcal{M}}$. Les interprétations des symboles fonctionnels et des prédicats "passent au quotient" (grâce aux axiomes 4 et 5). C'est-à-dire qu'à tout symbole fonctionnel f d'arité n , on peut associer une fonction $f_{\mathcal{M}'}$ de M'^n à valeurs dans M' définie par :

$$f_{\mathcal{M}'}(a_1 / =_{\mathcal{M}}, \dots, a_n / =_{\mathcal{M}}) = f_{\mathcal{M}}(a_1, \dots, a_n) / =_{\mathcal{M}}$$

(où $f_{\mathcal{M}}$ est l'interprétation de f dans $\mathcal{M}$).

De même, à tout symbole de prédicat p d'arité n , on peut associer une relation $p_{\mathcal{M}'}$ définie sur M'^n par :

$$p_{\mathcal{M}'}(a_1/=_{\mathcal{M}}, \dots, a_n/=_{\mathcal{M}}) \quad \text{ssi} \quad p_{\mathcal{M}}(a_1, \dots, a_n).$$

Le modèle quotient $\mathcal{M}'$ ainsi défini est équivalent à $\mathcal{M}$, c'est-à-dire qu'une formule close F de $\mathcal{L}$ est satisfaite dans $\mathcal{M}$ ssi elle l'est dans $\mathcal{M}'$.

On peut donc construire, pour tout modèle de la théorie de l'égalité, un modèle équivalent dans lequel **le prédicat d'égalité est interprété comme l'identité**. On se limitera par la suite à des modèles ayant cette propriété. On les appelle parfois modèles **égalitaires** ou **normaux**.

Notations : On note $\text{Her}(\mathcal{L})$ l'ensemble constitué par ces 9 axiomes (ou schémas d'axiomes). Les modèles de $\text{Her}(\mathcal{L})$ sont exactement les modèles de Herbrand de $\mathcal{L}$. Si Ax est un ensemble de formules de $\mathcal{L}$, on note $Ax^{\text{Her}} = Ax \cup \text{Her}(\mathcal{L})$. Les modèles de Ax^{Her} sont exactement les modèles de Herbrand de Ax .

Remarque : Si le langage $\mathcal{L}$ ne possède aucune constante, il n'a aucun modèle de Herbrand. C'est la raison pour laquelle on suppose par la suite que les langages considérés en contiennent au moins une.

L'importance des modèles de Herbrand apparaît lorsque l'on s'intéresse à la satisfiabilité d'ensembles de clauses :

Définition I.10 : *Un littéral est un atome ou la négation d'un atome.*

Définitions I.11 : *Une clause est une formule de la forme :*

$$\forall x_1 \dots \forall x_n (L_1 \vee \dots \vee L_m)$$

où chaque L_i est un littéral et où les x_i sont les variables apparaissant dans les littéraux L_i .

Une clause définie est une clause qui ne contient qu'un seul littéral positif.

Une clause unité est une clause définie qui ne contient aucun littéral négatif.

Une clause de Horn est une clause qui contient au plus un littéral positif.

Un but défini est une clause qui ne contient aucun littéral positif.

Un programme défini est un ensemble de clauses définies.

Théorème I.1 : *Soit S un ensemble de clauses.*

S a un modèle ssi S a un modèle de Herbrand.

S est insatisfiable ssi S n'a pas de modèle de Herbrand.

Démonstration : voir [Llo 87] par exemple. □

Mais si l'on ne suppose plus que S est un ensemble de clauses, ce résultat est faux en général comme le montre l'exemple suivant [Llo 87].

Soit $S = \{p(a), \exists x \neg p(x)\}$. S n'a pas de modèle de Herbrand si $\mathcal{L}$ ne contient que la constante a . Et pourtant, S est satisfiable.

Les notions de "formule vraie dans tous les modèles de Herbrand de S " et de "formule vraie dans tous les modèles de S " ne coïncident donc plus.

En fait, la plupart des systèmes d'axiomes que nous aurons à considérer sont constitués de formules non clausales et nous ne pourrions pas nous limiter aux modèles de Herbrand. Un autre cadre tout aussi naturel en programmation logique et moins restrictif est constitué par la **Théorie Equationnelle de Clark** [Cla 78].

I.4 Théorie équationnelle de Clark

On obtient la Théorie Equationnelle de Clark à partir des axiomes définissant les modèles de Herbrand en supprimant l'axiome de clôture.

Définition I.12 [Cla 78] : *On appelle Théorie Equationnelle de Clark du langage $\mathbf{L}$ la théorie notée $\text{CET}(\mathbf{L})^1$, obtenue en ajoutant éventuellement à $\mathbf{L}$ le symbole de prédicat '=' et constituée par l'ensemble des axiomes (ou schémas d'axiomes) de la théorie de l'égalité et de l'interprétation libre des symboles de fonctions (Axiomes numérotés de 1 à 8, §1.3).*

Notation : Si Ax est un ensemble de formules de $\mathbf{L}$, on note :

$$Ax^{\text{CET}} = Ax \cup \text{CET}(\mathbf{L}).$$

Remarque : Les modèles de Herbrand de $\mathbf{L}$ (resp. de Ax) sont donc des modèles de la $\text{CET}(\mathbf{L})$ (resp. de Ax^{CET}). Mais les domaines des modèles de la $\text{CET}(\mathbf{L})$ peuvent contenir des éléments qui ne sont pas des dénотations de termes du langage $\mathbf{L}$, ce qui est proscrit pour les modèles de Herbrand par l'axiome de clôture.

Définitions I.13 : *Soit $\mathbf{L}$ un langage du premier ordre avec égalité, comprenant au moins une constante et soit $\mathbf{L}'$ le sous-langage de $\mathbf{L}$ possédant les mêmes constantes et symboles fonctionnels que $\mathbf{L}$ mais dont le seul symbole de prédicat est '='.*

On appelle pré-modèle de la $\text{CET}(\mathbf{L})$ tout modèle de la $\text{CET}(\mathbf{L}')$.

On appelle formule équationnelle de $\mathbf{L}$ toute formule de $\mathbf{L}'$.

Remarque : On utilise la notion de pré-modèle lorsque l'on souhaite ne fixer que l'interprétation des termes du langage et de l'égalité à l'exception des autres prédicats. Tout modèle de la $\text{CET}(\mathbf{L})$ définit de manière canonique un pré-modèle unique de la $\text{CET}(\mathbf{L})$.

Définition I.14 : *On appelle pré-modèle standard de la $\text{CET}(\mathbf{L})$, le pré-modèle construit sur l'univers de Herbrand de $\mathbf{L}$ dans lequel, les fonctions et le prédicat '=' ont leur interprétation canonique.*

Notation : On notera $\mathbf{U}_{\mathbf{L}}$ (ou $\mathbf{U}$ s'il n'y a pas d'ambiguïté) le pré-modèle standard de la $\text{CET}(\mathbf{L})$.

¹pour Clark's Equational Theory.

Définition I.15 : Si $\mathcal{M}$ est un pré-modèle de la $CET(\mathcal{L})$, on note $U^{\mathcal{M}}$ le domaine de $\mathcal{M}$ et $H^{\mathcal{M}}$ l'ensemble $\{(p; \alpha_1, \dots, \alpha_n) ; \text{où } p \text{ est un symbole de prédicat } n\text{-aire de } \mathcal{L} \text{ et où } \alpha_1, \dots, \alpha_n \text{ sont des éléments de } U^{\mathcal{M}}\}$.

L'exposant $\mathcal{M}$ sera omis dans le cas du pré-modèle standard et l'ensemble H sera identifié à la base de Herbrand de $\mathcal{L}$ c.-à-d. $H = \{p(t_1, \dots, t_n) \text{ où } p \text{ est un symbole de prédicat } n\text{-aire de } \mathcal{L} \text{ et où } t_1, \dots, t_n \in U\}$.

On peut montrer [Llo 87, ex 10, p 107] que tout pré-modèle de la $CET(\mathcal{L})$ contient une copie isomorphe de $\mathcal{U}$. On peut donc ne considérer que les pré-modèles de la $CET(\mathcal{L})$ qui sont des extensions de $\mathcal{U}$.

Exemples :

1. Une situation classique en programmation logique [She 87] conduit à considérer l'axiome Ax suivant :

$$\forall x(p(x) \Leftrightarrow \exists y(x = s(y) \wedge p(y)))$$

où le langage sous-jacent $\mathcal{L}$ ne contient qu'une seule constante 'a' et 's' pour seul symbole fonctionnel.

Le système $Ax \cup \{\exists x p(x)\}$ n'a pas de modèle de Herbrand.

En effet, soit $\mathcal{M}$ un modèle de Herbrand et n le plus petit entier tel que :

$$\mathcal{M} \models p(s^n(a))$$

L'axiome Ax montre alors que l'on devrait alors avoir :

$$\mathcal{M} \models \exists y(s^n(a) = s(y) \wedge p(y))$$

et comme $s^n(a) = s(y)$ implique que $y = s^{n-1}(a)$ (et $n \geq 1$) :

$$\mathcal{M} \models p(s^{n-1}(a))$$

ce qui est contradictoire.

Soit $\mathcal{L}_1$ le sur-langage de $\mathcal{L}$ obtenu en lui adjoignant une nouvelle constante 'c' et un nouveau symbole fonctionnel 1-aire 'g'. Soit $U^{\mathcal{M}}$ l'ensemble quotient $U_1/\equiv_1$ où U_1 est l'univers de Herbrand associé à $\mathcal{L}_1$ et où $\equiv_1$ est la relation d'équivalence engendrée par les égalités $t = s(g(t))$ pour tout terme t possédant une occurrence de 'c'.

L'interprétation standard des termes de $\mathcal{L}$ dans son univers de Herbrand se prolonge naturellement à $U^{\mathcal{M}}$ et définit un pré-modèle de la $CET(\mathcal{L})$.

L'axiome du "test d'occurrence" : $t \neq x$ est bien respecté puisqu'il ne s'applique qu'à des termes de $\mathcal{L}$, c'est-à-dire, ne contenant pas d'occurrence de 'c' ni de 'g'.

Si maintenant, on interprète dans $\mathcal{M}$ le prédicat p par la relation :

$$p(t) \text{ est vrai ssi } t \text{ contient une occurrence de 'c',}$$

on obtient un modèle de $Ax \cup \{\exists x p(x)\}$.

On peut d'ailleurs remarquer que $Ax \cup \{\exists x p(x)\}$ n'a pas de modèle de Herbrand, *quelque soit le langage sous-jacent envisagé*.

On voit bien sur cet exemple que le cadre de la Théorie Equationnelle de Clark est très nettement moins restrictif que celui des modèles de Herbrand.

2. On peut également remarquer que chaque fois que le langage initial $\mathcal{L}$ est plongé dans un sur-langage $\mathcal{L}_1$, le plongement canonique de l'interprétation de Herbrand des symboles fonctionnels de $\mathcal{L}$ à l'univers de Herbrand de $\mathcal{L}_1$ fournit un pré-modèle de la $CET(\mathcal{L})$.

Remarque : Si le langage $\mathcal{L}$ est quelconque, la $CET(\mathcal{L})$ n'est en général pas complète. Il existe des formules équationnelles F de $\mathcal{L}$ indécidables (vraies dans un pré-modèle de la $CET(\mathcal{L})$ et fausses dans un autre).

Exemple : Soit F la formule : $\forall xy(x = y)$.

F est vraie dans le pré-modèle standard de la $CET(\mathcal{L})$ si $\mathcal{L}$ ne contient qu'une constante et fausse dans tout pré-modèle contenant au moins deux éléments.

Par contre, la $CET(\mathcal{L})$ est complète si le langage $\mathcal{L}$ contient suffisamment de symboles de fonctions. Le résultat suivant est dû à Kunen [Kun 87a].

Théorème I.2 : *Si le langage $\mathcal{L}$ contient une infinité de symboles de fonctions d'arité n pour tout entier n , alors la $CET(\mathcal{L})$ est complète. En particulier, une formule F est conséquence de la $CET(\mathcal{L})$ ssi elle est vraie dans son pré-modèle standard.*

Définition I.16 : *On appelle par la suite langage de Kunen tout langage du premier ordre vérifiant les hypothèses du théorème précédent.*

Notation : On notera généralement les langages de Kunen : $\mathcal{L}_k$

Définition I.17 : *Soit $\mathcal{M}$ un pré-modèle de la $CET(\mathcal{L})$. On appelle interprétation de $\mathcal{L}$ sur $\mathcal{M}$ toute partie de $H^{\mathcal{M}}$.*

Proposition I.3 : *L'ensemble des interprétations de $\mathcal{L}$ sur $\mathcal{M}$ muni de la relation d'inclusion forme un treillis complet.*

Démonstration : Immédiat puisque l'ensemble des interprétations est égal à l'ensemble des parties de $H^{\mathcal{M}}$.

□

Nous montrons dans la proposition suivante qu'une interprétation (suivant la définition I.17) définit un modèle de la $CET(\mathcal{L})$ et réciproquement. Nous avons choisi de représenter les interprétations de la $CET(\mathcal{L})$ comme des parties de $H^{\mathcal{M}}$ de façon à prolonger le formalisme utilisé habituellement pour les modèles de Herbrand. Cela permettra en particulier de définir la notion d'opérateur de conséquence immédiate sur l'ensemble des interprétations de la $CET(\mathcal{L})$ de manière analogue à la définition classique souvent restreinte aux interprétations de Herbrand.

Proposition I.4 : Soit $\mathbf{L}$ un langage du premier ordre, $\mathcal{M}$ un pré-modèle de la CET($\mathbf{L}$), soit $\text{Mod}_{\mathcal{M}}$ l'ensemble des modèles de la CET($\mathbf{L}$) de pré-modèle $\mathcal{M}$ et soit $I_{\mathcal{M}}$ l'ensemble des interprétations de $\mathbf{L}$ sur $\mathcal{M}$.

L'application :

$$\begin{array}{ccc} \text{Mod}_{\mathcal{M}} & \dashrightarrow & I_{\mathcal{M}} \\ \mathcal{N} & \dashrightarrow & \{(p; \alpha_1, \dots, \alpha_n) ; \mathcal{N} \models p[\alpha_1, \dots, \alpha_n]\} \end{array}$$

est une bijection.

Notation : On notera $\mathcal{M}_I$ (ou plus simplement I , s'il n'y a pas d'ambiguïté) le modèle de la CET($\mathbf{L}$) associé à l'interprétation I .

Remarque : Soit Ax un ensemble de formules du langage $\mathbf{L}$. Les univers de Herbrand de Ax^{CET} et de la CET($\mathbf{L}$) sont les mêmes. Comme on se limite aux modèles de la CET($\mathbf{L}$) qui sont des extensions du pré-modèle standard $\mathcal{U}$, on peut ne considérer que les interprétations de Ax^{CET} qui sont des extensions de $\mathcal{U}$.

I.5 Programmes généraux

La notion de programme logique se réfère certains types d'axiomes dont les plus généraux (du moins si l'on reste dans le cadre de la logique classique bivaluée) ont été désignés dans [Llo 87] sous le terme de **programmes généraux**.

Définition I.1.18 : On appelle **programme général** tout système fini d'axiomes P de la forme $\forall(A \Leftarrow F)$ où A est un atome (dont le prédicat est distinct de '=') et F est une formule d'un langage du premier ordre $\mathbf{L}$. L'atome A est appelé la **tête** et la formule F le **corps** de l'axiome $\forall(A \Leftarrow F)$.

Lorsque l'on donnera des exemples de programmes généraux, le quantificateur $\forall$ sera souvent sous-entendu.

Les *programmes définis* (Cf définition I.1.1) et les *programmes normaux* (dont les corps sont des conjonctions de littéraux positifs ou négatifs) sont des cas particuliers de programmes généraux.

Nous montrons dans les propositions suivantes que l'on peut supposer sans perte de généralité qu'un programme général est *normalisé*, c'est-à-dire que :

- toute tête d'axiome de P est de la forme $p(x_1, \dots, x_n)$ où p est un symbole de prédicat (distinct de '=') et où $x_1, \dots, x_n$ sont des variables de $\mathbf{L}$
- tout symbole de prédicat (distinct de '=') apparaît en tête d'une règle et d'une seule.

Remarque : L'hypothèse faite au paragraphe I.1 selon laquelle un langage du premier ordre ne contient qu'un nombre fini de symboles de prédicats garantit qu'un programme général normalisé ne contient qu'un nombre fini d'axiomes.

Définition I.19 : Si P est un programme général normalisé et p un symbole de prédicat de L , on appelle **axiome définissant** p le seul axiome de P dont le prédicat de tête est p .

Définition I.20 : Etant données m formules $H_1, \dots, H_m$ où chaque H_i est de la forme $p(t_{1,i}, \dots, t_{n,i}) \Leftarrow F_i$ et p est un symbole de prédicat n -aire, on appelle **forme normale** des formules $H_1, \dots, H_m$ la formule suivante :

$$p(x_1, \dots, x_n) \Leftarrow G_1 \vee \dots \vee G_m$$

où G_i est la formule :

$$\exists y_1 \dots \exists y_{p_i} x_1 = t_{1,i} \wedge \dots \wedge x_n = t_{n,i} \wedge F_i$$

où les y_i sont les variables libres figurant dans les termes $t_{j,i}$ et dans la formule F_i et où les x_i sont de nouvelles variables.

Proposition I.5 : Si H_{Norm} est la forme normale des formules $H_1, \dots, H_m$ alors les clôtures universelles de H_{Norm} et de $H_1 \wedge \dots \wedge H_m$ sont équivalentes pour la $\text{CET}(L)$, c'est-à-dire que :

$$\text{CET}(L) \models \forall H_{\text{Norm}} \Leftrightarrow \forall H_1 \wedge \dots \wedge \forall H_m.$$

Démonstration : Soit $\mathcal{M}$ un modèle de la $\text{CET}(L)$.

- Supposons que $\mathcal{M} \models \forall H_{\text{Norm}}$.

Soit $i \in \{1, \dots, m\}$ et supposons que :

$$\mathcal{M} \models F_i[\beta_1, \dots, \beta_{p_i}]$$

où $\beta_1, \dots, \beta_{p_i}$ est une assignation des variables $y_1, \dots, y_{p_i}$ dans $\mathcal{M}$.

On a alors :

$$\mathcal{M} \models G_i[\alpha_1, \dots, \alpha_n]$$

où $\alpha_j = t_{j,i}[\beta_1, \dots, \beta_{p_i}]$ et comme $\mathcal{M}$ est un modèle de $\forall H_{\text{Norm}}$,

$$\mathcal{M} \models p[\alpha_1, \dots, \alpha_n]$$

soit :

$$\mathcal{M} \models p(t_{1,i}, \dots, t_{n,i})[\beta_1, \dots, \beta_{p_i}].$$

$\mathcal{M}$ est donc un modèle de $\forall H_i$ et ceci étant vrai pour tout entier $i \in \{1, \dots, m\}$, $\mathcal{M}$ est un modèle de $\forall H_1 \wedge \dots \wedge \forall H_m$.

- Supposons que $\mathcal{M} \models \forall H_1 \wedge \dots \wedge \forall H_m$.

Soit $i \in \{1, \dots, m\}$ et $\alpha_1, \dots, \alpha_n$ une assignation des variables $x_1, \dots, x_n$ telle que :

$$\mathcal{M} \models G_i[\alpha_1, \dots, \alpha_n].$$

On a alors :

$$\mathcal{M} \models \exists y_1 \dots \exists y_{p_i} \alpha_1 = t_{1,i} \wedge \dots \wedge \alpha_n = t_{n,i} \wedge F_i.$$

Soit $\beta_1, \dots, \beta_{p_i}$ une assignation des variables $y_1, \dots, y_{p_i}$ dans $\mathcal{M}$ pour laquelle :

$$\mathcal{M} \models \alpha_1 = t_{1,i}[\beta_1, \dots, \beta_{p_i}] \wedge \dots \wedge \alpha_n = t_{n,i}[\beta_1, \dots, \beta_{p_i}] \wedge F_i[\beta_1, \dots, \beta_{p_i}].$$

On a :

$$\mathcal{M} \models p(t_{1,i}, \dots, t_{n,i})[\beta_1, \dots, \beta_{p_i}]$$

puisque $\mathcal{M}$ est un modèle de $\forall H_i$ et donc :

$$\mathcal{M} \models p[\alpha_1, \dots, \alpha_n].$$

$\mathcal{M}$ est donc un modèle de $\forall H_{\text{Norm}}$.

□

Définition 1.21 : Soit P un programme général. On appelle **programme général normalisé associé à P** , le programme P_{Norm} formé à partir de P en remplaçant tous les sous-ensembles d'axiomes de P de même prédicat de tête par leur forme normale et en introduisant pour chaque prédicat p de L ne figurant pas en tête d'un axiome de P l'axiome $\forall(p(x_1, \dots, x_n)) \Leftarrow \text{Faux}$.

Exemple : Soit P le programme :

$$\begin{cases} p(a) \\ p(s(x)) \Leftarrow p(x) \end{cases}$$

et supposons que le langage L contienne un autre prédicat 1-aire q .

Le programme général normalisé associé à P est alors le programme :

$$P_{\text{Norm}} \begin{cases} p(x) \Leftarrow x = a \vee \exists y(x = s(y) \wedge p(y)) \\ q(x) \Leftarrow \text{Faux} \end{cases}$$

Proposition 1.6 : Soit P un programme général et P_{Norm} le programme général normalisé associé à P . Alors P et P_{Norm} sont équivalents pour la $\text{CET}(L)$, c'est-à-dire que tout modèle de la $\text{CET}(L)$ qui est un modèle de P est également un modèle de P_{Norm} et réciproquement.

Démonstration : Les axiomes de la forme $\forall(p(x_1, \dots, x_n)) \Leftarrow \text{Faux}$ sont des tautologies. Cette proposition est donc un corollaire de la proposition précédente.

□

Il est naturel en programmation logique d'associer à un programme général P un opérateur T_P défini sur le treillis des interprétations de Herbrand du langage sous-jacent au programme ([Apt 82, 88, Del 87b, 88, Emd 76, Fer 86, Fit 85, Jaf 83, Las 84, Llo 87, Mah 86, Myc 84, She 87]). Informellement, si I est une interprétation de Herbrand, $T_P(I)$ est l'ensemble des atomes clos que l'on peut déduire du programme P en utilisant un et un seul axiome de P à partir de I . Cet opérateur est habituellement appelé "*opérateur de conséquence immédiate associé à P* ". La notion de **chaînage avant** peut alors être exprimée en itérant l'opérateur T_P à partir de l'interprétation vide :

Un atome clos A est déduit de P par chaînage avant si et seulement si il existe un entier n tel que A appartienne à $(T_P)^n(\emptyset)$.

Ou bien, en utilisant les notations définies au paragraphe 1.2 :

$$A \text{ est déduit de } P \text{ par chaînage avant ssi } A \in T_P \uparrow \omega.$$

On voit donc que l'ensemble $T_P \uparrow \omega$ a une signification opérationnelle naturelle.

Cette notion d'opérateur de conséquence immédiate a également été étendue au treillis des interprétations de $\mathbf{L}$ sur un pré-modèle de la $CET(\mathbf{L})$ (voir par exemple [Llo 87]). Nous en donnons la définition ci-dessous.

Définition I.22 : Si $\mathcal{M}$ est un pré-modèle de la $CET(\mathbf{L})$, on note $T_P^{\mathcal{M}}$ (ou seulement $T^{\mathcal{M}}$ s'il n'y a pas d'ambiguïté) l'opérateur de conséquence immédiate associé au programme général normalisé P défini sur le treillis des interprétations de $\mathbf{L}$ sur $\mathcal{M}$ (l'exposant $\mathcal{M}$ est omis lorsqu'il s'agit du pré-modèle standard $\mathcal{U}$). Cet opérateur est défini par :

$$(p; \alpha_1, \dots, \alpha_m) \in T^{\mathcal{M}}(I) \quad \text{ssi} \quad p(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m) \text{ est un axiome de } P \text{ et } \mathcal{M}_I \models F[\alpha_1, \dots, \alpha_m]$$

La notion de modèle peut être caractérisée à l'aide de l'opérateur de conséquence immédiate :

Proposition I.7 : Soit P un programme général normalisé et soit I une interprétation de $\mathbf{L}$ sur le pré-modèle $\mathcal{M}$ de la $CET(\mathbf{L})$. Alors,

$$I \text{ est un modèle de } P \quad \text{ssi} \quad T^{\mathcal{M}}(I) \subset I.$$

Démonstration : Soit $p(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m)$ un axiome de P .

I est un modèle de cet axiome ssi pour tous $\alpha_1, \dots, \alpha_m \in U^{\mathcal{M}}$,

Si $\mathcal{M}_I \models F[\alpha_1, \dots, \alpha_m]$ alors $\mathcal{M}_I \models p[\alpha_1, \dots, \alpha_m]$ c.-à-d. $(p; \alpha_1, \dots, \alpha_m) \in I$.

Et comme $\mathcal{M}_I \models F[\alpha_1, \dots, \alpha_m]$ ssi $(p; \alpha_1, \dots, \alpha_m) \in T^{\mathcal{M}}(I)$, on en déduit que I est un modèle de cet axiome ssi $T^{\mathcal{M}}(I) \subset I$.

□

Notation : Soit P un programme général normalisé, $\mathcal{M}$ un pré-modèle de la $CET(\mathbf{L})$ et n un entier. On notera $\mathcal{M}_n$ le modèle de la $CET(\mathbf{L})$ associé à l'interprétation $T^{\mathcal{M}} \uparrow n$ (voir proposition I.4).

Exemple : Soit P le programme :

$$\begin{cases} p(x) \Leftarrow x = a \vee \exists y(x = s(y) \wedge p(y)) \\ q(x) \Leftarrow \text{Faux} \end{cases}$$

et considérons l'opérateur de conséquence immédiate T_P défini sur le treillis des interprétations standard.

$T_P \uparrow 1 = \{p(a)\}$ et donc $\mathcal{U}_1 \models p(a)$.

Les programmes généraux que nous étudierons par la suite ont une particularité importante : le corps de leurs axiomes sont des formules **positives**, suivant la définition que nous en donnons ci-dessous.

Définitions I.23 :

i) On appelle **formule positive de $\mathcal{L}$** , toute formule construite inductivement à partir des règles suivantes :

- a) F est une formule équationnelle de $\mathcal{L}$
- b) F est un atome
- c) $F = G \vee H$ ou $F = G \wedge H$ où G et H sont des formules positives.
- d) $F = \forall xG$ ou $F = \exists xG$ où G est une formule positive.

ii) On appelle **formule négative de $\mathcal{L}$** , toute formule F tq $\neg F$ soit positive.

Exemple : $x \neq a \wedge \forall y(x \neq s(y) \vee p(y))$ est une formule positive.

Remarque : Une formule positive est donc une formule telle que le connecteur $\neg$ ne puisse s'appliquer (après simplifications éventuelles) qu'au prédicat ' $\neq$ '.

Définition I.24 : Si les prémisses des axiomes d'un programme général P sont des formules positives, on dit que P est un **programme général positif**.

La propriété des programmes généraux positifs qui nous servira le plus est que les opérateurs de conséquences immédiates qui leur sont associés sont monotones, ainsi que le montre la proposition ci-dessous.

Proposition I.8 : Soit $\mathcal{M}$ un pré-modèle de la CET($\mathcal{L}$)

a) Si F est une formule positive et si I et J sont deux interprétations de $\mathcal{L}$ sur $\mathcal{M}$ telles que $I \subset J$, alors :

$$\mathcal{M}_I \models F \text{ implique } \mathcal{M}_J \models F.$$

b) Si P est un programme général positif normalisé alors, l'opérateur de conséquence immédiate $T^{\mathcal{M}}$ est monotone. En particulier, la suite $(T^{\mathcal{M}} \uparrow n)_{n \in \omega}$ est croissante.

Démonstration :

a) La démonstration est immédiate par récurrence sur la complexité de la formule F .

b) Soient I et J deux interprétations de $\mathcal{L}$ sur $\mathcal{M}$ telles que $I \subset J$. Il faut démontrer que $T^{\mathcal{M}}(I) \subset T^{\mathcal{M}}(J)$. Soit $(p; \alpha_1, \dots, \alpha_n) \in T^{\mathcal{M}}(I)$ et soit $p(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m)$ l'axiome de P correspondant.

On a $\mathcal{M}_I \models F[\alpha_1, \dots, \alpha_n]$ et comme F est positive, $\mathcal{M}_J \models F[\alpha_1, \dots, \alpha_n]$ d'où l'on déduit que $(p; \alpha_1, \dots, \alpha_n) \in T^{\mathcal{M}}(J)$.

Comme $T^{\mathcal{M}} \uparrow 0 = \emptyset \subset T^{\mathcal{M}} \uparrow 1$, on en déduit par récurrence sur n que la suite $(T^{\mathcal{M}} \uparrow n)_{n \in \omega}$ est croissante.

□

La monotonie de l'opérateur $T^{\mathcal{M}}$ a pour conséquence une propriété de "monotonie logique" : une formule positive déduite à une certaine étape du calcul est déduite définitivement.

Proposition I.9 : Soit P un programme général normalisé. Si l'opérateur $T^{\mathcal{M}}$ est monotone alors, pour tous les entiers m et n tels que $m < n$:

a) Si F est une formule positive alors :

$$\mathcal{M}_m \models F \text{ implique } \mathcal{M}_n \models F.$$

b) Si F est une formule négative alors :

$$\mathcal{M}_n \models F \text{ implique } \mathcal{M}_m \models F.$$

Démonstration :

a) En effet, si $T^{\mathcal{M}}$ est monotone, $T^{\mathcal{M}} \uparrow m \subset T^{\mathcal{M}} \uparrow n$. On peut donc appliquer la proposition précédente.

b) $[\mathcal{M}_n \models F \text{ implique que } \mathcal{M}_m \models F]$ équivaut à $[\mathcal{M}_m \models \neg F \text{ implique que } \mathcal{M}_n \models \neg F]$ et $\neg F$ est une formule positive. On peut donc appliquer le a).

□

Chapitre II :

Préliminaires à propos du langage Prolog

Nous commençons par donner dans ce chapitre (§ 1, 2 et 3) les définitions permettant de décrire le fonctionnement des interpréteurs Prolog. Elles concernent essentiellement les notions d'**unification** et de **résolution SLD**. Nous montrons ensuite (§ 4) qu'il est non seulement possible mais aussi très naturel, si l'on prend en compte les implémentations réelles des interpréteurs Prolog, de définir d'autres sémantiques opérationnelles pour un programme Prolog que celle envisagée habituellement. Le but de ce travail étant de caractériser de manière axiomatique ces nouvelles sémantiques, nous concluons ce chapitre en rappelant les résultats (très classiques pour certains, moins pour d'autres) déjà obtenus à ce sujet.

II.1 Programmes Prolog et langages du premier ordre

Les programmes logiques (et Prolog en particulier) sont souvent définis comme des systèmes d'axiomes d'un calcul des prédicats du premier ordre. La sémantique procédurale qui leur est associée est pourtant souvent très différente de leur sémantique logique. Nous préférons, pour marquer cette différence, introduire les définitions suivantes :

Définitions II.1 : *Soit L un langage du premier ordre.*

Un programme Prolog est une suite finie de règles Prolog.

Une règle Prolog est une suite finie de la forme :

$$A \text{ :- } B_1, \dots, B_n \quad (n \geq 0)$$

où ':-' est un symbole n'appartenant pas au langage L et $A, B_1, \dots, B_n$ sont des formules atomiques (ou atomes) de L dont le prédicat est différent de '='.

*Si $n = 0$, la règle est simplement notée A et est appelée **règle unité**.*

Un but est une suite finie de la forme :

$$:- B_1, \dots, B_n \quad (n \geq 0)$$

où les B_i sont des atomes dont le prédicat est différent de '='.

La définition que nous donnons correspond à ce qui est généralement appelé "**programme Prolog défini pur**". En particulier, nous n'autorisons pas l'utilisation du CUT, de la négation ni de prédicats "non logiques" tels VAR, ASSERT, ...

Définition II.2 : *On appelle variable locale d'une règle Prolog $A :- B_1, \dots, B_n$ toute variable figurant dans un des atomes $B_1, \dots, B_n$ et ne figurant pas dans A .*

Un programme Prolog sans variable locale est alors un programme Prolog dont aucune règle ne possède de variable locale.

II.2 Unification

Nous rappelons quelques définitions et résultats sur la notion d'unification [Rob 65] qui est le mécanisme uniforme utilisé en programmation logique pour effectuer les passages de paramètres [Llo 87].

Soit $\mathcal{L}$ un langage du premier ordre. Notons $V_{\mathcal{L}}$ (resp. $T_{\mathcal{L}}$) l'ensemble des variables (resp. des termes) de $\mathcal{L}$.

Définitions II.3 : *Une substitution est une application $\theta : V_{\mathcal{L}} \dashrightarrow T_{\mathcal{L}}$ dont le domaine $\text{Dom}(\theta) = \{x / \theta(x) \neq x\}$ est fini.*

On utilisera la notation $\theta = (x_1/r_1, \dots, x_n/r_n)$ où $\text{Dom}(\theta) = \{x_1, \dots, x_n\}$ et $r_i = \theta(x_i)$ pour $i \in \{1, \dots, n\}$.

On note $\text{Im}(\theta) = \theta(\text{Dom}(\theta))$.

La substitution θ est close si pour tout $x \in \text{Dom}(\theta)$, $\theta(x)$ est un terme clos.

La composition des substitutions est définie de manière standard et induit un pré-ordre sur l'ensemble des substitutions :

$$\alpha \leq \beta \quad \text{ssi} \quad \exists \gamma / \beta = \alpha\gamma$$

Si $\alpha \leq \beta$ on dit que α est plus générale que β .

Exemple : $(x \backslash s(y)) \leq (x \backslash s^2(a))$.

Notation : Si E est une *expression* (c'est-à-dire un terme, une formule sans quantificateur, une règle Prolog ou un n-uplet d'expressions) et si θ est une substitution, on note $E\theta$ l'expression obtenue en remplaçant simultanément les variables figurant dans E par leur image par θ .

Exemple : Si $E = \{p(s(x)), p(s^2(y))\}$ et si $\theta = (x \backslash s(a), y \backslash a)$, alors $E\theta = \{p(s^2(a))\}$.

Définition II.4 : Soit E une expression. Une substitution θ est une **substitution de renommage pour E** si :

- θ est à valeurs dans V_L
- θ est injective
- $(\text{Var}(E) \setminus \text{Dom}(\theta)) \cap \text{Im}(\theta) = \emptyset$ où $\text{Var}(E)$ désigne l'ensemble des variables figurant dans E .

Définition II.5 : Deux expressions E et F sont des **variantes** s'il existe des substitutions α et β telles que : $E\alpha = F$ et $F\beta = E$.

Dans ce cas, α (resp. β) est une substitution de renommage pour E (resp. pour F). On dit également que les expressions E et F sont égales *au renommage des variables près*. Ceci définit une relation d'équivalence sur l'ensemble des expressions.

Exemples :

- $p(x)$ et $p(y)$ sont des variantes
- $p(x,x)$ et $p(x,y)$ n'en sont pas

La notion **d'unification** a été définie par Herbrand [Her 30] et redécouverte par Robinson [Rob 65].

Définition II.6 : Deux expressions E et F sont **unifiables** s'il existe une substitution θ telle que $E\theta = F\theta$. On dit alors que θ est un **unificateur** de E et F . D'une manière plus générale, si S est un ensemble d'expressions, on appelle **unificateur** de S , toute substitution θ telle que $S\theta$ soit un singleton. Si de plus, θ est minimum (parmi les unificateurs), on dit que θ est un **unificateur le plus général** (upg¹ en abrégé) de S .

Exemple : $p(f(x), g(y))$ et $p(y, g(f(a)))$ sont unifiables et $\theta = (x \backslash a, y \backslash f(a))$ est un upg.

Les upg sont uniques modulo une substitution de renommage. Si θ est un upg de S et si α est un unificateur de S alors il existe une substitution β telle que $\alpha = \theta\beta$ ([Llo 87] par exemple).

L'algorithme d'unification défini dans [Rob 65, Llo 87] prend en entrée un ensemble fini d'expressions simples (c'est-à-dire de termes ou d'atomes) et rend en sortie un upg si l'ensemble est unifiable. Sinon, il mentionne que l'ensemble n'est pas unifiable. De plus, si θ est l'upg de l'ensemble d'expressions $\{E_1, \dots, E_n\}$ fourni par l'algorithme d'unification, l'ensemble $\text{Var}(\text{Im}(\theta))$ est contenu dans l'ensemble des variables des expressions $E_1, \dots, E_n$. C'est-à-dire que l'algorithme d'unification n'introduit pas de nouvelles variables par rapport à celles figurant déjà dans les expressions à unifier. Sauf mention contraire, nous supposons que tous les upg considérés vérifient cette propriété.

Clark a démontré [Cla 78] que la $\text{CET}(L)$ est **complète pour l'unification**.

¹ou mgu en anglais, pour "most general unifier"

Proposition II.1 : Soient E et F deux expressions simples d'un langage $\mathcal{L}$.

a) Si E et F sont unifiables, alors $\text{CET}(\mathcal{L}) \models \exists(E = F)$.

b) Si E et F ne sont pas unifiables, alors $\text{CET}(\mathcal{L}) \models \forall(E \neq F)$.

Dans le cas où E et F sont unifiables, on a même le résultat plus précis suivant.

Proposition II.2 : Soit $\theta = (x_1 \backslash r_1, \dots, x_n \backslash r_n)$ l'upg de E et F calculé par l'algorithme d'unification. Alors :

$$\text{CET}(\mathcal{L}) \models \forall(E = F \Leftrightarrow x_1 = r_1 \wedge \dots \wedge x_n = r_n).$$

Si l'on rapproche ce résultat du théorème I.2, on obtient :

Théorème II.1 [Kun 87a] : Si $\mathcal{L}_k$ est un langage de Kunen, alors la $\text{CET}(\mathcal{L}_k)$ est complète et décidable.

C'est-à-dire qu'une formule équationnelle de $\mathcal{L}_k$ est conséquence logique de la $\text{CET}(\mathcal{L}_k)$ ssi elle est vraie dans le pré-modèle standard $\mathcal{U}$ de la $\text{CET}(\mathcal{L}_k)$ et qu'il existe un algorithme permettant de décider si une formule équationnelle est vraie dans $\mathcal{U}$.

Notations : Nous utiliserons fréquemment les notations suivantes afin de rendre certaines démonstrations plus lisibles : si s (resp. t , resp. x , resp. a) désignent le n -uplet de termes $(s_1, \dots, s_n)$ (resp. de termes $(t_1, \dots, t_n)$, resp. de variables $(x_1, \dots, x_n)$, resp. d'éléments $(a_1, \dots, a_n)$) du domaine d'un modèle $\mathcal{M}$ de $\mathcal{L}$, on note :

$$s = t \text{ la formule } s_1 = t_1 \wedge \dots \wedge s_n = t_n,$$

$$s \neq t \text{ la formule } s_1 \neq t_1 \vee \dots \vee s_n \neq t_n$$

et

$$[x \backslash a] \text{ l'assignation dans } \mathcal{M} [x_1 \backslash a_1, \dots, x_n \backslash a_n].$$

Nous utiliserons plusieurs fois les deux lemmes suivants :

Lemme II.1 : Soient s et t deux n -uplets de termes et F une formule sans quantificateur. Alors :

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F)$$

$$\text{CET}(\mathcal{L}) \models \neg \exists(s = t \wedge F)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ (calculé par l'algorithme d'unification) alors,

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F) \Leftrightarrow \forall F\theta.$$

$$\text{CET}(\mathcal{L}) \models \exists(s = t \wedge F) \Leftrightarrow \exists F\theta.$$

Démonstration :

a) Puisque la $\text{CET}(\mathcal{L})$ est complète pour l'unification (Prop. II.1),

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t)$$

et en particulier,

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F).$$

On remarque que $\neg\exists(s = t \wedge F)$ est équivalent à $\forall(s \neq t \vee \neg F)$ et l'on peut appliquer la première partie.

b) Soit $\theta = (x_1 \setminus r_1, \dots, x_p \setminus r_p)$ où les variables x_i et celles des termes r_i figurent dans les termes t_i et s_i . La proposition II.2 montre que :

$$\text{CET}(\mathcal{L}) \models \forall(s = t \Leftrightarrow x = r)$$

On a également :

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F) \Leftrightarrow \forall(s = t \Rightarrow F)$$

soit :

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F) \Leftrightarrow \forall(x = r \Rightarrow F)$$

et comme $\forall(x = r \Rightarrow F) \Leftrightarrow \forall F\theta$ est universellement valide, on en déduit que :

$$\text{CET}(\mathcal{L}) \models \forall(s \neq t \vee F) \Leftrightarrow \forall F\theta.$$

La deuxième partie se montre de manière analogue. □

Lemme II.2 : Soient s et t deux n -uplets de termes unifiables et soit $\theta = (x_1 \setminus r_1, \dots, x_p \setminus r_p)$ leur upg (calculé par l'algorithme d'unification). Soit $V = \{x_1, \dots, x_q\}$ un ensemble de variables contenant les variables des termes s_i et t_i et soit $[x \setminus a]$ une assignation des variables de V dans un pré-modèle $\mathcal{M}$ de la $\text{CET}(\mathcal{L})$ telle que :

$$\mathcal{M} \models (s = t)[x \setminus a].$$

Alors :

a) Pour toute variable x_j de V , on a : $x_j\theta[x \setminus a] = a_j$.

b) Plus généralement, pour toute expression simple (n -uplet de termes ou formule sans quantificateur) E ne comportant que des variables de V , on a :

$E\theta[x \setminus a] = E[x \setminus a]$ si E est un terme ou un n -uplet de termes.

$\mathcal{M} \models E\theta[x \setminus a] \Leftrightarrow E[x \setminus a]$ si E est une formule.

Démonstration :

a) Si $x_j \notin \text{Dom}(\theta)$, on a $x_j\theta = x_j$ et donc $x_j\theta[x \setminus a] = a_j$.

Sinon, comme $\text{CET}(\mathcal{L}) \models \forall(s = t \Leftrightarrow x_1 = r_1 \wedge \dots \wedge x_p = r_p)$ et d'après l'hypothèse sur l'assignation des variables de V , on a en particulier : $\mathcal{M} \models (x_j = r_j)[x \setminus a]$, soit :

$$\mathcal{M} \models (x_j = x_j\theta)[x \setminus a]$$

et comme l'égalité est interprétée dans $\mathcal{M}$ comme l'identité, on a bien :

$$x_j\theta[x \setminus a] = a_j.$$

b) Immédiat d'après le a) et la définition de $E\theta$. □

II.3 Résolution SLD

La notion de résolution SLD¹ [Apt 82, Hil 74, Kow 71] est fondée sur le principe de résolution de Robinson [Rob 65] appliqué aux clauses définies (c.-à-d. aux règles de la forme de celles d'un programme Prolog).

Définition II.7 : Soit P un programme Prolog, soit $G = :- A_1, A_2, \dots, A_q$ un but et $C = B :- B_1, B_2, \dots, B_n$ une variante d'une règle de P telle que C et G n'aient pas de variable commune. Supposons que B et A_m soient unifiables et soit θ leur upg.

On dit que le but $G' = :- (A_1, A_2, \dots, A_{m-1}, B_1, \dots, B_n, A_{m+1}, \dots, A_q)\theta$ est dérivable à partir de G en dérivant le m -ième atome par la règle C , ce que l'on note par :

$$G \xrightarrow[C]{\theta} G'.$$

On obtient, par application répétée de cette étape, une **dérivation SLD** :

$$\delta : G \xrightarrow[C_1]{\theta_1} G_1 \xrightarrow[C_2]{\theta_2} G_2 \quad \dots \quad G_{n-1} \xrightarrow[C_n]{\theta_n} G_n$$

que l'on peut encore noter plus brièvement

$$\delta : G \xrightarrow[C_1 \dots C_n]{\theta} G_n \text{ avec } \theta = \theta_1 \dots \theta_n.$$

Si G_n est vide, on dit que la dérivation δ **réussit**. On appelle une telle dérivation une **réfutation** pour $P \cup \{G_0\}$. La restriction de θ aux variables de G_0 s'appelle alors une **réponse calculée** pour $P \cup \{G_0\}$.

Si l'atome sélectionné dans G_n ne s'unifie avec aucune tête de règle de P , on dit que la dérivation δ **échoue**.

Exemple : Soit P le programme Prolog

$$\begin{cases} A(x) :- B(x), C(x) & C_1 \\ B(f(x)) :- B(x) & C_2 \\ B(a) & C_3 \\ C(a) & C_4 \end{cases}$$

Dans les dérivations suivantes, l'atome sélectionné est souligné.

$$:- \underline{A(x)} \xrightarrow[C_1]{h} :- \underline{B(x)}, C(x) \xrightarrow[C_3]{(x \backslash a)} :- \underline{C(a)} \xrightarrow[C_4]{h} \emptyset$$

est une dérivation réussie dont la réponse calculée est $(x \backslash a)$.

¹pour résolution Linéaire avec fonction de Sélection appliquée aux clauses Définies.

$$\vdash \underline{A(x)} \xrightarrow[C_1]{H} \vdash \underline{B(x)}, C(x) \xrightarrow[C_2]{(x\forall(y))} \vdash B(y), \underline{C(f(y))}$$

est une dérivation qui échoue.

Définition II.8 : On appelle **arbre de dérivation SLD** de racine G_0 pour le programme P tout arbre dont les noeuds sont des couples (G, m) où G est un but et m un entier vérifiant :

1. la racine est (G_0, r)
2. si (G', m') est un fils de (G, m) alors G' est dérivable à partir de G en dérivant le m -ième atome de G par une règle de P .
3. pour toute règle permettant de dériver le m -ième atome de G , le noeud (G, m) a un fils correspondant; les différents fils étant ordonnés comme les règles utilisées pour les obtenir.

Lorsque G est la liste vide, le couple $(G, 0)$ est noté $\emptyset$ et appelé **clause vide**.

Nous supposons dans la suite qu'un interpréteur Prolog (fonctionnant en chaînage arrière) peut être déterminé par :

- une stratégie de choix des coefficients m (choix d'un atome dans chaque liste) qui engendre certains arbres SLD.
- une stratégie de choix, à chaque noeud de l'arbre, de l'ordre des règles du programme avec les têtes desquelles s'unifie l'atome sélectionné.
- une stratégie d'exploration des arbres SLD déterminés précédemment.

Toute branche d'un arbre SLD de racine G pour le programme P dont la feuille terminale est la clause vide détermine une réfutation pour $P \cup \{G\}$.

Définitions II.9 :

Une dérivation SLD est **équitable** si elle échoue ou si, pour chacun de ses atomes A , une version instanciée de A est choisie en un nombre fini d'étapes [Las 84].

Un arbre SLD est **équitable** si toute branche de l'arbre détermine une dérivation équitable [Las 84].

On appelle **arbre d'échecs finis** tout arbre SLD fini ne contenant pas la clause vide.

Un interpréteur Prolog est **équitable** s'il n'engendre que des arbres SLD équitables.

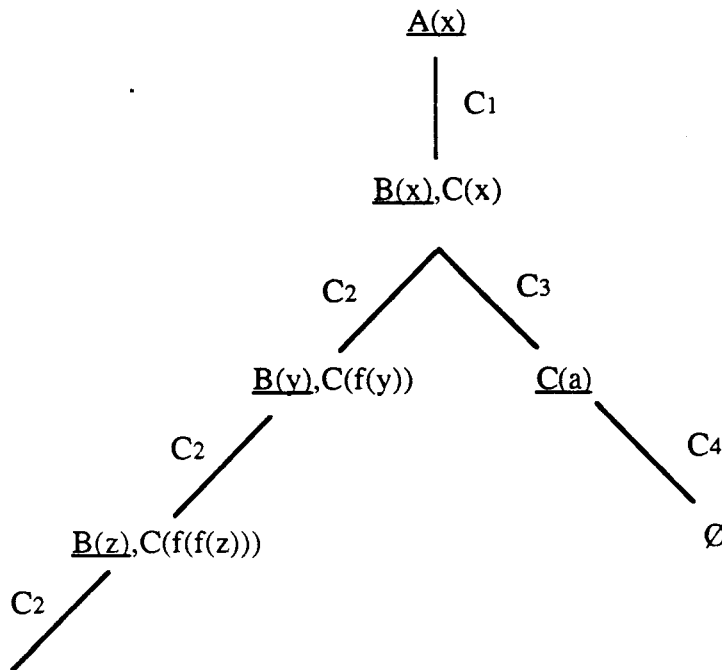
Un interpréteur Prolog "**en largeur d'abord**" parcourt les arbres SLD qu'il engendre "**en largeur d'abord**".

Un interpréteur Prolog **idéal** est un interpréteur équitable et "**en largeur d'abord**" [Del 88a].

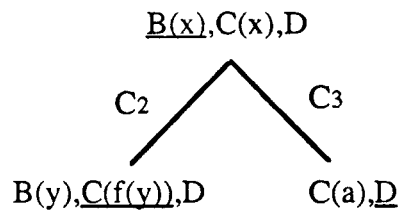
On appelle **arbre SLD standard** tout arbre SLD engendré par la stratégie de choix : $m=1$ (c.-à-d. dérivation du premier atome dans chaque liste), les fils de chaque noeud étant ordonnés comme les règles du programme [Del 88a].

Un interpréteur Prolog **standard** n'engendre que des arbres SLD standards et les explore en "**profondeur d'abord**".

Exemple : Considérons à nouveau le programme ci-dessus. A chaque noeud des arbres SLD ci-dessous, l'atome sélectionné a été souligné.



est (le début d') un arbre SLD standard non équitable (si le deuxième atome de chaque noeud de la branche de gauche n'est jamais sélectionné).



est un arbre SLD (non standard) d'échecs finis.

II.4 Sémantiques opérationnelles d'un programme Prolog

On définit une **sémantique opérationnelle** d'un programme Prolog en sélectionnant certaines propriétés de son comportement opérationnel lorsqu'il est soumis à certains interpréteurs. On peut distinguer les sémantiques opérationnelles pouvant être représentées par des sous-ensembles de la base de Herbrand du langage sous-jacent au programme (celles que nous définissons ci-après appartiennent toutes à cette catégorie) d'autres sémantiques plus riches représentées par des ensembles plus complexes. Par exemple : l'ensemble des couples (A, θ) où A est un atome (pas nécessairement clos) et θ une réponse calculée pour le but $:- A$ [Lev 88, 90, Fal 89a, 89b].

Nous définissons ci-dessous quatre sémantiques opérationnelles d'un programme Prolog.

II.4.1 La sémantique opérationnelle d'un interpréteur Prolog idéal

C'est la sémantique considérée habituellement. Soit P un programme Prolog.

- SS (Set of Success) est l'ensemble des atome clos A tels que $P \cup \{:- A\}$ admette une réfutation.

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

Si l'un des arbres SLD de racine $:-A$ contient la clause vide, alors tous la contiennent.

SS est l'ensemble des atomes clos pour lesquels un interpréteur Prolog "en largeur d'abord" retourne un succès (c'est-à-dire, explore un noeud égal à la clause vide). Ceci est vrai, plus généralement, de tout interpréteur explorant tous les noeuds des arbres SLD qu'il engendre.

Remarque : Le résultat de [Apt 82] cité précédemment peut être généralisé.

Soit L une liste d'atomes et σ une substitution des variables de L telle que pour toute instantiation close $L\sigma$ de $L\sigma$, $P \cup \{:-L\sigma\}$ admette une réfutation. Alors si le langage sous-jacent L contient suffisamment de constantes n'apparaissant pas dans P pour instancier les variables libres de $L\sigma$ (une constante différente par variable), $P \cup \{:-L\}$ a une réfutation de réponse calculée θ moins générale que σ , *quelle que soit la règle de sélection utilisée.*

Ce résultat est un corollaire du Théorème d'indépendance de la règle de sélection [Llo 87].

- FF (Finite Failure set) est l'ensemble des atomes clos A tels que toute dérivation équitale de racine $:- A$ échoue.

Les résultats de [Emd 76, Las 84] montrent que :

FF est l'ensemble des atomes clos A pour lesquels un arbre SLD équitale de racine $:- A$ est un arbre d'échec finis. Et dans ce cas, tous les arbres SLD équitales de racine $:-A$ sont des arbres d'échecs finis.

FF l'ensemble des atomes clos pour lesquels un interpréteur Prolog équitale retourne un échec (en un temps fini).

Les ensembles SS et FF décrivent la sémantique opérationnelle d'un interpréteur Prolog idéal. Cette sémantique a été très fréquemment étudiée [Apt 82, Cla 78, Emd 76, Jaf 83, Las 84, Llo 87]. C'est à elle que l'on fait référence lorsque l'on parle de sémantique "classique".

II.4.2 La sémantique opérationnelle d'un interpréteur Prolog standard

Comme nous l'avons mentionné dans l'introduction, il n'est pas réaliste d'envisager l'utilisation d'interpréteurs idéaux. L'exploration des arbres "en largeur

d'abord" est très coûteuse en occupation mémoire et paraît donc particulièrement inadaptée à un langage comme Prolog dont le mode de calcul consiste justement à engendrer des arbres.

De fait, les interpréteurs Prolog standards utilisent une stratégie d'exploration en "profondeur d'abord". De plus, le fait que ce soit toujours le premier atome du but courant qui soit sélectionné a pour conséquence que les arbres SLD engendrés ne sont pas toujours équitables.

Reprenons l'exemple donné dans l'introduction :

$$\begin{cases} A :- A \\ A \\ B :- A, C \end{cases}$$

Un interpréteur Prolog Standard ne retourne ni un succès pour le but $:- A$, ni un échec fini pour le but $:- B$. Il faut donc donner une autre définition de la sémantique opérationnelle d'un interpréteur standard.

- SS_{st} est l'ensemble des atomes clos A tels que l'arbre SLD standard de racine $:- A$ possède une branche de succès, à gauche de laquelle toute branche est finie.

C'est l'ensemble des atomes clos pour lesquels un interpréteur Prolog standard retourne un succès. On appelle SS_{st} l'ensemble des **succès standards** de P .

- FF_{st} est l'ensemble des atomes clos A tels que l'arbre SLD standard de racine $:- A$ soit un arbre d'échecs finis.

C'est l'ensemble des atomes clos pour lesquels un interpréteur Prolog standard retourne un échec. On appelle FF_{st} l'ensemble des **échecs standards finis** de P .

Les ensembles SS_{st} et FF_{st} décrivent la sémantique opérationnelle d'un interpréteur Prolog standard. On a donné de cette sémantique des caractérisations dénotationnelles [Arb 87, Bil 88, Deb 87, Jon 84, Ric 89], par point fixe [Fit 85] mais, à notre connaissance, elle n'a été étudiée d'un point de vue axiomatique que dans [Bal 90, Cer 89,90, Del 88a, Reg 88] (voir § II.5).

II.4.3 La sémantique opérationnelle standard finie

La notion de succès standard n'est pas très satisfaisante d'un point de vue pratique. Considérons par exemple, le programme suivant :

$$\begin{cases} A \\ A :- A \\ B :- A, C \end{cases}$$

Pour un interpréteur standard, le but $:-A$ donne un succès (et même une infinité !) et le programme boucle indéfiniment. Il est rarement souhaitable qu'un système informatique boucle après avoir donné une réponse (les processus infinis tels les systèmes d'exploitation étant une exception).

De plus, ce prédicat n'est pas réutilisable : le programme boucle également sur le but :- B et ne retourne pas un échec fini. Les seuls succès "fiabes" sont donc les succès **standards finis** que nous définissons maintenant.

- SS_{fst} est l'ensemble des atomes clos A tels que l'arbre SLD standard de racine :- A soit fini et possède une branche de succès.

On appelle SS_{fst} l'ensemble des succès standards finis de P.

Les ensembles SS_{fst} et FF_{st} décrivent la sémantique opérationnelle finie d'un interpréteur Prolog standard [Den 89]. On peut remarquer que deux programmes Prolog qui ne se distinguent que par l'ordre de leurs règles ont la même sémantique standard finie.

II.4.4 La sémantique opérationnelle invariante finie

Un des attraits de la sémantique opérationnelle idéale réside dans ses propriétés de symétrie (ou d'invariance) :

- par rapport à l'ordre des règles dans le programme
- par rapport à l'ordre des atomes dans chaque clause

La sémantique que nous définissons maintenant possède également ces bonnes propriétés de symétrie ainsi qu'une propriété "d'universalité" par rapport aux interpréteurs considérés.

- SS_f est l'ensemble des atomes clos A tels que tous les arbres SLD de racine :- A soient finis et possèdent une branche de succès.

On appelle SS_f l'ensemble des succès invariants finis de P.

- FF_f est l'ensemble des atomes clos A tels que tous les arbres SLD de racine :- A soient des arbres d'échecs finis.

On appelle FF_f l'ensemble des échecs invariants finis de P.

Les ensembles SS_f et FF_f décrivent la sémantique opérationnelle invariante finie d'un programme Prolog (indépendamment du choix d'un interpréteur) [Den 90a].

Cette sémantique représente en fait une intersection : SS_f (resp. FF_f) est l'ensemble des atomes clos pour lesquels *tout* interpréteur retournera un succès (resp. un échec) fini. Elle paraît bien adaptée aux traitements dans les *bases de données* pour lesquelles on s'attend à ce que la réponse à une requête ne dépendent pas de la manière dont sont gérés les règles et les faits qui les composent.

Nous regrouperons souvent dans la suite sous le terme de **sémantiques opérationnelles finies** les sémantiques standard finie et invariante finie.

II.4.5 Remarques

a) Mycroft a proposé dans [Myc 84] de classer les atomes clos A relativement à un programme Prolog P suivant cinq familles :

1. Un arbre SLD pour $P \cup \{:-A\}$ contient la clause vide (et par conséquent tous la contiennent)

1.a - et tous les arbres SLD pour $P \cup \{:-A\}$ sont finis.

1.b - et au moins un arbre SLD pour $P \cup \{:-A\}$ est infini.

2. Aucun arbre SLD pour $P \cup \{:-A\}$ ne contient la clause vide

2.a - et tous les arbres SLD pour $P \cup \{:-A\}$ sont finis.

2.b - et tous les arbres SLD pour $P \cup \{:-A\}$ sont infinis.

2.c - et certains arbres SLD pour $P \cup \{:-A\}$ sont infinis, tandis que d'autres sont finis.

Ces familles peuvent être représentées au moyen de nos notations :

$$\begin{array}{lll} 1.a \longleftrightarrow SS_f & 1.b \longleftrightarrow SS \setminus SS_f & \\ 2.a \longleftrightarrow FF_f & 2.b \longleftrightarrow H \setminus (FF \cup SS) & 2.c \longleftrightarrow FF \setminus FF_f. \end{array}$$

b) L'exemple suivant montre que tous les ensembles définis précédemment peuvent être différents.

Exemple :

$$\left\{ \begin{array}{l} A \\ A :- A \\ B(a) :- B(a) \\ B(x) :- \\ C :- D(x), B(x) \\ D(b) \\ E :- A, G \\ F :- G, A \end{array} \right. \quad \begin{array}{l} SS_f = \{B(b), D(b)\} \\ SS_{fst} = \{B(b), D(b), C\} \\ SS_{st} = \{B(b), D(b), C, A\} \\ SS = \{B(b), D(b), C, A, B(a)\} \\ FF_f = \{D(a)\} \\ FF_{st} = \{D(a), F\} \\ FF = \{D(a), F, E\} \end{array}$$

Soit H l'ensemble des atomes clos du langage L sous-jacent à un programme Prolog P . On obtient immédiatement à partir des définitions précédentes la chaîne d'inclusion suivante :

$$SS_f \subseteq SS_{fst} \subseteq SS_{st} \subseteq SS \subseteq H \setminus FF \subseteq H \setminus FF_{st} \subseteq H \setminus FF_f$$

SS_{st} est en général distinct de SS car, en raison de la stratégie d'exploration en "profondeur d'abord", une branche de succès d'un arbre SLD standard peut ne pas être trouvée.

SS_{fst} est distinct de SS_{st} si SS_{st} contient un atome dont l'arbre SLD standard est infini.

SS_f est distinct de SS_{fst} si SS_{fst} contient un atome dont un arbre SLD est infini.

FF_{st} est en général distinct de FF car un arbre SLD standard peut ne pas être équitale.

FF_f est distinct de FF_{st} si FF_{st} contient un atome dont un arbre SLD n'est pas équitale.

c) On peut supposer que l'ordre dans lequel sont utilisées les règles du programme ne dépend que de l'ordre que l'on donne aux règles dans le programme. Dans ce cas, la sémantique standard finie ne dépend pas de cet ordre, mais la réciproque est fautive. En effet, il existe un programme Prolog P et un atome clos A tel que $A \in SS_{st}$ quelque soit l'ordre des règles dans le programme P et tel que $A \notin SS_{fst}$. L'exemple le plus simple illustrant cette situation que nous avons trouvé est :

Exemple :

$$\left\{ \begin{array}{ll} A(a,x) :- B(x,y,z), A(y,z) & (C_1) \\ A(a,x) :- C(x) & (C_2) \\ A(b,x) & \\ B(a,a,b) & \\ B(b,b,b) & \\ C(a) & \\ C(b) :- C(b) & \end{array} \right.$$

Si (C_1) est avant (C_2) , la réfutation suivante pour $:- A(a,a)$ est calculée par un interpréteur standard :

$$:- A(a,a) \rightarrow :- B(a,y,z), A(y,z) \rightarrow :- A(a,b) \rightarrow :- B(b,u,v), A(u,v) \rightarrow :- A(b,b) \rightarrow \emptyset$$

Si (C_2) est avant (C_1) , la réfutation suivante pour $:- A(a,a)$ est calculée par un interpréteur standard :

$$A(a,a) \rightarrow C(a) \rightarrow \emptyset$$

et l'arbre SLD standard de $:- A(a,a)$ contient la dérivation infinie suivante :

$$A(a,a) \rightarrow B(a,y,z), A(y,z) \rightarrow A(a,b) \rightarrow C(b) \rightarrow C(b) \rightarrow \dots$$

Par conséquent, $A(a,a) \notin SS_{fst}$.

La sémantique standard finie possède une propriété "d'universalité" bien plus faible que la sémantique invariante finie : SS_{fst} est l'ensemble des atomes clos pour lesquels tout interpréteur sélectionnant le premier atome du but courant mais pouvant à *chaque étape du calcul* utiliser les règles du programme dans un ordre quelconque retourne un succès fini.

d) Ces sémantiques finies peuvent paraître très restrictives. Ce n'est pourtant pas le cas d'un point de vue pratique. En effet, étant donné la spécification d'un problème, un programmeur Prolog (il est vrai, n'écrivant que des programmes purs!) écrira un programme pour lequel les ensembles SS et FF correspondent à la spécification et qui *en plus* ne boucle pour aucune des requêtes qui lui sont soumises (et c'est en cela que Prolog ne peut pas être considéré comme un langage entièrement logique).

C'est-à-dire que l'on cherche à écrire des programmes pour lesquels :

$$SS = SS_{fst} \text{ et } FF = FF_f \text{ (et, encore mieux } SS = SS_f).$$

Ces égalités assurent que le sens "intentionnel" classique du programme correspond à ce qui est réellement calculé par les implémentations standard de Prolog. Les programmes hiérarchiques (c.-à-d. sans définition récursive), mais d'une manière plus générale, tous les programmes pour lesquels il a été prouvé d'une manière ou d'une autre que tous les calculs sont finis vérifient ces égalités.

II.5 Sémantiques axiomatiques d'un programme Prolog

D'une manière générale, les études sur *la sémantique des langages à base de règles* (et donc Prolog en particulier) traitent des problèmes suivants :

- caractériser de manière logique des sémantiques opérationnelles.
- caractériser de manière logique ce que l'on attend d'un système de programmation logique, c'est-à-dire le sens souhaité ou intentionnel ("intended meaning") d'un programme.
- trouver des classes de programmes les plus générales possibles pour lesquelles un système de programmation calcule effectivement ce que l'on en attend.

Le sens le plus fort que l'on puisse donner à : "caractériser un ensemble de manière logique" est celui envisagé dans les études sur les sémantiques axiomatiques des langages à base de règles : "associer un système d'axiomes à un programme dont l'ensemble des formules conséquences (ou du moins, certaines classes de formules conséquences) soit égal à l'ensemble étudié d'un point de vue opérationnel".

Le problème de l'adéquation d'une sémantique axiomatique à une sémantique opérationnelle est classiquement décomposé en deux sous-problèmes :

- la **correction** de la sémantique opérationnelle par rapport à la sémantique axiomatique. C'est-à-dire, ce qui est calculé est-il conséquence du système d'axiomes envisagé ?

- la **complétude** de la sémantique opérationnelle par rapport à la sémantique axiomatique. Tout ce qui est conséquence peut-il être calculé ?

On peut remarquer la dualité de ces notions : à une permutation des deux sémantiques correspond une permutation des termes de correction et complétude. Mais la définition que nous en avons donné correspond à l'usage.

Nous avons mentionné au chapitre I le fait qu'en programmation logique deux "cadres logiques généraux" sont privilégiés : la restriction de la notion de modèle du langage sous jacent L :

- aux seuls modèles de Herbrand de L .
- aux modèles de la théorie équationnelle de Clark de L .

A chacun de ces cadres logiques correspond bien entendu des notions de correction et de complétude différentes.

Nous aurons également besoin d'une notion plus faible : celle de correction et de complétude par rapport à "l'itéré $T \uparrow \omega$ " où T est l'opérateur de conséquence immédiate associé à un système d'axiomes.

Définition II.10 : Soit L un langage du premier ordre, P_{Op} un sous-ensemble de la base de Herbrand de L représentant une "sémantique opérationnelle", P_{Ax} un programme général positif et T_{Ax} l'opérateur de conséquence immédiate associé à P_{Ax} défini sur les interprétations de Herbrand de P_{Ax} . Nous dirons que :

$$P_{Op} \text{ est } T_{Ax} \uparrow \omega\text{-correcte ssi } P_{Op} \subset T_{Ax} \uparrow \omega$$

et que :

$$P_{Op} \text{ est } T_{Ax} \uparrow \omega\text{-complète ssi } T_{Ax} \uparrow \omega \subset P_{Op}.$$

Remarque :

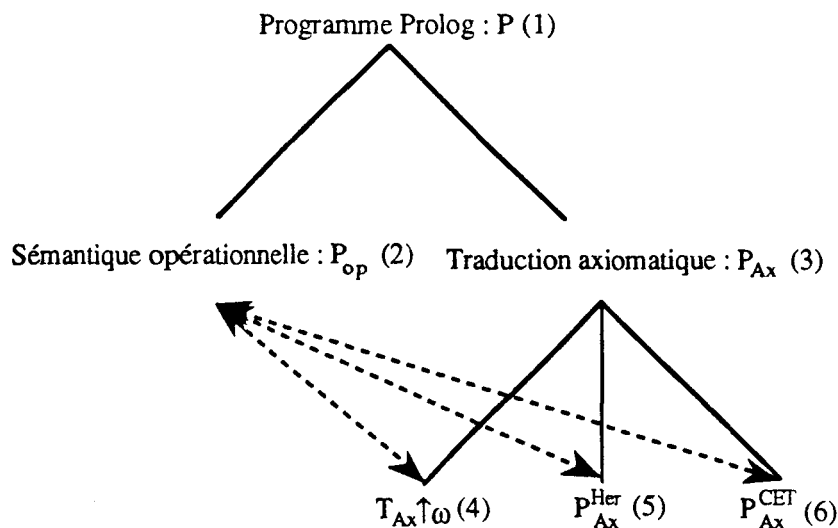
La notion de $T_{Ax} \uparrow \omega$ -correction implique en fait la correction par rapport à P_{Ax}^{Her} puisque $T_{Ax} \uparrow \omega$ est contenu dans tous les modèles de Herbrand de P_{Ax} .

Par contre, la notion de $T_{Ax} \uparrow \omega$ -complétude n'est équivalente à la notion de complétude par rapport à P_{Ax}^{Her} que si $T_{Ax} \uparrow \omega$ est un point fixe de T_{Ax} ce qui n'est pas vrai en général.

L'égalité $T_{Ax} \uparrow \omega = P_{Op}$ a la signification suivante : l'ensemble P_{Op} peut être calculé par un algorithme de "chaînage avant" à partir du programme général positif P_{Ax} .

Elle prouve l'adéquation du système d'axiomes P_{Ax} et de la sémantique opérationnelle étudiée mais elle ne constitue pas encore une caractérisation *logique* de cette adéquation.

Le schéma ci-dessous décrit la méthode que nous avons suivie dans cette thèse pour caractériser axiomatiquement une sémantique opérationnelle P_{Op} : définir une traduction P_{Ax} puis déterminer les propriétés éventuelles de $T_{Ax} \uparrow \omega$ -correction et $T_{Ax} \uparrow \omega$ -complétude, de P_{Ax}^{Her} -correction et P_{Ax}^{Her} -complétude et de P_{Ax}^{CET} -correction et P_{Ax}^{CET} -complétude.



Nous présentons maintenant les principaux systèmes d'axiomes qui ont déjà été associés à un programme Prolog en rappelant pour chacun d'eux, les principaux résultats obtenus. Nous utiliserons, à titre d'exemple, le programme Prolog P suivant :

$$\begin{cases} A \\ A :- A \\ B(a) \\ C(x) :- A, B(s(y)) \end{cases}$$

II.5.1 La traduction clausale [Emd 76, Apt 82]

C'est la traduction la plus simple. Elle revient à remplacer dans le programme Prolog P le symbole non logique ':' par le symbole d'implication ' $\Leftarrow$ ', le symbole ',' par $\wedge$ puis à supposer que chaque axiome ainsi obtenu est quantifié universellement. On notera P_{SI} cette traduction (elle est notée Ax_α dans [Del 88a]).

Exemple :

$$P_{SI} \quad \begin{cases} A \\ A \Leftarrow A \\ B(a) \\ C(x) \Leftarrow A \wedge B(s(y)) \end{cases}$$

Soit T_{SI} l'opérateur de conséquence immédiate associé à P_{SI} . T_{SI} est monotone, continu. Son plus petit point fixe (qui est égal à $T_{SI} \uparrow \omega$) est aussi le plus petit modèle de Herbrand de P_{SI} ainsi que l'ensemble SS. Pour tout atome clos A, on a :

$$A \in SS \quad \text{ssi} \quad A \in T_{SI} \uparrow \omega \quad \text{ssi} \quad P_{SI} \models A.$$

Le théorème II.1 permet de montrer les équivalences suivantes :

$$A \in SS \quad \text{ssi} \quad P_{SI} \models A \quad \text{ssi} \quad P_{SI}^{CET} \models A \quad \text{ssi} \quad P_{SI}^{Her} \models A.$$

Ces résultats peuvent être précisés afin de tenir compte des réponses calculées :

Théorème II.2 : [Cla 78] Soit P un programme Prolog et $L = :- A_1, \dots, A_n$ un but.

a) Si θ est une réponse calculée pour $P \cup \{:-L\}$, alors $\forall ((A_1 \wedge \dots \wedge A_n)\theta)$ est conséquence logique de P_{SI} (et aussi de P_{SI}^{CET} et P_{SI}^{Her}).

b) S'il existe une substitution θ des variables de L telle que $\forall ((A_1 \wedge \dots \wedge A_n)\theta)$ soit conséquence logique de P_{SI} (ou de P_{SI}^{CET} ou P_{SI}^{Her}) alors il existe une réponse calculée σ pour $P \cup \{:-L\}$ moins générale que θ .

Cette traduction fournit donc une caractérisation axiomatique de l'ensemble des succès d'un interpréteur "en largeur d'abord".

II.5.2 Le programme complété de Clark [Cla 78]

Cette traduction axiomatique a été définie afin de rendre compte de la *règle de négation par l'échec*. Selon cette règle, on peut déduire $\neg A$ (où A est un atome clos) d'un programme Prolog P si l'interpréteur Prolog a retourné un échec pour $P \cup \{:-A\}$.

On construit cette traduction :

- en normalisant la traduction clausale de P
- en remplaçant chaque symbole d'implication ' $\Leftarrow$ ' par le symbole d'équivalence ' $\Leftrightarrow$ '.

On obtient alors le système P_{SSI}

Notations : P_{SSI} est noté Ax_γ dans [Del 88a], P_{SSI}^{CET} est noté $\text{comp}(P)$ dans [Llo 87] et CDB dans [She 87]).

Exemple :

$$P_{SSI} \quad \begin{cases} A \\ B(x) \Leftrightarrow x = a \\ C(x) \Leftrightarrow \exists y(A \wedge B(s(y))) \end{cases}$$

(si l'on suppose que le langage L ne contient que les trois symboles de prédicats A , B et C).

Le plus petit modèle de Herbrand de P_{SSI} est SS . Le plus grand modèle de Herbrand de P_{SSI} est le plus grand point fixe de T_{SSI} . FF est contenu dans son complémentaire; plus précisément $FF = HNT_{SSI} \downarrow \omega$. [Apt 82, Cla 78, Llo 87]. On a en fait le résultat plus fort suivant [Jaf 83, Llo 87]:

$$\begin{array}{ll} A \in SS & ssi \quad P_{SSI}^{CET} \models A \\ A \in FF & ssi \quad P_{SSI}^{CET} \models \neg A \end{array}$$

qui est lui même corollaire du :

Théorème II.3 [Cla 78, Jaf 83] : Soit P un programme Prolog et $:-L = :-A_1, \dots, A_n$ un but.

- a) Si θ est une réponse calculée pour $P \cup \{:-L\}$, alors $\forall ((A_1 \wedge \dots \wedge A_n)\theta)$ est conséquence logique de P_{SSI} (et aussi de P_{SSI}^{CET} et P_{SSI}^{Her}).
- b) S'il existe une substitution θ des variables de L telle que $\forall ((A_1 \wedge \dots \wedge A_n)\theta)$ soit conséquence logique de P_{SSI} (ou de P_{SSI}^{CET} ou P_{SSI}^{Her}) alors il existe une réponse calculée σ pour $P \cup \{:-L\}$ moins générale que θ .
- c) La formule $\neg \exists (A_1 \wedge \dots \wedge A_n)$ est conséquence logique de P_{SSI}^{CET} ssi tout arbre équitale pour $P \cup \{:-L\}$ est un arbre d'échec finis.

Le programme complété de Clark fournit donc une **caractérisation axiomatique de la sémantique opérationnelle d'un interpréteur Prolog idéal**.

II.5.3 La traduction SSI dédoublée

Cette traduction, définie dans [Del 88a], utilise une technique de dédoublement des symboles de prédicats qui a été reprise en programmation logique dans [Ble 90, Den 89, Den 90a, Gil 89] mais qui avait également été utilisée dans l'étude des paradoxes sémantiques et logiques [Fef 82, Gilm 74]. Elle permet, comme les techniques reposant sur la logique trivaluée, d'exprimer *la négation par l'échec* de manière souvent plus précise que ne le permet la logique classique (voir chapitre VII).

Notation : Soit $\mathbf{L}$ un langage du premier ordre. Rappelons que l'on note $\mathbf{L}_d$ le langage possédant les mêmes symboles de fonctions (et donc de constantes) que $\mathbf{L}$ et qui, pour tout symbole de prédicat p de $\mathbf{L}$, possède deux symboles de prédicats : p et p' . On note H_d la base de Herbrand associée à $\mathbf{L}_d$.

Si E est un ensemble d'atomes clos de $\mathbf{L}$, on note E' l'ensemble $\{A' / A \in E\}$.

La traduction SSI dédoublée d'un programme Prolog P se construit de la manière suivante :

- normaliser la traduction P_{SI}
- ajouter à chaque axiome de P_{SI} de la forme : $p(x_1, \dots, x_n) \Leftarrow F$ l'axiome réciproque $p'(x_1, \dots, x_n) \Leftarrow F'$ où F' est obtenue à partir de la formule F en remplaçant :

- chaque symbole de prédicat p par p'
- chaque symbole $=$ par $\neq$
- chaque symbole $\neq$ par $=$
- chaque symbole $\wedge$ par $\vee$
- chaque symbole $\vee$ par $\wedge$
- chaque symbole $\forall$ par $\exists$
- chaque symbole $\exists$ par $\forall$.

Notation : P_{SSI_d}

Exemple :

$$\left\{ \begin{array}{l} A \Leftarrow \text{Vrai} \\ A' \Leftarrow \text{Faux} \\ B(x) \Leftarrow x = a \\ B'(x) \Leftarrow x \neq a \\ C(x) \Leftarrow \exists y (A \wedge B(s(y))) \\ C'(x) \Leftarrow \forall y (A' \vee B'(s(y))) \end{array} \right.$$

soit après simplification :

$$\begin{cases} A \\ B(x) \Leftarrow x = a \\ B'(x) \Leftarrow x \neq a \\ C(x) \Leftarrow \exists y (A \wedge B(s(y))) \\ C'(x) \Leftarrow \forall y (A' \vee B'(s(y))) \end{cases}$$

Remarque : si l'on substitue dans cette traduction $\neg p$ à p' pour tout symbole de prédicat p , on obtient un système d'axiomes équivalent à P_{SSI} .

Soit T_{SSI_d} l'opérateur de conséquence immédiate associé à P_{SSI_d} . T_{SSI_d} est monotone (puisque P_{SSI_d} est un programme général positif) mais non continu en général. Le plus petit point fixe de T_{SSI_d} peut donc être différent de $T_{SSI_d} \uparrow \omega$. Il a été montré dans [Del 88a] que :

$$a) T_{SSI_d} \uparrow \omega = SS \cup FF'$$

b) $\text{ppm}(P_{SSI_d}) = SS \cup (H \setminus \text{pgm}(P_{SSI}))'$ où ppm (resp. pgm) désigne le plus petit (resp. le plus grand) modèle de Herbrand.

Ces résultats ont été précisés dans [Ble 90, Gil 89] :

Théorème II.4 : Soit P un programme Prolog, A un atome clos et $L = A_1, \dots, A_n$ une liste d'atomes.

a)

$$\begin{array}{lll} A \in SS & ssi & P_{SSI_d}^{CET} \models A. \\ A \in FF' & ssi & P_{SSI_d}^{CET} \models A' \end{array}$$

b) Si θ est une réponse calculée pour $P \cup \{:-L\}$, alors :

$$\forall ((A_1 \wedge \dots \wedge A_n)\theta) \text{ est conséquence logique de } P_{SSI_d} \text{ (et aussi de } P_{SSI_d}^{CET} \text{ et } P_{SSI_d}^{Her}).$$

c) Si tout arbre équitale pour $P \cup \{:-L\}$ est un arbre d'échecs finis, alors :

$$\forall (A'_1 \vee \dots \vee A'_n) \text{ est conséquence logique de } P_{SSI_d}^{CET}$$

où $P_{SSI_d}^{CET}$ désigne la théorie obtenue en ajoutant à P_{SSI_d} l'ensemble des axiomes de la $CET(L_d)$.

La traduction SSI dédoublée fournit donc, d'une autre manière que le complété de Clark, une caractérisation axiomatique de la sémantique opérationnelle d'un interpréteur Prolog idéal.

II.5.4 La traduction P_θ :

J.-P. Delahaye a proposé dans [Del 88a] une traduction axiomatique des programmes Prolog qui approche leur sémantique opérationnelle standard. Cette traduction utilise également la technique de dédoublement des prédicats. Elle s'effectue, à partir d'un programme Prolog P , de la manière suivante :

Notations :

Soit $L = A_1, A_2, \dots, A_n$ une liste d'atomes. On note :

$$- [L]_\theta^S = A_1 \wedge A_2 \wedge \dots \wedge A_n \quad \text{et} \quad []_\theta^S = \text{Vrai.}$$

$$- [L]_\theta^{EF} = A_1' \vee (A_1 \wedge [A_2, \dots, A_n]_\theta^{EF}) \quad \text{et} \quad []_\theta^{EF} = \text{Faux.}$$

Si les A_j sont des atomes clos, la sémantique souhaitée pour :

$[L]_\theta^S$ est : un interpréteur Prolog standard retourne un succès pour $P \cup \{:- L\}$.

$[L]_\theta^{EF}$ est : un interpréteur Prolog standard retourne un échec pour $P \cup \{:- L\}$.

Exemple : $[A_1, A_2]_\theta^{EF} = A_1' \vee (A_1 \wedge A_2')$

A chaque règle $C \equiv p(t) :- L$ de P , où $t = (t_1, \dots, t_n)$, on associe les formules suivantes :

$$C^S(x) \equiv [\exists y(x = t \wedge [L]_\theta^S)]$$

$$C^{EF}(x) \equiv [\forall y(x \neq t \vee [L]_\theta^{EF})]$$

dans lesquelles les y_i sont les variables apparaissant dans les termes t_i et les atomes de la liste L .

- Si $C_1, \dots, C_n$ sont les règles de P dont le prédicat de tête est p , on considère les axiomes suivants (que l'on suppose quantifiés universellement) :

$$p(x) \Leftarrow [C_1^S \vee (C_1^{EF} \wedge C_2^S) \vee \dots \vee (C_1^{EF} \wedge \dots \wedge C_{p-1}^{EF} \wedge C_p^S)](x)$$

$$p'(x) \Leftarrow [C_1^{EF} \wedge \dots \wedge C_p^{EF}](x)$$

- Si aucune règle de P n'admet le symbole p comme prédicat de tête, on considère l'axiome :

$$\forall x \, p'(x)$$

Notation : P_θ (noté Ax_θ dans [Del 88a]) est la réunion de ces axiomes pour tout symbole de prédicat p du programme.

Le sens opérationnel souhaité pour $C_i^S(t)$ (resp. $C_i^{EF}(t)$) est que la i -ème règle de P conduise à un succès (resp. à un échec) pour l'atome $p(t)$ où $t = (t_1, \dots, t_n)$. Le système P_θ signifie alors que :

- $p(t)$ donne un succès si
 - la première règle donne un succès ou
 - la première règle donne un échec et la deuxième un succès ou
 -
 - les $p-1$ premières règles donnent un échec et la p -ième donne un succès.
- $p(t)$ donne un échec si toutes les règles donnent un échec.

Exemple :

$$\begin{cases} A \Leftarrow \text{Vrai} \vee (\text{Faux} \wedge A) \\ A' \Leftarrow \text{Faux} \wedge A' \\ B(x) \Leftarrow x = a \\ B'(x) \Leftarrow x \neq a \\ C(x) \Leftarrow \exists y (A \wedge B(s(y))) \\ C'(x) \Leftarrow \forall y (A' \vee (A \wedge B'(s(y)))) \end{cases}$$

soit après simplification

$$\begin{cases} A \\ B(x) \Leftarrow x = a \\ B'(x) \Leftarrow x \neq a \\ C(x) \Leftarrow \exists y (A \wedge B(s(y))) \\ C'(x) \Leftarrow \forall y (A' \vee (A \wedge B'(s(y)))) \end{cases}$$

Soit T_θ l'opérateur de conséquence immédiate associé à P_θ . T_θ est monotone (puisque P_θ est un programme général positif) mais non continu en général. Le plus petit point fixe de T_θ peut donc être différent de $T_\theta \uparrow \omega$. Il a été montré dans [Del 88a] que :

$$SS_{st} \cup FF'_{st} \subset T_\theta \uparrow \omega \subset SS \cup FF' \text{ et que les inclusions sont strictes en général.}$$

La traduction P_θ fournit donc une **approximation axiomatique de la sémantique opérationnelle d'un interpréteur Prolog standard**.

II.5.5 Les sémantiques axiomatiques de Prolog en logique trivaluée

L'introduction de la négation en programmation logique pose de nombreux problèmes. Le mécanisme du fonctionnement de Prolog (basé sur la résolution SLD) peut être généralisé de manière à prendre en compte les programmes **normaux** (au sens de Lloyd [Llo 87]), c'est-à-dire composés de règles dont le corps contient des littéraux positifs ou négatifs. Il est alors basé sur la résolution SLDNF [Cla 78, Llo 87]. On montre que la résolution SLDNF est **correcte** pour le complété de Clark d'un programme normal [Cla 78] mais la **complétude** par contre n'est vérifiée que par une classe très restreinte de programmes (contenant les programmes **hiérarchiques**, c.-à-d. sans règle récursive).

L'exemple suivant suggère une des principales raisons de cette inadéquation :

$$\begin{cases} A :- B \\ A :- \neg B \\ B :- B \end{cases}$$

Un interpréteur Prolog ne retournera pas un succès pour le but A alors que toute traduction axiomatique "raisonnable" *en logique classique* doit permettre de déduire A . Le recours à la **logique trivaluée ou partielle** (pour laquelle $B \vee \neg B$ n'est pas un théorème logique) semble donc assez naturelle.

Inaugurée par Fitting [Fit 85], l'application de la logique trivaluée (ou partielle) à la programmation logique a permis depuis d'obtenir des résultats de complétude de la résolution SLDNF pour une classe assez générale de programmes normaux [Kun 87a, 89]. La traduction axiomatique la plus étudiée est le **complété de Fitting** d'un programme normal obtenu en considérant le complété de Clark de ce programme puis en substituant le connecteur d'équivalence trivalué faible de Kleene ' $\leftrightarrow$ ' au connecteur d'équivalence bivalué ' $\Leftrightarrow$ ' ($A \leftrightarrow B$ est vrai dans une interprétation trivaluée ssi A et B ont la même valeur de vérité : Vrai, Faux ou Indéterminé). Nous mentionnons ces résultats à la fin du chapitre VII. Notons également les nombreuses sémantiques basées sur la logique trivaluée appliquées au traitement de la négation dans les bases de données déductives [Bid 89].

Nous détaillons un peu plus l'approche trivaluée au chapitre VII.

En ce qui concerne les programmes Prolog que nous étudions ici (les programmes définis [Llo 87]), les programmes complétés de Clark et de Fitting sont équivalents [Del 87b]. L'approche trivaluée n'apporte donc rien de plus à l'approche classique *si l'on se limite à l'étude de la sémantique opérationnelle "idéale" des programmes Prolog*. A notre connaissance, la logique trivaluée n'a jamais été utilisée pour rendre compte des autres sémantiques opérationnelles que nous avons définies. Nous montrons au chapitre VII que l'approche utilisant la technique de dédoublement des symboles de prédicats peut être représentée de manière naturelle en logique trivaluée et que par conséquent, les traductions axiomatiques "dédoublées" que nous donnons ont une traduction "trivaluée" immédiate.

II.5.6 Les sémantiques axiomatiques de Prolog en logique linéaire, en logique modale et en logique intuitionniste

La sémantique de Prolog a été étudiée du point de vue de la logique linéaire dans [Reg 88], [Cer 89] et [Cer 90]. S. Cerrito a pris en compte la stratégie des interpréteurs standard dans [Cer 89] et a obtenu une caractérisation correcte et complète de la sémantique opérationnelle standard pour des programmes Prolog normaux *propositionnels* (c'est-à-dire sans variable). Dans [Cer 90], un résultat de complétude de la SLDNF analogue à celui de [Kun 89] a été obtenu dans le cadre de la logique linéaire. L'apport majeur de la logique linéaire réside dans le fait qu'elle possède un système de déduction qui fait encore défaut (en toute généralité) à la logique trivaluée (voir [Thi 90]).

Citons également les travaux de Ph. Balbiani [Bal 90] en logique modale dans lesquels la stratégie standard des interpréteurs Prolog est prise en compte.

Enfin, la logique intuitionniste paraît être également une approche possible [She 85, 87] des sémantiques de Prolog mais nous n'avons connaissance d'aucune publication se rapportant aux sémantiques opérationnelles définies ici¹.

¹J.-P. JOUANNAUD a attiré mon attention sur le fait qu'en logique intuitionniste, la négation n'est pas involutive ($\neg\neg A$ n'est pas équivalent à A). Elle ne semble donc pas pouvoir rendre compte de la règle de la négation par l'échec ($\neg\neg A$ réussit ssi $\neg A$ échoue ssi A réussit).

Chapitre III :

La sémantique standard

Nous étudions dans ce chapitre la **sémantique opérationnelle standard** d'un programme Prolog, définie au paragraphe II.4.1. Nous commençons par analyser en quoi la traduction axiomatique P_θ proposée par J.-P. Delahaye [Del 88a] est insuffisante à rendre compte de cette sémantique dans le cas général et nous en déduisons les caractéristiques d'une classe de programmes pour lesquels cette traduction est satisfaisante (§ III.1).

Cette analyse nous a conduit à définir la notion de **succès fini** d'un interpréteur standard et, grâce à elle, à élaborer une nouvelle et meilleure approximation axiomatique P_{st} de la sémantique opérationnelle standard (III.2). Nous montrons que $T_{st} \uparrow \omega$ (où T_{st} est l'opérateur de conséquence immédiate associé à P_{st} défini sur les interprétations de Herbrand du langage sous-jacent à P) représente "au plus près" cette sémantique puis nous en tirons les conséquences en termes de correction et complétude dans le cadre des modèles de Herbrand. En fait, des trois sémantiques que nous étudions, la sémantique standard est celle pour laquelle les résultats sont les moins satisfaisants et il est possible de lire directement le chapitre IV.

On suppose dans tout ce paragraphe que la notion de succès ou d'échec fini se réfère à un **interpréteur Prolog Standard**. On dira donc, étant donné un programme P , qu'un but G **échoue** ssi l'arbre SLD standard pour $P \cup \{G\}$ est un arbre d'échecs finis, que G **réussit** ssi l'arbre SLD standard pour $P \cup \{G\}$ a une branche de succès à la gauche de laquelle ne se trouve aucune branche infinie et que G **réussit finiment** si G réussit et si l'arbre SLD standard pour $P \cup \{G\}$ est fini.

III.1 Programmes décrits par $T_\theta \uparrow \omega$

C'est la traduction P_θ qui a constitué le point de départ du présent travail. Nous commençons par donner une liste d'hypothèses suffisantes définissant une classe de programmes pour lesquels l'égalité $SS_{st} \cup FF_{st}^i = T_\theta \uparrow \omega$ est vraie.

Ce paragraphe sert donc à faire le lien entre les résultats contenus dans [Del 88a] et les notions développées dans cette thèse. Il en explique la genèse mais il n'apporte que peu de résultats théoriques importants.

III.1.1 Conditions nécessaires pour que $SS_{st} \cup FF'_{st} = T_{\theta} \uparrow \omega$

Le programme suivant (voir [Del 88a]) donne un exemple pour lequel :

$$SS_{st} \cup FF'_{st} = T_{\theta} \uparrow \omega \subset SS \cup FF' = T_{SSId} \uparrow \omega,$$

l'inclusion étant stricte.

$$\begin{cases} A(x) :- B(x), C(x) \\ B(x) :- B(x) \\ B(a) \\ B(c) \\ C(a) \\ C(b) \end{cases}$$

$$T_{SSId} \uparrow \omega = SS \cup FF' = \begin{array}{c} \{A(a), B(a), C(a), C(b), A'(c), C'(c)\} \\ \text{I-----I I-----I} \\ \quad SS \qquad \qquad FF' \end{array}$$

$$\text{et } T_{\theta} \uparrow \omega = SS_{st} \cup FF'_{st} = \begin{array}{c} \{C(a), C(b), C'(c)\} \\ \text{I-----I I-----I} \\ \quad SS_{st} \quad FF'_{st} \end{array}$$

Mais dans de nombreux cas, l'inclusion de $SS_{st} \cup FF'_{st}$ dans $T_{\theta} \uparrow \omega$ est stricte.

Exemple III.1 :

$$P \quad \begin{cases} A \\ A :- A \\ B(b) \\ C :- A, B(a) \end{cases}$$

$$P_{\theta} \quad \begin{cases} A \Leftarrow \text{Vrai} \vee (\text{Faux} \wedge A) \\ A' \Leftarrow \text{Faux} \wedge A' \\ B(x) \Leftarrow x = b \\ B'(x) \Leftarrow x \neq b \\ C \Leftarrow A \wedge B(a) \\ C' \Leftarrow A' \vee (A \wedge B'(a)) \end{cases}$$

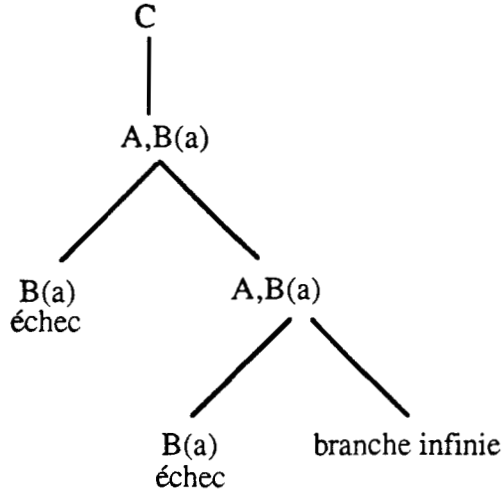
Ce système se simplifie en :

$$\begin{cases} A \\ B(x) \Leftarrow x = b \\ B'(x) \Leftarrow x \neq b \\ C \Leftarrow A \wedge B(a) \\ C' \Leftarrow A' \vee (A \wedge B'(a)) \end{cases}$$

$$T_{\theta} \uparrow 1 = \{A, B(b), B'(a)\}$$

$$T_{\theta} \uparrow 2 = T_{\theta} \uparrow 1 \cup \{C\} = T_{\theta} \uparrow \alpha \text{ pour tout ordinal } \alpha \geq 2.$$

Mais comme l'arbre SLD-standard de $\neg A$ n'est pas fini, le but $\neg C$ n'échoue pas finiment pour un interpréteur Prolog Standard.



$$SS_{st} \cup FF'_{st} = \{A, B(b), B'(a)\} \text{ est strictement inclus dans } T_{\theta} \uparrow \omega = \{A, B(b), B'(a), C\}$$

Remarque : On peut écrire un programme propositionnel pur qui a la même propriété :

$$\begin{cases} A \\ A :- A \\ C :- A, B \end{cases}$$

Définition III.1 : On dit qu'un programme P vérifie l'hypothèse H_1 si tout succès de P pour un interpréteur Prolog Standard est un succès fini, c.-à-d. si

$$SS_{st} = SS_{fst}.$$

Remarque : Il est en général impossible de décider si un programme P vérifie ou non cette hypothèse (voir § III.3).

Exemple III.2 :

$$P \quad \begin{cases} A(y) :- B(x, y) \\ B(b, y) \\ B(a, y) :- C(a, y) \\ C(a, a) \\ C(a, s(x)) :- C(a, x) \\ D(b) \\ E :- A(x), D(a) \end{cases}$$

On suppose que le langage L ne contient que les symboles fonctionnels 'a', 'b' et 's'.

$SS_{st} = \{ A(t), B(b,t), B(a,t), C(a,s^n(a)), D(b) \mid n \in \omega, t \text{ terme clos} \}$

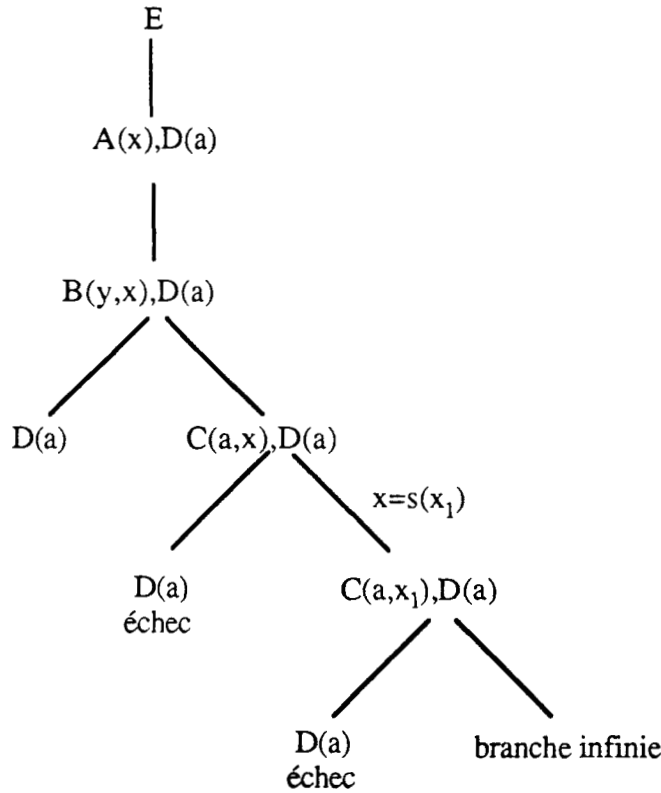
$FF_{st} = \{ B(s^n(t),t'), C(b,t), C(s^n(t),t'), D(a), D(s^n(t)) \mid n \geq 1, t \text{ et } t' \text{ termes clos} \}$

Nous n'écrivons qu'une partie de la traduction P_θ de ce programme.

$$P_\theta \begin{cases} A(x) \Leftarrow \exists y B(y,x) \\ B(x,y) \Leftarrow (x=b) \vee [(x \neq b) \wedge (x=a) \wedge C(a,y)] \\ D'(x) \Leftarrow x \neq b \\ E' \Leftarrow \forall x (A'(x) \vee (A(x) \wedge D'(a))) \end{cases}$$

Pour tout terme clos t , $B(b,t) \in T_\theta \uparrow 1$ donc, pour tout terme clos t , $A(t) \in T_\theta \uparrow 2$.

Comme $D'(a) \in T_\theta \uparrow 1$, on en déduit que $E' \in T_\theta \uparrow 3$. Et pourtant $E \notin FF_{st}$.

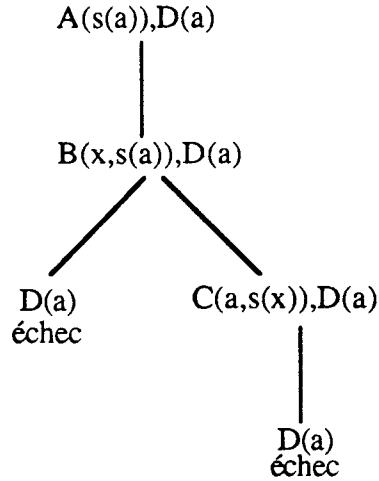


L'inadéquation entre $SS_{st} \cup FF'_{st}$ et $T_\theta \uparrow \omega$ provient ici du fait que pour tout terme clos t , $A(t) \in SS_{st} \cap T_\theta \uparrow 2$ alors que la profondeur des arbres SLD standard des buts $:-A(t)$ n'est pas bornée.

$$\sup_{t \text{ terme clos}} (\text{prof}(\text{arbre SLD}(A(t)))) = +\infty.$$

Pour toute substitution close σ du but $:-A(x), D(a)$, le but $:- (A(x), D(a))\sigma$ échoue pour un interpréteur standard, et pourtant, le but $:- A(x), D(a)$ boucle indéfiniment.

Exemple :



Définition III.2 : On dit qu'un programme Prolog P vérifie l'hypothèse H_2 s'il vérifie la propriété suivante :

Pour toute règle de P de corps L et pour toute substitution close σ qui instancie les variables non locales de cette règle, si pour toute substitution close τ des variables de $L\sigma$ le but $:- L\sigma\tau$ échoue pour un interpréteur standard alors le but $:- L\sigma$ échoue également.

Autrement dit :

$$\forall (A :- L) \in P, \forall \sigma \text{ substitution close tq } \text{Dom}(\sigma) = \text{Var}(A),$$

$$[\forall \tau \text{ substitution tq } L\sigma\tau \text{ soit clos } :- L\sigma\tau \text{ échoue}] \Rightarrow [-: L\sigma \text{ échoue}]$$

Cette hypothèse revient à exiger d'un programme P qu'il vérifie la propriété suivante pour certains buts G : si toutes les instances closes de G échouent finiment alors G échoue finiment. Ce qui est raisonnable d'un point de vue logique, mais beaucoup moins d'un point de vue opérationnel.

Remarque : Tout programme sans variable locale vérifie cette hypothèse.

Exemple III.3 :

$$P \quad \begin{cases} A :- B(x) \\ B(a) :- B(a) \\ B(b) \end{cases}$$

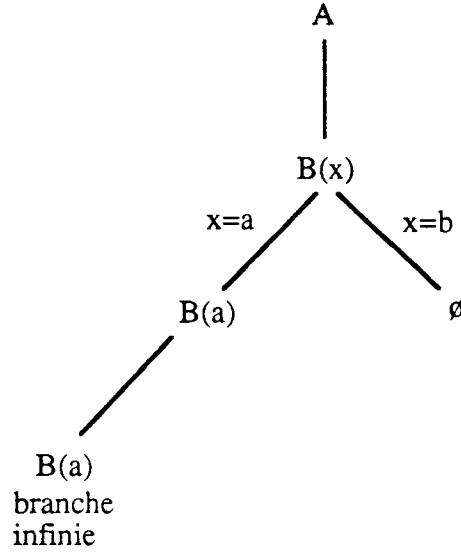
Le système P_θ associé est :

$$P_\theta \quad \begin{cases} A \Leftarrow \exists x B(x) \\ A' \Leftarrow \forall x B'(x) \\ B(x) \Leftarrow (x=a \wedge B(a)) \vee [(x \neq a \vee B'(a)) \wedge (x=b)] \\ B'(x) \Leftarrow (x \neq a \vee B'(a)) \wedge (x \neq b) \end{cases}$$

Soit, après simplification :

$$\begin{cases} A \Leftarrow \exists x B(x) \\ A' \Leftarrow \forall x B'(x) \\ B(x) \Leftarrow (x=a \wedge B(a)) \vee (x=b) \\ B'(x) \Leftarrow (x \neq b) \wedge B'(a) \end{cases}$$

$T_0 \uparrow 1 = \{B(b)\}$, $T_0 \uparrow 2 = \{B(b), A\} = T_0 \uparrow \alpha$ pour $\alpha \geq 2$ et $A \notin SS_{st}$.



Le but $\vdash B(b)$ réussit alors que $\vdash B(x)$ boucle.

Définition III.3 : On dit qu'un programme P vérifie l'hypothèse H_3 s'il vérifie la propriété suivante :

Pour toute règle de P de corps L et pour toute substitution σ qui instancie les variables non locales de cette règle, si pour une substitution close τ le but $\vdash L\sigma\tau$ réussit, alors le but $\vdash L\sigma$ réussit également.

Autrement dit :

$$\begin{aligned} &\forall (A \vdash L) \in P, \forall \sigma \text{ substitution close tq } \text{Dom}(\sigma) = \text{Var}(A), \\ &[\exists \tau \text{ substitution tq } L\sigma\tau \text{ soit clos et } \vdash L\sigma\tau \text{ réussisse}] \Rightarrow [\vdash L\sigma \text{ réussit}] \end{aligned}$$

Remarques :

1. Un programme sans variable locale vérifie l'hypothèse H_3 .
2. L'hypothèse H_3 est très naturelle puisqu'elle revient à exiger d'un programme qu'il vérifie l'axiome logique :

$$F(a) \text{ implique } \exists x F(x)$$

(si toutefois, l'on interprète la requête $F(x)$ par $\exists x F(x)$).

3. Il est en général impossible de décider si un programme vérifie cette hypothèse (voir § III.3).

III.1.2 Une classe de programmes pour lesquels $T_\theta \uparrow \omega = SS_{st} \cup FF'_{st}$

Nous montrons dans ce paragraphe que les conditions H_1 , H_2 et H_3 sont suffisantes à assurer l'adéquation de $T_\theta \uparrow \omega$ et de la sémantique opérationnelle standard d'un programme Prolog.

Lemme III.1 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P , soit s un n -uplet de termes et soit X une interprétation de Herbrand de L_d .

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$X \models \forall C^{EF}(s) \text{ et } X \models \neg \exists C^S(s)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$X \models \forall C^{EF}(s) \Leftrightarrow \forall [L\theta]_\theta^{EF},$$

$$X \models \exists C^S(s) \Leftrightarrow \exists [L\theta]_\theta^S.$$

Démonstration : Il suffit d'appliquer le lemme II.1 en prenant pour formule F successivement : $[L]_\theta^{EF}$ et $[L]_\theta^S$ qui sont des formules sans quantificateur. Il montre que les différentes formules citées dans le lemme sont des conséquences logiques de la $CET(L_d)$ et sont donc en particulier vraies dans le modèle de la $CET(L_d)$ associé à X (prop. I.4). $\square$

Théorème III.1 : Si un programme P vérifie les conditions H_1 , H_2 et H_3 , alors

$$T_\theta \uparrow \omega = SS_{st} \cup FF'_{st}.$$

Démonstration :

On sait que $SS_{st} \cup FF'_{st} \subset T_\theta \uparrow \omega$ [Del 88a]. Il reste à montrer l'inclusion inverse.

Montrons par récurrence sur n que $T_\theta \uparrow n \subset SS_{st} \cup FF'_{st}$ pour tout entier naturel n .

- Pour $n = 0$, c'est évident puisque $T_\theta \uparrow 0 = \emptyset$.
- Supposons l'inclusion établie pour un entier n .

Soit p un symbole de prédicat et soient $C_1, \dots, C_p$ les règles de P de prédicat de tête p .

1) Soit $p(s) \in T_\theta \uparrow (n+1)$ où s est un n -uplet de termes clos.

Alors :

$$T_\theta \uparrow n \models [C_1^S \vee (C_1^{EF} \wedge C_2^S) \vee \dots \vee (C_1^{EF} \wedge C_2^{EF} \wedge \dots \wedge C_{p-1}^{EF} \wedge C_p^S)](s)$$

(voir définition de P_θ § II.5.4).

Soit $j \in \{1, \dots, p\}$ tel que :

$$T_\theta \uparrow n \models [C_1^{EF} \wedge C_2^{EF} \wedge \dots \wedge C_{j-1}^{EF} \wedge C_j^S](s).$$

a) La formule $C_i^{EF}(s)$ est vraie dans $T_\theta \uparrow n$ pour $i \in \{1, \dots, j-1\}$.

Cette formule peut s'écrire :

$$\forall (s \neq t \vee (A'_1 \vee (A_1 \wedge A'_2) \vee \dots \vee (A_1 \wedge A_2 \dots \wedge A_{q-1} \wedge A'_q)))$$

Dire que $C_i^{EF}(s)$ est vraie dans $T_\theta \uparrow n$ signifie que :

- $p(s)$ n'est pas unifiable avec la tête de la règle C_i

ou que

- l'unification est possible (soit σ l'upg associé) et que :

$$T_\theta \uparrow n \models \forall [A'_1 \vee (A_1 \wedge A'_2) \vee \dots \vee (A_1 \wedge A_2 \dots \wedge A_{q-1} \wedge A'_q)]\sigma.$$

d'après le lemme III.1.

Autrement dit, pour toute substitution close τ dont le domaine contient les variables de l'expression précédente,

$$T_\theta \uparrow n \models [A'_1 \vee (A_1 \wedge A'_2) \vee \dots \vee (A_1 \wedge A_2 \dots \wedge A_{q-1} \wedge A'_q)]\sigma\tau.$$

Soit $k \in \{1, \dots, q\}$ tel que $(A_1 \wedge A_2 \dots \wedge A_{k-1} \wedge A'_k)\sigma\tau$ soit vraie dans $T_\theta \uparrow n$.

Par hypothèse de récurrence, les buts $:- A_1 \sigma\tau, \dots, :- A_{k-1} \sigma\tau$ réussissent finiment (puisque P vérifie l'hypothèse H_1), et le but $:- A_k \sigma\tau$ échoue finiment.

Donc le but $:- (A_1, A_2, \dots, A_q)\sigma\tau$ échoue finiment.

L'hypothèse H_2 montre alors que le but $:- (A_1, A_2, \dots, A_q)\sigma$ échoue.

La résolution de l'atome $p(s)$ avec la règle C_i échoue.

b) La formule $C_j^S(s)$ est vraie dans $T_\theta \uparrow n$.

Elle peut s'écrire sous la forme :

$$\exists (s = t \wedge (A_1 \wedge \dots \wedge A_q))$$

L'unification de $p(s)$ avec la tête de la j -ème règle de P est possible. Soit σ l'upg associé.

$$T_\theta \uparrow n \models \exists (A_1 \wedge \dots \wedge A_q)\sigma$$

c.-à-d. qu'il existe une substitution close τ dont le domaine contient les variables de l'expression précédente telle que

$$T_\theta \uparrow n \models (A_1 \wedge \dots \wedge A_q)\sigma\tau.$$

L'hypothèse H_3 assure alors que le but $:- (A_1, A_2, \dots, A_q)\sigma$ réussit; la règle C_j de P donne un succès standard pour $p(s)$ qui appartient donc à SS_{st} .

2) Soit $p'(s) \in T_\theta \uparrow (n+1)$ où s est un n -uplet de termes clos.

Alors $[C_1^{EF} \wedge C_2^{EF} \wedge \dots \wedge C_p^{EF}](s)$ est vraie dans $T_\theta \uparrow n$ donc, sous les hypothèses H_1, H_2 et H_3 et d'après ce qui précède, les p règles de P donnent un échec Prolog pour $p'(s)$ qui appartient donc à FF_{st} .

□

Corollaire III.1 :

1. Si un programme P sans variable locale, vérifie la condition H_1 , alors :

$$T_\theta \uparrow \omega = SS_{st} \cup FF'_{st}.$$

2. Si un programme P propositionnel pur vérifie la condition H_1 , alors :

$$pppf(T_\theta) = T_\theta \uparrow \omega = SS_{st} \cup FF'_{st}.$$

3. Si un programme P sans symbole de fonction vérifie les conditions H_1 , H_2 et H_3 alors :

$$pppf(T_\theta) = T_\theta \uparrow \omega = SS_{st} \cup FF'_{st}.$$

Démonstration :

1. En effet, les hypothèses H_2 et H_3 sont vérifiées par tout programme sans variable locale.
2. et 3. Pour tout atome A contenant une occurrence d'un symbole fonctionnel ne figurant pas dans P , $A' \in T_\theta \uparrow 1$ puisque A n'est unifiable avec aucune tête de règles de P . Et comme les autres atomes sont en nombre fini, $T_\theta \uparrow \omega$ est le plus petit point fixe de T_θ .

□

Remarque : Nous sommes donc parvenu à caractériser une classe de programmes Prolog pour lesquels $T_\theta \uparrow \omega$ constitue une représentation de leur sémantique opérationnelle standard. Ce résultat n'est pas très satisfaisant pour au moins deux raisons :

- 1- Les conditions H_1 , H_2 et H_3 sont des conditions très fortes et en général indécidables (nous le montrons au paragraphe III.3 pour H_1 et H_3).
- 2- Même sous ces hypothèses, on n'obtient pas une caractérisation axiomatique (au moins si l'on se limite aux modèles de Herbrand) puisque l'opérateur T_θ n'est pas toujours continu et que par conséquent $T_\theta \uparrow \omega$ n'est ni un point fixe de T_θ ni un modèle de P_θ au sens des modèles de Herbrand.

L'exemple III.1 montre bien en quoi la traduction P_θ est insuffisamment précise et suggère une idée permettant de l'améliorer : essayer de caractériser axiomatiquement la notion de succès standard fini.

Plus précisément, considérons la règle : $C :- A, B$.

Alors que le système P_θ tente d'exprimer :

" C échoue si [A échoue ou si [A réussit et B échoue]]",

il faut plutôt arriver à exprimer :

" C échoue si [A échoue ou si [A réussit *finiment* et B échoue]]".

C'est cette idée que nous exploitons dans le paragraphe suivant.

III.2 La traduction axiomatique standard triple

III.2.1 Définition de la traduction standard triple

De même que la technique de dédoublement des symboles de prédicats a été introduite afin d'exprimer "directement" la notion de négation par l'échec, nous utilisons dans ce paragraphe une technique de "triplement" des symboles de prédicats afin d'exprimer en plus la notion de finitude.

Plus précisément, soit $\mathcal{L}$ un langage du premier ordre. On note $\mathcal{L}_t$ le langage possédant les mêmes symboles de fonctions (et donc de constantes) que $\mathcal{L}$ et qui, pour tout symbole de prédicat p de $\mathcal{L}$, possède trois symboles de prédicats : p , p' et p'' . $\mathcal{L}_t$ est donc un sur-langage de $\mathcal{L}$ (et de $\mathcal{L}_d$). On note H_t la base de Herbrand associée à $\mathcal{L}_t$.

Si E est un ensemble d'atomes clos de $\mathcal{L}$, on note E'' l'ensemble $\{A'' / A \in E\}$.

Si A est un atome clos, la sémantique souhaitée est :

- pour A : un interpréteur Prolog standard retourne un succès pour $P \cup \{:- A\}$
- pour A' : un interpréteur Prolog standard retourne un échec pour $P \cup \{:- A\}$
- pour A'' : l'arbre SLD standard de $P \cup \{:- A\}$ est fini.

Notations :

Soit $L = A_1, A_2, \dots, A_n$ une liste d'atomes. On note :

- $[L]_{st}^S = A_1 \wedge A_2 \wedge \dots \wedge A_n$ et $[]_{st}^S = \text{Vrai}$.
- $[L]_{st}^{EF} = A'_1 \vee (A_1 \wedge A''_1 \wedge [A_2, \dots, A_n]_{st}^{EF})$ et $[]_{st}^{EF} = \text{Faux}$.
- $[L]_{st}^F = A'_1 \vee (A_1 \wedge A''_1 \wedge [A_2, \dots, A_n]_{st}^F)$ et $[]_{st}^F = \text{Vrai}$.

Si les A_j sont des atomes clos, la sémantique souhaitée pour :

- $[L]_{st}^S$ est : un interpréteur Prolog standard retourne un succès pour $P \cup \{:- L\}$
- $[L]_{st}^{EF}$ est : un interpréteur Prolog standard retourne un échec pour $P \cup \{:- L\}$
- $[L]_{st}^F$ est : l'arbre SLD standard de $P \cup \{:- L\}$ est fini.

Exemple III.4 :

$$[A_1, A_2]_{st}^{EF} = A'_1 \vee (A_1 \wedge A''_1 \wedge A'_2)$$

$$[A_1, A_2]_{st}^F = A'_1 \vee (A_1 \wedge A''_1 \wedge (A'_2 \vee (A_2 \wedge A''_2)))$$

Lemme III.2 : Si X est une partie de H_t et L une liste d'atomes clos,

- a) $[L]_{st}^{EF}$ vrai dans X implique que $[L]_{st}^F$ est vrai dans X
b) $[L]_{st}^F$ vrai dans X implique que $([L]_{st}^S \vee [L]_{st}^{EF})$ est vrai dans X .

Démonstration : Immédiat par récurrence sur la longueur de L . □

La traduction axiomatique standard triple d'un programme Prolog P s'effectue de la manière suivante :

A chaque règle $C \equiv p(t) :- L$ de P , on associe les formules suivantes :

$$\begin{aligned} C^S(x) &\equiv [\exists y(x = t \wedge [L]_{st}^S)] \\ C^{EF}(x) &\equiv [\forall y(x \neq t \vee [L]_{st}^{EF})] \\ C^F(x) &\equiv [\forall y(x \neq t \vee [L]_{st}^F)] \end{aligned}$$

dans lesquelles les y_i sont les variables apparaissant dans les termes t_i et les atomes de la liste L .

Remarque : Nous utilisons les mêmes notations C^S et C^{EF} que pour la traduction P_θ afin de ne pas les surcharger par un indice supplémentaire. Nous ferons de même pour les deux sémantiques suivantes. Le contexte permet toujours de savoir de quelle formule il s'agit.

- Si $C_1, \dots, C_n$ sont les règles de P dont le prédicat de tête est p , on considère les axiomes suivants (que l'on suppose quantifiés universellement) :

$$\begin{aligned} p(x) &\Leftarrow [C_1^S \vee (C_1^{EF} \wedge C_2^S) \vee \dots \vee (C_1^{EF} \wedge \dots \wedge C_{p-1}^{EF} \wedge C_p^S)](x) \\ p'(x) &\Leftarrow [C_1^{EF} \wedge \dots \wedge C_p^{EF}](x) \\ p''(x) &\Leftarrow [C_1^F \wedge \dots \wedge C_p^F](x) \end{aligned}$$

- Si aucune règle de P n'admet le symbole p comme prédicat de tête, on considère les axiomes :

$$\forall x p'(x) \text{ et } \forall x p''(x).$$

Notation : P_{st} est la réunion de ces axiomes pour tout symbole de prédicat p du langage. T_{st} est l'opérateur de conséquence immédiate associé à P_{st} défini sur les interprétations de Herbrand de L .

Proposition III.1 :

1. T_{st} est monotone.
2. Pour toute partie X de H_t , $T_{st}(X) \cap H_d \subset T_\theta(X \cap H_d)$.
En particulier, pour tout ordinal α , $T_{st} \uparrow \alpha \cap H_d \subset T_\theta \uparrow \alpha$.

3. *A est un atome clos et α un ordinal.*

- a) Si $A' \in T_{st} \uparrow \alpha$, alors $A'' \in T_{st} \uparrow \alpha$.
- b) Si $A'' \in T_{st} \uparrow \alpha$, alors $A \in T_{st} \uparrow \alpha$ ou $A' \in T_{st} \uparrow \alpha$.
- c) A et A' ne peuvent appartenir simultanément à $T_{st} \uparrow \alpha$.

Démonstration :

1. En effet, P_{st} est un programme général positif. On peut donc appliquer la proposition 1.8.

2. En effet, $T_{st}(X) \subset T_{st}(X \cup H'')$ et

$$T_{st}(X) \cap H_d \subset T_{st}(X \cup H'') \cap H_d = T_\theta(X \cap H_d)$$

puisque si l'on remplace tout atome A'' par vrai dans la définition de p' , on retrouve la définition associée à T_θ .

3. Soit $A = p(s)$ un atome clos et soient $C_1, \dots, C_p$ les règles de P de prédicat de tête p . Nous montrons les propositions a) et b) par induction sur α .

a) Soit $A' \in T_{st} \uparrow \alpha$.

Supposons que pour tout ordinal $\beta < \alpha$ et tout atome clos B , si B' appartient à $T_{st} \uparrow \beta$ alors B'' appartient aussi à $T_{st} \uparrow \beta$.

- Si α est un *ordinal limite*, il existe $\beta < \alpha$ tq $A' \in T_{st} \uparrow \beta$, donc $A'' \in T_{st} \uparrow \beta$ et comme T_{st} est monotone, $A'' \in T_{st} \uparrow \alpha$.

- Si α est un *ordinal successeur*, $\alpha = \beta + 1$ et :

$$T_{st} \uparrow \beta \models [C_1^{EF} \wedge \dots \wedge C_p^{EF}](s).$$

Comme pour toute liste close L et toute partie X de H_t :

$$\text{si } X \models [L]_{st}^{EF} \text{ alors } X \models [L]_{st}^F \text{ (lemme III.1)}$$

on en déduit que pour tout entier $i \in \{1, \dots, p\}$,

$$\text{si } X \models C_i^{EF}(s) \text{ alors } X \models C_i^F(s).$$

Donc $T_{st} \uparrow \beta \models [C_1^F \wedge \dots \wedge C_p^F](s)$ et $A'' \in T_{st} \uparrow \alpha$.

b) Soit $A'' \in T_{st} \uparrow \alpha$.

Supposons que pour tout ordinal $\beta < \alpha$ et tout atome clos B , si B'' appartient à $T_{st} \uparrow \beta$ alors B ou B' appartiennent à $T_{st} \uparrow \beta$.

- Si α est un *ordinal limite*, il existe $\beta < \alpha$ tq $A'' \in T_{st} \uparrow \beta$, donc $A \in T_{st} \uparrow \beta$ ou $A' \in T_{st} \uparrow \beta$ et comme T_{st} est monotone, la même propriété est vérifiée par $T_{st} \uparrow \alpha$.

- Si α est un *ordinal successeur*, $\alpha = \beta + 1$ et :

$$T_{st} \uparrow \beta \models [C_1^F \wedge \dots \wedge C_p^F](s).$$

Si $T_{st} \uparrow \beta \models [C_1^{EF} \wedge \dots \wedge C_p^{EF}](s)$, alors $A' \in T_{st} \uparrow \alpha$.

Sinon, soit i le premier entier tel que $C_i^{EF}(s)$ soit faux dans $T_{st} \uparrow \beta$.

Soit C_i la règle $p(t) :- L$. La formule $C_i^{EF}(s)$ est donc de la forme :

$$[\forall y (s \neq t \vee [L]_{st}^{EF})]$$

On a donc : $T_{st} \uparrow \beta \models [\exists y (s = t \wedge \neg [L]_{st}^{EF})]$.

Soit σ une substitution close des variables $y_1 \dots y_m$ telle que :

$$T_{st} \uparrow \beta \models (s = t\sigma \wedge \neg [L\sigma]_{st}^{EF}).$$

$$T_{st} \uparrow \beta \models [L\sigma]_{st}^F \text{ puisque } A'' \in T_{st} \uparrow \alpha.$$

Comme pour toute liste close M et toute partie X de H_1 , si $[M]_{st}^F$ vrai dans X alors $[M]_{st}^{EF}$ ou $[M]_{st}^S$ est vrai dans X (lemme III.1), on en déduit que :

$$T_{st} \uparrow \beta \models [L\sigma]_{st}^S$$

et donc que

$$T_{st} \uparrow \beta \models [C_1^{EF} \wedge \dots \wedge C_{i-1}^{EF} \wedge C_i^S](s)$$

soit $A \in T_{st} \uparrow \alpha$.

c) Supposons cette propriété fausse. Soit α le premier ordinal tel qu'il existe un atome clos $A = p(s)$ tq A et A' appartiennent à $T_{st} \uparrow \alpha$.

- α ne peut pas être un ordinal limite (sinon, il ne serait pas le plus petit à vérifier cette propriété).

- α est un ordinal successeur : $\alpha = \beta + 1$. On a alors :

$$T_{st} \uparrow \beta \models [C_1^{EF} \wedge \dots \wedge C_p^{EF}](s)$$

et

$$T_{st} \uparrow \beta \models [C_1^{EF} \wedge \dots \wedge C_{i-1}^{EF} \wedge C_i^S](s) \text{ pour un entier } i \in \{1, \dots, p\}.$$

D'après les définitions de C_i^S et C_i^{EF} , si L est le corps de la règle C_i , il existe une substitution close σ telle que $[L\sigma]_{st}^{EF}$ et $[L\sigma]_{st}^S$ soient vrais dans $T_{st} \uparrow \beta$. Comme par définition de α , pour tout entier $j \in \{1, \dots, q\}$, $A_j\sigma$ et $A'_j\sigma$ n'appartiennent pas simultanément à $T_{st} \uparrow \beta$, on obtient donc une contradiction.

□

Exemple III.5 :

$$P \quad \begin{cases} A \\ A :- A \\ C :- A, B \end{cases}$$

La traduction P_{st} de ce programme est :

$$P_{st} \quad \begin{cases} A \Leftarrow \text{Vrai} \vee (\text{Faux} \wedge A) \\ A' \Leftarrow \text{Faux} \wedge A' \\ A'' \Leftarrow \text{Vrai} \wedge (A' \vee (A \wedge A'')) \\ B \Leftarrow \text{Faux} \\ B' \\ B'' \\ C \Leftarrow A \wedge B \\ C' \Leftarrow A' \vee (A \wedge A'' \wedge B') \\ C'' \Leftarrow A' \vee (A \wedge A'' \wedge (B' \vee (B \wedge B''))) \end{cases}$$

soit après simplification :

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

$T_{st} \uparrow 1 = \{A, B', B''\} = T_{st} \uparrow \alpha$ pour tout ordinal $\alpha \geq 1$.

$T_\theta \uparrow 1 = \{A, B'\}$ $T_\theta \uparrow 2 = \{A, B', C'\} = T_\theta \uparrow \alpha$ pour tout ordinal $\alpha \geq 2$.

$SS_{st} \cup FF'_{st} = \{A, B'\}$.

On voit que sur cet exemple de programme qui ne vérifie pas l'hypothèse H_1 (A est un succès non fini), l'opérateur T_{st} donne une meilleure description de $SS_{st} \cup FF'_{st}$ que T_θ . Nous montrons dans les deux paragraphes suivants que ce résultat est vrai dans le cas général.

Nous aurons besoin plusieurs fois du lemme suivant.

Lemme III.3 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P , soit s un n -uplet de termes et soit X une interprétation de Herbrand de L_1 .

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$X \models \forall C^{EF}(s), \quad X \models \forall C^F(s) \quad \text{et} \quad X \models \neg \exists C^S(s)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$X \models \forall C^{EF}(s) \Leftrightarrow \forall [L\theta]_{st}^{EF},$$

$$X \models \forall C^F(s) \Leftrightarrow \forall [L\theta]_{st}^F \text{ et}$$

$$X \models \exists C^S(s) \Leftrightarrow \exists [L\theta]_{st}^S.$$

Démonstration : Il suffit d'appliquer le lemme II.1 en prenant pour formule F successivement : $[L]_{st}^{EF}$, $[L]_{st}^F$ et $[L]_{st}^S$ qui sont des formules sans quantificateur. Il montre que les différentes formules citées dans le lemme sont des conséquences logiques de la $CET(L_d)$ et sont donc en particulier vraies dans le modèle de la $CET(L_d)$ associé à X (prop. I.4).

□

III.2.2 $T_{st} \uparrow \omega$ -correction et P_{st}^{Her} -correction

Définitions III.4 : Etant donné un programme Prolog P , la **profondeur standard** d'un but G est :

- la profondeur de l'arbre SLD standard de G (c'est-à-dire la longueur de la plus longue branche, un arbre réduit à sa racine étant de profondeur 0) si G échoue finiment.

- la profondeur du sous-arbre situé à gauche de la première branche de succès (branche comprise) si G réussit.

Remarque : La profondeur standard d'un but G n'est donc définie que si G échoue finiment ou réussit.

Notation : On note SS_{st}^n (resp. SS_{fst}^n , resp. FF_{st}^n) les atomes de SS_{st} (resp. SS_{fst} , resp. FF_{st}) de profondeur standard au plus égale à n .

Lemme III.4 : Si un but de profondeur standard n échoue, alors pour toute substitution σ , le but $G\sigma$ a une profondeur standard au plus égale à n et échoue.

Démonstration : En effet, l'arbre SLD standard de $G\sigma$ est isomorphe à un sous-arbre de celui de G . □

Lemme III.5 : Si un but G de profondeur standard n réussit, alors il existe une substitution σ close telle que $G\sigma$ ait une profondeur standard au plus égale à n et réussisse.

Démonstration : Si θ est l'upg correspondant à la première branche de succès de G , il suffit de prendre σ égale à $\theta\tau$ où τ est une substitution close de $G\theta$. □

Lemme III.6 : Soit P un programme Prolog et n un entier.
On suppose que $SS_{st}^n \cup (FF_{st}^n)' \cup (SS_{fst}^n)'' \cup (FF_{st}^n)''' \subset T_{st} \uparrow n$.

Pour tout but $:- L$ de profondeur standard au plus égale à n :

- a) si $:- L$ réussit, alors $\exists [L]_{st}^S$ est vrai dans $T_{st} \uparrow n$.
- b) si $:- L$ échoue, alors $\forall [L]_{st}^{EF}$ est vrai dans $T_{st} \uparrow n$.
- c) si l'arbre SLD standard de $P \cup \{:- L\}$ est fini, alors $\forall [L]_{st}^F$ est vrai dans $T_{st} \uparrow n$.

Démonstration : Soit $L = [A_1, A_2, \dots, A_q]$

La démonstration se fait par récurrence sur q .

- si $q = 0$, c'est évident.
- Supposons le résultat démontré à l'ordre $q-1$ ($q \geq 1$).

a) Il existe une substitution close σ telle que $:- L\sigma$ réussisse et ait une profondeur standard au plus égale à n (lemme III.4). $A_1\sigma \in T_{st} \uparrow n$ puisque $A_1\sigma \in SS_{st}^n$ et par hypothèse de récurrence, $[A_2, \dots, A_q]_{st}^S \sigma$ est vrai dans $T_{st} \uparrow n$ puisque $:- (A_2, \dots, A_q)\sigma$ réussit. Par définition, $[L]_{st}^S \sigma$ et par conséquent $\exists [L]_{st}^S$ est vrai dans $T_{st} \uparrow n$.

b) Pour toute substitution close σ , le but $:- L\sigma$ échoue, et a une profondeur standard inférieure ou égale à n (lemme III.4).

- Si $A_1\sigma$ échoue, alors $A_1'\sigma \in T_{st}\uparrow n$ et par définition de $[L]_{st}^{EF}$, $[L]_{st}^{EF}\sigma$ est vrai dans $T_{st}\uparrow n$.
 - Si $A_1\sigma$ réussit, alors $A_1\sigma \in SS_{fst}^n$ et donc $A_1\sigma \in T_{st}\uparrow n$ et $A_1'\sigma \in T_{st}\uparrow n$.
- Le but :- $(A_2, \dots, A_q)\sigma$ échoue donc et $A_1\sigma \wedge A_1'\sigma \wedge [A_2, \dots, A_q]_{st}^{EF}\sigma$ et par conséquent $[L]_{st}^{EF}\sigma$ est vrai dans $T_{st}\uparrow n$.

Ceci démontre que $\forall [L]_{st}^{EF}$ est vrai dans $T_{st}\uparrow n$.

c) Si le but :-L échoue alors, $\forall [L]_{st}^{EF}$ est vrai dans $T_{st}\uparrow n$ et d'après le lemme III.3, $\forall [L]_{st}^F$ est vrai dans $T_{st}\uparrow n$.

Si :- L réussit finiment, alors pour toute substitution close σ , le but :- $L\sigma$ réussit finiment ou échoue. S'il échoue, on est ramené au cas précédent. S'il réussit finiment, il en est alors de même de :- $A_1\sigma$ et de :- $(A_2, \dots, A_q)\sigma$. Comme $A_1\sigma$ et $A_1'\sigma$ appartiennent à $T_{st}\uparrow n$ et d'après l'hypothèse de récurrence, $[A_2, \dots, A_q]_{st}^F\sigma$ est vrai dans $T_{st}\uparrow n$. Par conséquent, $\forall [L]_{st}^F$ est vrai dans $T_{st}\uparrow n$. □

Le résultat suivant prouve que les calculs d'un interpréteur standard appliqués à un programme Prolog P sont $T_{st}\uparrow\omega$ -corrects, c'est-à-dire qu'ils peuvent être calculés par "chaînage avant" à partir de la traduction axiomatique standard de P.

Théorème III.2 :

$$SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst} \subset T_{st}\uparrow\omega.$$

Démonstration : On démontre par récurrence sur n, et en utilisant les résultats du lemme III.6 que

$$SS_{st}^n \cup (FF_{st}^n)' \cup (FF_{st}^n)'' \cup (SS_{fst}^n)'' \subset T_{st}\uparrow n. \quad (1)$$

Si $n=0$, c'est évident puisque tous les ensembles ci-dessus sont vides.

Montrons par exemple que $SS_{st}^{n+1} \subset T_{st}\uparrow(n+1)$ si l'on suppose l'inclusion (1) vérifiée pour n.

Soit $A = p(s) \in SS_{st}^{n+1}$. Il existe un entier j tel que la résolution de A avec les j-1 premières règles $C_1, \dots, C_{j-1}$ de prédicat de tête p échouent et avec la j-ème règle C_j réussisse.

Soit $i \in \{1, \dots, j-1\}$ et soit L_i le corps de la règle C_i .

- si A n'est pas unifiable avec la tête de C_i , alors : $T_{st}\uparrow n \models C_i^{EF}(s)$ d'après le lemme III.3.

- si A lui est unifiable, avec la substitution σ , alors :- L_i échoue et donc par le lemme précédent, $\forall [L_i\sigma]_{st}^{EF}$ est vrai dans $T_{st}\uparrow n$. Par conséquent : $T_{st}\uparrow n \models C_i^{EF}(s)$ d'après le lemme III.3.

A est unifiable avec la tête de la règle C_j de corps L_j , avec la substitution σ . Le but $:- L_j\sigma$ réussit et donc $\exists[L_i\sigma]_{st}^S$ est vrai dans $T_{st}\uparrow n$ et $: T_{st}\uparrow n \models C_i^S(s)$ d'après le lemme III.3.

En conclusion, $[C_1^{EF} \wedge \dots \wedge C_{j-1}^{EF} \wedge C_j^S](s)$ est vraie dans $T_{st}\uparrow n$ et donc $A \in T_{st}\uparrow(n+1)$.

□

Corollaire III.2 : Soit P un programme Prolog et n un entier.

Pour tout but $:- L$ de profondeur standard au plus égale à n.

- a) si $:- L$ réussit, alors $\exists[L]_{st}^S$ est vrai dans $T_{st}\uparrow n$.
- b) si $:- L$ échoue, alors $\forall[L]_{st}^{EF}$ est vrai dans $T_{st}\uparrow n$.
- c) si l'arbre SLD standard de $P \cup \{:- L\}$ est fini, alors $\forall[L]_{st}^F$ est vrai dans $T_{st}\uparrow n$.

Démonstration : C'est l'énoncé du lemme III.6 sans l'hypothèse :

$$SS_{st}^n \cup (FF_{st}^n)' \cup (FF_{st}^n)'' \cup (SS_{fst}^n)'' \subset T_{st}\uparrow n. \quad (1)$$

qui a été démontrée dans le théorème III.2.

On déduit du Théorème III.2 un résultat de **correction** de la sémantique opérationnelle standard pour l'axiomatique P_{st}^{Her} .

Théorème III.3 : Soit $G = :- L$ un but.

- a) Si G réussit avec la réponse calculée σ , alors $P_{st}^{Her} \models \forall[L\sigma]_{st}^S$.
- b) Si G échoue finiment alors, $P_{st}^{Her} \models \forall[L]_{st}^{EF}$.
- c) Si l'arbre SLD standard de G est fini, alors $P_{st}^{Her} \models \forall[L]_{st}^F$.

Démonstration :

a) Si G réussit avec la réponse calculée σ , alors pour toute substitution close τ , le but $:- L\sigma\tau$ réussit. Le corollaire III.2 montre alors que $[L\sigma\tau]_{st}^S$ est vrai dans $T_{st}\uparrow \omega$ donc dans le plus petit point fixe de T_{st} (qui est égal au plus petit modèle de Herbrand de P_{st}) puisque c'est une formule positive et que T_{st} est monotone et par conséquent, que $[L\sigma\tau]_{st}^S$ est conséquence logique de P_{st}^{Her} . Ceci étant vrai pour toute substitution close τ , on a bien :

$$P_{st}^{Her} \models \forall[L\sigma]_{st}^S.$$

b) Si G échoue finiment alors, le corollaire III.2 montre que $\forall[L]_{st}^{EF}$ est vrai dans $T_{st}\uparrow \omega$, donc aussi dans le plus petit modèle de Herbrand de P_{st} donc $\forall[L]_{st}^{EF}$ est conséquence logique de P_{st}^{Her} .

Démonstration analogue pour le c).

□

Corollaire III.3 : Soit A un atome clos.

- a) Si $A \in SS_{st}$ alors $P_{st}^{Her} \models A$.
 b) Si $A' \in FF'_{st}$ alors $P_{st}^{Her} \models A'$.
 c) Si $A'' \in FF''_{st} \cup SS''_{fst}$ alors $P_{st}^{Her} \models A''$.

Démonstration : Il suffit d'appliquer le théorème III.3. □

III.2.3 $T_{st} \uparrow \omega$ -complétude et P_{st}^{Her} -complétude

Lemme III.7 : Soit le but $:- L$. S'il existe un entier N tel que pour toute substitution σ telle que la liste $L\sigma$ soit close, le but $:- L\sigma$ ait un arbre fini de profondeur standard inférieure ou égale à N , alors $:- L$ a également un arbre fini de profondeur standard inférieure ou égale à N .

Démonstration : Supposons la conclusion fausse. Il suffit d'instancier une branche de longueur supérieure à N pour obtenir une contradiction. □

Définition III.4 : Pour un programme Prolog P , on note M_P le nombre maximum d'atomes figurant dans un corps d'une règle.

$$\begin{aligned} \text{Soit } d_n &= \frac{(M_P)^n - 1}{M_P - 1} & \text{si } M_P \geq 2 \\ &= n & \text{si } M_P = 1 \\ &= 1 & \text{si } M_P = 0. \end{aligned}$$

d_n vérifie la relation de récurrence $d_{n+1} = M_P d_n + 1$.

Lemme III.8 : Supposons que pour tout atome A , si $A'' \in T_{st} \uparrow n$ alors le but $:- A$ a un arbre de profondeur standard inférieure ou égale à d_n .

Soit L une liste d'atomes telle que $\forall [L]_{st}^F$ soit vrai dans $T_{st} \uparrow n$.

Alors le but $:- L$ a un arbre de profondeur standard inférieure ou égale à qd_n .

Démonstration : Soit $L = A_1, A_2, \dots, A_q$. On montre ce lemme par récurrence sur q .
 Si $q = 0$, c'est évident.

Sinon, pour toute substitution σ telle que $L\sigma$ soit close, $[L\sigma]_{st}^F$ est vrai dans $T_{st} \uparrow n$.

- si $A_1'\sigma \in T_{st} \uparrow n$, alors $A_1'\sigma \in T_{st} \uparrow n$ et par hypothèse, le but $:- A_1\sigma$ a un arbre SLD standard de profondeur standard inférieure ou égale à d_n . Le but $:- A_1\sigma$ ne peut pas réussir puisque sinon on aurait : $A_1'\sigma$ et $A_1\sigma \in T_{st} \uparrow n$ d'après le théorème III.2.

Donc le but $:- L\sigma$ échoue et a un arbre de profondeur standard inférieure ou égale à d_n .

- si $A_1'\sigma \notin T_{st}\uparrow n$, alors $A_1\sigma$ et $A_1'\sigma \in T_{st}\uparrow n$. Le but :- $A_1\sigma$ a un arbre de profondeur standard inférieure ou égale à d_n . $[A_2, \dots, A_q]_{st}^F\sigma$ est vrai dans $T_{st}\uparrow n$. L'hypothèse de récurrence montre que :- $(A_2, \dots, A_q)\sigma$ a un arbre de profondeur standard inférieure ou égale à $(q-1)d_n$.
Le but :- $L\sigma$ a donc bien un arbre de profondeur standard inférieure ou égale à qd_n .

Le lemme III.7 montre alors que :- L a un arbre de profondeur standard inférieure ou égale à qd_n . □

Remarque : Une des raisons de l'échec de T_θ à rendre compte exactement des succès et des échecs pour un interpréteur standard provenait du fait que l'on pouvait avoir :

$A(t) \in T_\theta\uparrow 2$ pour tout terme clos t alors que la profondeur standard de l'arbre SLD standard de $A(t)$ était non bornée (voir exemple III.2). La proposition suivante montre que ce cas ne peut pas se produire avec l'opérateur T_{st} . L'appartenance d'un atome à $T_{st}\uparrow n$ implique que son arbre SLD standard a une profondeur standard bornée en fonction de n .

Proposition III.2 : Si $A'' \in T_{st}\uparrow n$ alors, le but :- A a un arbre fini de profondeur standard inférieure ou égale à d_n .

Démonstration : On démontre cette proposition par récurrence sur n .

Si $n = 1$, les seules règles qui peuvent s'appliquer à A ont un corps vide. Le résultat est donc vrai dans ce cas.

Supposons la proposition vraie pour l'entier n .

Soit $A'' \in T_{st}\uparrow (n+1)$.

Supposons que A soit unifiable avec la tête de la i -ème règle de corps L_i , avec la substitution σ . Alors $\forall [L_i\sigma]_{st}^F$ est vrai dans $T_{st}\uparrow n$. L'hypothèse de récurrence permet d'appliquer le lemme III.8.

Le but :- $L_i\sigma$ a un arbre de profondeur standard ou égale à $qd_n \leq M_{pd_n}$ et A a donc un arbre de profondeur standard ou égale à $M_{pd_n} + 1 = d_{n+1}$. □

Corollaire III.4 : Pour tout programme,

$$T_{st}\uparrow \omega \cap H'' = SS_{fst}'' \cup FF_{st}''.$$

Démonstration : On a déjà montré que $SS_{fst}'' \cup FF_{st}'' \subset T_{st}\uparrow \omega \cap H''$.

L'inclusion inverse est une application immédiate de la proposition III.2 puisque tout atome dont l'arbre SLD standard est fini est soit un échec, soit un succès fini. □

Remarque : Ce résultat montre que $T_{st}\uparrow \omega$ caractérise exactement l'ensemble des échecs et des succès finis d'un programme P soumis à un interpréteur Prolog Standard.

En effet,

$$\begin{aligned} SS_{fst} &= \{ A \in H \mid A \text{ et } A'' \in T_{st} \uparrow \omega \} \text{ et} \\ FF_{st} &= \{ A \in H \mid A' \in T_{st} \uparrow \omega \}. \end{aligned}$$

Exemple III.6 : Soit P le programme :

$$\begin{cases} A \\ A :- A \\ C :- A, B \end{cases}$$

$$T_{st} \uparrow 1 = \{ A, B', B'' \} = T_{st} \uparrow \alpha \text{ pour tout ordinal } \alpha \geq 1.$$

B est un échec fini.

A n'est pas un succès fini puisque $A'' \notin T_{st} \uparrow \omega$.

Corollaire III.5 : Soit n un entier.

a) si $\forall [L]_{st}^F$ est vraie dans $T_{st} \uparrow n$, alors le but $:- L$ a un arbre fini de profondeur standard inférieure ou égale à qd_n où q est le nombre d'atomes de la liste L.

b) si $\forall [L]_{st}^{EF}$ est vraie dans $T_{st} \uparrow n$, alors le but $:- L$ échoue finiment et a un arbre fini de profondeur standard inférieure ou égale à qd_n .

Démonstration : Evident d'après le lemme III.7 et la proposition précédente. □

Théorème III.4 : Si un programme Prolog P vérifie l'hypothèse H_3 , alors

$$T_{st} \uparrow \omega = SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst}.$$

Démonstration :

Montrons par récurrence sur n que $T_{st} \uparrow n \subset SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst}$.

Pour $n = 0$, c'est évident.

Supposons l'inclusion établie pour un entier n.

1) Soit $p(s) \in T_{st} \uparrow (n+1)$ où s est un n-uplet de termes clos et soient $C_1, \dots, C_p$ les règles de P de prédicat de tête p.

Alors : $T_{st} \uparrow n \models [C_1^S \vee (C_1^{EF} \wedge C_2^S) \vee \dots \vee (C_1^{EF} \wedge \dots \wedge C_{p-1}^{EF} \wedge C_p^S)](s)$.

Soit $j \in \{1, \dots, p\}$ tel que : $T_{st} \uparrow n \models [C_1^{EF} \wedge \dots \wedge C_{j-1}^{EF} \wedge C_j^S](s)$.

a) $C_i^{EF}(s)$ est vrai dans $T_{st} \uparrow n$ pour $i \in \{1, \dots, j-1\}$. Cette formule peut s'écrire :

$$\forall y(s \neq t \vee \forall [L_i]_{st}^{EF})$$

où les y_i sont les variables apparaissant dans la tête de la règle C_i .

Le lemme III.3 montre que si $C_i^{EF}(s)$ est vraie dans $T_{st} \uparrow n$ alors :

- $p(s)$ n'est pas unifiable avec la tête de la i -ème règle de P

ou :

- l'unification est possible (soit σ l'upg associé) et que $\forall [L_i \sigma]_{st}^{EF}$ est vraie dans $T_{st} \uparrow n$. Le Corollaire III.5 montre alors que le but :- $L \sigma$ échoue.

La règle C_i de P donne donc un échec standard pour l'atome $p(s)$.

b) $C_j^S(s)$ est vrai dans $T_{st} \uparrow n$.

$C_j^S(s)$ peut s'écrire sous la forme :

$\exists y(s = t \wedge \exists [L_j]_{st}^S)$ où les y_i sont les variables apparaissant dans la tête de

C_j .

L'atome $p(s)$ et la tête de la règle C_j sont unifiables. Soit σ la substitution associée.

$$T_{st} \uparrow n \models \exists [L_j \sigma]_{st}^S$$

c.-à-d. qu'il existe une substitution τ telle que $L_j \sigma \tau$ soit close et :

$$T_{st} \uparrow n \models [L_j \sigma \tau]_{st}^S.$$

D'après l'hypothèse de récurrence, le but :- $L_j \sigma \tau$ réussit et l'hypothèse H_3 assure alors que le but :- $L_j \sigma$ réussit; la règle C_j donne un succès standard pour $p(s)$ qui appartient donc à SS_{st} .

2) Soit $p'(s) \in T_{st} \uparrow (n+1)$. Alors :

$$T_{st} \uparrow n \models [C_1^{EF} \wedge \dots \wedge C_p^{EF}](s)$$

donc, sous l'hypothèse H_3 et d'après ce qui précède, les règles de P de prédicat de tête p donnent un échec Prolog pour $p(s)$ et $p'(s)$ appartient donc à FF_{st}' .

3) Soit $p''(s) \in T_{st} \uparrow (n+1)$. Alors :

$$T_{st} \uparrow n \models [C_1^F \wedge \dots \wedge C_p^F].$$

$C_i^F(s)$ est vrai dans $T_{st} \uparrow n$ pour $i \in \{1, \dots, p\}$.

La formule $C_i^F(s)$ peut s'écrire :

$$\forall y(s \neq t \vee \forall [L_i]_{st}^F)$$

où les y_i sont les variables apparaissant dans la tête de C_i .

Le lemme III.3 montre que si $C_i^F(s)$ est vrai dans $T_{st} \uparrow n$ alors :

- $p(s)$ n'est pas unifiable avec la tête de C_i

ou :

- l'unification est possible (soit σ l'upg associé) et que $\forall [L_i \sigma]_{st}^F$ est vraie dans $T_{st} \uparrow n$. Le corollaire III.5 montre alors que le but :- $L_i \sigma$ a un arbre fini de profondeur standard inférieure ou égale à Mpd_n .

La règle C_i donne donc soit un échec, soit un succès fini pour l'atome $p(s)$. Comme ceci est vrai pour toutes les règles de P , $p''(s)$ appartient donc à $SS_{st}'' \cup FF_{st}''$.

□

Corollaire III.6 :

1. Si P est un programme sans variable locale,

$$T_{st} \uparrow \omega = SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst}.$$

2. Si P est un programme propositionnel pur,

$$pppf(T_{st}) = T_{st} \uparrow \omega = SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst}.$$

3. Si P est un programme sans variable locale et sans symbole de fonction,

$$pppf(T_{st}) = T_{st} \uparrow \omega = SS_{st} \cup FF'_{st} \cup FF''_{st} \cup SS''_{fst}.$$

Démonstration :

1. En effet, les programmes sans variables locales vérifient l'hypothèse H_3 .
 2. et 3. Pour tout atome A contenant une occurrence d'un symbole fonctionnel ne figurant pas dans P , $A' \in T_{st} \uparrow 1$ puisque A n'est unifiable avec aucune tête de règles de P . Et comme les autres atomes sont en nombre fini, $T_{st} \uparrow \omega$ est le plus petit point fixe de T_{st} .

□

La condition H_3 est donc suffisante pour qu'un programme soit décrit par $T_{st} \uparrow \omega$. Ce résultat est à rapprocher du théorème III.1. Nous voyons que la traduction P_{st} constitue donc une meilleure approximation axiomatique de l'ensemble des succès et des échecs finis d'un interpréteur Prolog Standard que P_θ . Il n'en demeure pas moins que H_3 est indécidable et qu'en général, $T_{st} \uparrow \omega$ n'est pas le plus petit modèle de Herbrand de P_{st} .

On peut néanmoins énoncer le résultat de **complétude** suivant :

Théorème III.5 : Si P est un programme qui vérifie l'hypothèse H_3 (par exemple sans variable locale) et pour lequel $T_{st} \uparrow \omega$ est le plus petit point fixe de T_{st} (par exemple sans symbole de fonction), alors pour tout atome clos A , on a :

- a) $A \in SS_{st} \quad ssi \quad P_{st}^{Her} \models A.$
 b) $A' \in FF'_{st} \quad ssi \quad P_{st}^{Her} \models A'.$
 c) $A'' \in FF''_{st} \cup SS''_{fst} \quad ssi \quad P_{st}^{Her} \models A''.$

Démonstration :

- a) Si $P_{st}^{Her} \models A$, alors $A \in T_{st} \uparrow \omega$ donc $A \in SS_{st}$.
 b) Si $P_{st}^{Her} \models A'$, alors $A' \in T_{st} \uparrow \omega$ donc $A' \in FF'_{st}$.
 c) Si $P_{st}^{Her} \models A''$, alors $A'' \in T_{st} \uparrow \omega$ donc $A'' \in FF''_{st} \cup SS''_{fst}$.

□

On montre au paragraphe IV.3 qu'un programme sans variable locale est tel que $T_{st} \uparrow \omega$ est le plus petit point fixe de T_{st} . Pour ces programmes, P_{st}^{Her} est donc une caractérisation axiomatique correcte et complète de leur sémantique opérationnelle standard.

Il n'est pas possible d'avoir de résultats de complétude plus généraux. Par exemple, il est faux que si $P_{st}^{Her} \models \forall A \sigma$, alors le but $\vdash A$ a une réponse calculée τ telle que $\sigma \geq \tau$.

Contre-exemple : [Llo 87]

Soit P le programme $\{p(a)\}$. Si l'on suppose que le langage L ne contient que la constante a alors $P_{st}^{Her} \models \forall x p(x)\sigma$ où σ est la substitution identité et la seule réponse calculée est $\tau = (x/a)$. On n'a évidemment pas $\sigma \geq \tau$.

On pourrait penser qu'il suffit que le langage sous-jacent L contienne suffisamment de constantes pour que l'on puisse obtenir un résultat de complétude plus général, comme c'est le cas pour la sémantique d'un interpréteur idéal. L'exemple suivant montre que ce n'est pas le cas :

Soit $P \begin{cases} B(a) :- B(a) \\ B(b) \end{cases}$ et L un langage quelconque contenant les constantes a et b .

P vérifie l'hypothèse H_3 et T_{st} est continu puisque P ne contient pas de variables locales. Soit $\sigma = (x \setminus b)$. On a $T_{st} \uparrow 1 \models \forall (B(x)\sigma)$ et pourtant, le but $:- B(x)$ n'a pas de réponse calculée standard¹. Ceci provient du fait que l'hypothèse H_3 n'est pas encore assez forte. Il faudrait la remplacer par H'_3 : pour tout but $:- L$ et toute substitution close τ des variables de L , si le but $:- L\tau$ réussit alors le but $:- L$ réussit avec une réponse calculée (standard) θ telle que $\theta \leq \tau$. Sous cette hypothèse, on peut démontrer le résultat suivant :

Proposition III.3 : *Soit P un programme Prolog vérifiant l'hypothèse H'_3 et pour lequel $T_{st} \uparrow \omega$ est un point fixe de T_{st} , L une liste d'atomes et σ une substitution des variables de L . Alors, si L contient assez de constantes n'apparaissant ni dans P ni dans $L\sigma$ pour instancier les variables libres de $L\sigma$ et si $P_{st}^{Her} \models \forall [L\sigma]_{st}^S$ alors le but $:- L$ a une réponse calculée standard θ telle que $\theta \leq \sigma$.*

Démonstration :

Comme $P_{st}^{Her} \models \forall [L\sigma]_{st}^S$ et comme $T_{st} \uparrow \omega$ est un point fixe de T_{st} (et donc aussi un plus petit modèle de Herbrand de P_{st}), on a : $T_{st} \uparrow \omega \models \forall [L\sigma]_{st}^S$. Soient $\{\theta_i\}_{i \in I}$ les réponses calculées standards pour $P \cup \{:- L\}$. Supposons que pour tout $i \in I$, θ_i ne soit pas plus générale que σ . Soit τ une substitution close des variables de $L\sigma$ par des constantes ne figurant pas dans P . Le but $:- L\sigma\tau$ réussit d'après le théorème III.4 et aucune des réponses θ_i n'est plus générale que $\sigma\tau$ ce qui contredit l'hypothèse H'_3 .

□

Par contre, même si P vérifie les hypothèses de la proposition précédente, $P_{st}^{Her} \models \forall [L]_{st}^{EF}$ (resp. $P_{st}^{Her} \models \forall [L]_{st}^F$) n'implique pas que le but $:- L$ échoue finiment (resp. ait un arbre SLD standard fini). Il suffit pour s'en assurer de considérer le programme suivant :

$\{p(s(x)) :- p(x)\}$

¹c'est-à-dire calculée par un interpréteur standard.

Quelque soit le langage L , on a $P_{st}^{Her} \models \forall x p'(x)$ puisque pour tout terme clos t , $p(t) \in FF_{st}$ et l'arbre SLD standard de $P \cup \{-p(x)\}$ n'est pas un arbre d'échecs finis.

Rappelons néanmoins l'énoncé du Corollaire III.5.

Soit P un programme Prolog et L une liste d'atomes.

- a) S'il existe un entier n pour lequel $T_{st} \uparrow n \models \forall [L]_{st}^F$ alors, l'arbre SLD standard de $P \cup \{:- L\}$ est fini.
- b) S'il existe un entier n pour lequel $T_{st} \uparrow n \models \forall [L]_{st}^{EF}$ alors, l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échec fini.

Pour résumer : La traduction axiomatique P_{st}^{Her} réalise une meilleure approximation de la sémantique opérationnelle standard que la traduction P_{θ}^{Her} . Cette traduction est correcte et complète pour les programmes sans variable locale et sans symbole de fonction mais dans le cas général, trois "points noirs" subsistent :

- 1- Le fait que l'on ait obtenu un résultat de complétude limité (le théorème III.4) pour une classe partielle (et indécidable) de programmes : les sémantiques finies que nous étudions par la suite ne présentent pas le même inconvénient.
- 2- Même si le programme considéré P vérifie l'hypothèse H_3 (ou H_3'), la complétude n'est obtenue qu'à certaines conditions sur le langage sous-jacent et surtout, sur la continuité de l'opérateur de conséquence immédiate associé.
- 3- Même si l'opérateur T_{st} est continu, nous n'obtenons des résultats de complétude entièrement satisfaisants que pour des buts constitués d'atomes clos. Ces résultats se généralisent assez mal à des buts contenant des variables.

Nous retrouverons ces deux derniers problèmes lors de l'étude des sémantiques finies et nous montrerons au chapitre VI qu'il est dû au cadre choisi : la restriction de la notion de modèle aux seuls modèles de Herbrand. La solution à ce problème consiste à se placer dans le cadre de la Théorie Equationnelle de Clark.

III.3 Indécidabilité des hypothèses H_1 , H_2 et H_3 .

Il est démontré dans [Llo 84] que pour toute fonction n -aire récursive partielle f , il existe un programme Prolog P_f et un prédicat $(n+1)$ -aire p_f tel que le but $:- p_f(k_1, \dots, k_n, x)$ ait pour réponse calculée $x = k$ si et seulement si $f(k_1, \dots, k_n) = k$.

Il est facile de vérifier dans cette démonstration que pour tous les programmes P_f ci-dessus, les réponses calculées par un interpréteur standard sont les mêmes que celles calculées par un interpréteur "en largeur d'abord et équitable".

D'autre part, si L est un langage de programmation dans lequel il est possible de calculer toutes les fonctions récursives partielles, il n'existe pas d'algorithme permettant

de décider si un programme quelconque donné P de L boucle ou non pour une donnée d (voir par exemple [Del 87d]).

Nous utiliserons ce résultat ici sous la forme suivante : il n'existe pas d'algorithme permettant de décider si un but quelconque :- A (où A est un atome clos) boucle ou non pour un programme Prolog P soumis à un interpréteur Standard.

Proposition III.4 : *Il n'existe pas d'algorithme permettant de décider si un succès A d'un programme Prolog P quelconque est un succès fini.*

Démonstration : Soit P un programme Prolog quelconque et A un atome clos. On forme le programme P' défini à partir de P par :

$$P' \begin{cases} B \\ B :- A \\ P \end{cases} \quad \text{où } B \text{ est un symbole de prédicat ne figurant pas dans } P.$$

B est un succès de P'.

B est un succès fini de P' si et seulement si :- A ne boucle pas pour P.

Un algorithme permettant de décider si B est un succès fini de P' permettrait alors de décider si A boucle ou non pour P. Un tel algorithme n'existe donc pas. □

Proposition III.5 : *Il n'existe pas d'algorithme permettant de décider si, pour un programme Prolog P et un corps de règle :- L de P, l'échec fini de toute instance close de :- L implique l'échec fini de :- L (pour un interpréteur standard).*

Démonstration : Soit P un programme Prolog défini par la suite de règles (R₁, ..., R_p). Pour chaque règle R, on définit la règle R⁺ par :

$$\begin{aligned} R &= A(t) :- B_1(t_1), \dots, B_n(t_n) \\ &\quad \text{ssi} \\ R^+ &= A(t, s(z)) :- B_1(t_1, z), \dots, B_n(t_n, z) \end{aligned}$$

où z (resp. s) est une variable (resp. un symbole de fonction) n'apparaissant pas dans P.

Soit G le but :- B₁(t₁), ..., B_n(t_n). On définit le but G⁺ par :- B₁(t₁, z), ..., B_n(t_n, z).

Soit P⁺ le programme (R₁⁺, ..., R_p⁺, X :- G⁺, fail) où X et fail sont deux nouveaux symboles de prédicats 0-aires.

• Pour toute instance close τ des variables de G⁺, l'arbre SLD standard de P⁺ ∪ {:- G⁺} est fini. En effet, τ instancie en particulier la nouvelle variable z et chaque application d'une règle R⁺ diminue d'une unité la hauteur de ce paramètre.

On peut montrer que si zτ = sⁿ(a) où a n'est pas une image de s et si M est la longueur maximale des corps des règles de P alors, la profondeur de l'arbre SLD standard de P⁺ ∪ {:- A(t, sⁿ(a))} est inférieure ou égale à d_n (cf définition III.4).

- Pour toute instance close τ des variables de G^+ , $P^+ \cup \{:- G^+\tau, \text{fail}\}$ échoue finiment (pour un interpréteur standard).
- $P^+ \cup \{:- G^+, \text{fail}\}$ échoue finiment ssi $P^+ \cup \{:- G^+\}$ ne boucle pas.
- $P^+ \cup \{:- G^+\}$ ne boucle pas ssi $P \cup \{:- G\}$ ne boucle pas. En effet, la nouvelle variable z n'impose aucune contrainte.
- Un algorithme permettant de décider si $P^+ \cup \{:- G^+, \text{fail}\}$ échoue finiment permettrait du même coup de décider si $P \cup \{:- G\}$ boucle.

□

Proposition III.6 : *Il n'existe pas d'algorithme permettant de décider si pour un programme Prolog P et un corps de règle $:- A_1, \dots, A_n$ de P , le succès d'une instance close de $:- A_1, \dots, A_n$ implique le succès de $:- A_1, \dots, A_n$.*

Démonstration : Soit P un programme, A un prédicat de P . Soient B et C deux nouveaux symboles de prédicat, a et b deux nouvelles constantes et P' le programme obtenu en adjoignant à P les règles :

$$\begin{cases} C :- B(x) \\ B(a) :- A \\ B(b) \end{cases}$$

Le but $:- B(b)$ réussit.

Le but $:- B(x)$ réussit si et seulement si le programme P s'arrête pour $:- A$.

Un algorithme permettant de décider si le but $:- B(x)$ réussit permettrait alors de décider si le but $:- A$ boucle ou non pour P .

□

Chapitre IV :

Les sémantiques finies

IV.1. La sémantique standard finie

Des considérations opérationnelles (les seuls programmes pratiquement admissibles sont ceux qui ne bouclent pas) et théoriques (le corollaire III.3 montre que l'axiomatique standard *restreinte aux échecs et succès tous deux finis* est satisfaisante) nous ont conduits à définir la sémantique opérationnelle standard finie (§ II.4.3) d'un programme Prolog P et à tenter d'en donner une caractérisation axiomatique P_{fst} en utilisant une méthode analogue à celle développée au chapitre précédent. Nous montrons que $T_{fst} \uparrow \omega$ (où T_{fst} est l'opérateur de conséquence immédiate associé à P_{fst} défini sur les interprétations de Herbrand du langage sous-jacent) représente exactement la sémantique opérationnelle standard finie de P puis nous en tirons les conséquences en termes de correction et de complétude dans le cadre des modèles de Herbrand. Nous montrerons au chapitre VI que c'est dans le cadre de la Théorie Equationnelle de Clark que ce résultat a la signification logique la plus forte. Cette traduction s'exprime dans le langage dédoublé L_d (§ I.1).

IV.1.1 Définition de la traduction axiomatique standard finie

On suppose dans tout ce paragraphe que la notion de succès fini ou d'échec fini se réfère à un **interpréteur Prolog Standard**. On dira donc, étant donné un programme P , qu'un but G **échoue** ssi l'arbre SLD standard pour $P \cup \{G\}$ est un arbre d'échecs finis et que G **réussit finiment** ssi l'arbre SLD standard pour $P \cup \{G\}$ est fini et contient la clause vide.

Considérons un programme Prolog propositionnel dont les règles ayant pour symbole de prédicat de tête A sont (dans cet ordre) :

$$\begin{cases} A :- B, C \\ A :- D \end{cases}$$

$[-: A \text{ réussit finiment}]$ ssi $[(-: B, C \text{ réussit finiment et } -: D \text{ est fini}) \text{ ou } (-: B, C \text{ échoue finiment et } -: D \text{ réussit finiment})]$

ou, d'une manière équivalente :

$[-: A \text{ réussit finiment}]$ ssi $[(-: B, C \text{ et } -: D \text{ sont finis}) \text{ et } (-: B, C \text{ réussit finiment ou } -: D \text{ réussit finiment})]$.

$[-: A \text{ échoue finiment}]$ ssi $[-: B, C \text{ échoue finiment et } -: D \text{ échoue finiment}]$.

$[-: B, C \text{ réussit finiment}]$ ssi $[-: B \text{ et } -: C \text{ réussissent finiment}]$.

$[-: B, C \text{ échoue finiment}]$ ssi $[-: B \text{ échoue finiment ou } (-: B \text{ réussit finiment et } -: C \text{ échoue finiment})]$.

La notion de **but fini** peut être caractérisée par :

$[-: A \text{ est fini}]$ ssi $[(-: B \text{ échoue finiment ou } (-: B \text{ réussit finiment et } -: C \text{ est fini})) \text{ et } (-: D \text{ est fini})]$.

Cet exemple conduit à introduire les définitions suivantes.

Soit $L = A_1, A_2, \dots, A_n$ une liste d'atomes. On définit :

$$- [L]_{fst}^{SF} = A_1 \wedge A_2 \wedge \dots \wedge A_n \text{ et } []_{fst}^{SF} = \text{Vrai.}$$

$$- [L]_{fst}^{EF} = A_1' \vee (A_1 \wedge [A_2, \dots, A_n]_{fst}^{EF}) \text{ et } []_{fst}^{EF} = \text{Faux.}$$

$$- [L]_{fst}^F = A_1' \vee (A_1 \wedge [A_2, \dots, A_n]_{fst}^F) \text{ et } []_{fst}^F = \text{Vrai.}$$

Si les A_j sont des atomes clos, le sens souhaité de :

$[L]_{fst}^{SF}$ est : le but $[-: L \text{ réussit finiment}]$ (c.-à-d. l'arbre SLD standard pour $P \cup \{ -: L \}$ est fini et contient la clause vide $\emptyset$)

$[L]_{fst}^{EF}$ est : le but $[-: L \text{ échoue finiment}]$ (c.-à-d. l'arbre SLD standard pour $P \cup \{ -: L \}$ est un arbre d'échecs finis)

$[L]_{fst}^F$ est : l'arbre SLD standard pour $P \cup \{ -: L \}$ est fini.

Exemple IV.1 :

$$[A_1, A_2]_{fst}^{EF} = A_1' \vee (A_1 \wedge A_2')$$

$$[A_1, A_2]_{fst}^F = A_1' \vee (A_1 \wedge (A_2' \vee A_2))$$

Lemme IV.1 : Soit $X \subset H_d$ (base de Herbrand associée à L_d) et soit L une liste d'atomes clos.

$$[L]_{fst}^F \text{ est vrai dans } X \text{ ssi } ([L]_{fst}^{SF} \vee [L]_{fst}^{EF}) \text{ est vrai dans } X.$$

Démonstration : Soit $L = A_1, A_2, \dots, A_q$.

Ce lemme se démontre par récurrence sur q .

Si $q = 0$, c'est évident puisque $[]_{fst}^F$ et $[]_{fst}^{SF}$ sont vrais (dans X).

Soit $q \geq 1$ et supposons le résultat démontré pour toute liste de longueur $q - 1$.

- Supposons que $[L]_{fst}^F$ soit vrai dans X .

Si $A'_1 \in X$, alors $[L]_{fst}^{EF}$ est vrai dans X .

Sinon, $A_1 \in X$ et $[A_2, \dots, A_q]_{fst}^F$ est vrai dans X .

Si $[A_2, \dots, A_q]_{fst}^{EF}$ est vrai dans X alors $[L]_{fst}^{EF}$ est vrai dans X .

Si $[A_2, \dots, A_q]_{fst}^{SF}$ est vrai dans X alors $[L]_{fst}^{SF}$ est vrai dans X .

La réciproque se montre de manière analogue.

□

On associe à toute règle $C \equiv p(t) :- L$ du programme les formules suivantes de L_d :

$$\begin{aligned} C^S(x) &\equiv [\exists y(x = t \wedge [L]_{fst}^{SF})] \\ C^{EF}(x) &\equiv [\forall y(x \neq t \vee [L]_{fst}^{EF})] \\ C^F(x) &\equiv [\forall y(x \neq t \vee [L]_{fst}^F)] \end{aligned}$$

où les y_i sont les variables apparaissant dans les termes t_i et les atomes de L et où les x_i sont de nouvelles variables.

Remarque : Nous utilisons à nouveau les mêmes notations que pour les traductions P_θ et P_{st} afin de ne pas les alourdir par des indices supplémentaires.

- Si $C_1, C_2, \dots, C_p$ sont les règles de P dont le symbole de prédicat de tête est p , on considère les axiomes suivants (que l'on suppose quantifiés universellement) :

$$\begin{aligned} p(x) &\Leftarrow ((C_1^F \wedge \dots \wedge C_p^F) \wedge (C_1^S \vee \dots \vee C_p^S))(x) \\ p'(x) &\Leftarrow (C_1^{EF} \wedge \dots \wedge C_p^{EF})(x) \end{aligned}$$

- Si aucune règle de P n'admet le symbole p comme prédicat de tête, on considère l'axiome :

$$\forall x \, p'(x).$$

Notation : P_{fst} est la réunion de ces axiomes pour tout symbole de prédicat du langage.

Exemple IV.2 : Soit P le programme Prolog :

$$\begin{cases} A(a) :- A(a) \\ A(x) \\ B :- C(x), A(x) \\ C(b) \\ D :- E, A(x) \end{cases}$$

La traduction standard finie de P est alors le système P_{fst} :

$$\begin{cases} A(x) \Leftarrow (x \neq a \vee A(a) \vee A'(a)) \wedge \text{Vrai} \wedge ((x = a \wedge A(a)) \vee \text{Vrai}) \\ A'(x) \Leftarrow (x \neq a \vee A'(a)) \wedge \text{Faux} \\ B \Leftarrow \forall x(C'(x) \vee (C(x) \wedge A'(x)) \vee (C(x) \wedge A(x))) \wedge \exists x(C(x) \wedge A(x)) \\ B' \Leftarrow \forall x(C'(x) \vee (C(x) \wedge A'(x))) \\ \begin{cases} C(x) \Leftarrow x = b \\ C'(x) \Leftarrow x \neq b \end{cases} \\ D \Leftarrow \forall x(E' \vee (E \wedge A'(x)) \vee (E \wedge A(x))) \wedge \exists x(E \wedge A(x)) \\ D' \Leftarrow \forall x(E' \vee (E \wedge A'(x))) \\ E' \end{cases}$$

qui se simplifie en :

$$\begin{cases} A(x) \Leftarrow x \neq a \vee A(a) \vee A'(a) \\ B \Leftarrow \forall x(C'(x) \vee (C(x) \wedge (A'(x) \vee A(x)))) \wedge \exists x(C(x) \wedge A(x)) \\ B' \Leftarrow \forall x(C'(x) \vee (C(x) \wedge A'(x))) \\ \begin{cases} C(x) \Leftarrow x = b \\ C'(x) \Leftarrow x \neq b \end{cases} \\ D \Leftarrow \forall x(E' \vee (E \wedge (A'(x) \vee A(x)))) \wedge \exists x(E \wedge A(x)) \\ D' \Leftarrow \forall x(E' \vee (E \wedge A'(x))) \\ E' \end{cases}$$

Notation : On note T_{fst} l'opérateur de conséquence immédiate associé à P_{fst} défini sur les interprétations de Herbrand de L_d . On omet la référence à P dans cette notation, le contexte permettant de savoir de quel programme il s'agit.

Exemple IV.2 (suite) :

$$T_{fst} \uparrow 1 = \{A(b), C(b), C'(a), E'\}$$

$$T_{fst} \uparrow 2 = T_{fst} \uparrow 1 \cup \{B, D'\} = \text{plus petit point fixe de } T_{fst} = SS_{fst} \cup FF_{st}'$$

Proposition IV.1 : Soit P un programme Prolog.

a) P_{fst} est un programme général positif

b) T_{fst} est monotone

c) Pour tout ordinal α et tout atome clos A, les atomes A et A' ne peuvent pas appartenir simultanément à $T_{fst} \uparrow \alpha$.

Démonstration :

a) Immédiat d'après la définition I.24.

b) En effet, P_{fst} est un programme général positif. On peut appliquer la proposition I.8.

c) Supposons cette propriété fausse. Soit α le premier ordinal tel qu'il existe un atome clos $A = p(s)$ tq A et A' appartiennent à $T_{fst} \uparrow \alpha$. Soient $C_1, \dots, C_q$ les règles de P de prédicat de tête p .

- α n'est pas un ordinal limite sinon, il existerait un ordinal β tel que A et A' appartiennent à $T_{fst} \uparrow \beta$.

- α est un ordinal successeur : $\alpha = \beta + 1$.

Les formules $(C_1^{EF} \wedge \dots \wedge C_q^{EF})(s)$ et $(C_1^S \vee \dots \vee C_q^S)(s)$ sont vraies dans $T_{fst} \uparrow \beta$.

Soit $i \in \{1, \dots, q\}$ tel que $C_i^S(s)$ soit vraie dans $T_{fst} \uparrow \beta$.

D'après les définitions de C_i^S et C_i^{EF} , si L_i est le corps de la règle C_i , il existe une substitution close σ telle que $[L_i\sigma]_{fst}^{EF}$ et $[L_i\sigma]_{fst}^{SF}$ soient vrais dans $T_{fst} \uparrow \beta$.

On en déduit qu'il existe un atome B de L_i tel que B et B' appartiennent simultanément à $T_{fst} \uparrow \beta$, ce qui est contradictoire.

□

Lemme IV.2 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P , soit s un n -uplet de termes et soit X une interprétation de Herbrand de L_d .

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$X \models \forall C^{EF}(s), \quad X \models \forall C^F(s) \quad \text{et} \quad X \models \neg \exists C^S(s)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$X \models \forall C^{EF}(s) \Leftrightarrow \forall [L\theta]_{fst}^{EF},$$

$$X \models \forall C^F(s) \Leftrightarrow \forall [L\theta]_{fst}^F \quad \text{et}$$

$$X \models \exists C^S(s) \Leftrightarrow \exists [L\theta]_{fst}^{SF}.$$

Démonstration : Il suffit d'appliquer le lemme II.1 en prenant pour formule F successivement : $[L]_{fst}^{EF}$, $[L]_{fst}^F$ et $[L]_{fst}^{SF}$ qui sont des formules sans quantificateur. Il montre que les différentes formules citées dans le lemme sont des conséquences logiques de la $CET(L_d)$ et sont donc en particulier vraies dans le modèle de la $CET(L_d)$ associé à X (prop. I.4).

□

IV.1.2 $T_{fst} \uparrow \omega$ -correction et P_{fst}^{Her} -correction

Nous commençons par montrer que la sémantique opérationnelle standard finie est $T_{fst} \uparrow \omega$ -correcte, c.-à-d. : $SS_{fst} \cup FF'_{st} \subseteq T_{fst} \uparrow \omega$.

Cela signifie que pour tout programme Prolog P et tout atome clos A , si $P \cup \{:-A\}$ donne lieu à un calcul fini pour un interpréteur standard alors A ou A' est calculé par "chaînage avant" à partir de P_{fst} .

Définition IV.1 : Soit P un programme Prolog. La **profondeur standard finie** d'un but G pour le programme P est la profondeur de l'arbre SLD standard de $P \cup \{G\}$ (c'est-à-dire la longueur de sa branche la plus longue) si celui-ci est fini.

Notation : Soit P un programme Prolog. On note SS_{fst}^n et FF_{fst}^n respectivement les succès finis et les échecs finis standards de profondeur standard finie au plus égale à n .

Remarque : Nous commettons un abus de notation en notant de la même manière les succès et échecs finis de profondeur *standard* et *standard finie*. Ces notions sont pourtant différentes, mais le contexte permet chaque fois de savoir desquelles il s'agit.

Lemme IV.3 : Soit P un programme Prolog et n un entier.

Supposons que : $SS_{fst}^n \cup (FF_{fst}^n)' \subset T_{fst} \uparrow n$.

Pour tout but G :- L de profondeur standard finie pour P inférieure ou égale à n :

- a) Si G réussit finiment, alors $\exists [L]_{fst}^{SF}$ est vrai dans $T_{fst} \uparrow n$.
- b) Si G échoue finiment, alors $\forall [L]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$.
- c) $\forall [L]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$

Démonstration : Soit $L = A_1, A_2, \dots, A_q$. La démonstration se fait par récurrence sur q .

- C'est évident si $q = 0$.

- Supposons que le résultat soit vrai pour $q-1$ ($q \geq 1$), et soit G le but G :- L .

a) Il existe une instance close $G\sigma$ de G telle que $G\sigma$ réussisse finiment. La profondeur standard finie de $G\sigma$ pour P est inférieure ou égale à n . Comme $A_1\sigma \in SS_{fst}^n$, par hypothèse $A_1\sigma \in T_{fst} \uparrow n$ et puisque G réussit et a une profondeur standard finie pour P inférieure ou égale à n , on peut déduire de l'hypothèse de récurrence que $[A_2\sigma, \dots, A_q\sigma]_{fst}^{SF}$ est vrai dans $T_{fst} \uparrow n$. Par définition, $[L\sigma]_{fst}^{SF}$ et aussi par conséquent $\exists [L]_{fst}^{SF}$ sont vrais dans $T_{fst} \uparrow n$.

b) Toute instance close $G\sigma$ de G échoue et a une profondeur standard finie pour P inférieure ou égale à n .

- Si $A_1\sigma$ échoue, alors $A_1'\sigma \in T_{fst} \uparrow n$ et par définition, $[L\sigma]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$.

- Si $A_1\sigma$ réussit, alors $A_1\sigma \in T_{fst} \uparrow n$.

Le but G :- $(A_2, \dots, A_q)\sigma$ échoue et $[A_2\sigma, \dots, A_q\sigma]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$ d'après l'hypothèse de récurrence. $A_1\sigma \wedge [A_2\sigma, \dots, A_q\sigma]_{fst}^{EF}$ et par conséquent $[L\sigma]_{fst}^{EF}$ sont vrais dans $T_{fst} \uparrow n$.

Ceci montre que $\forall [L]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$.

c) Toute instance close $G\sigma$ de G échoue ou réussit et a une profondeur standard finie pour P inférieure ou égale à n .

- Si $G\sigma$ échoue, $[L\sigma]_{fst}^{EF}$ et donc $[L\sigma]_{fst}^F$ sont vrais dans $T_{fst} \uparrow n$ (voir le lemme IV.1).

- Si $G\sigma$ réussit alors, $A_1\sigma \in T_{fst} \uparrow n$ et, d'après l'hypothèse de récurrence, $[A_2\sigma, \dots, A_q\sigma]_{fst}^F$ et par conséquent $[L\sigma]_{fst}^F$ sont vrais dans $T_{fst} \uparrow n$.
On en déduit que $\forall [L]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$.

□

Lemme IV.4 : Soit P un programme Prolog. Pour tout entier n ,

$$SS_{fst}^n \cup (FF_{st}^n)' \subseteq T_{fst} \uparrow n \quad (1)$$

Démonstration : (par récurrence sur n)

C'est évident si $n = 0$ puisque tous les ensembles ci-dessus sont vides.

Démontrons par exemple que $SS_{fst}^{n+1} \subseteq T_{fst} \uparrow (n+1)$ si la relation (1) est supposée vraie pour n .

Soit $p(t) \in SS_{fst}^{n+1}$ et soit $C \equiv p(s) :- L$ une règle de P .

- Si t et s ne sont pas unifiables, alors $C^F(t)$ est vrai dans $T_{fst} \uparrow n$ d'après le lemme IV.2.

- Si t et s sont unifiables, et si $\sigma = \text{upg}(t, s)$, alors

. le but $:- L\sigma$ a une profondeur standard finie pour P inférieure ou égale à n et donc, d'après le lemme IV.3, $\forall [L\sigma]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$. La formule $C^F(t)$ est par conséquent vraie dans $T_{fst} \uparrow n$ (lemme IV.2).

. si le but $:- L\sigma$ réussit finiment (et ceci est le cas pour au moins une règle de P), alors $\exists [L\sigma]_{fst}^{SF}$ est vrai dans $T_{fst} \uparrow n$ et $C^S(t)$ est vrai dans $T_{fst} \uparrow n$.

En conclusion, si $C_1, C_2, \dots, C_p$ sont les règles P de prédicat de tête p , la formule :

$$((C_1^F \wedge \dots \wedge C_p^F) \wedge (C_1^S \vee \dots \vee C_p^S))(t)$$

est vraie dans $T_{fst} \uparrow n$ et $p(t) \in T_{fst} \uparrow (n+1)$.

On démontre de manière analogue que $(FF_{st}^{n+1})' \subseteq T_{fst} \uparrow (n+1)$ si la relation (1) est supposée vraie pour n .

Théorème IV.1 : Soit P un programme Prolog. La sémantique opérationnelle standard finie est $T_{fst} \uparrow \omega$ -correcte pour la traduction axiomatique associée.

$$SS_{fst} \cup FF_{st}' \subseteq T_{fst} \uparrow \omega.$$

Démonstration : En effet, si A est un atome clos appartenant à SS_{fst} (resp. FF_{st}), il existe un entier n tel que A appartienne à SS_{fst}^n (resp. FF_{st}^n) et l'on peut appliquer le lemme précédent.

□

Corollaire IV.1 : Soit P un programme Prolog et n un entier.

Pour tout but $:- L$ de profondeur standard finie pour P inférieure ou égale à n :

a) Si $:- L$ réussit finiment, alors $\exists [L]_{fst}^{SF}$ est vrai dans $T_{fst} \uparrow n$.

b) Si $:- L$ échoue finiment, alors $\forall [L]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$.

c) $\forall [L]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$.

Démonstration :

C'est l'énoncé du lemme IV.3 sans l'hypothèse : $SS_{fst}^n \cup (FF_{st}^n)' \subseteq T_{fst} \uparrow n$ qui a été démontrée au lemme IV.4.

□

On déduit du Théorème IV.1 un résultat de **correction** de la sémantique opérationnelle standard finie pour l'axiomatique P_{fst}^{Her} .

Théorème IV.2 : Soit P un programme Prolog et $G = :- L$ un but dont l'arbre SLD standard est fini.

- a) Si G réussit avec la réponse calculée σ , alors $P_{fst}^{Her} \models \forall [L\sigma]_{fst}^{SF}$.
- b) Si G échoue finiment alors, $P_{fst}^{Her} \models \forall [L]_{fst}^{EF}$.
- c) Dans tous les cas, $P_{fst}^{Her} \models \forall [L]_{fst}^F$.

Démonstration :

a) Si G réussit avec la réponse calculée σ alors, pour toute substitution close τ , le but $:- L\sigma\tau$ réussit. Le corollaire IV.1 montre alors que $[L\sigma\tau]_{fst}^{SF}$ est vrai dans $T_{fst} \uparrow \omega$ donc dans le plus petit point fixe de T_{fst} (qui est égal au plus petit modèle de Herbrand de P_{fst}) puisque c'est une formule positive et que T_{fst} est monotone. Par conséquent, $[L\sigma\tau]_{fst}^{SF}$ est conséquence logique de P_{fst}^{Her} .

Ceci étant vrai pour toute substitution close τ , on a bien : $P_{fst}^{Her} \models \forall [L\sigma]_{fst}^{SF}$.

b) Si G échoue finiment alors, le corollaire IV.1 montre que $\forall [L]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow \omega$, donc aussi dans le plus petit modèle de Herbrand de P_{fst} donc $\forall [L]_{fst}^{EF}$ est conséquence logique de P_{fst}^{Her} .

Démonstration analogue pour le c).

□

Corollaire IV.2 : Soit P un programme Prolog et A un atome clos.

- a) Si $A \in SS_{fst}$ alors $P_{fst}^{Her} \models A$.
- b) Si $A' \in FF_{fst}'$ alors $P_{fst}^{Her} \models A'$.

Démonstration : En effet, $\forall [A]_{fst}^{SF} = A$ et $\forall [A]_{fst}^{EF} = A'$.

□

IV.1.3 $T_{fst} \uparrow \omega$ -complétude et P_{fst}^{Her} -complétude

Nous commençons par montrer que la sémantique opérationnelle standard finie est $T_{fst} \uparrow \omega$ -complète, c.-à-d. : $T_{fst} \uparrow \omega \subseteq SS_{fst} \cup FF_{st}'$.

Cela signifie que pour tout programme Prolog P et tout atome clos A , si A ou A' est calculé par "chaînage avant" à partir de P_{fst} alors $P \cup \{:-A\}$ donne lieu à un calcul fini pour un interpréteur standard.

Lemme IV.5 : *Soit P un programme Prolog. Supposons que pour tout entier n , il existe un entier d_n tel que pour tout atome clos A : $[A \in T_{fst} \uparrow n \text{ ou } A' \in T_{fst} \uparrow n]$ implique que la profondeur standard finie de $:-A$ pour P est inférieure ou égale à d_n .*

a) *Pour toute suite finie d'atome clos L de longueur q , si $[L]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$ alors la profondeur standard finie du but $:-L$ pour P est inférieure ou égale à qd_n .*

b) *Supposons qu'il existe un entier N tel que les profondeurs standards finies de toute instance close du but $:-L$ soient inférieures ou égales à N .*

Alors, la profondeur standard finie du but $:-L$ pour P est inférieure ou égale à N .

c) *Soit A un atome clos. Si pour toute règle $B :- L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$, alors la profondeur standard finie du but $:-A$ pour P est inférieure ou égale à $Mpd_n + 1$ où Mp est le nombre maximal d'atomes figurant dans le corps d'une règle de P .*

Démonstration : Soit $L = A_1, A_2, \dots, A_q$.

a) Démontrons ce résultat par récurrence sur q .

Le résultat est évident si $q = 1$. Soit $q \geq 2$.

Si $[A_1, A_2, \dots, A_q]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$ alors $A_1 \in T_{fst} \uparrow n$ ou $A_1' \in T_{fst} \uparrow n$ et donc, la profondeur standard finie du but $:-A_1$ est inférieure ou égale d_n .

Si $:-A_1$ échoue, alors la profondeur standard finie de $:-A_1$ est égale à la profondeur standard finie de $:-A_1, A_2, \dots, A_q$ qui est inférieure ou égale d_n .

Si $:-A_1$ réussit, alors $A_1 \in T_{fst} \uparrow n$ (d'après le lemme IV.4), $A_1' \notin T_{fst} \uparrow n$ (d'après le lemme IV.1) et donc $[A_2, \dots, A_q]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$ (d'après la définition de $[L]_{fst}^F$), et la profondeur standard finie de $:-A_2, \dots, A_q$ est inférieure ou égale à $(q-1)d_n$ d'après l'hypothèse de récurrence.

En conclusion, la profondeur standard finie de $:-A_1, A_2, \dots, A_q$ est inférieure ou égale qd_n .

b) Supposons que la conclusion soit fausse. Il suffit d'instancier une branche de longueur strictement supérieure à N pour obtenir une contradiction.

c) Soit A un atome clos tel que pour toute règle $B :- L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_{fst}^F$ soit vrai dans $T_{fst} \uparrow n$. Les lemmes précédents montrent que la profondeur standard finie de $:-L\sigma$ est inférieure ou égale à qd_n où q est le

nombre d'atomes de la liste L. Comme $q \leq M_P$, la profondeur standard finie de $\vdash A$ est inférieure ou égale $M_{Pd_n} + 1$.

□

Théorème IV.3 : *Pour tout programme Prolog P, la sémantique opérationnelle standard finie est $T_{fst} \uparrow \omega$ -complète pour la traduction axiomatique associée :*

$$SS_{fst} \cup FF'_{st} = T_{fst} \uparrow \omega.$$

Démonstration : Soit $(d_n)_{n \in \omega}$ la suite définie¹ par la relation $d_{n+1} = M_{Pd_n} + 1$ où M_P est le nombre maximum d'atomes figurant dans le corps des règles de P et $d_1 = 1$.

Nous montrons que si $[A \in T_{fst} \uparrow n$ ou $A' \in T_{fst} \uparrow n]$ alors la profondeur standard finie de $\vdash A$ pour P est inférieure ou égale à d_n .

Cela démontrera que $T_{fst} \uparrow \omega$ est contenu dans $SS_{fst} \cup FF'_{st}$.

Soit $n = 1$.

Les seules règles dont les têtes peuvent s'unifier avec A ont un corps vide. Le résultat est donc vrai dans ce cas.

Supposons que la proposition soit vraie pour un entier $n \geq 1$.

Soit A un atome clos tel que $A \in T_{fst} \uparrow (n+1)$ ou $A' \in T_{fst} \uparrow (n+1)$. Pour toute règle B $\vdash L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_{fst}^F$ ou $\forall [L\sigma]_{fst}^{EF}$ est vrai dans $T_{fst} \uparrow n$. Le lemme IV.1 montre alors que $\forall [L\sigma]_{fst}^F$ est vrai dans $T_{fst} \uparrow n$ et le lemme IV.5 montre que la profondeur standard finie de $\vdash A$ est inférieure ou égale à $M_{Pd_n} + 1 = d_{n+1}$ d'après le lemme précédent.

□

Le Théorème IV.3 montre donc qu'en un certain sens, la traduction axiomatique P_{fst} est une caractérisation exacte de la *sémantique standard finie* d'un programme Prolog. Mais cette caractérisation n'est pas encore complète du point de vue de la logique (dans le cadre des modèles de Herbrand) puisqu'en général, $T_{fst} \uparrow \omega$ n'est pas le plus petit modèle de Herbrand de P_{fst} .

On peut néanmoins énoncer le résultat de **complétude** suivant :

Théorème IV.4 : *Si P est un programme pour lequel $T_{fst} \uparrow \omega$ est le plus petit point fixe de T_{fst} , alors pour tout atome A clos, on a :*

- a) $A \in SS_{fst} \quad ssi \quad P_{fst}^{Her} \models A.$
- b) $A' \in FF'_{st} \quad ssi \quad P_{fst}^{Her} \models A'.$

¹Voir aussi la définition III.4

Démonstration :

- a) Si $P_{fst}^{Her} \models A$, alors $A \in T_{fst} \uparrow \omega$ puisque $T_{fst} \uparrow \omega$ est le plus petit modèle de Herbrand de P_{fst} et donc $A \in SS_{fst}$.
- b) De même, si $P_{fst}^{Her} \models A'$, alors $A' \in T_{fst} \uparrow \omega$ et donc $A' \in FF'_{st}$.

□

Corollaire IV.3 : *Si P est un programme sans variable locale (et en particulier, propositionnel pur) ou sans symbole de fonction,*

- a) $A \in SS_{fst}$ ssi $P_{fst}^{Her} \models A$.
- b) $A' \in FF'_{st}$ ssi $P_{fst}^{Her} \models A'$.

Démonstration :

On montre au paragraphe IV.3 que l'opérateur T_{fst} associé à un programme sans variable locale est continu et donc, tel que $T_{fst} \uparrow \omega$ est le plus petit point fixe de T_{fst} .

La base de Herbrand d'un programme sans symbole de fonction est finie donc, comme T_{fst} est monotone, $T_{fst} \uparrow \omega$ est le plus petit point fixe de T_{fst} .

Pour ces programmes, P_{fst}^{Her} est donc une **caractérisation axiomatique correcte et complète** de leur sémantique opérationnelle standard finie.

□

Remarque :

Comme au chapitre précédent, l'exemple :

$$P : \quad p(s(x)) :- p(x)$$

montre que $P_{fst}^{Her} \models \forall [L]_{fst}^{EF}$ (resp. $P_{fst}^{Her} \models \forall [L]_{fst}^F$) n'implique pas que le but $:- L$ ait un arbre SLD standard d'échecs finis (resp. fini). En effet, pour tout terme clos t , le but $:- p(t)$ échoue finiment et par contre, le but $:- p(x)$ a un arbre SLD standard infini.

On peut énoncer toutefois la proposition suivante qui servira par la suite :

Proposition IV.2 : *Soit P un programme Prolog et L une liste d'atomes.*

- a) *S'il existe un entier n pour lequel $T_{fst} \uparrow n \models \forall [L]_{fst}^F$ alors, l'arbre SLD standard pour $P \cup \{:- L\}$ est fini de profondeur inférieure ou égale à qd_n où q est le nombre d'atomes de la liste L.*
- b) *S'il existe un entier n pour lequel $T_{fst} \uparrow n \models \forall [L]_{fst}^{EF}$ alors, l'arbre SLD standard pour $P \cup \{:- L\}$ est un arbre d'échecs finis de profondeur inférieure ou égale à qd_n .*
- c) *Soit σ une substitution des variables de L. Alors, si L contient assez de constantes n'apparaissant pas ni dans P ni dans $L\sigma$ pour instancier les variables libres de $L\sigma$ (une constante différente par variable) et si $P_{fst}^{Her} \models \forall [L\sigma]_{fst}^{SF} \wedge \forall [L]_{fst}^F$ alors le but $:- L$ a un arbre SLD standard fini et une réponse calculée θ telle que $\theta \leq \sigma$.*

Démonstration :

- a) Pour toute substitution close τ des variables de L, on a : $T_{fst} \uparrow n \models [L\tau]_{fst}^F$. Le théorème IV.3 et le lemme IV.5 permettent alors de conclure.

b) Le a) montre que l'arbre SLD standard de $\text{:- } L$ est fini. Si ce n'était pas un arbre d'échecs finis, il existerait une substitution close τ des variables de L telle que le but $\text{:- } L\tau$ réussisse. On aurait alors d'après le théorème IV.2 : $T_{fst} \uparrow n \models [L\tau]_{fst}^{SF}$ ce qui contredit $T_{fst} \uparrow n \models \forall [L]_{fst}^{EF}$ d'après le lemme IV.1.

c) Le a) montre que l'arbre SLD standard de $P \cup \{\text{:- } L\}$ est fini. Le théorème IV.4 montre que pour toute instantiation close $L\sigma$ de L , le but $\text{:- } L\sigma$ réussit. La remarque du § II.4.1 permet alors de conclure.

□

Remarques :

- Le programme $\{p(a)\}$ et un langage L ne possédant que la constante 'a' montre comme dans le chapitre III que l'hypothèse sur le langage est indispensable. En effet, si σ est la substitution identique, $P_{fst}^{Her} \models \forall [p(x)\sigma]_{fst}^{SF} \wedge \forall [p(x)]_{fst}^F$ et le but $\text{:- } p(x)$ n'admet pas de réponse calculée plus générale que σ .

- Le programme $P \begin{cases} A(a) \text{ :- } A(a) \\ A(b) \end{cases}$ montre que dans le c), l'hypothèse $P_{fst}^{Her} \models \forall [p(x)]_{fst}^F$, c'est-à-dire le fait que l'arbre SLD standard de $\text{:- } L$ soit fini, est indispensable.

En résumé, la traduction axiomatique standard finie P_{fst} est plus satisfaisante que la traduction standard en ce sens que $T_{fst} \uparrow \omega$ représente **exactement** la sémantique opérationnelle standard finie de **tout** programme Prolog P : c'est ce que nous avons appelé la $T_{fst} \uparrow \omega$ -correction et la $T_{fst} \uparrow \omega$ -complétude.

Pour les programmes sans variable locale ou sans symbole de fonction¹, le système d'axiomes P_{fst}^{Her} constitue une caractérisation axiomatique exacte de cette sémantique opérationnelle.

Mais dans le cas général, les deux derniers problèmes, mentionnés au paragraphe III.2, subsistent :

- L'opérateur T_{fst} n'est en général pas continu et par conséquent $T_{fst} \uparrow \omega$ n'est pas un point fixe de T_{fst} : il peut exister des conséquences atomiques de P_{fst} qui ne sont pas prises en compte dans la sémantique opérationnelle standard finie.

- La proposition IV.2 énonce des conditions encore plus restrictives dès lors que l'on veut un résultat de complétude prenant en compte un but quelconque (c.-à-d. pouvant contenir des variables).

Nous verrons au chapitre VI que ces difficultés disparaissent lorsque l'on se place dans le cadre logique de la Théorie Equationnelle de Clark.

¹c'est-à-dire pour les programmes utilisés généralement dans les bases de données

IV.2. La sémantique invariante finie

La sémantique opérationnelle standard finie d'un programme Prolog a été introduite afin de décrire les résultats des calculs (finis) effectués par un *interpréteur Prolog standard*. Elle ne dépend pas de l'ordre des règles du programme mais elle dépend, en général, de l'ordre des atomes dans chaque règle.

La sémantique opérationnelle invariante finie (§ II.4.4) est à la fois plus générale et plus homogène. Elle possède les mêmes **propriétés d'invariance** que la sémantique d'un interpréteur idéal :

- par rapport à l'ordre des règles dans le programme
- par rapport à l'ordre des atomes dans les buts et chaque corps des règles du programme.

Elle a également une **propriété "d'universalité"** :

- tout atome clos qu'elle définit donne lieu à **un calcul fini quelque soit interpréteur Prolog utilisé** et réciproquement.

Ces deux propriétés sont bien sûr très liées.

Les propriétés d'invariance laissent espérer qu'une traduction axiomatique de cette sémantique bénéficiera des mêmes propriétés d'homogénéité. La propriété d'universalité montre que cette sémantique décrit un noyau commun à tout interpréteur. Elle paraît bien adaptée en particulier au cadre des bases de données pour lesquelles, les connaissances (faits et règles) doivent être gérées "en vrac" et non suivant un ordre arbitraire précis.

IV.2.1 Définition de la traduction axiomatique invariante finie

Nous donnons dans ce paragraphe, une caractérisation axiomatique P_f de la sémantique invariante finie d'un programme Prolog P , en utilisant une méthode analogue à celle développée au paragraphe précédent. Nous montrons que $T_f \uparrow \omega$ (où T_f est l'opérateur de conséquence immédiate associé à P_f défini sur l'ensemble des interprétations de Herbrand du langage sous-jacent) représente exactement cette sémantique opérationnelle puis nous en tirons les conséquences en termes de correction et de complétude dans le cadre des modèles de Herbrand. Comme pour la sémantique standard finie, nous montrerons au chapitre VI que c'est dans le cadre de la Théorie Equationnelle de Clark que ce résultat a la signification logique la plus forte.

Cette traduction s'exprime dans le langage dédoublé L_d .

Remarque : Soit P un programme Prolog et $:-L$ un but.

Si tous les arbres SLD de $P \cup \{:-L\}$ sont finis alors :

- si l'un des arbres SLD de $P \cup \{:-L\}$ contient la clause vide, tous la contiennent¹ et *tout interpréteur Prolog* trouve une réfutation pour $P \cup \{:-L\}$. Nous dirons simplement dans ce cas que le but $:-L$ réussit.
- si l'un des arbres SLD de $P \cup \{:-L\}$ est un arbre d'échecs finis, tous sont également des arbres d'échecs finis¹ et *tout interpréteur Prolog* donne un échec pour $P \cup \{:-L\}$. Nous dirons simplement dans ce cas que le but $:-L$ échoue.

¹Voir les résultats de Apt et Van Emden au § II.4.1.

Notations : Soit $L = A_1, A_2, \dots, A_n$ une liste d'atomes. On définit :

$$- [L]_f^{SF} = A_1 \wedge A_2 \wedge \dots \wedge A_n \quad \text{et} \quad []_f^{SF} = \text{Vrai.}$$

$$- [L]_f^F = (A'_1 \vee A_1) \wedge \dots \wedge (A'_n \vee A_n) \quad \text{et} \quad []_f^F = \text{Vrai.}$$

$$- [L]_f^{EF} = [L]_f^F \wedge (A'_1 \vee \dots \vee A'_n) \quad \text{et} \quad []_f^{EF} = \text{Faux.}$$

Lemme IV.6 : Soit $X \subset H_d$ (base de Herbrand associée à L_d) et L une liste d'atomes clos :

- $[L]_f^F$ est vrai dans X ssi $([L]_f^{SF} \vee [L]_f^{EF})$ est vrai dans X
- $[L]_f^{EF}$ est vrai dans X implique que $[L]_{fst}^{EF}$ est vrai dans X
- $[L]_f^F$ est vrai dans X implique que $[L]_{fst}^F$ est vrai dans X .

Démonstration : Immédiate par récurrence sur la longueur de L . □

Comme dans le paragraphe précédent, nous associons les formules suivantes à chaque règle $C = p(t) :- L$ du programme :

$$\begin{aligned} C^S(x) &\equiv [\exists y(x = t \wedge [L]_f^{SF})] \\ C^{EF}(x) &\equiv [\forall y(x \neq t \vee [L]_f^{EF})] \\ C^F(x) &\equiv [\forall y(x \neq t \vee [L]_f^F)] \end{aligned}$$

dans lesquelles les y_i sont les variables apparaissant dans les termes t_i et les atomes de L et où les x_i sont de nouvelles variables.

Remarque : Nous utilisons à nouveau les mêmes notations que pour les sémantiques précédentes pour ne pas avoir à rajouter un indice supplémentaire.

- Si $C_1, C_2, \dots, C_p$ sont les règles de P dont le symbole de prédicat de tête est p , on considère les axiomes suivants (que l'on suppose quantifiés universellement) :

$$\begin{aligned} p(x) &\Leftarrow ((C_1^F \wedge \dots \wedge C_p^F) \wedge (C_1^S \vee \dots \vee C_p^S))(x) \\ p'(x) &\Leftarrow (C_1^{EF} \wedge \dots \wedge C_p^{EF})(x) \end{aligned}$$

- Si aucune règle de P n'admet le symbole p comme prédicat de tête, on considère l'axiome :

$$\forall x \, p'(x).$$

Notation : P_f est la réunion de ces axiomes pour tout symbole de prédicat p du programme.

Exemple IV.3 : Soit P le programme Prolog suivant :

$$\begin{cases} A(a) :- A(a) \\ A(x) \\ B :- C(x), A(x) \\ C(b) \\ D :- E, A(x) \end{cases}$$

La traduction invariante finie de P est P_f :

$$\begin{cases} A(x) \Leftarrow (x \neq a \vee A(a) \vee A'(a)) \wedge \text{Vrai} \wedge ((x = a \wedge A(a)) \vee \text{Vrai}) \\ A'(x) \Leftarrow (x \neq a \vee A'(a)) \wedge \text{Faux} \\ B \Leftarrow \forall x((C(x) \vee C'(x)) \wedge (A(x) \vee A'(x))) \wedge \exists x(C(x) \wedge A(x)) \\ B' \Leftarrow \forall x((C(x) \vee C'(x)) \wedge (A(x) \vee A'(x))) \wedge \exists x(C'(x) \wedge A'(x)) \\ C(x) \Leftarrow x = b \\ \begin{cases} C'(x) \Leftarrow x \neq b \\ D \Leftarrow \forall x((E \vee E') \wedge (A(x) \vee A'(x))) \wedge \exists x(E \wedge A(x)) \\ D' \Leftarrow \forall x((E \vee E') \wedge (A(x) \vee A'(x))) \wedge (E' \vee A'(x)) \\ E' \end{cases} \end{cases}$$

Le premier axiome peut être simplifié en $A(x) \Leftarrow x \neq a \vee A(a) \vee A'(a)$ et le deuxième peut être supprimé.

Notation : On note T_f l'opérateur de conséquence immédiate défini sur l'ensemble des interprétations de Herbrand du langage L_d associé à la traduction P_f .

Exemple IV.3 (suite) :

$$T_f \uparrow 1 = \{A(b), C(b), C'(a), E'\} = \text{plus petit point fixe de } T_f = SS_f \cup FF_f$$

Proposition IV.3 : Soit P un programme Prolog.

- P_f est un programme général positif.
- T_f est monotone.
- Pour tout ordinal α et tout atome clos A , les atomes A et A' ne peuvent pas appartenir simultanément à $T_f \uparrow \alpha$.

Démonstration :

- Immédiat d'après la définition I.24.
- En effet, P_f est un programme général positif et l'on peut appliquer la proposition I.8.
- Supposons cette propriété fautive.

Soit α le premier ordinal tel qu'il existe un atome clos $A = p(s)$ tq A et A' appartiennent à $T_f \uparrow \alpha$.

– α n'est pas un ordinal limite sinon, il existerait un ordinal β tel que A et A' appartiennent à $T_f \uparrow \beta$.

– α est un ordinal successeur : $\alpha = \beta + 1$.

Soient $C_1, \dots, C_p$ les règles du programme de symbole de prédicat de tête p .

Les formules $[C_1^{EF} \wedge \dots \wedge C_p^{EF}](s)$ et $[C_1^S \vee \dots \vee C_p^S](s)$ sont vraies dans $T_f \uparrow \beta$.

Soit $i \in \{1, \dots, p\}$ tel que $C_i^S(s)$ soit vrai dans $T_f \uparrow \beta$.

D'après les définitions de C_i^S et C_i^{EF} , si $L = A_1, A_2, \dots, A_q$ est le corps de la règle C_i , il existe une substitution close σ telle que $[L\sigma]_f^{EF}$ et $[L\sigma]_f^{SF}$ soient vrais dans $T_f \uparrow \beta$.

Comme par définition de α , pour tout entier $j \in \{1, \dots, q\}$, $A_j\sigma$ et $A'_j\sigma$ n'appartiennent pas simultanément à $T_f \uparrow \beta$, on obtient donc une contradiction.

□

Nous montrons un lemme analogue au lemme IV.2.

Lemme IV.7 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P , soit s un n -uplet de termes et soit X une interprétation de Herbrand de L_d .

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$X \models \forall C^{EF}(s), \quad X \models \forall C^F(s) \quad \text{et} \quad X \models \neg \exists C^S(s)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$X \models \forall C^{EF}(s) \Leftrightarrow \forall [L\theta]_f^{EF},$$

$$X \models \forall C^F(s) \Leftrightarrow \forall [L\theta]_f^F \text{ et}$$

$$X \models \exists C^S(s) \Leftrightarrow \exists [L\theta]_f^{SF}.$$

Démonstration : Il suffit d'appliquer le lemme II.1 en prenant pour formule F successivement : $[L]_f^{EF}$, $[L]_f^F$ et $[L]_f^{SF}$ qui sont des formules sans quantificateur. Il montre que les différentes formules citées dans le lemme sont des conséquences logiques de la $CET(L_d)$ et sont donc en particulier vraies dans le modèle de la $CET(L_d)$ associé à X (prop. I.4).

□

IV.2.2 $T_f \uparrow \omega$ -correction et P_f^{Her} -correction.

Comme pour la sémantique précédente, nous montrons que la sémantique opérationnelle invariante finie est $T_f \uparrow \omega$ -correcte, c.-à-d. : $SS_f \cup FF_f \subseteq T_f \uparrow \omega$.

Cela signifie que pour tout programme Prolog P et tout atome clos A , si $P \cup \{:-A\}$ donne lieu à un calcul fini pour *tout* interpréteur alors A ou A' est calculé par "chaînage avant" à partir de P_f .

Définition IV.2 : La **profondeur maximale** d'un but G pour un programme Prolog P est la borne supérieure de l'ensemble des profondeurs des arbres SLD de $P \cup \{G\}$.

Remarque : D'après la définition précédente, il semble possible qu'un but dont tous les arbres SLD sont finis ait une profondeur maximale infinie. La proposition suivante montre qu'en fait, il n'en est rien.

Proposition IV.4 : *Si la profondeur maximale d'un but G pour un programme P est infinie alors $P \cup \{G\}$ a un arbre SLD infini.*

Démonstration : En effet, si pour tout atome A de G , la profondeur maximale des buts dérivés de G en sélectionnant A était finie, alors G serait également de profondeur maximale finie. Il existe donc au moins un atome A de G tel que la profondeur maximale d'un des buts dérivés de G en sélectionnant A soit infinie. Ceci montre par induction que G a un arbre SLD infini. □

Notation : Soit P un programme Prolog. On note SS_f^n et FF_f^n respectivement les succès finis et les échecs finis de profondeur maximale au plus égale à n .

Lemme IV.8 : *Soit P un programme Prolog et n un entier.*

Supposons que $SS_f^n \cup (FF_f^n)' \subset T_f \uparrow n$.

Pour tout but L de profondeur maximale pour P inférieure ou égale à n :

- a) $\forall [L]_f^F$ est vrai dans $T_f \uparrow n$.
- b) Si L réussit, alors $\exists [L]_f^{SF}$ est vrai dans $T_f \uparrow n$.
- c) Si L échoue finiment, alors $\forall [L]_f^{EF}$ est vrai dans $T_f \uparrow n$.

Démonstration : Soit $L = A_1, A_2, \dots, A_q$ et $G = :- L$.

a) Pour toute instance close $G\sigma$ de G et tout $i \in \{1, \dots, q\}$, le but $A_i\sigma$ a également une profondeur maximale pour P inférieure ou égale à n . Comme par hypothèse, un des deux atomes $A_i\sigma$ ou $A_i'\sigma$ appartient à $T_f \uparrow n$ (puisque $A_i\sigma \in SS_f^n$ ou $A_i\sigma \in FF_f^n$), la formule:

$$(A_1\sigma \vee A_1'\sigma) \wedge \dots \wedge (A_q\sigma \vee A_q'\sigma)$$

est vraie dans $T_f \uparrow n$.

Donc, $\forall [L]_f^F$ est vrai dans $T_f \uparrow n$.

b) Il existe une instance close $G\sigma$ de G telle que $G\sigma$ réussisse quelle que soit la règle de sélection utilisée. La profondeur maximale de $G\sigma$ est inférieure ou égale à n donc, pour $i \in \{1, \dots, q\}$, $A_i\sigma \in SS_f^n$. Par hypothèse, $A_i\sigma \in T_f \uparrow n$ et donc, $\exists [L]_f^{SF}$ est vrai dans $T_f \uparrow n$.

c) Toute instance close $G\sigma$ de G échoue finiment quelle que soit la règle de sélection utilisée et a une profondeur maximale inférieure ou égale à n . Pour toute instance close $G\sigma$ de G , $[L\sigma]_f^F$ est vrai dans $T_f \uparrow n$ et il existe $i \in \{1, \dots, q\}$ tel que $A_i\sigma$ échoue finiment quelle que soit la règle de sélection utilisée, c.-à-d. appartienne à FF_f^n et donc à $T_f \uparrow n$. Par conséquent, $\forall [L]_f^{EF}$ est vrai dans $T_f \uparrow n$. □

Lemme IV.9 : Soit P un programme Prolog. Pour tout entier n ,

$$SS_f^n \cup (FF_f^n)' \subseteq T_f \uparrow n \quad (1)$$

Démonstration : (par récurrence sur n)

C'est évident si $n = 0$ puisque tout est vide.

Démontrons par exemple que $SS_f^{n+1} \subseteq T_f \uparrow (n+1)$ si la relation (1) est supposée vraie pour n .

Soit $p(t) \in SS_f^{n+1}$ et soit $C \equiv p(s) :- L$ une règle de P .

- Si t et s ne sont pas unifiables, alors $C^F(t)$ est vrai dans $T_f \uparrow n$ d'après le lemme IV.7.

- Si t et s sont unifiables, et si $\sigma = \text{upg}(t, s)$, alors

. le but $:- L\sigma$ a une profondeur maximale pour P inférieure ou égale à n et donc, d'après le lemme précédent, $\forall [L\sigma]_f^F$ est vrai dans $T_f \uparrow n$. La formule $C^F(t)$ est par conséquent vraie dans $T_f \uparrow n$ (lemme IV.7).

. si le but $:- L\sigma$ réussit finiment (et ceci est le cas pour au moins une règle de P quelle que soit la règle de sélection utilisée), alors $\exists [L\sigma]_f^{SF}$ est vrai dans $T_f \uparrow n$ et $C^S(t)$ est vraie dans $T_f \uparrow n$ (lemme IV.7).

En conclusion, si $C_1, C_2, \dots, C_p$ sont les règles P dont le symbole de prédicat de tête est p , la formule :

$$((C_1^F \wedge \dots \wedge C_p^F) \wedge (C_1^S \vee \dots \vee C_p^S))(t)$$

est vraie dans $T_f \uparrow n$ et $p(t) \in T_f \uparrow (n+1)$.

On démontre de manière analogue que $(FF_f^{n+1})' \subseteq T_f \uparrow (n+1)$ si la relation (1) est vraie pour n .

□

Théorème IV.5 : Pour tout programme Prolog P , la sémantique opérationnelle invariante finie est $T_f \uparrow \omega$ -correcte pour la traduction axiomatique associée :

$$SS_f \cup FF_f' \subseteq T_f \uparrow \omega.$$

Démonstration : En effet, si A est un atome clos appartenant à SS_f (resp. FF_f), il existe un entier n tel que A appartienne à SS_f^n (resp. FF_f^n) d'après la proposition IV.4 et l'on peut appliquer le lemme précédent.

□

Corollaire IV.4 : Soit P un programme Prolog et n un entier.

Pour tout but $:- L$ de profondeur maximale pour P inférieure ou égale à n :

a) $\forall [L]_f^F$ est vrai dans $T_f \uparrow n$.

b) Si $:- L$ réussit, alors $\exists [L]_f^{SF}$ est vrai dans $T_f \uparrow n$.

c) Si $:- L$ échoue finiment, alors $\forall [L]_f^{EF}$ est vrai dans $T_f \uparrow n$.

Démonstration : C'est l'énoncé du lemme IV.8 sans l'hypothèse $SS_f^n \cup (FF_f^n)' \subseteq T_f \uparrow n$, démontrée dans le lemme IV.9.

□

On déduit du Théorème IV.5 un résultat de correction de la sémantique opérationnelle standard finie pour l'axiomatique P_f^{Her} .

Théorème IV.6 : Soit P un programme Prolog et $G = :- L$ un but de profondeur maximale finie.

a) Si G réussit avec la réponse calculée σ , alors $P_f^{Her} \models \forall [L\sigma]_f^{SF}$.

b) Si G échoue finiment alors, $P_f^{Her} \models \forall [L]_f^{EF}$.

c) Dans tous les cas, $P_f^{Her} \models \forall [L]_f^F$.

Démonstration :

a) Si G réussit avec la réponse calculée σ (pour une règle de sélection) alors, pour toute substitution close τ , le but $:- L\sigma\tau$ réussit (pour toute règle de sélection).

Le lemme IV.9 montre alors que $[L\sigma\tau]_f^{SF}$ est vrai dans $T_f \uparrow \omega$ donc dans le plus petit point fixe de T_f (qui est égal au plus petit modèle de Herbrand de P_f) puisque c'est une formule positive et que T_f est monotone. Par conséquent, que $[L\sigma\tau]_f^{SF}$ est conséquence logique de P_f^{Her} . Ceci étant vrai pour toute substitution close τ , on a bien : $P_f^{Her} \models \forall [L\sigma]_f^{SF}$.

b) Si G échoue finiment (quelle que soit la règle de sélection utilisée) alors, le lemme IV.9 montre que $\forall [L]_f^{EF}$ est vrai dans $T_f \uparrow \omega$, donc aussi dans le plus petit modèle de Herbrand de P_f donc $\forall [L]_f^{EF}$ est conséquence logique de P_{st}^{Her} .

Démonstration analogue pour le c).

□

Corollaire IV.5 : Soit P un programme Prolog et A un atome clos.

a) Si $A \in SS_f$ alors $P_f^{Her} \models A$.

b) Si $A' \in FF_f'$ alors $P_f^{Her} \models A'$.

Démonstration : en effet, si A (resp. A') $\in SS_f$ (resp. FF_f'), alors A (resp. A') $\in T_f \uparrow \omega$ et donc aussi au plus petit modèle de Herbrand de P_f .

□

IV.2.3 $T_f \uparrow \omega$ -complétude et P_f^{Her} -complétude.

Comme au § IV.1.3, nous montrons que la sémantique opérationnelle invariante finie est $T_f \uparrow \omega$ -complète, c.-à-d. : $T_f \uparrow \omega \subseteq SS_f \cup FF_f$.

Cela signifie que pour tout programme Prolog P et tout atome clos A , si A ou A' est calculé par "chaînage avant" à partir de P_f alors $P \cup \{:-A\}$ donne lieu à un calcul fini pour tout interpréteur Prolog.

Lemme IV.10 : *Soit P un programme Prolog. Supposons que pour tout entier n il existe un entier d_n tel que pour tout atome clos A : $[A \in T_f \uparrow n \text{ ou } A' \in T_f \uparrow n]$ implique que la profondeur maximale de $:-A$ pour P soit inférieure ou égale à d_n . Alors,*

a) *Pour toute suite finie d'atomes clos L , si $[L]_f^F$ est vrai dans $T_f \uparrow n$ alors la profondeur maximale de $:-L$ pour P est inférieure ou égale à qd_n .*

b) *Supposons qu'il existe un entier N tel que les profondeurs maximales de toutes les instances closes du but $:-L$ soient inférieures ou égales à N . Alors, la profondeur maximale de $:-L$ pour P est inférieure ou égale à N .*

c) *Soit A un atome clos. Si pour toute règle $B :- L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_f^F$ est vrai dans $T_f \uparrow n$, alors la profondeur maximale de $:-A$ pour P est inférieure ou égale à $Mpd_n + 1$ où M_p est le nombre maximal d'atomes figurant dans le corps d'une règle de P .*

Démonstration :

a) $[A_1, A_2, \dots, A_q]_f^F$ est vrai dans $T_f \uparrow n$ implique que pour $i \in \{1, \dots, q\}$, $A_i \in T_f \uparrow n$ ou $A_i' \in T_f \uparrow n$ et donc que la profondeur maximale de $:-A_i$ est inférieure ou égale à d_n . Par conséquent, la profondeur maximale de $:-A_1, A_2, \dots, A_q$ est inférieure ou égale à qd_n .

b) Supposons que la conclusion soit fausse. Il suffit d'instancier une branche de longueur strictement supérieure à N pour obtenir une contradiction.

c) Soit A un atome clos tel que pour toute règle $B :- L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_f^F$ soit vrai dans $T_f \uparrow n$. Les points a) et b) montrent que la profondeur maximale de $:-L\sigma$ est inférieure ou égale à qd_n où q est le nombre d'atomes de la liste L . Comme $q \leq M_p$, la profondeur maximale de $:-A$ pour P est inférieure ou égale à $Mpd_n + 1$.

Théorème IV.7 : *Pour tout programme Prolog P , la sémantique opérationnelle invariante finie est $T_f \uparrow \omega$ -complète pour la traduction axiomatique associée :*

$$SS_f \cup FF_f = T_f \uparrow \omega.$$

Démonstration : Nous reprenons la démonstration du théorème IV.3. Nous démontrons plus précisément que si d_n est la suite définie¹ par la relation $d_{n+1} = Mpd_n + 1$ où Mp est le nombre maximum d'atomes figurant dans le corps des règles de P et $d_1 = 1$, alors $[A \in T_f \uparrow n \text{ ou } A' \in T_f \uparrow n]$ implique que la profondeur maximale de $:- A$ pour P est inférieure ou égale à d_n .

Soit $n = 1$.

Les seules règles dont les têtes peuvent s'unifier avec A ont un corps vide. Le résultat est donc vrai dans ce cas.

Supposons que la proposition soit vraie pour un entier $n \geq 1$.

Soit A un atome clos tel que $A \in T_f \uparrow (n+1)$ ou $A' \in T_f \uparrow (n+1)$. Pour toute règle $B :- L$ de P telle que A et B soient unifiables avec σ comme upg, $\forall [L\sigma]_f^F$ ou $\forall [L\sigma]_f^{EF}$ est vrai dans $T_f \uparrow n$. Le lemme IV.6 montre alors que $\forall [L\sigma]_f^F$ est vrai dans $T_{fst} \uparrow n$ et le lemme IV.10 montre que la profondeur maximale de $:- A$ est alors inférieure ou égale à $Mpd_n + 1 = d_{n+1}$.

□

Le Théorème IV.7 montre donc qu'en un certain sens, la traduction axiomatique P_{fst} est une caractérisation exacte de la *sémantique invariante finie* d'un programme Prolog. Mais cette caractérisation n'est pas encore complète du point de vue de la logique (dans le cadre des modèles de Herbrand) puisqu'en général, $T_f \uparrow \omega$ n'est pas le plus petit modèle de Herbrand de P_f .

On peut néanmoins énoncer le résultat de **complétude** suivant :

Théorème IV.8 : Si P est un programme pour lequel $T_f \uparrow \omega$ est le plus petit point fixe de T_f , alors pour tout atome A clos, on a :

- a) $A \in SS_f$ ssi $P_f^{Her} \models A$.
- b) $A' \in FF_f$ ssi $P_f^{Her} \models A'$.

Démonstration :

a) Si $P_f^{Her} \models A$, alors $A \in T_f \uparrow \omega$ puisque $T_f \uparrow \omega$ est le plus petit modèle de Herbrand de P_f et donc $A \in SS_f$.

b) De même, si $P_f^{Her} \models A'$, alors $A' \in T_f \uparrow \omega$ et donc $A' \in FF_f$.

□

Corollaire IV.6 : Si P est un programme sans variable locale (et en particulier, propositionnel pur) ou sans symbole de fonction,

- a) $A \in SS_f$ ssi $P_f^{Her} \models A$.
- b) $A' \in FF_f$ ssi $P_f^{Her} \models A'$.

Démonstration :

On montre au paragraphe IV.3 que l'opérateur T_f associé à un programme sans variable locale est continu et donc, tel que $T_f \uparrow \omega$ est le plus petit point fixe de T_f .

¹Voir aussi la définition III.4.

La base de Herbrand d'un programme sans symbole de fonction est finie donc, comme T_f est monotone, $T_f \uparrow \omega$ est le plus petit point fixe de T_f .

Pour ces programmes, P_f^{Her} est donc une **caractérisation axiomatique correcte et complète** de leur sémantique opérationnelle invariante finie.

□

Remarque :

Comme au paragraphe précédent, l'exemple :

$$P \quad \{p(s(x)) :- p(x)\}$$

montre que $P_f^{\text{Her}} \models \forall [L]_f^{\text{EF}}$ ou $P_f^{\text{Her}} \models \forall [L]_f^F$ n'implique pas que les arbres SLD du but $:- L$ soient finis. En effet, pour tout terme clos t , le but $:-p(t)$ échoue finiment et par contre, le but $:- p(x)$ a un arbre SLD (standard) infini.

On peut énoncer toutefois la proposition suivante qui servira par la suite :

Proposition IV.5 : Soit P un programme Prolog et L une liste d'atomes.

a) S'il existe un entier n pour lequel $T_f \uparrow n \models \forall [L]_f^F$, alors tous les arbres SLD de $:- L$ sont finis de profondeur inférieure ou égale à qd_n où q est le nombre d'atomes de L .

b) S'il existe un entier n pour lequel $T_f \uparrow n \models \forall [L]_f^{\text{EF}}$ alors, tous les arbres SLD de $:- L$ sont des arbres d'échecs finis de profondeur inférieure ou égale à qd_n .

c) Soit σ une substitution des variables de L . Alors, si L contient assez de constantes n'apparaissant ni dans P , ni dans $L\sigma$ pour instancier les variables libres de $L\sigma$ (une constante différente par variable) et si $P_f^{\text{Her}} \models \forall [L\sigma]_f^{\text{SF}} \wedge \forall [L]_f^F$ alors les arbres SLD de $:- L$ sont finis et le but $:- L$ a une réponse calculée θ (quelle que soit la stratégie utilisée) telle que $\theta \leq \sigma$.

Démonstration :

a) Pour toute substitution close τ des variables de L , on a : $T_f \uparrow n \models [L\tau]_f^F$. Le théorème IV.7 et le lemme IV.10 permettent alors de conclure.

b) Le a) montre que les arbres SLD de $:- L$ sont finis. Si un de ces arbre n'était pas un arbre d'échecs finis, il existerait une substitution close τ des variables de L telle que le but $:- L\tau$ réussisse pour une stratégie particulière (et donc pour toutes). On aurait alors d'après le théorème IV.5 : $T_f \uparrow n \models [L\tau]_f^{\text{SF}}$ ce qui contredit $T_f \uparrow n \models \forall [L]_f^{\text{EF}}$ d'après le lemme IV.6.

c) Le a) montre que tous les arbres SLD de $P \cup \{:- L\}$ est fini. Le théorème IV.7 montre que pour toute instanciation close $L\sigma\tau$ de L , le but $:-L\sigma\tau$ réussit. La remarque du § II.4.1 permet alors de conclure.

□

Remarque : Comme au paragraphe précédent :

- le programme $\{p(a)\}$ et un langage L ne possédant que la constante 'a' montre que l'hypothèse sur le langage est indispensable.
- le programme $P \begin{cases} A(a) :- A(a) \\ A(b) \end{cases}$ montre que dans le c), il est indispensable que $P_f^{Her} \models \forall [L]_f^F$, c.-à-d. que le but $:-L$ soit de profondeur maximale finie.

En résumé, nous obtenons des résultats analogues à ceux du paragraphe précédent. La traduction axiomatique invariante finie P_f d'un programme Prolog P est optimale en ce sens que $T_f \uparrow \omega$ représente exactement la sémantique opérationnelle invariante finie de P : c'est ce que nous avons appelé la $T_f \uparrow \omega$ -correction et la $T_f \uparrow \omega$ -complétude

Pour les programmes sans variable locale ou sans symbole de fonction, le système d'axiomes P_f^{Her} constitue une caractérisation exacte de cette sémantique opérationnelle. Bien que limitée, cette classe de programmes recouvre quand même l'ensemble des programmes utilisés habituellement dans les bases de données.

Mais les deux derniers problèmes, mentionnés au paragraphe III.2, subsistent :

- L'opérateur T_f n'est en général pas continu et par conséquent $T_f \uparrow \omega$ n'est pas un point fixe de T_f : il peut exister des conséquences atomiques de P_f qui ne sont pas "calculées".
- La proposition IV.4 énonce des conditions encore plus restrictives dès lors que l'on veut un résultat de complétude prenant en compte un but quelconque (c.-à-d. pouvant contenir des variables).

Encore une fois, nous verrons au chapitre VI que ces difficultés disparaissent dans le cadre logique de la CET.

IV.3. Continuité des opérateurs T_θ , T_{st} , T_{fst} et T_f pour les programmes sans variable locale.

Les opérateurs T_θ , T_{st} , T_{fst} et T_f ne sont pas toujours continus. Ceci est dû à la présence de quantificateurs universels dans les corps des systèmes d'axiomes qui les définissent. En fait, il nous suffirait que les ensembles $T_\theta \uparrow \omega$, $T_{st} \uparrow \omega$, $T_{fst} \uparrow \omega$ et $T_f \uparrow \omega$ soient des points fixes de ces opérateurs et donc, des plus petits modèles de Herbrand des systèmes d'axiomes qui les définissent. Mais cela est également faux en général.

Exemple IV.4 (pour les traductions standard finie et invariante finie) :

Soit P le programme Prolog :

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

Les traductions axiomatiques standard finie P_{fst} et invariante finie P_f de P sont :

$$\begin{cases} P(x) \Leftarrow \exists y(x = s(y) \wedge P(y)) \wedge \forall y(x \neq s(y) \vee P(y) \vee P'(y)) \\ P'(x) \Leftarrow \forall y(x \neq s(y) \vee P'(y)) \\ Q(x) \Leftarrow \exists y(x = a \wedge P(y)) \wedge \forall y(x \neq a \vee P(y) \vee P'(y)) \\ Q'(x) \Leftarrow \forall y(x \neq a \vee P'(y)) \end{cases}$$

$$T_{fst} \uparrow 1 = T_f \uparrow 1 = \{P'(a), Q'(s^n(a))\}_{n \geq 1}$$

$$T_{fst} \uparrow m = T_f \uparrow m = \{P'(s^n(a)), Q'(s^n(a))\}_{n \leq m} \text{ pour tout entier } m$$



$$T_{fst} \uparrow \omega = T_f \uparrow \omega = \{P'(s^n(a)), Q'(s^n(a))\}_{n \geq 0}$$

$$T_{fst} \uparrow (\omega+1) = T_f \uparrow (\omega+1) = \{P'(s^n(a)), Q'(s^n(a))\}_{n \geq 0} \neq T_{fst} \uparrow \omega.$$

Par contre, si P est un programme Prolog dont les règles n'ont pas de variable locale¹, les opérateurs T_θ , T_{st} , T_{fst} et T_f sont continus.

Définition IV.3 : On dit que la formule F d'un langage $\mathcal{L}$ vérifie la propriété C si :
- pour toute partie dirigée X de 2^H (ensemble des interprétations de Herbrand de $\mathcal{L}$),
si F est vraie dans $\bigcup_{I \in X} I$ alors il existe $J \in X$ telle que F soit vraie dans J .

Lemme IV.11 : Soit F une formule sans quantificateur. Alors la clôture existentielle de F vérifie la propriété C .

Démonstration : Soit X une partie dirigée de 2^H telle que $\bigcup_{I \in X} I \models \exists F$ et soit $(x\forall)$ une instantiation des variables libres de F par des termes clos telle que : $\bigcup_{I \in X} I \models F(x\forall)$.

Il existe dans la forme normale disjonctive de $F(x\forall)$ une conjonction de littéraux :

$L_1 \wedge \dots \wedge L_m$ vraie dans $\bigcup_{I \in X} I$.

Soient $\{A_1, \dots, A_n\}$ l'ensemble des littéraux positifs de $\{L_1, \dots, L_m\}$.

Pour tout entier $i \in \{1, \dots, n\}$, il existe un élément I_i de X tel que $I_i \models A_i$.

Soit $J = \bigcup_{i=1..n} I_i$. Comme X est dirigée, $J \in X$.

Pour $j \in \{1, \dots, m\}$,

- si L_j est négatif, $J \models L_j$ puisque $J \subset \bigcup_{I \in X} I$ et que $\bigcup_{I \in X} I \models L_j$.

- si L_j est positif, alors $J \models L_j$ puisque $I_i \subset J$.

Donc, $J \models L_1 \wedge \dots \wedge L_m$, soit $J \models F(x\forall)$ et encore : $J \models \exists F$.

□

Lemme IV.12 : Soit G la formule close :

$$\forall y (t \neq s \vee F)$$

où F est une formule sans quantificateur et t (resp. s) est un n -uplet de termes (resp. de termes clos). On suppose de plus que toutes les variables de G ont une occurrence dans t . Alors, G vérifie la propriété C .

¹ Voir définition II.2.

Démonstration : Soit X une partie dirigée de 2^H telle que G soit vraie dans $\bigcup_{I \in X} I$.

- Si t et s ne sont pas unifiables, alors pour toute interprétation de Herbrand J (et en particulier appartenant à X , $J \models G$ (d'après le lemme II.1).

- Si t et s sont unifiables, soit θ leur upg. Le lemme II.1 montre que :

$$\bigcup_{I \in X} I \models G \quad \text{ssi} \quad \bigcup_{I \in X} I \models F\theta \quad (\theta \text{ est une substitution close}).$$

Le lemme IV.11 montre alors qu'il existe $J \in X$ tel que $J \models G$.

G vérifie bien la propriété C.



□

Lemme IV.13 : *L'ensemble des formules vérifiant la propriété C est stable par conjonction et disjonction.*

Démonstration : Soient F et G deux formules closes vérifiant la propriété C. Soit X une partie dirigée de 2^H telle que $F \wedge G$ soit vraie dans $\bigcup_{I \in X} I$. Alors, F et G sont toutes deux

vraies dans $\bigcup_{I \in X} I$. Soient J_1 et J_2 deux éléments de X telles que F (resp. G) soit vraie dans J_1 (resp. J_2).

Alors, $J = J_1 \cup J_2$ appartient à X puisque X est dirigée.

Comme $J_1 \subset J \subset \bigcup_{I \in X} I$ et comme $J_1 \models F$ et $\bigcup_{I \in X} I \models F$, on en déduit que $J \models F$. De

même, $J \models G$ et finalement $J \models F \wedge G$.

Démonstration analogue pour $F \vee G$.

□

Théorème IV.9 : *Si P est un programme Prolog sans variable locale, alors T_δ , T_θ , T_{st} , T_{fst} et T_f sont continus.*

Démonstration :

Soit T un des opérateurs précédents, associé au système d'axiomes Ax et soit X une partie dirigée de H .

Il faut montrer que :

$$T\left(\bigcup_{I \in X} I\right) = \bigcup_{I \in X} (T(I)).$$

- L'inclusion $\bigcup_{I \in X} (T(I)) \subset T\left(\bigcup_{I \in X} I\right)$ est vraie puisque T est monotone.

- Démontrons l'inclusion inverse.

$$\text{Soit } p(t) \in T\left(\bigcup_{I \in X} I\right).$$

p désigne ici un symbole de prédicat muni éventuellement d'un prime (') ou d'un caractère seconde ('').

Il existe une instance d'axiome de Ax dont le corps F est vrai dans $\bigcup_{I \in X} I$. La

formule F est une conjonction / disjonction de formules de la forme :

- $\exists y(t = s \wedge G)$ où G est une formule sans quantificateur, et l'on peut appliquer le lemme IV.11.

- $\forall y(t \neq s \vee G)$ où G est une formule sans quantificateur. Puisque P est sans variable locale, les variables y_i apparaissent toutes dans les termes s et comme les termes t sont clos, il existe au plus une instanciation des variables y_i par des termes clos pour laquelle $t = s$. On peut donc appliquer le lemme IV.12.

Le lemme IV.13 montre alors que F vérifie la propriété C et donc qu'il existe un élément J de X telle que F soit vraie dans J et donc tel que $p(t)$ appartienne à $T(J) \subset \bigcup_{I \in X} (T(I))$.

□

Conséquence : Si P est un programme sans variable locale, alors :

$$\begin{aligned} \text{pppf}(T_\theta) &= T_\theta \uparrow \omega = \text{ppm}(P_\theta^{\text{Her}})^1 \\ \text{pppf}(T_{st}) &= T_{st} \uparrow \omega = \text{ppm}(P_{st}^{\text{Her}}) \\ \text{pppf}(T_{fst}) &= T_{fst} \uparrow \omega = \text{ppm}(P_{fst}^{\text{Her}}) \\ \text{pppf}(T_f) &= T_f \uparrow \omega = \text{ppm}(P_f^{\text{Her}}). \end{aligned}$$

On en déduit les résultats déjà énoncés plus haut. Si P est un programme sans variable locale, alors :

$$A \in \text{SS}_{fst} \text{ ssi } P_{fst}^{\text{Her}} \models A \text{ et } A' \in \text{FF}'_{st} \text{ si } P_{fst}^{\text{Her}} \models A'.$$

$$A \in \text{SS}_f \text{ ssi } P_f^{\text{Her}} \models A \text{ et } A' \in \text{FF}'_f \text{ ssi } P_f^{\text{Her}} \models A'.$$

Si de plus, P vérifie l'hypothèse H_3^2 , alors :

$$A \in \text{SS}_{st} \text{ ssi } P_{st}^{\text{Her}} \models A \text{ et } A' \in \text{FF}'_{st} \text{ ssi } P_{st}^{\text{Her}} \models A'.$$

La classe des programmes Prolog sans variable locale est assez restreinte, du moins si l'on se réfère aux programmes écrits habituellement. Mentionnons néanmoins les résultats de Blair [Bla 87] et Maher [Mah 86] qui montrent que pour tout programme Prolog P , on peut construire de manière effective un programme Prolog Q sans variable locale tel que :

$$\text{SS}(P) = \text{SS}(Q) \text{ et } \text{FF}(P) = \text{FF}(Q).$$

Il pourrait être intéressant de savoir si l'on peut obtenir un résultat analogue pour les sémantiques opérationnelles standard, standard finie et invariante finie.

¹pppf = plus petit point fixe et ppm = plus petit modèle.

²voir p 54.

Chapitre V :

Un théorème de complétude

V.1 Introduction

Ainsi que nous l'avons déjà vu, un des principaux problèmes qui se pose en programmation logique peut être formulé en ces termes : "établir une relation simple entre un ensemble défini d'un point de vue opérationnel et une partie de l'ensemble des formules conséquences d'un système d'axiomes associé au programme".

La formulation de ce problème peut être simplifiée en ce qui nous concerne puisque les ensembles "opérationnels" que l'on étudie sont représentés par des parties de la base de Herbrand du langage sous-jacent au programme (ou des langages associés : L_d et L_l). On peut l'énoncer de la manière suivante : "associer un système d'axiomes à un programme dont l'ensemble des atomes clos conséquences soit égal à l'ensemble opérationnel étudié".

Les résultats obtenus aux chapitres III et IV, ainsi d'ailleurs que les résultats antérieurs rappelés au chapitre II suggèrent un certain nombre de remarques :

- les systèmes d'axiomes Ax associés aux programmes sont des **programmes généraux positifs** (à l'exception des complétés de Clark et de Fitting, mais les résultats de [Del 86, Ble 90, Gil 89] montrent que le premier est équivalent en un certain sens à un programme positif : la traduction SSI dédoublée; nous montrons un résultat du même type pour le complété de Fitting au chapitre VII).

- si T est l'opérateur de conséquence immédiate associé défini sur les interprétations de Herbrand du langage sous-jacent, la sémantique opérationnelle étudiée est (presque) **chaque fois**¹ représentée par l'itéré $T \uparrow \omega$.

- l'opérateur T est monotone mais en général, il n'est pas continu et par conséquent, $T \uparrow \omega$ n'est en général pas le plus petit point fixe de T ni le plus petit modèle de Herbrand de Ax .

- si l'on limite la notion de modèles aux seuls modèles de Herbrand, on obtient des résultats de **correction** (les atomes clos définis par la sémantique opérationnelle sont

¹sauf pour la sémantique standard dans le cas général.

conséquences de Ax^{Her}) mais pas de résultats de **complétude** (il existe des atomes clos conséquences qui ne sont pas retenus par la sémantique opérationnelle), sauf dans certains cas particuliers importants (les programmes sans variable locale ou sans symbole de fonction).

- les résultats de complétude deviennent encore plus mauvais lorsque l'on essaie de prendre en compte des conséquences (ou des requêtes) plus générales (par exemple : des buts contenant des variables).

Ces dernières remarques ne sont pas étonnantes dans la mesure où les systèmes d'axiomes associés Ax ne sont pas sous forme clausale, et que dans ce cas, la restriction aux modèles de Herbrand semble un peu arbitraire et n'a pas de sens logique très naturel.

On rencontre des problèmes équivalents lorsque l'on essaie de rendre compte de *la négation par l'échec* même en ne considérant que des programmes définis (que nous avons appelés programmes Prolog ici) et des interpréteurs idéaux (ne générant que des arbres SLD équitables). En effet,

$$A \in FF \text{ ssi } A \in H \setminus T \downarrow \omega \text{ [Emd 76]}$$

et $T \downarrow \omega$ n'est pas un point fixe de T , ni un modèle de P_{SI} (voir §II.2.5). Ce problème a été résolu [Cla 78, Jaf 83] en abandonnant le cadre des modèles de Herbrand pour se placer dans le cadre de la Théorie Equationnelle de Clark. Nous rappelons ces résultats :

$$\begin{aligned} A \in SS & \text{ ssi } A \in T \uparrow \omega \text{ ssi } P_{SSI}^{CET} \models A \\ A \in FF & \text{ ssi } A \in HNT \downarrow \omega \text{ ssi } P_{SSI}^{CET} \models \neg A. \end{aligned}$$

Kunen a montré un résultat du même type dans [Kun 87a] reliant l'ensemble $T \uparrow \omega$ à la notion de conséquence trivaluée dans le cadre de la Théorie Equationnelle de Clark.

Plus précisément,

- si P est un programme normal (c'est-à-dire dont les règles sont de la forme $A :- L$ où A est un atome et L une suite de littéraux positifs ou négatifs), si P_{Fit} est le complété de Fitting de P^1 et si T est l'opérateur de conséquence immédiate associé à P_{Fit} , alors pour tout atome clos A , on a :

$$A \in T \uparrow \omega \text{ ssi } P_{Fit}^{CET} \models_t A$$

(où ici, la notion de conséquence logique est une notion trivaluée² et où le langage sous-jacent $\mathcal{L}_K$ est un langage de Kunen³)

En fait, ce dernier résultat est un corollaire du théorème plus général suivant. Sous les mêmes hypothèses, si F est une formule close, on a :

$$\exists n \text{ tq } T \uparrow n \models F \text{ ssi } P_{Fit}^{CET} \models_t F$$

¹voir § II.5.5 et Définition VII.14

²voir Définition VII.5

³voir Définition I.16

Notre but dans ce chapitre est d'établir les résultats suivants :

- si P est un *programme général positif* et si le langage sous-jacent est un langage de Kunen, alors pour toute formule positive close F , on a :

$$\exists n \text{ tq } T \uparrow n \models F \quad \text{ssi} \quad P^{\text{CET}} \models F$$

- En particulier, si A est un atome clos :

$$A \in T \uparrow \omega \quad \text{ssi} \quad P^{\text{CET}} \models A.$$

Ce résultat permettra au chapitre VI d'obtenir des résultats de complétude satisfaisants pour les traductions axiomatiques standard, standard finie et invariante finie. Nous verrons également au chapitre VII qu'il a une interprétation naturelle en logique trivaluée et qu'il permet de retrouver comme corollaire le résultat de [Kun 87a].

V.2. Le Théorème de complétude

V.2.1 Quelques éléments de la théorie des modèles

La démonstration du théorème principal de ce chapitre utilise la notion de modèle saturé dont se sert d'ailleurs aussi Kunen dans sa démonstration. Nous rappelons dans ce paragraphe les définitions et résultats de théorie des modèles, en particulier concernant les ultraproducts (puisés dans "Model Theory" de Chang et Keisler [Cha 73]) dont nous avons besoin pour cette démonstration. Les nombres entre crochets désignent les numéros des pages de ce livre.

Définition V.1 : [12, 37] Une *théorie* T est un ensemble de formules. Soit $\mathcal{U}$ un modèle du langage $\mathcal{L}$. La *théorie de* $\mathcal{U}$ (notée $\text{Th}(\mathcal{U})$) est l'ensemble des formules de $\mathcal{L}$ vraies dans $\mathcal{U}$.

Définition V.2 : [78] Soit T une théorie et $\Sigma(x_1, \dots, x_n)$ un ensemble de formules. On dit que $\Sigma(x_1, \dots, x_n)$ est *consistant avec* T si T a un modèle qui réalise Σ^1 .

Proposition V.1 : [78] Soit T une théorie et $\Sigma(x_1, \dots, x_n)$ un ensemble de formules. $\Sigma(x_1, \dots, x_n)$ est consistant avec T ssi tout sous ensemble fini de Σ est réalisé dans un modèle de T .

Remarque : Cette proposition est une conséquence du théorème de compacité.

Définition V.3 : [68] Soit $\mathcal{U}$ un modèle du langage $\mathcal{L}$ de domaine A et soit $Y \subset A$. On définit $\mathcal{L}_Y = \mathcal{L} \cup \{c_y ; y \in Y\}$ où les c_y sont de nouvelles constantes. $\mathcal{U}_Y$ est alors le modèle construit à partir de $\mathcal{U}$ en interprétant chaque constante c_y par y .

¹Voir la définition I.1

Définition V.4 : [164] Soit I un ensemble non vide et soit $D \subset 2^I$ (où 2^I est l'ensemble des parties de I).

D est un **filtre** sur I ssi

- $I \in D$
- si $X, Y \in D$ alors $X \cap Y \in D$
- si $X \in D, Y \in 2^I$ et si $X \subset Y$ alors $Y \in D$

On dit que D est un filtre **propre** si de plus $\emptyset \notin D$ (c.-à-d. $D \neq 2^I$).

Définition V.5 : [166] D est un **ultrafiltre** sur I ssi

- D est un filtre propre sur I
- pour toute partie X de I , $X \in D$ ou $I \setminus X \in D$

Définition V.6 : [201] Un filtre D est **non clos par intersection dénombrable** (countably incomplete) ssi il existe une partie dénombrable E de D tq $(\bigcap E) \notin D$.

Proposition V.2 : [201] Il existe des ultrafiltres non clos par intersection dénombrable.

Soit D un filtre propre sur I et pour tout $i \in I$, soit A_i un ensemble non vide et soit $C = \prod_{i \in I} A_i$.

On définit la relation $=_D$ sur C par :

$$f =_D g \text{ ssi } \{i \in I; f(i) = g(i)\} \in D$$

Proposition V.3 : [168] La relation $=_D$ est une relation d'équivalence sur C .

Notation : On note f_D la classe d'équivalence de f dans cette relation.

Définition V.7 : On note $\prod_D A_i$ l'ensemble quotient $C/_D$. C'est le **produit réduit** des A_i modulo D . Si D est un ultrafiltre, le produit réduit est appelé **ultraproduit**. Si de plus, tous les A_i sont identiques, ce produit réduit est appelé **ultrapuissance** (ultrapower).

Définition V.8 : [168] Soit L un langage du premier ordre, D un filtre propre sur I , et pour tout $i \in I$, soit $\mathcal{U}_i$ un modèle de L . On définit un nouveau modèle de L noté $\prod_D \mathcal{U}_i$ par :

- Le domaine du modèle est $\prod_D A_i$ (où les A_i sont les domaines des interprétations $\mathcal{U}_i$).

- Soit p un symbole de prédicat n -aire. p est interprété par la relation P définie par :

$$P(f_{1D}, \dots, f_{nD}) \quad \text{ssi} \quad \{i \in I; P_i(f_1(i), \dots, f_n(i))\} \in D$$

où P_i est l'interprétation de p dans $\mathcal{U}_i$.

- Soit f un symbole de fonction n -aire ($n \geq 1$). f est interprété par la fonction F définie par :

$$F(f_{1D}, \dots, f_{nD}) = \langle F_i(f_1(i), \dots, f_n(i)) ; i \in I \rangle_D$$

où F_i est l'interprétation dans $\mathcal{U}_i$ de la fonction F .

- Soit c une constante de $\mathcal{L}$. c est interprétée par l'élément

$$C = \langle C_i ; i \in I \rangle_D$$

où C_i est l'interprétation de c dans $\mathcal{U}_i$.

Théorème V.1 (Théorème fondamental sur les ultraproduits) : [170]

Soit $\mathcal{V}$ l'ultraproduit $\prod_D \mathcal{U}_i$ de domaine V , et soit I l'ensemble des indices. Alors :

- Pour tout terme $t(x_1, \dots, x_n)$ de $\mathcal{L}$ et tout n -uplet $(f_{1D}, \dots, f_{nD}) \in V^n$, on a

$$t_{\mathcal{V}}[f_{1D}, \dots, f_{nD}] = \langle t_{\mathcal{U}_i}[f_1(i), \dots, f_n(i)] ; i \in I \rangle_D$$

où $t_{\mathcal{U}}$ désigne l'interprétation de t dans le modèle $\mathcal{U}$.

- Pour toute formule $F(x_1, \dots, x_n)$ de $\mathcal{L}$ et tout n -uplet $(f_{1D}, \dots, f_{nD}) \in V^n$, on a

$$\mathcal{V} \models F[f_{1D}, \dots, f_{nD}] \text{ ssi } \{i \in I ; \mathcal{U}_i \models F[f_1(i), \dots, f_n(i)]\} \in D.$$

- Pour tout énoncé clos F de $\mathcal{L}$, on a

$$\mathcal{V} \models F \text{ ssi } \{i \in I ; \mathcal{U}_i \models F\} \in D.$$

Proposition V.4 : [172] Soit $\mathcal{U}$ un modèle de $\mathcal{L}$ et soit $\prod_D \mathcal{U}$ une ultrapuissance de $\mathcal{U}$. Alors $\mathcal{U}$ et $\prod_D \mathcal{U}$ sont des modèles équivalents, c.-à-d. que les formules de $\mathcal{L}$ vraies dans les deux modèles sont les mêmes.

Définition V.9 : [96] Un modèle $\mathcal{U}$ de $\mathcal{L}$ de domaine A est ω -saturé ssi pour toute partie finie Y de A , tout ensemble $\Gamma(x)$ de formules de $\mathcal{L}_Y$ consistant avec $\text{Th}(\mathcal{U}_Y)$ est réalisé dans $\mathcal{U}_Y$.

Théorème V.2 : [305] Soit $\mathcal{L}$ un langage dénombrable du premier ordre et soit D un ultrafiltre non clos par intersection dénombrable sur un ensemble I . Alors, pour toute famille $(\mathcal{U}_i)_{i \in I}$ de modèles de $\mathcal{L}$, l'ultraproduit $\prod_D \mathcal{U}_i$ est ω -saturé.

V.2.2 Dépliage d'une formule

Soit $\mathcal{L}$ un langage du premier ordre avec égalité et P un programme général normalisé sur $\mathcal{L}$. Le but de ce paragraphe est d'associer à toute formule $F(x)$ de $\mathcal{L}$ une suite de formules équationnelles $(F^n(x))_{n \in \omega}$ telle que pour tout pré-modèle $\mathcal{M}$ de la $\text{CET}(\mathcal{L})$ et toute assignation $(x \mapsto \alpha)$ des variables x dans le domaine de $\mathcal{M}$, $F[\alpha]$ soit vraie dans $\mathcal{M}_{\mathcal{L}_n}^1$ ssi $F^n[\alpha]$ est vraie dans $\mathcal{M}$. Cette suite est définie de la manière suivante :

1- Soit n un entier quelconque.

Si $F = (t = t')$ où t et t' sont des termes de $\mathcal{L}$, alors $F^n = F$.

¹Nous rappelons que $\mathcal{M}_n$ est le modèle de la $\text{CET}(\mathcal{L})$ associé à l'interprétation $T^n \uparrow n$ (voir § 1.5 et Proposition II.4).

2- Si $F = p(x_1, \dots, x_m)$ où p est un symbole de prédicat m -aire (distinct de '=') de $\mathcal{L}$, alors $F^0 = \text{Faux}$ (ou $a \neq a$ si a est une constante de $\mathcal{L}$).

3- Soit n un entier et supposons F^n définie pour toute formule de la forme $t = t'$ où $p(x_1, \dots, x_m)$. On définit alors F^n pour toute formule F de $\mathcal{L}$ par application des règles suivantes :

$$\text{a) } (\neg F)^n = \neg F^n$$

$$\text{b) } (F \vee G)^n = F^n \vee G^n$$

$$\text{c) } (F \wedge G)^n = F^n \wedge G^n$$

$$\text{d) } (\forall x F)^n = \forall x F^n$$

$$\text{e) } (\exists x F)^n = \exists x F^n$$

f) Pour toute substitution θ des variables libres de F (telle qu'aucune variable de $\text{Im}(\theta)$ ne soit une variable liée de F), $(F\theta)^n = F^n\theta$

4- Soit n un entier et supposons F^n définie pour toute formule F de $\mathcal{L}$.

Si $F = p(x_1, \dots, x_m)$ où p est un symbole de prédicat m -aire (distinct de '=') de $\mathcal{L}$ et si $p(x_1, \dots, x_m) \Leftarrow G(x_1, \dots, x_m)$ est l'axiome de P correspondant, on pose $F^{n+1} = G^n$.

Exemple :

Soit P le système composé du seul axiome :

$$\forall x(p(x) \Leftarrow \forall y(x \neq s(y) \vee p(y)))$$

et soit $F(x) = p(x)$

$$F^0(x) = \text{Faux}$$

$$F^1(x) = \forall y(x \neq s(y) \vee \text{Faux})$$

$$F^2(x) = \forall y(x \neq s(y) \vee \forall z(y \neq s(z) \vee \text{Faux}))$$

Si le langage $\mathcal{L}$ ne contient que la constante '0' et le symbole fonctionnel 's', $F^n(x)$ est équivalente à la formule $x = 0 \vee x = s(0) \vee \dots \vee x = s^{n-1}(0)$ dans le pré-modèle standard $\mathcal{U}$ de la CET($\mathcal{L}$).

Remarque : Si F est une formule équationnelle, alors $F = F^n$ pour tout entier n .

Définition V.10 : On appelle la suite $(F^n(x))_{n \in \omega}$ la suite des formules dépliées de F par rapport à P .

Théorème V.2 : Soit P un programme général normalisé, $F(x)$ une formule de $\mathcal{L}$, n un entier, $\mathcal{M}$ un pré-modèle de la $CET(\mathcal{L})$ et $[x\alpha]$ une assignation des variables x_i dans $U\mathcal{M}^1$:

$$\mathcal{M}_n \models F[\alpha] \quad \text{ssi} \quad \mathcal{M} \models F^n[\alpha]$$

Démonstration : La démonstration se fait en 4 étapes.

a) Le théorème est vrai pour toute formule équationnelle. En effet :

$$\begin{array}{ccc} \mathcal{M}_n \models F[\alpha] & \text{ssi} & \mathcal{M} \models F[\alpha] \\ & \text{ssi} & \mathcal{M} \models F^n[\alpha] \text{ puisque } F = F^n. \end{array}$$

b) Si $n=0$, le théorème est vrai pour toute formule atomique. En effet, chaque membre de l'équivalence est faux.

c) Si pour un entier n donné, le théorème est vrai pour toute de la forme $p(x_1, \dots, x_q)$, alors il est vrai pour cet entier n et toute formule F de $\mathcal{L}$.

La démonstration se fait par récurrence sur la complexité k de la formule F .

Supposons que $k=0$.

Si F est de la forme $t = t'$, on applique le a).

Si $F(x_1, \dots, x_m)$ est de la forme $p(t_1, \dots, t_q)$ où p est un symbole de prédicat de $\mathcal{L}$ alors soit $\theta = (y_1/t_1, \dots, y_q/t_q)$.

$F(x_1, \dots, x_m) = p(y_1, \dots, y_q)\theta$ et donc $F^n = (p\theta)^n = p^n\theta$.

$$\begin{array}{ccc} \mathcal{M}_n \models F[\alpha] & \text{ssi} & \mathcal{M}_n \models p[t_1[\alpha], \dots, t_q[\alpha]] \\ & \text{ssi} & \mathcal{M} \models p^n[t_1[\alpha], \dots, t_q[\alpha]] \\ & \text{ssi} & \mathcal{M} \models p^n\theta[\alpha] \\ & \text{ssi} & \mathcal{M} \models (p\theta)^n[\alpha] \\ & \text{ssi} & \mathcal{M} \models F^n[\alpha] \end{array}$$

Supposons le théorème démontré pour toute formule de complexité k . Soit F une formule de complexité $k+1$.

- Si $F = \neg G$,

$$\begin{array}{ccc} \mathcal{M}_n \models F[\alpha] & \text{ssi} & G[\alpha] \text{ est faux dans } \mathcal{M}_n \\ & \text{ssi} & G^n[\alpha] \text{ est faux dans } \mathcal{M} \\ & \text{ssi} & \mathcal{M} \models \neg G^n[\alpha] \\ & \text{ssi} & \mathcal{M} \models (\neg G)^n[\alpha] \\ & \text{ssi} & \mathcal{M} \models F^n[\alpha] \end{array}$$

¹Où $U\mathcal{M}$ est le domaine de $\mathcal{M}$.

- Si $F = G \vee H$,

$$\begin{array}{ll}
 \mathfrak{M}_n \models F[\alpha] & \text{ssi} \quad \mathfrak{M}_n \models (G \vee H)[\alpha] \\
 & \text{ssi} \quad \mathfrak{M}_n \models G[\alpha] \quad \text{ou} \quad \mathfrak{M}_n \models H[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models G^n[\alpha] \quad \text{ou} \quad \mathfrak{M} \models H^n[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models (G^n \vee H^n)[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models (G \vee H)^n[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models F^n[\alpha]
 \end{array}$$

- Démonstration analogue si $F = G \wedge H$

- Si $F = \forall x G(x, x_1, \dots, x_m)$,

$$\begin{array}{ll}
 \mathfrak{M}_n \models F[\alpha] & \text{ssi} \quad \mathfrak{M}_n \models \forall x G[x, \alpha] \\
 & \text{ssi} \quad \text{pour tout } \beta \in U^{\mathfrak{M}}, \mathfrak{M}_n \models G[\beta, \alpha] \\
 & \text{ssi} \quad \text{pour tout } \beta \in U^{\mathfrak{M}}, \mathfrak{M} \models G^n[\beta, \alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models \forall x G^n[x, \alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models (\forall x G(x))^n[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models F^n[\alpha]
 \end{array}$$

- Démonstration analogue si $F = \exists x G(x)$.

d) Supposons le théorème démontré pour un entier n donné. Soit $F = p(x_1, \dots, x_m)$ et soit $p(x_1, \dots, x_m) \Leftarrow G(x_1, \dots, x_m)$ l'axiome de P associé.

$$\begin{array}{ll}
 \mathfrak{M}_{n+1} \models F[\alpha] & \text{ssi} \quad \mathfrak{M}_n \models G[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models G^n[\alpha] \\
 & \text{ssi} \quad \mathfrak{M} \models F^{n+1}[\alpha].
 \end{array}$$

□

Remarque : La suite des formules dépliées d'une formule par rapport à un programme général P permet d'exprimer par une formule équationnelle la proposition :

"la formule F est obtenue à la n -ième étape d'un "chaînage avant" effectué sur P ".

V.2.3 Construction d'un modèle saturé d'un programme général positif

Exemple : Soit P un programme général positif normalisé. Nous construisons dans ce paragraphe un pré-modèle $\mathcal{V}$ de la $CET(\mathcal{L})$ équivalent au pré-modèle standard $\mathcal{U}$ et ω -saturé. C'est-à-dire :

- pour toute formule close F de $\mathcal{L}$,

$$\mathcal{V} \models F \quad \text{ssi} \quad \mathcal{U} \models F$$

- si $\Gamma(x)$ est une famille de formules équationnelles ayant x pour seule variable libre et si toute partie finie de $\Gamma(x)$ est réalisée dans $\mathcal{V}$ alors, $\Gamma(x)$ est réalisée dans $\mathcal{V}$.

Nous rappelons que l'on note $\mathcal{V}_n = T^{\mathcal{V}} \uparrow n$ pour tout entier n et $\mathcal{V}_\omega = T^{\mathcal{V}} \uparrow \omega$. Montrons sur un exemple en quoi ces propriétés de $\mathcal{V}$ permettent d'obtenir le résultat de complétude énoncé dans l'introduction.

Soit P le programme composé du seul axiome :

$$\forall x(p(x) \Leftarrow \forall y(x \neq s(y) \vee p(y)))$$

et soit $F = \forall x p(x)$.

On suppose que $\mathcal{L}$ ne contient comme symboles fonctionnels que 's' (1-aire) et '0' (constante).

Nous avons vu que $p^n(x)$ est équivalente dans le pré-modèle standard $\mathcal{U}$ à la formule :

$$x = 0 \vee x = s(0) \vee \dots \vee x = s^{n-1}(0).$$

Pour tout entier n ,

$$\mathcal{U} \models \neg p^n(s^m(0)) \text{ pour } m \geq n$$

donc,

$$\mathcal{U} \models \neg \forall x p^n(x)$$

soit encore,

$$\mathcal{U} \models (\neg \forall x p(x))^n$$

et d'après le théorème V.2,

$$\mathcal{U}_n \models (\neg \forall x p(x)).$$

F est donc faux dans tous les $\mathcal{U}_n$.

Comme $\mathcal{U}_\omega = \{p(s^n(0)); n \geq 0\}$, F est vraie dans $\mathcal{U}_\omega$.

Comme $\mathcal{U}$ et $\mathcal{V}$ sont équivalents, on a :

$$\mathcal{V} \models \neg p^n(s^m(0)) \text{ pour } m \geq n.$$

Soit $\Gamma(x)$ la famille de formules $\{\neg p^n(x)\}_{n \in \omega}$. Pour toute partie finie A de ω , $\{\neg p^n(x)\}_{n \in A}$ est réalisée dans $\mathcal{V}$ en prenant par exemple $x = s^q(0)$ où $q = \sup(A)$.

La famille $\Gamma(x)$ est donc réalisée dans $\mathcal{V}$ par un élément α . On a :

$$\mathcal{V} \models \neg p^n(\alpha) \text{ pour tout entier } n.$$

Donc, par définition des formules dépliées,

$$\mathcal{V}_n \models \neg p(\alpha) \text{ pour tout entier } n$$

et par conséquent,

$$\mathcal{V}_\omega \models \neg p(\alpha) \text{ et } \mathcal{V}_\omega \models \neg \forall x p(x).$$

F est fausse dans $\mathcal{V}_\omega$.

Il suffira alors de montrer que $\mathcal{V}_\omega$ est un modèle de $PCET$ pour prouver que F n'est pas conséquence logique de $PCET$. Par contraposée, si F est conséquence logique de $PCET$, il existe un entier n pour lequel F est vraie dans $\mathcal{U}_n$, c.-à-d. $T \uparrow n$.

Donnons maintenant des définitions et des démonstrations plus formelles.

Nous supposons donné pour toute la suite du § V.2.3 un programme général positif normalisé P .

On rappelle que $\mathcal{U}$ est le pré-modèle standard de la $CET(\mathcal{L})$ et que pour tout entier n , $\mathcal{U}_n$ est le modèle défini par l'interprétation $T\uparrow n$ de $\mathcal{L}$ sur $\mathcal{U}$.

Notation : Soit D un ultrafiltre non clos par intersection dénombrable sur un ensemble I . On pose $V = \prod_D U$ (où U est l'espace de Herbrand associé à $\mathcal{L}$), $\mathcal{V} = \prod_D \mathcal{U}$ et, pour tout entier n , $\mathcal{V}_n = \prod_D \mathcal{U}_n$.

Lemme V.1 : $\mathcal{V}$ est un modèle ω -saturé de la $CET(\mathcal{L})$.

Démonstration : C'est une application du théorème V.2. □

Notation : On définit également un modèle $\mathcal{V}_\omega$ de la $CET(\mathcal{L})$ de la manière suivante : $\mathcal{V}_\omega$ a pour domaine V , les constantes et les fonctions sont interprétées dans $\mathcal{V}_\omega$ comme dans les $\mathcal{V}_n$ (c'est-à-dire, comme dans $\mathcal{V}$) et pour tout symbole de prédicat p m -aire (différent de '=') et tout m -uplet $(f_{1D}, \dots, f_{mD}) \in V^m$:

$$\mathcal{V}_\omega \models p[f_{1D}, \dots, f_{mD}] \text{ ssi il existe un entier } n \text{ tq } \mathcal{V}_n \models p[f_{1D}, \dots, f_{mD}]$$

Lemme V.2 : $\mathcal{V}$, $\mathcal{V}_n$ (pour tout entier n) et $\mathcal{V}_\omega$ sont des pré-modèles de la $CET(\mathcal{L})$ équivalents à $\mathcal{U}$.

Démonstration : Les pré-modèles défini par chaque $\mathcal{V}_n$ et par $\mathcal{V}_\omega$ sont égaux à $\mathcal{V}$ et la proposition V.4 montre que $\mathcal{V}$ est équivalent à $\mathcal{U}$. □

Nous montrons, dans la proposition suivante, que la notation choisie pour $\mathcal{V}_n$ est compatible avec la notation définie au § I.5.

Proposition V.5 : Pour tout entier n , $\mathcal{V}_n$ (resp. $\mathcal{V}_\omega$) est le modèle de la $CET(\mathcal{L})$ défini par $T\mathcal{V}\uparrow n$ (resp. $T\mathcal{V}\uparrow \omega$).

Démonstration : (par récurrence sur n)

Soit p un symbole de prédicat m -aire, $p(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m)$ l'axiome de P associé et $(f_{1D}, \dots, f_{mD}) \in V^m$.

Si $n = 0$:

$$\begin{array}{lll} \mathcal{V}_0 \models p[f_{1D}, \dots, f_{mD}] & \text{ssi} & \{i \in I; \mathcal{U}_0 \models p(f_1(i), \dots, f_m(i))\} \in D \\ & \text{ssi} & \emptyset \in D \text{ (ce qui est faux) (puisque } T\uparrow 0 = \emptyset) \\ & \text{ssi} & (p; f_{1D}, \dots, f_{mD}) \in T\mathcal{V}\uparrow 0 (= \emptyset). \end{array}$$

Supposons la propriété établie pour l'entier n .

$$\begin{aligned}
 \mathcal{V}_{n+1} \models p[f_{1D}, \dots, f_{mD}] & \text{ ssi } \{i \in I; \mathcal{U}_{n+1} \models p(f_1(i), \dots, f_m(i))\} \in D \\
 & \text{ ssi } \{i \in I; \mathcal{U}_n \models F(f_1(i), \dots, f_m(i))\} \in D \\
 & \text{ ssi } \mathcal{V}_n \models F[f_{1D}, \dots, f_{mD}] \text{ d'après le théorème } \\
 & \text{ fondamental des ultraproducts.} \\
 & \text{ ssi } (p; f_{1D}, \dots, f_{mD}) \in T^{\mathcal{V}}(T^{\mathcal{V}} \uparrow n) \\
 & \text{ ssi } (p; f_{1D}, \dots, f_{mD}) \in T^{\mathcal{V}} \uparrow (n+1)
 \end{aligned}$$

La première partie de la proposition est donc démontrée.

$$\begin{aligned}
 \mathcal{V}_{\omega} \models p(f_{1D}, \dots, f_{mD}) & \text{ ssi } \text{il existe un entier } n \text{ tq } \mathcal{V}_n \models p[f_{1D}, \dots, f_{mD}] \\
 & \text{ ssi } \text{il existe un entier } n \text{ tq } (p; f_{1D}, \dots, f_{mD}) \in T^{\mathcal{V}} \uparrow n \\
 & \text{ ssi } (p; f_{1D}, \dots, f_{mD}) \in T^{\mathcal{V}} \uparrow \omega.
 \end{aligned}$$

□

Le théorème suivant décrit une propriété fondamentale du modèle $\mathcal{V}_{\omega}$.

Théorème V.3 : *Pour toute formule positive $F(x_1, \dots, x_m)$ et tout m -uplet $(f_{1D}, \dots, f_{mD})$ d'éléments de V :*

$$\mathcal{V}_{\omega} \models F[f_{1D}, \dots, f_{mD}] \text{ ssi il existe un entier } n \text{ tq } \mathcal{V}_n \models F[f_{1D}, \dots, f_{mD}].$$

Démonstration : (par récurrence sur la complexité de F).

- Si F est une formule atomique de symbole de prédicat différent de '=', c'est vrai par définition de $\mathcal{V}_{\omega}$.

- Si F est une formule équationnelle, c'est vrai parce que $\mathcal{V}_{\omega}$ et $\mathcal{V}_n$ (pour tout entier n) sont des pré-modèles équivalents de la CET(L).

- Soit $F(x_1, \dots, x_m) = G(x_1, \dots, x_m) \vee H(x_1, \dots, x_m)$ et soit $(f_{1D}, \dots, f_{mD}) \in V^m$ tq :

$$\mathcal{V}_{\omega} \models F[f_{1D}, \dots, f_{mD}].$$

Alors $\mathcal{V}_{\omega} \models G[f_{1D}, \dots, f_{mD}]$ ou $\mathcal{V}_{\omega} \models H[f_{1D}, \dots, f_{mD}]$. En appliquant l'hypothèse de récurrence, on voit donc qu'il existe un entier n tq $\mathcal{V}_n \models F[f_{1D}, \dots, f_{mD}]$.

- De même si $F(x_1, \dots, x_n) = G(x_1, \dots, x_n) \wedge H(x_1, \dots, x_n)$.

- Soit $F(x_1, \dots, x_m) = \exists x G(x, x_1, \dots, x_m)$ et soit $(f_{1D}, \dots, f_{mD}) \in V^m$ tq :

$$\mathcal{V}_{\omega} \models F[f_{1D}, \dots, f_{mD}].$$

On a alors $\mathcal{V}_{\omega} \models \exists x G[x, f_{1D}, \dots, f_{mD}]$ et il existe donc $f_D \in V$ tq :

$$\mathcal{V}_{\omega} \models G[f_D, f_{1D}, \dots, f_{mD}].$$

Par hypothèse de récurrence, on sait qu'il existe un entier n tq $\mathcal{V}_n \models G[f_D, f_{1D}, \dots, f_{mD}]$ et donc $\mathcal{V}_n \models \exists x G[x, f_{1D}, \dots, f_{mD}]$, soit finalement $\mathcal{V}_n \models F[f_{1D}, \dots, f_{mD}]$.

- Soit $F(x_1, \dots, x_m) = \forall x G(x, x_1, \dots, x_m)$ et soit $(f_{1D}, \dots, f_{mD}) \in V^m$ tq :

$$\mathcal{V}_\omega \models F[f_{1D}, \dots, f_{mD}].$$

C'est ce cas (le plus difficile) qui nécessite la construction précédente de $\mathcal{V}_\omega$ au moyen d'ultrafiltres.

Supposons qu'il n'existe pas d'entier n pour lequel $\mathcal{V}_n \models F[f_{1D}, \dots, f_{mD}]$. Pour tout entier n , il existe donc un élément f_{Dn} de V tq $\mathcal{V}_n \models \neg G[f_{Dn}, f_{1D}, \dots, f_{mD}]$.

Soit $(H^n(x, x_1, \dots, x_m))_{n \in \omega}$ la suite des formules dépliées de $\neg G(x, x_1, \dots, x_m)$.

$$\mathcal{V}_n \models \neg G[f_{Dn}, f_{1D}, \dots, f_{mD}] \text{ ssi } \mathcal{V} \models H^n[f_{Dn}, f_{1D}, \dots, f_{mD}].$$

Comme $T^\mathcal{V}$ est monotone et comme $\neg G$ est une formule négative¹, si $n < p$ alors :

$$\mathcal{V}_p \models \neg G[f_{Dp}, f_{1D}, \dots, f_{mD}] \text{ implique } \mathcal{V}_n \models \neg G[f_{Dp}, f_{1D}, \dots, f_{mD}],$$

soit encore,

$$\mathcal{V} \models H^p[f_{Dp}, f_{1D}, \dots, f_{mD}] \text{ implique } \mathcal{V} \models H^n[f_{Dp}, f_{1D}, \dots, f_{mD}].$$

Soit $Y = \{f_{1D}, \dots, f_{mD}\}$ et soient $c_{1D}, \dots, c_{mD}$ les constantes de L_Y associée à $f_{1D}, \dots, f_{mD}$ ².

Si A est une partie finie non vide de ω , l'ensemble de formules $\{H^n(x, c_{1D}, \dots, c_{mD})\}_{n \in A}$ est donc réalisé dans $\mathcal{V}_Y$ en prenant $x = f_{Dp}$ où $p = \sup(A)$.

La proposition V.1 montre alors que l'ensemble $\Gamma(x) = \{H^n[x, c_{1D}, \dots, c_{mD}]\}_{n \in \omega}$ est consistant avec $\text{Th}(\mathcal{V}_Y)$.

Comme $\mathcal{V}$ est ω -saturé (Théorème V.2), $\Gamma(x)$ est réalisable dans $\mathcal{V}_Y$. C'est-à-dire qu'il existe $f_D \in V$ tq $\mathcal{V} \models H^n(f_D, f_{1D}, \dots, f_{mD})$ pour tout entier n .

On a donc $\mathcal{V}_n \models \neg G(f_D, f_{1D}, \dots, f_{mD})$ pour tout entier n .

Or $\mathcal{V}_\omega \models G(f_D, f_{1D}, \dots, f_{mD})$. Par hypothèse de récurrence, il existe donc un entier n pour lequel $\mathcal{V}_n \models G(f_D, f_{1D}, \dots, f_{mD})$, ce qui est contradictoire.

□

Corollaire V.1 : Pour toute formule positive close F de L , F est vraie dans $\mathcal{V}_\omega$ ssi il existe un entier n tq F soit vraie dans $\mathcal{V}_n$.

Démonstration : C'est un cas particulier du théorème précédent.

□

Corollaire V.2 : $T^\mathcal{V} \uparrow \omega$ est le plus petit point fixe de $T^\mathcal{V}$. $\mathcal{V}_\omega$ est donc le plus petit modèle de PCET de pré-modèle $\mathcal{V}$.

¹c.-à-d. la négation d'une formule positive (voir Définition I.23 et Proposition I.9).

² Voir Définition V.3.

Démonstration :

Soit $(p; f_{1D}, \dots, f_{mD}) \in T^{\nu} \uparrow (\omega+1)$ et soit $p(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m)$ l'axiome de P associé.

$F[f_{1D}, \dots, f_{mD}]$ est vrai dans $T^{\nu} \uparrow \omega$ et comme F est une formule positive, on peut appliquer le théorème précédent. Il existe un entier n tq $F[f_{1D}, \dots, f_{mD}]$ soit vrai dans $\mathcal{V}_n$. On en déduit que $(p; f_{1D}, \dots, f_{mD}) \in T^{\nu} \uparrow (n+1) \subset T^{\nu} \uparrow \omega$.

□

Le théorème suivant énonce le résultat principal de ce chapitre. C'est un *théorème de complétude* en ce sens qu'il montre qu'un "chaînage avant" appliqué à un programme général positif P "donne" toute les conséquences positives du système d'axiomes $PCET$.

Théorème V.4 : *Soit P un programme général positif normalisé. Si F est une formule positive close de L et si F est une conséquence logique de $PCET$ alors il existe un entier n tel que F soit vraie dans $T \uparrow n$.*

Démonstration : Comme F est une conséquence logique de $PCET$, F est vraie dans tous les modèles de $PCET$ et en particulier dans $\mathcal{V}_\omega$. On en déduit, par le théorème précédent qu'il existe un entier n tel que F soit vraie dans $\mathcal{V}_n$. Comme $\mathcal{V}_n$ est équivalent à $\mathcal{U}_n$ (proposition V.4), F est vraie dans $\mathcal{U}_n$, c'est-à-dire dans $T \uparrow n$.

□

Remarque : Le fait que F soit une formule positive est indispensable, comme le montre l'exemple suivant. Soit P le programme général positif :

$$\forall x [p(x) \Leftarrow \exists y (x=s(y) \wedge p(y)) \vee x = a]$$

et soit $F = P$.

On suppose que L ne contient que les deux symboles fonctionnels 'a' et 's'.

Pour tout entier n, $T \uparrow n = \{p(s^m(a)) ; 0 \leq m \leq n-1\}$.

La formule F est une conséquence logique de $PCET$ et F est fausse dans $T \uparrow n$ pour tout entier n.

Corollaire V.3 : *Si A est un atome clos et si A est une conséquence logique de $PCET$ alors $A \in T \uparrow \omega$.*

Démonstration : C'est un cas particulier du théorème précédent.

□

Remarque : La réciproque est évidemment fausse dans le cas général comme le montre l'exemple suivant :

P : $A(a) \Leftarrow \forall x \forall y (x=y)$ et L ne contient que la constante a.

$A(a) \in T \uparrow 1$ et $A(a)$ n'est pas une conséquence logique de P.

Le théorème suivant montre que si $\mathbf{L}_k$ est un langage de Kunen, la réciproque est vraie.

Théorème V.5 : *Soit P un programme général positif normalisé. Si F est une formule positive close de $\mathbf{L}_k$ et si $\mathbf{L}_k$ est un langage de Kunen alors :*
F est une conséquence logique de $\mathbf{P}^{\text{CET}}$ ssi il existe un entier n tel que F soit vraie dans $T\uparrow n$.

Démonstration :

Soit $\mathcal{M}$ un modèle quelconque de $\mathbf{P}^{\text{CET}}$ et supposons que $T\uparrow n \models F$, c.-à-d. $\mathcal{U}_n \models F$.

Le théorème V.2 montre que : $\mathcal{U} \models F^n$ c.-à-d. $\text{CET}(\mathbf{L}_k) \models F^n$ puisque F^n est une formule équationnelle de $\mathbf{L}_k^1$.

Comme $\mathcal{M}$ est un modèle de $\mathbf{P}^{\text{CET}}$ donc de la $\text{CET}(\mathbf{L}_k)$, on a $\mathcal{M} \models F^n$.

Le théorème V.2 montre alors que $\mathcal{M}_n \models F$ et comme pour tout entier n, $T^{\mathcal{M}}\uparrow n$ est contenu dans l'interprétation définie par $\mathcal{M}$ (puisque $T^{\mathcal{M}}$ est monotone), et comme F est positive, on en déduit que $\mathcal{M} \models F$.

Donc, $T\uparrow n \models F$ implique $\mathbf{P}^{\text{CET}} \models F$.

□

Corollaire V.4 : *Soit P un programme général positif normalisé. Si $\mathbf{L}_k$ est un langage de Kunen et si A est un atome clos alors :*

$$\mathbf{P}^{\text{CET}} \models A \quad \text{ssi} \quad A \in T\uparrow\omega.$$

Démonstration : C'est une conséquence immédiate du théorème précédent.

□

Corollaire V.5 : *Soit P un programme général positif normalisé. Si $\mathbf{L}_k$ est un langage de Kunen, alors pour toute formule positive close F de $\mathbf{L}$:*

F est vraie dans $\mathcal{V}_\omega$ ssi F est conséquence logique de $\mathbf{P}^{\text{CET}}$.

(c.-à-d. $\mathcal{V}_\omega$ est un modèle générique de $\mathbf{P}^{\text{CET}}$ pour les formules positives).

Démonstration :

Si F est conséquence logique de $\mathbf{P}^{\text{CET}}$, F est évidemment vraie dans $\mathcal{V}_\omega$ puisque c'est un modèle de $\mathbf{P}^{\text{CET}}$.

Si F est vraie dans $\mathcal{V}_\omega$, il existe un entier n tel que F soit vraie dans $T\uparrow n$. Le théorème V.4 montre alors que F est conséquence logique de $\mathbf{P}^{\text{CET}}$.

□

¹Voir le théorème I.2.

Remarques : Etant donné un programme général positif P , quel langage $\mathcal{L}$ considérer ?

- le langage $\mathcal{L}_P$ défini par le programme P (c'est-à-dire ne contenant pas d'autres symboles de prédicat (autres que '=') et de fonctions que ceux ayant une occurrence dans P) peut sembler le plus naturel, mais la remarque précédente montre que dans ce cas, l'opérateur de conséquence immédiate n'est plus un opérateur de déduction logique; c'est-à-dire que l'on peut avoir $A \in T\uparrow n$ sans que A soit une conséquence logique de $\mathcal{P}^{CET}$.

- les langages de Kunen fournissent les propriétés logiques les plus fortes mais ils peuvent sembler quelque peu "démessurés". Encore que du point de vue de l'interrogation d'une base de donnée ou d'un programme Prolog, leur utilisation soit justifiée : tout symbole fonctionnel doit pouvoir être utilisé pour composer une requête.

La question suivante se pose naturellement (et personne à notre connaissance n'y a répondu) :

- Peut-on associer à P un langage $\mathcal{L}$ de manière "naturelle et effective" tel que : $A \in T\uparrow n$ implique $\mathcal{P}^{CET} \models A$ et qui soit minimal (en un sens à préciser) ?

Pour un tel langage, on aurait alors le théorème suivant : pour tout atome clos A ,

$$\mathcal{P}^{CET} \models A \text{ ssi il existe un entier } n \text{ tel que } A \in T\uparrow n.$$

Conclusion : Les théorèmes de complétude démontrés dans ce chapitre donnent un sens logique fort à des ensembles (les $T\uparrow n$) qui par ailleurs ont une interprétation calculatoire très naturelle. En ce sens, ils permettent de mieux comprendre les rapports entre la logique et le calcul, dont nous parlions dans l'introduction.

Notons cependant que la construction du modèle $\mathcal{V}_\omega$ est hautement ineffective et que la démonstration du théorème V.3 ne nous fournit aucune intuition sur le "pourquoi" de cette adéquation (qui peut sembler un peu miraculeuse) entre les ensembles $T\uparrow n$ et les conséquences positives de $\mathcal{P}^{CET}$.

Ces théorèmes montrent¹ également sans ambiguïté que le cadre logique de la Théorie Equationnelle de Clark est celui qui convient le mieux à la programmation logique contrairement à l'idée première qui assignait ce rôle aux modèles de Herbrand.

¹ou confirment : les résultats de [Jaf 83, Kun 87, 89] ont tous été obtenus dans ce cadre.

Chapitre VI :

Application aux sémantiques des programmes Prolog dans le cadre de la CET.

Nous avons vu au chapitre III que si l'on restreint les modèles des traductions axiomatiques standard, standard finie et invariante finie d'un programme Prolog P aux modèles de Herbrand, les sémantiques opérationnelles correspondantes sont *correctes* vis à vis de leur traduction. Par contre la *complétude* n'est vérifiée que sous une forme faible et à certaines conditions (très fortes) sur le programme P . Nous montrons dans ce chapitre que si l'on se place dans le cadre de la théorie équationnelle de Clark les sémantiques opérationnelles standard, standard finie et invariante finie d'un programme Prolog P sont *correctes* et *complètes* vis à vis de leur traduction axiomatique (en conservant la condition H_3 ou H_3' dans le cas standard). Nous obtenons donc des résultats analogues à ceux de [Del 88a, Gil 89, Ble 90] (Théorème II.6) qui sont eux mêmes une version "dédoublée" des résultats classiques de correction et de complétude de la résolution SLDNF appliquée aux programmes définis pour le complété de Clark de ces programmes (Théorème II.5).

VI.1 Complétude dans le cadre de la CET(L)

Nous appliquons dans ce paragraphe le théorème de complétude démontré au chapitre précédent aux traductions axiomatiques standard, standard finie et invariante finie.

Soit L un langage du premier ordre avec égalité.

Théorème VI.1 : Soit P un programme Prolog.

Si une formule positive close F de L_d est conséquence logique de P_{fst}^{CET} (resp. P_f^{CET}), alors il existe un entier n pour lequel F est vraie dans $T_{fst} \uparrow n$ (resp. $T_f \uparrow n$).

Si une formule positive close F de L_i est conséquence logique de P_{st}^{CET} , alors il existe un entier n pour lequel F est vraie dans $T_{st} \uparrow n$.

Démonstration : Puisque les traductions axiomatiques standard, standard finie et invariante finie d'un programme Prolog P sont des programmes généraux positifs normalisés, on peut appliquer le théorème V.4 qui fournit le résultat. $\square$

VI.1.1 La sémantique standard

Rappelons tout d'abord les conditions H_3 et H_3' que nous avons énoncées au chapitre III.

Soit P un programme Prolog.

H_3 : Pour toute règle de P de corps L et pour toute substitution σ qui instancie les variables non locales de cette règle, si pour une substitution close τ le but $:- L\sigma\tau$ réussit, alors le but $:- L\sigma$ réussit également.

H_3' : Pour tout but $:- L$ et toute substitution close τ des variables de L , si le but $:- L\tau$ réussit alors le but $:- L$ réussit avec une réponse calculée (standard) θ telle que $\theta \leq \tau$.

Rappel : Les notions de succès ou d'échec se réfèrent à un interpréteur standard (comme chaque fois qu'il est question de la sémantique opérationnelle standard).

Théorème VI.2 : (Complétude de la sémantique opérationnelle standard pour la traduction axiomatique standard P_{st}^{CET})

Soit P un programme Prolog et L une liste d'atomes de L .

- a) Si $P_{st}^{CET} \models \forall [L]_{st}^F$ alors l'arbre SLD standard de $P \cup \{:- L\}$ est fini.
- b) Si $P_{st}^{CET} \models \forall [L]_{st}^{EF}$ alors l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis.
- c) On suppose de plus que P vérifie la condition H_3' . Alors, si σ est une substitution des variables de L telle que $P_{st}^{CET} \models \forall [L\sigma]_{st}^S$, il existe une réponse calculée standard¹ θ pour le but $:- L$ telle que $\theta \leq \sigma$.

Démonstration :

a) D'après le théorème VI.1, il existe un entier n tel que $T_{st} \uparrow n \models \forall [L]_{st}^F$ et le corollaire III.4 permet de conclure.

b) Démonstration analogue.

c) Soit L_1 un sur-langage de L contenant une infinité de constantes et soit $\mathcal{U}_1$ le pré-modèle standard de la $CET(L_1)$.

Puisque tout modèle de $P_{st}^{CET(L_1)}$ induit naturellement un modèle de $P_{st}^{CET(L)}$, la formule $\forall [L\sigma]_{st}^S$ est également conséquence logique de $P_{st}^{CET(L_1)}$. Il existe donc, d'après le théorème VI.1, un entier n pour lequel, $T_{st}^{\mathcal{U}_1} \uparrow n \models \forall [L\sigma]_{st}^S$.

On a alors $P_{st}^{Her(L_1)} \models \forall [L\sigma]_{st}^S$ et la proposition III.3 montre alors que le but $:- L$ a une réponse calculée standard θ telle que $\theta \leq \sigma$. $\square$

¹c.-à-d. trouvée par un interpréteur standard.

On a, en particulier, le résultat suivant :

Corollaire VI.1 : Soit P un programme Prolog et A un atome clos de L .

- a) Si $P_{st}^{CET} \models A$ alors l'arbre SLD standard de $P \cup \{:- A\}$ est fini, soit encore :
 $A \in SS_{fst} \cup FF_{st}$.
 b) Si $P_{st}^{CET} \models A$ alors $A \in FF_{st}$.
 c) Si de plus, P vérifie la condition H_3 et si $P_{st}^{CET} \models A$ alors $A \in SS_{st}$.

Démonstration :

- a) Il existe un entier n tel que $T_{st} \uparrow n \models A$. Comme $\forall [A]_{st}^F = A' \vee (A \wedge A'')$ et comme d'après la proposition III.1, si $T_{st} \uparrow n \models A$ alors $T_{st} \uparrow n \models A' \vee (A \wedge A'')$. On en déduit que l'arbre SLD standard de $P \cup \{:- A\}$ est fini.
 b) En effet, $\forall [A]_{st}^{EF} = A'$. Il suffit d'appliquer la proposition précédente.
 c) Il existe un entier n tel que $T_{st} \uparrow n \models A$ et donc $A \in SS_{st}$ d'après le théorème III.4.

Remarque : Nous ne sommes donc pas arrivé à associer à *tout* programme P un système d'axiome Ax pour lequel pour tout atome clos A : $A \in SS_{st}$ ssi $Ax \models A$. Il est naturel de se demander si une telle traduction axiomatique existe. La réponse est oui.

Soit P un programme Prolog. Il est possible d'écrire un programme Prolog Q_P (en utilisant par exemple un interpréteur Prolog standard écrit en Prolog) tel que pour tout atome clos A , $Q_P \cup \{:- A\}$ ait une réfutation ssi $A \in SS_{st}$. Pour un tel programme, on a donc : $SS(Q_P) : SS_{st}(P)$ et par conséquent, $A \in SS_{st}$ ssi $(Q_P)_{SI} \models A$. Mais le système d'axiome $(Q_P)_{SI}$, même s'il peut être construit récursivement à partir de P , n'a pas de rapport syntaxique simple avec P . L'intérêt des traductions axiomatiques standard, standard finie et invariante finie réside dans le fait qu'elles expriment des propriétés calculatoires complexes "à peu de frais".

VI.1.2 La sémantique standard finie

Théorème VI.3 : (Complétude de la sémantique opérationnelle standard finie pour la traduction axiomatique standard finie P_{fst}^{CET})

Soit P un programme Prolog et L une liste d'atomes de L .

- a) Si $P_{fst}^{CET} \models \forall [L]_{fst}^F$ alors l'arbre SLD standard de $P \cup \{:- L\}$ est fini.
 b) Si $P_{fst}^{CET} \models \forall [L]_{fst}^{EF}$ alors l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis.
 c) Si σ est une substitution des variables de L telle que $P_{fst}^{CET} \models \forall [L\sigma]_{fst}^{SF} \wedge \forall [L]_{fst}^F$ alors l'arbre SLD standard de $P \cup \{:- L\}$ est fini et il existe une réponse calculée θ pour $:- L$ telle que $\theta \leq \sigma$.

Démonstration :

- a) D'après le théorème VI.1, il existe un entier n tel que $T_{fst} \uparrow n \models \forall [L]_{fst}^F$ et la proposition IV.2 permet de conclure.
 b) Démonstration analogue.
 c) Le a) montre que l'arbre SLD standard de $:- L$ est fini.

Soit L_1 un sur-langage de L contenant une infinité de constantes et soit $\mathcal{U}_1$ le pré-modèle standard de la $CET(L_1)$. Comme dans le théorème VI.2, on montre qu'il existe un entier n tel que $T_{fst}^{\mathcal{U}_1} \uparrow n \models \forall [L\sigma]_{fst}^{SF}$. On a alors $P_{fst}^{Her(L_1)} \models \forall [L\sigma]_{fst}^{SF}$ et la proposition IV.2 montre alors qu'il existe une réponse calculée θ pour $:- L$ telle que $\theta \leq \sigma$.

□

On a, en particulier, le résultat suivant :

Corollaire VI.2 : Soit P un programme Prolog et A un atome clos de L .

- a) Si $P_{fst}^{CET} \models A \vee A'$ alors l'arbre SLD standard de $P \cup \{:- A\}$ est fini.
- b) Si $P_{fst}^{CET} \models A'$ alors $A \in FF_{st}$.
- c) Si $P_{fst}^{CET} \models A$ alors $A \in SS_{fst}$.

Démonstration :

- a) En effet, $\forall [A]_{fst}^F = A \vee A'$ et l'on peut appliquer la proposition précédente.
- b) En effet, $\forall [A]_{fst}^{EF} = A'$.
- c) Si $P_{fst}^{CET} \models A$ alors $P_{fst}^{CET} \models \forall [A]_{fst}^{SF} \wedge \forall [A]_{fst}^F$, donc $A \in SS_{fst}$.

□

VI.1.3 La sémantique invariante finie

Théorème VI.4 : (Complétude de la sémantique opérationnelle invariante finie pour la traduction axiomatique invariante finie P_f^{CET}).

Soit P un programme Prolog et L une liste d'atomes de L .

- a) Si $P_f^{CET} \models \forall [L]_f^F$ alors tous les arbres SLD de $P \cup \{:- L\}$ sont finis.
- b) Si $P_f^{CET} \models \forall [L]_f^{EF}$ alors tous les arbres SLD de $P \cup \{:- L\}$ sont des arbres d'échecs finis.
- c) Si σ est une substitution des variables de L telle que $P_f^{CET} \models \forall [L\sigma]_f^{SF} \wedge \forall [L]_f^F$ alors tous les arbres SLD de $P \cup \{:- L\}$ sont finis et il existe une réponse calculée θ pour $:- L$ telle que $\theta \leq \sigma$.

Démonstration :

- a) D'après le théorème VI.1, il existe un entier n tel que $T_f \uparrow n \models \forall [L]_f^F$ et la proposition IV.5 permet de conclure.
- b) Démonstration analogue.
- c) Le a) montre que tous les arbres SLD de $:-L$ sont finis.

Soit L_1 un sur-langage de L contenant une infinité de constantes et soit $\mathcal{U}_1$ le pré-modèle standard de la $CET(L_1)$. Comme dans le théorème VI.2, on montre qu'il existe un entier n tel que $T_f^{\mathcal{U}_1} \uparrow n \models \forall [L\sigma]_f^{SF}$. On a alors $P_f^{Her(L_1)} \models \forall [L\sigma]_f^{SF}$ et la proposition IV.5 montre alors qu'il existe une réponse calculée θ pour $:- L$ telle que $\theta \leq \sigma$.

□

On a, en particulier, le résultat suivant :

Corollaire VI.3 : Soit P un programme Prolog et A un atome clos de $\mathcal{L}$.

- a) Si $P_f^{\text{CET}} \models A \vee A'$ alors tous les arbres SLD de $P \cup \{:- A\}$ sont finis.
- b) Si $P_f^{\text{CET}} \models A'$ alors $A \in \text{FF}_f$.
- c) Si $P_f^{\text{CET}} \models A$ alors $A \in \text{SS}_f$.

Démonstration :

- a) En effet, $\forall [A]_f^F = A \vee A'$ et l'on peut appliquer la proposition précédente.
- b) En effet, $\forall [A]_f^{\text{EF}} = A'$.
- c) Si $P_f^{\text{CET}} \models A$ alors $P_f^{\text{CET}} \models \forall [A]_f^{\text{SF}} \wedge \forall [A]_f^F$ donc $A \in \text{SS}_f$.

□

Les résultats de complétude obtenus dans ce paragraphe sont tout à fait satisfaisants, au moins en ce qui concerne les sémantiques finies. Pour la sémantique standard, les conditions $\mathbf{H}_3$ et $\mathbf{H}_3'$ (toujours aussi fortes qu'au paragraphe III!), entachent la généralité de ces résultats.

Nous aurions pu énoncer (et démontrer) des résultats analogues pour les traductions axiomatiques P_{SSId} et P_θ . Mais la sémantique P_θ ne présente pas un grand intérêt et la complétude de la sémantique opérationnelle d'un interpréteur idéal pour la traduction $P_{\text{SSId}}^{\text{CET}}$ a déjà été démontrée directement dans [Ble 90, Gil 89].

Dans tous les cas, on peut constater que le cadre de la Théorie Equationnelle de Clark est beaucoup mieux adapté à ces sémantiques axiomatiques que celui des modèles de Herbrand. En particulier, il n'est plus nécessaire de faire d'hypothèses supplémentaires sur la continuité des opérateurs associés ni sur le langage sous-jacent comme c'était le cas dans les théorèmes III.5, IV.4 et IV.8 et les propositions III.3, IV.2 et IV.5..

Il reste, pour obtenir une caractérisation axiomatique correcte et complète, à prouver la *correction* des différentes sémantiques opérationnelles étudiées par rapport aux traductions axiomatiques associées lorsqu'on les considère dans le cadre de la CET.

VI.2. Correction et complétude dans le cadre de la CET($\mathcal{L}_k$).

Si l'on suppose que le langage sous-jacent des programmes Prolog considérés est un langage de Kunen $\mathcal{L}_k$, le théorème V.5 permet d'établir immédiatement des résultats de correction. Comme par ailleurs, les résultats de complétude énoncés plus haut sont indépendants du langage sous-jacent au programme, nous établissons dans ce paragraphe que les traductions axiomatiques standard¹, standard finie et invariante finie sont des caractérisations exactes des sémantiques opérationnelles correspondantes dans le cadre logique de la CET($\mathcal{L}_k$).

¹Avec une restriction importante dans ce cas.

VI.2.1 La sémantique standard

Théorème VI.5 : *(Correction et complétude de la sémantique opérationnelle standard pour la traduction axiomatique standard pour un langage de Kunen).*

Soit P un programme Prolog et L une liste d'atomes de $\mathcal{L}_k$.

- a) $P_{st}^{CET} \models \forall [L]_{st}^F$ ssi l'arbre SLD standard de $P \cup \{:- L\}$ est fini.
- b) $P_{st}^{CET} \models \forall [L]_{st}^{EF}$ ssi l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis.
- c) Supposons de plus que P vérifie la condition H_3 .
 - c₁) - Si σ est une réponse calculée standard de $:- L$, alors $P_{st}^{CET} \models \forall [L\sigma]_{st}^S$.
 - c₂) - Si σ est une substitution des variables de L telle que $P_{st}^{CET} \models \forall [L\sigma]_{st}^S$, alors il existe une réponse calculée standard θ pour $:- L$ telle que $\theta \leq \sigma$.

Démonstration : Il suffit bien sur de prouver les parties SI des équivalences a) et b) et le point c₁), puisque les autres points concernant la complétude ont été énoncés au paragraphe précédent.

- a) En effet, si l'arbre SLD standard de $P \cup \{:- L\}$ est fini de profondeur standard n alors, $T_{st} \uparrow n \models \forall [L]_{st}^F$ (corollaire III.2) et le théorème V.5 permet de conclure.
- b) De même, si l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis de profondeur standard n alors, $T_{st} \uparrow n \models \forall [L]_{st}^{EF}$ et le théorème V.5 permet de conclure.
- c₁) Si l'arbre SLD standard de $P \cup \{:- L\}$ est de profondeur standard n , alors pour toute substitution close τ , l'arbre SLD standard de $P \cup \{:- L\sigma\tau\}$ est de profondeur standard inférieure ou égale à n . D'après le corollaire III.2, $T_{st} \uparrow n \models \forall [L\sigma]_{st}^S$ et donc $P_{st}^{CET} \models \forall [L\sigma]_{st}^S$ d'après le théorème V.5.

□

Et, en particulier :

Corollaire VI.4 : Soit P un programme Prolog et A un atome clos de $\mathcal{L}$.

- a) $P_{st}^{CET} \models A''$ ssi l'arbre SLD standard de $P \cup \{:- A\}$ est fini (c.-à-d. $A \in SS_{fst} \cup FF_{st}$).
- b) $P_{st}^{CET} \models A'$ ssi $A \in FF_{st}$.
- c) Si de plus, P vérifie la condition H_3 , alors $P_{st}^{CET} \models A$ ssi $A \in SS_{st}$.

Démonstration :

- a) Si l'arbre SLD standard de $P \cup \{:- A\}$ est fini, alors $P_{st}^{CET} \models \forall [A]_{st}^F$ donc, il existe un entier n tel que $T_{st} \uparrow n \models \forall [A]_{st}^F$ et comme $\forall [A]_{st}^F = A' \vee (A \wedge A'')$ et que si $A' \in T_{st} \uparrow n$ alors $A'' \in T_{st} \uparrow n$ (proposition III.1), on en déduit que $P_{st}^{CET} \models A''$.
- b) Immédiat puisque $\forall [L]_{st}^{EF} = A'$.
- c) Si $A \in SS_{st}$, il existe un entier n tel que $T_{st} \uparrow n \models A$ donc $P_{st}^{CET} \models A$.

VI.2.2 La sémantique standard finie

Théorème VI.6 : *(Correction et complétude de la sémantique opérationnelle standard finie pour la traduction axiomatique standard finie pour un langage de Kunen).*

Soit P un programme Prolog et L une liste d'atomes de $\mathcal{L}$.

- a) $P_{fst}^{CET} \models \forall [L]_{fst}^F$ ssi l'arbre SLD standard de $P \cup \{:- L\}$ est fini.
- b) $P_{fst}^{CET} \models \forall [L]_{fst}^{EF}$ ssi l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis.
- c₁) Si σ est une réponse calculée du but $:- L$ et si l'arbre SLD standard de $P \cup \{:- L\}$ est fini alors $P_{fst}^{CET} \models \forall [L\sigma]_{fst}^{SF} \wedge \forall [L]_{fst}^F$.
- c₂) Si σ est une substitution des variables de L telle que $P_{fst}^{CET} \models \forall [L\sigma]_{fst}^{SF} \wedge \forall [L]_{fst}^F$ alors l'arbre SLD standard du but $:- L$ est fini et il existe une réponse calculée θ pour $:- L$ telle que $\theta \leq \sigma$.

Démonstration : Il suffit de prouver les parties SI et le point c₁).

- a) En effet, si l'arbre SLD standard de $P \cup \{:- L\}$ est fini de profondeur n , le corollaire IV.4 montre que $T_{fst} \uparrow n \models \forall [L]_{fst}^F$ et le théorème V.5 permet de conclure.
- b) De même, si l'arbre SLD standard de $P \cup \{:- L\}$ est un arbre d'échecs finis de profondeur n alors, $T_{fst} \uparrow n \models \forall [L]_{fst}^{EF}$.
- c₁) Si l'arbre SLD standard de $P \cup \{:- L\}$ est fini de profondeur n , alors pour toute substitution close τ , l'arbre SLD standard de $P \cup \{:- L\sigma\tau\}$ est de profondeur inférieure ou égale à n . D'après le corollaire IV.4, $T_{fst} \uparrow n \models \forall [L\sigma]_{fst}^S$ et donc $P_{fst}^{CET} \models \forall [L\sigma]_{fst}^{SF}$.

□

Et, en particulier :

Corollaire VI.5: Soit P un programme Prolog et A un atome clos de $\mathcal{L}$.

- a) $P_{fst}^{CET} \models A \vee A'$ ssi l'arbre SLD standard de $P \cup \{:- A\}$ est fini.
- b) $P_{fst}^{CET} \models A'$ ssi $A \in FF_{fst}$.
- c) $P_{fst}^{CET} \models A$ ssi $A \in SS_{fst}$.

Démonstration :

- a) Si l'arbre SLD standard de $:- A$ est fini, $P_{fst}^{CET} \models \forall [A]_{fst}^F$ et $\forall [A]_{fst}^F = A \vee A'$.
- b) De même, $\forall [A]_{fst}^{EF} = A'$.
- c) De même, $\forall [A]_{fst}^{SF} = A$.

□

VI. 2.3 La sémantique invariante finie

Théorème VI.7 : *(Correction et complétude de la sémantique opérationnelle invariante finie pour la traduction axiomatique invariante finie pour un langage de Kunen).*

Soit P un programme Prolog et L une liste d'atomes de $\mathcal{L}$.

- a) $P_f^{CET} \models \forall [L]_f^F$ ssi tous les arbres SLD de $P \cup \{:- L\}$ sont finis.
- b) $P_f^{CET} \models \forall [L]_f^{EF}$ ssi tous les arbres SLD de $P \cup \{:- L\}$ sont des arbres d'échecs finis.
- c1) Si σ est une réponse calculée du but $:- L$ et si tous les arbres SLD de $P \cup \{:- L\}$ sont finis alors $P_f^{CET} \models \forall [L\sigma]_f^{SF} \wedge \forall [L]_f^F$.
- c2) Si σ est une substitution des variables de L telle que $P_f^{CET} \models \forall [L\sigma]_f^{SF} \wedge \forall [L]_f^F$ alors tous les arbres SLD de $P \cup \{:- L\}$ sont finis et il existe une réponse calculée θ pour le but $:- L$ telle que $\theta \leq \sigma$.

Démonstration :

- a) En effet, si $:- L$ est de profondeur maximale n , $T_f \uparrow n \models \forall [L]_f^F$ (corollaire IV.4) et le théorème V.5 permet de conclure.
- b) De même, si tous les arbres SLD de $P \cup \{:- L\}$ sont des arbres d'échecs finis de profondeur maximale inférieure ou égale à n alors, $T_f \uparrow n \models \forall [L]_f^{EF}$.
- c) Si $P \cup \{:- L\}$ est de profondeur maximale n , alors pour toute substitution close τ , le but $P \cup \{:- L\sigma\tau\}$ est de profondeur inférieure ou égale à n donc d'après le corollaire IV.4, $T_f \uparrow n \models \forall [L\sigma]_f^S$ et donc $P_f^{CET} \models \forall [L\sigma]_f^{SF}$.

□

Et, en particulier :

Corollaire VI.6 : Soit P un programme Prolog et A un atome clos de $\mathcal{L}$.

- a) $P_f^{CET} \models A \vee A'$ ssi tous les arbres SLD de $:- L$ sont finis.
- b) $P_f^{CET} \models A'$ ssi $A \in FF_f$.
- c) $P_f^{CET} \models A$ ssi $A \in SS_f$.

Démonstration :

- a) Si tous les arbres SLD de $:- L$ sont finis, $P_f^{CET} \models \forall [A]_f^F$ et $\forall [A]_f^F = A \vee A'$.
- b) De même, $\forall [A]_f^{EF} = A'$.
- c1) De même, $\forall [A]_f^{SF} = A$.

□

En conclusion, nous avons montré que les traductions axiomatiques standard, standard finie et invariante finie sont des caractérisations axiomatiques correctes et complètes des sémantiques opérationnelles de même nom dans le cadre de la $CET(\mathcal{L}_k)$ (sous réserve pour la sémantique standard que les programmes vérifient l'hypothèse H_3^1 ou H_3).

VI.3. Correction (et complétude) dans le cadre de la $CET(\mathcal{L})$.

VI.3.1 Introduction

Nous avons déjà indiqué qu'il est naturel de considérer les langages de Kunen dans la mesure où un utilisateur d'une base de données ou d'un programme Prolog doit pouvoir utiliser n'importe quel symbole pour formuler ses requêtes. Néanmoins, les résultats de correction classiques (de la résolution SLDNF par rapport au complété de Clark [Cla 78], à la traduction dédoublée [Ble 90, Gil 89]) ont été prouvés sans hypothèse supplémentaire sur le langage sous-jacent.

Nous montrons dans ce paragraphe qu'il en est de même pour les sémantiques finies.

Nous avons d'abord tenté de définir une classe générale $\mathcal{P}$ de programmes généraux positifs (contenant ceux que l'on considère ici) pour lesquels on aurait un tel résultat de correction (au moins pour certaines classes de formules $\mathcal{F}_P$). Plus précisément, nous souhaitions un résultat du type :

si $P \in \mathcal{P}$, si $F \in \mathcal{F}_P$ et si n est un entier alors,

$$T_P \uparrow n \models F \text{ implique } P^{CET} \models F.$$

Ce problème semble assez difficile et nous nous sommes rabattus sur des preuves adaptées à chaque traduction axiomatique finie¹.

Il semble probable que ces résultats de correction soient aussi vrais pour la sémantique standard mais nous n'avons pas tenté de le prouver en raison à la fois de la difficulté de ce type de démonstration et du fait que les résultats sont de tout façon partiels (nécessité de l'hypothèse H_3^1 ou H_3) et donc moins intéressants pour cette sémantique.

Les démonstrations qui suivent étant assez longues et techniques, nous utilisons systématiquement les notations compactes définies au § II.2.

VI.3.2. La sémantique standard finie

On applique le lemme II.1 dans les cas particuliers suivants :

Lemme VI.1 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P où p est un prédicat n -aire et soit s un n -uplet de termes.

a) Si les n -uplets t et s ne sont pas unifiables alors,

$$CET(\mathcal{L}) \models \forall C^{EF}(s), \quad CET(\mathcal{L}) \models \forall C^F(s) \quad \text{et} \quad CET(\mathcal{L}) \models \neg \exists C^S(s)$$

¹sans renoncer pour autant à essayer de trouver une preuve plus générale (et donc plus élégante).

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$\begin{aligned} \text{CET}(\mathbf{L}) &\models \forall C^{\text{EF}}(s) \Leftrightarrow \forall [\mathbf{L}\theta]_{\text{fst}}^{\text{EF}}, \\ \text{CET}(\mathbf{L}) &\models \forall C^{\text{F}}(s) \Leftrightarrow \forall [\mathbf{L}\theta]_{\text{fst}}^{\text{F}} \text{ et} \\ \text{CET}(\mathbf{L}) &\models \exists C^{\text{S}}(s) \Leftrightarrow \exists [\mathbf{L}\theta]_{\text{fst}}^{\text{SF}}. \end{aligned}$$

Les deux lemmes suivants sont intuitivement évidents (pour une liste d'atomes clos $\mathbf{L}$) si l'on donne :

à $[\mathbf{L}]_{\text{fst}}^{\text{F}}$ le sens : l'arbre SLD standard de $P \cup \{:-\mathbf{L}\}$ est fini

à $[\mathbf{L}]_{\text{fst}}^{\text{EF}}$ le sens : l'arbre SLD standard de $P \cup \{:-\mathbf{L}\}$ est un arbre d'échecs finis

à $[\mathbf{L}]_{\text{fst}}^{\text{SF}}$ le sens : l'arbre SLD standard de $P \cup \{:-\mathbf{L}\}$ est fini et contient la clause vide.

Le fait qu'ils soient vrais contribue à montrer que cette intuition est fondée.

Lemme VI.2 : Soient $\mathbf{L}$ et $\mathbf{M}$ deux listes d'atomes. On note $\mathbf{L},\mathbf{M}$ la juxtaposition de ces deux listes. Alors,

a) $\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Leftrightarrow [\mathbf{L}]_{\text{fst}}^{\text{EF}} \vee ([\mathbf{L}]_{\text{fst}}^{\text{SF}} \wedge [\mathbf{M}]_{\text{fst}}^{\text{F}}))$ est universellement valide

b) $\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Rightarrow \forall ([\mathbf{L}]_{\text{fst}}^{\text{F}}))$ est universellement valide.

Démonstration :

a) Par récurrence sur la longueur de la liste $\mathbf{L}$.

- C'est immédiat si $\mathbf{L}$ est la liste vide puisque $[\]_{\text{fst}}^{\text{EF}} = \text{Faux}$ et $[\]_{\text{fst}}^{\text{SF}} = \text{Vrai}$.

- Supposons le résultat vrai pour toute liste $\mathbf{L}$ de longueur $\leq n$ et soit $\mathbf{L}$ une liste de longueur $n+1$. On peut écrire $\mathbf{L} = [\mathbf{A}], \mathbf{N}$ où $\mathbf{A}$ est un atome.

$$[\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} = \mathbf{A}' \vee (\mathbf{A} \wedge [\mathbf{N},\mathbf{M}]_{\text{fst}}^{\text{F}}).$$

$$\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Leftrightarrow \mathbf{A}' \vee (\mathbf{A} \wedge ([\mathbf{N}]_{\text{fst}}^{\text{EF}} \vee ([\mathbf{N}]_{\text{fst}}^{\text{SF}} \wedge [\mathbf{M}]_{\text{fst}}^{\text{F}}))))$$

par hypothèse de récurrence.

$$\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Leftrightarrow \mathbf{A}' \vee (\mathbf{A} \wedge [\mathbf{N}]_{\text{fst}}^{\text{EF}}) \vee (\mathbf{A} \wedge [\mathbf{N}]_{\text{fst}}^{\text{SF}} \wedge [\mathbf{M}]_{\text{fst}}^{\text{F}}))$$

soit encore :

$$\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Leftrightarrow [\mathbf{L}]_{\text{fst}}^{\text{EF}} \vee ([\mathbf{L}]_{\text{fst}}^{\text{SF}} \wedge [\mathbf{M}]_{\text{fst}}^{\text{F}})).$$

b) On déduit du a) que :

$$\forall ([\mathbf{L},\mathbf{M}]_{\text{fst}}^{\text{F}} \Rightarrow \forall ([\mathbf{L}]_{\text{fst}}^{\text{EF}} \vee [\mathbf{L}]_{\text{fst}}^{\text{SF}}))$$

est universellement valide ce qui fournit la conclusion.

□

Lemme VI.3 : Soit $\mathfrak{M}$ un modèle de la $\text{CET}(\mathbf{L})$ dans lequel pour tout atome $\mathbf{A}$ on a :

$\mathfrak{M} \models \neg \exists (\mathbf{A} \wedge \mathbf{A}')$. Alors :

a) pour toute liste d'atomes $\mathbf{L}$:

$$\mathfrak{M} \models \forall [\mathbf{L}]_{\text{fst}}^{\text{F}} \text{ implique } \mathfrak{M} \models \forall ([\mathbf{L}]_{\text{fst}}^{\text{EF}} \Leftrightarrow \neg [\mathbf{L}]_{\text{fst}}^{\text{SF}}).$$

b) Si C est une règle de la forme $p(t) :- L$ où p est un prédicat n -aire, si s est un n -uplet de termes et si $\mathcal{M} \models \forall C^F(s)$ alors $\mathcal{M} \models \forall (C^{EF}(s) \Leftrightarrow \neg C^{SF}(s))$.

Démonstration :

a) Par récurrence sur la longueur de L .

- Si L est la liste vide, c'est immédiat puisque $[]_{fst}^{EF} = \text{Faux}$ et $[]_{fst}^{SF} = \text{Vrai}$.

- Soit $L = [A_1, \dots, A_p]$ avec $p \geq 1$ et supposons le résultat démontré pour toute liste de longueur $< p$.

Soit $\{x_1, \dots, x_q\}$ les variables de L et soit $[x \backslash a]$ une assignation des variables de L dans $\mathcal{M}$.

Si $\mathcal{M} \models A_1[x \backslash a]$ alors $\mathcal{M} \models [L]_{fst}^{EF}[x \backslash a]$ et d'après l'hypothèse sur le modèle $\mathcal{M}$, on a $\mathcal{M} \models \neg A_1[x \backslash a]$ et donc $\mathcal{M} \models \neg [L]_{fst}^{SF}[x \backslash a]$.

Sinon, comme $\mathcal{M} \models \forall [L]_{fst}^F$, on a :

$$\mathcal{M} \models A_1[x \backslash a] \text{ et } \mathcal{M} \models \neg A_1[x \backslash a].$$

On en déduit que :

$$\mathcal{M} \models ([L]_{fst}^{EF} \Leftrightarrow [A_2, \dots, A_p]_{fst}^{EF})[x \backslash a] \text{ et}$$

$$\mathcal{M} \models ([L]_{fst}^{SF} \Leftrightarrow [A_2, \dots, A_p]_{fst}^{SF})[x \backslash a]$$

ce qui permet de conclure d'après l'hypothèse de récurrence.

b) Soit $\{x_1, \dots, x_q\}$ l'ensemble des variables figurant dans les termes s_i, t_i et dans la liste L et soit $[x \backslash a]$ une assignation de ces variables dans $\mathcal{M}$.

Si $\mathcal{M} \models (s \neq t)[x \backslash a]$ alors :

$$\mathcal{M} \models C^{EF}(s)[x \backslash a]$$

et

$$\mathcal{M} \models \neg C^{SF}(s)[x \backslash a].$$

Sinon :

$$\mathcal{M} \models C^{EF}(s)[x \backslash a] \Leftrightarrow [L]_{fst}^{EF}[x \backslash a]$$

et

$$\mathcal{M} \models C^{SF}(s)[x \backslash a] \Leftrightarrow [L]_{fst}^{SF}[x \backslash a]$$

et la partie a) permet de conclure.

La démonstration de la proposition suivante est assez longue. Peut-être en existe-t-il de plus simples ? Elle permet de montrer un résultat de correction concernant la finitude.

Proposition VI.1 : Soit P un programme Prolog, $\mathcal{M}'$ un pré-modèle de la CET(L) et soit $\mathcal{M} = \text{pppf}(T_{fst}^{\mathcal{M}'})$. Soit L une liste d'atomes. Si l'arbre SLD-standard de $P \cup \{:-L\}$ est fini, alors $\mathcal{M} \models \forall [L]_{fst}^F$.

Démonstration : On démontre que si $\mathcal{M} \models \neg \forall [L]_{fst}^F$, alors le but $:-L$ admet au moins un but dérivé (non vide) $:-L'$ pour lequel $\mathcal{M} \models \neg \forall [L']_{fst}^F$, ce qui prouve, par induction, que l'arbre SLD standard de $:-L$ est infini.

Soit $L = [A_1, \dots, A_p]$, $M = [A_2, \dots, A_p]$ et $A_1 = p(s)$ où $A_1, \dots, A_p$ sont des atomes, $s = (s_1, \dots, s_n)$ et p est un symbole de prédicat n -aire.

Par définition, $\forall[L]_{fst}^F = \forall(A_1' \vee (A_1 \wedge [M]_{fst}^F))$.

Soient $C_1, \dots, C_q$ les q règles de P de prédicat de tête p .

$$\forall(C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s) \Rightarrow A_1')$$

et

$$\forall(C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \Rightarrow A_1)$$

sont des instances d'axiomes de P_{fst}^1 et donc des conséquences logiques de P_{fst} . Donc :

$$\forall((C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)) \vee (C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \wedge [M]_{fst}^F) \Rightarrow [L]_{fst}^F)$$

est une conséquence logique de P_{fst} .

Soit $i \in \{1, \dots, q\}$ et soit $C_i = p(t) :- L_i$.

- Si A_1 ne s'unifie pas avec la tête de C_i , alors :

$$CET(L) \models \forall(s \neq t \vee [L_i]_{fst}^F) \text{ et en particulier } \mathcal{M} \models C_i^F(s).$$

(Lemme VI.1)

- Sinon, soit θ_i l'upg des n -uplets s et t .

D'après le lemme VI.1, on a :

$$CET(L) \models \forall C_i^F(s) \Leftrightarrow \forall [L_i \theta_i]_{fst}^F.$$

Nous pouvons remarquer que le but $:- L_i \theta_i, M \theta_i$ est le but dérivé de $:- L$ obtenu en sélectionnant le premier atome de L et en choisissant la i -ème règle de P de prédicat de tête p .

Supposons que pour toutes ces règles C_i , on ait $\mathcal{M} \models \forall [L_i \theta_i, M \theta_i]_{fst}^F$. On a alors *a fortiori*, $\mathcal{M} \models \forall [L_i \theta_i]_{fst}^F$ (lemme VI.2).

Pour tout $i \in \{1, \dots, q\}$, on a donc : $\mathcal{M} \models \forall C_i^F(s)$.

On en déduit que :

$$\mathcal{M} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s)), \text{ et}$$

$$\mathcal{M} \models \forall((C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)) \vee (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \wedge [M]_{fst}^F \Rightarrow [L]_{fst}^F)$$

puisque $\mathcal{M}$ est un modèle de P_{fst} .

$$\mathcal{M} \models \forall(C^{EF}(s) \Leftrightarrow \neg C^{SF}(s))$$

d'après le lemme VI.3. En effet, il existe un ordinal α tel que $T_{fst}^{\mathcal{M}} \uparrow \alpha = \text{pppf}(T_{fst}^{\mathcal{M}'})$ et la proposition IV.1 montre que pour tout ordinal α et tout atome A , on a :

¹Dont la définition générale se trouve au § IV.1.1.

$$T_{fst}^{\mathcal{M}} \uparrow \alpha \models \neg \exists (A \wedge A').$$

Donc :

$$\mathcal{M} \models \forall (C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s) \Leftrightarrow \neg (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s))).$$

Soit $G = C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)$.

On a donc :

$$\mathcal{M} \models \forall (G \vee [M]_{fst}^F \Rightarrow [L]_{fst}^F).$$

Soit $\{x_1, \dots, x_q\}$ les variables de L .

Montrons que pour toute assignation $[x \backslash a]$ de $\{x_1, \dots, x_q\}$ dans $\mathcal{M}$, on a :

$$\mathcal{M} \models (G \vee [M]_{fst}^F)[x \backslash a].$$

Ceci prouvera que $\mathcal{M} \models \forall [L]_{fst}^F$ ce qui est contradictoire.

On en déduira donc qu'il existe au moins une règle C_i de P avec la tête de laquelle A_1 s'unifie et pour laquelle $\mathcal{M} \models \neg \forall [L_i \theta_i, M \theta_i]_{fst}^F$; ce que l'on devait démontrer.

Soit $[x \backslash a]$ une assignation des variables de L dans $\mathcal{M}$.

Supposons que $\mathcal{M} \models \neg [M]_{fst}^F[x \backslash a]$.

Soit $C_i = p(t) :- L_i$, une règle de P et $\{y_1, \dots, y_k\}$ les variables des termes t_i et de la liste L_i .

On peut supposer que les ensembles $\{x_1, \dots, x_q\}$ et $\{y_1, \dots, y_k\}$ sont disjoints.

- Si $\mathcal{M} \models \forall y_1 \dots y_k (s \neq t)[x \backslash a]$ alors

$$\mathcal{M} \models C_i^{EF}(s)[x \backslash a].$$

- Sinon, $\mathcal{M} \models \exists y_1 \dots y_k (s = t)[x \backslash a]$. Les n-uplets s et t sont donc unifiables (d'upg θ_i).

Soit $[y \backslash b]$ une assignation dans $\mathcal{M}$ de $\{y_1, \dots, y_k\}$ telle que :

$$\mathcal{M} \models (s = t)[x \backslash a][y \backslash b].$$

Comme on a supposé que $\mathcal{M} \models \forall [L_i \theta_i, M \theta_i]_{fst}^F$, on a en particulier :

$$\mathcal{M} \models [L_i \theta_i, M \theta_i]_{fst}^F[x \backslash a][y \backslash b]$$

soit, d'après le lemme VI.2 :

$$\mathcal{M} \models ([L_i \theta_i]_{fst}^{EF} \vee ([L_i \theta_i]^S \wedge [M \theta_i]_{fst}^F))[x \backslash a][y \backslash b].$$

Le lemme II.2 montre que :

$$\mathcal{M} \models [M \theta_i]_{fst}^F[x \backslash a][y \backslash b] \Leftrightarrow [M]_{fst}^F[x \backslash a][y \backslash b]$$

soit :

$$\mathcal{M} \models [M \theta_i]_{fst}^F[x \backslash a][y \backslash b] \Leftrightarrow [M]_{fst}^F[x \backslash a]$$

puisque les variables de M appartiennent à $\{x_1, \dots, x_q\}$.

Par hypothèse,

$$\mathcal{M} \models \neg [M]_{fst}^F[x \backslash a]$$

donc

$$\mathcal{M} \models \neg[M\theta_i]_{fst}^F[x \setminus a][y \setminus b]$$

et donc

$$\mathcal{M} \models [L_i\theta_i]_{fst}^{EF}[x \setminus a][y \setminus b].$$

On en déduit que :

$$\mathcal{M} \models C_i^{EF}(s)[x \setminus a]$$

et comme i est quelconque :

$$\mathcal{M} \models G[x \setminus a].$$

Finalement,

$$\mathcal{M} \models (G \vee M)[x \setminus a]$$

ce qu'il fallait démontrer. □

Proposition VI.2 : Soit P un programme Prolog et $:-L$ un but. Si l'arbre SLD standard de $P \cup \{:-L\}$ est fini alors :

$$P_{fst}^{CET} \models \forall[L]_{fst}^F.$$

Démonstration : Soit $\mathcal{M}$ un modèle de P_{fst}^{CET} de pré-modèle $\mathcal{M}'$ et soit $\mathcal{N} = \text{pppf}(T_{fst}^{\mathcal{M}'}).$ On a $\mathcal{N} \models \forall[L]_{fst}^F$ d'après la proposition précédente, et comme $\forall[L]_{fst}^F$ est une formule positive, on a aussi $\mathcal{M} \models \forall[L]_{fst}^F$ puisque $\mathcal{M}$ est une expansion de $\mathcal{N}$. □

Lemme VI.4 : Soit P un programme Prolog et $L = [A_1, \dots, A_p]$ une liste d'atomes telle que le but $:-L$ ait un arbre SLD standard fini et $A_1 = p(s)$. Soient $C_1, \dots, C_q$ les règles de P de prédicat de tête p . Alors, $P_{fst}^{CET} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s)).$

Démonstration : En effet, soit $C_i = p(t) :- L_i$. Le lemme VI.1 montre que si les n -uplets s et t ne sont pas unifiables alors :

$$P_{fst}^{CET} \models \forall C_i^F(s).$$

Sinon, soit θ_i leur upg. Le même lemme montre que :

$$P_{fst}^{CET} \models \forall C_i^F(s) \Leftrightarrow \forall[L\theta_i]_{fst}^F$$

et la proposition précédent permet alors de conclure.

Proposition VI.3 : Soit P un programme Prolog. Si le but $:-L$ a un arbre SLD standard fini et réussit avec la réponse calculée θ , alors $P_{fst}^{CET} \models \forall[L\theta]_{fst}^{SF}.$

Démonstration : Par récurrence sur la profondeur de $:-L$.

Si la profondeur est nulle, L est la liste vide et le résultat est évident.

Sinon, soit $A_1 = p(s)$ le premier atome de L , M la liste telle que $L = [A_1], M$ et soient $C_1, \dots, C_q$ les règles de P de prédicat de tête p .

On a :

$$P_{fst}^{CET} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \Rightarrow A_1)$$

mais également, d'après le lemme précédent :

$$P_{fst}^{CET} \models \forall (C_1^F(s) \wedge \dots \wedge C_q^F(s)).$$

On en déduit que :

$$P_{fst}^{CET} \models \forall (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s) \Rightarrow A_1)$$

et

$$P_{fst}^{CET} \models \forall ((C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \wedge [M]_{fst}^{SF} \Rightarrow [L]_{fst}^{SF}).$$

Soit $C_i = p(t) :- L_i$ une règle qui permet d'obtenir la réponse calculée θ . On a :

$$P_{fst}^{CET} \models \forall (C_i^{SF}(s) \wedge [M]_{fst}^{SF} \Rightarrow [L]_{fst}^{SF}).$$

Soit θ_i l'upg des n-uplets s et t et σ une réponse calculée du but $:- L_i\theta_i, M\theta_i$ telle que $L\theta = L\theta_i\sigma$.

Soient $\{y_1, \dots, y_k\}$ les variables figurant dans les termes t_i et la liste L_i et $\{x_1, \dots, x_m\}$ les variables figurant dans les termes s_i (on suppose que ces deux ensembles sont disjoints).

On a :

$$P_{fst}^{CET} \models \forall x_1 \dots x_m (\exists y_1 \dots y_k (s = t \wedge [L_i]_{fst}^{SF} \wedge [M]_{fst}^{SF}) \Rightarrow [L]_{fst}^{SF}).$$

Supposons qu'il existe un modèle $\mathcal{M}$ de P_{fst}^{CET} tel que $\mathcal{M} \models \neg \forall [L\theta]_{fst}^{SF}$ et soit $[x \backslash a]$ une assignation des variables x_i satisfaisant l'équation associée à θ et pour laquelle on a :

$$\mathcal{M} \models \neg [L\theta]_{fst}^{SF} [x \backslash a]$$

ou encore (d'après le lemme II.2)

$$\mathcal{M} \models \neg [L]_{fst}^{SF} [x \backslash a].$$

On a :

$$\mathcal{M} \models \neg \exists y_1 \dots y_k (s = t \wedge [L_i]_{fst}^{SF} \wedge [M]_{fst}^{SF}) [x \backslash a]$$

soit :

$$\mathcal{M} \models \neg \exists y_1 \dots y_k (s = t \wedge [L_i, M]_{fst}^{SF}) [x \backslash a]$$

soit encore :

$$\forall y_1 \dots y_k (s \neq t \vee \neg ([L_i, M]_{fst}^{SF})) [x \backslash a]$$

et d'après le lemme II.2 :

$$\forall y_1 \dots y_k \neg ([L_i\theta_i, M\theta_i]_{fst}^{SF}) [x \backslash a].$$

Mais d'autre part, par hypothèse de récurrence, on a :

$$\mathcal{M} \models \forall [L_i\theta_i\sigma, M\theta_i\sigma]_{fst}^{SF}$$

et en particulier :

$$\forall y_1 \dots y_k [L_i\theta_i\sigma, M\theta_i\sigma]_{fst}^{SF} [x \backslash a]$$

ce qui est contradictoire.

On en déduit que pour tout modèle $\mathcal{M}$ de P_{fst}^{CET} , on a :

$$\mathcal{M} \models \forall [L\theta]_{fst}^{SF}$$

soit encore :

$$P_{fst}^{CET} \models \forall [L\theta]_{fst}^{SF}.$$

□

Proposition VI.4 : Soit P un programme Prolog. Si l'arbre SLD standard du but $:- L$ est un arbre d'échecs finis, alors $P_{fst}^{CET} \models \forall [L]_{fst}^{EF}$.

Démonstration : soit Q le programme $P \cup \{A :- A\}$ où A est un symbole de prédicat ne figurant pas dans P et soit M le but $:- L, A$. Comme l'arbre SLD standard de ce but est fini, on a :

$$Q_{fst}^{CET} \models \forall [L, A]_{fst}^F.$$

Mais on a également :

$$\forall [L, A]_{fst}^F \Leftrightarrow \forall ([L]_{fst}^{EF} \vee ([L]_{fst}^{SF} \wedge [A]_{fst}^F))$$

d'où :

$$Q_{fst}^{CET} \models \forall ([L]_{fst}^{EF} \vee ([L]_{fst}^{SF} \wedge [A]_{fst}^F)).$$

Soit $\mathcal{M}$ un pré-modèle de la $CET(L)$. On vérifie aisément que pour tout ordinal α , $T_{fst}^{\mathcal{M}} \uparrow \alpha \models \neg [A]_{fst}^F$.

Donc, $pppf(T_{fst}^{\mathcal{M}}) \models \forall [L]_{fst}^{EF}$ et comme $\forall [L]_{fst}^{EF}$ est une formule positive, pour tout modèle $\mathcal{M}'$ de Q_{fst}^{CET} de pré-modèle $\mathcal{M}$, on a $\mathcal{M}' \models \forall [L]_{fst}^{EF}$. Donc $Q_{fst}^{CET} \models \forall [L]_{fst}^{EF}$.

Et comme A ne figure pas dans cette formule :

$$P_{fst}^{CET} \models \forall [L]_{fst}^{EF}.$$

□

Les trois propositions précédentes établissent la correction de la sémantique opérationnelle standard finie pour la sémantique axiomatique standard finie dans le cadre de la CET. Nous résumons ces résultats dans le théorème suivant :

Théorème VI.8 : Soit P un programme Prolog et $:- L$ un but dont l'arbre SLD standard est fini. Alors,

- a) $P_{fst}^{CET} \models \forall [L]_{fst}^F$.
- b) Si ce but échoue finiment, alors $P_{fst}^{CET} \models \forall [L]_{fst}^{EF}$.
- c) S'il réussit avec la réponse calculée θ , alors $P_{fst}^{CET} \models \forall [L\theta]_{fst}^{SF}$.

Les théorèmes VI.3 et VI.8 répondent à la question posée dans l'introduction :

"Les interpréteurs Prolog standard admettent une sémantique axiomatique : la traduction standard finie d'un programme P dans le cadre logique de la Théorie Equationnelle de Clark du langage sous-jacent à P ".

VI.3.2 La sémantique invariante finie

On appliquera comme au paragraphe précédent le lemme II.1 dans les cas particuliers suivants :

Lemme VI.5 : Soit $C = p(t) :- L$ une règle d'un programme Prolog P et soit s un n -uplet de termes.

- a) Si les n -uplets t et s ne sont pas unifiables alors,

$$CET(L) \models \forall C^{EF}(s), \quad CET(L) \models \forall C^F(s) \quad \text{et} \quad CET(L) \models \neg \exists C^S(s)$$

b) Si les n -uplets t et s sont unifiables avec l'upg θ alors,

$$\begin{aligned} \text{CET}(\mathbf{L}) &\models \forall \text{C}^{\text{EF}}(s) \Leftrightarrow \forall [\mathbf{L}\theta]_{\mathbf{f}}^{\text{EF}}, \\ \text{CET}(\mathbf{L}) &\models \forall \text{C}^{\text{F}}(s) \Leftrightarrow \forall [\mathbf{L}\theta]_{\mathbf{f}}^{\text{F}} \text{ et} \\ \text{CET}(\mathbf{L}) &\models \exists \text{C}^{\text{S}}(s) \Leftrightarrow \exists [\mathbf{L}\theta]_{\mathbf{f}}^{\text{SF}}. \end{aligned}$$

Lemme VI.6 : Soient $\mathbf{L}$ et $\mathbf{M}$ deux listes d'atomes. On note $\mathbf{L},\mathbf{M}$ la juxtaposition de ces deux listes. Alors,

- a) $\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Leftrightarrow [\mathbf{L}]_{\mathbf{f}}^{\text{EF}} \vee ([\mathbf{L}]_{\mathbf{f}}^{\text{SF}} \wedge [\mathbf{M}]_{\mathbf{f}}^{\text{F}}))$ est universellement valide
b) $\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Rightarrow \forall ([\mathbf{L}]_{\mathbf{f}}^{\text{F}}))$ est universellement valide.

Démonstration :

a) Par récurrence sur la longueur de la liste $\mathbf{L}$.

- C'est immédiat si $\mathbf{L}$ est la liste vide puisque $[\]_{\mathbf{f}}^{\text{EF}} = \text{Faux}$ et $[\]_{\mathbf{f}}^{\text{SF}} = \text{Vrai}$.

- Supposons le résultat vrai pour toute liste $\mathbf{L}$ de longueur $\leq n$ et soit $\mathbf{L}$ une liste de longueur $n+1$. On peut écrire $\mathbf{L} = [\mathbf{A}], \mathbf{N}$ où $\mathbf{A}$ est un atome.

$$[\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} = \mathbf{A}' \vee (\mathbf{A} \wedge [\mathbf{N},\mathbf{M}]_{\mathbf{f}}^{\text{F}}).$$

$$\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Leftrightarrow \mathbf{A}' \vee (\mathbf{A} \wedge ([\mathbf{N}]_{\mathbf{f}}^{\text{EF}} \vee ([\mathbf{N}]_{\mathbf{f}}^{\text{SF}} \wedge [\mathbf{M}]_{\mathbf{f}}^{\text{F}}))))$$

par hypothèse de récurrence.

$$\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Leftrightarrow \mathbf{A}' \vee (\mathbf{A} \wedge [\mathbf{N}]_{\mathbf{f}}^{\text{EF}}) \vee (\mathbf{A} \wedge [\mathbf{N}]_{\mathbf{f}}^{\text{SF}} \wedge [\mathbf{M}]_{\mathbf{f}}^{\text{F}})),$$

soit encore :

$$\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Leftrightarrow [\mathbf{L}]_{\mathbf{f}}^{\text{EF}} \vee ([\mathbf{L}]_{\mathbf{f}}^{\text{SF}} \wedge [\mathbf{M}]_{\mathbf{f}}^{\text{F}})).$$

b) On déduit du a) que $\forall ([\mathbf{L},\mathbf{M}]_{\mathbf{f}}^{\text{F}} \Rightarrow \forall ([\mathbf{L}]_{\mathbf{f}}^{\text{EF}} \vee [\mathbf{L}]_{\mathbf{f}}^{\text{SF}}))$ est universellement valide, ce qui fournit la conclusion.

□

Lemme VI.7 : Soient $\mathbf{L}$, $\mathbf{M}$ et $\mathbf{N}$ trois listes d'atomes. Alors,

- a) $\forall ([\mathbf{L},\mathbf{M},\mathbf{N}]_{\mathbf{f}}^{\text{F}} \Rightarrow \forall ([\mathbf{M}]_{\mathbf{f}}^{\text{F}}))$ est universellement valide.

Démonstration : Il suffit de remarquer que $\forall ([\mathbf{L},\mathbf{M},\mathbf{N}]_{\mathbf{f}}^{\text{F}} \Leftrightarrow [\mathbf{M},\mathbf{L},\mathbf{N}]_{\mathbf{f}}^{\text{F}})$ et d'appliquer le lemme précédent.

Lemme VI.8 : Soit $\mathfrak{M}$ un modèle de la $\text{CET}(\mathbf{L})$ dans lequel pour tout atome $\mathbf{A}$ on a : $\mathfrak{M} \models \neg \exists (\mathbf{A} \wedge \mathbf{A}')$. Alors :

a) pour toute liste d'atomes $\mathbf{L}$:

$$\mathfrak{M} \models \forall [\mathbf{L}]_{\mathbf{f}}^{\text{F}} \text{ implique } \mathfrak{M} \models \forall ([\mathbf{L}]_{\mathbf{f}}^{\text{EF}} \Leftrightarrow \neg [\mathbf{L}]_{\mathbf{f}}^{\text{SF}}).$$

b) Si C est une règle de la forme $p(t) :- L$ où p est un prédicat n -aire, si s est un n -uplet de termes et si $\mathcal{M} \models \forall C^F(s)$ alors $\mathcal{M} \models \forall (C^{EF}(s) \Leftrightarrow \neg C^{SF}(s))$.

Démonstration :

a) Par récurrence sur la longueur de L .

- Si L est la liste vide, c'est immédiat puisque $[]_f^{EF} = \text{Faux}$ et $[]_f^{SF} = \text{Vrai}$.

- Soit $L = [A_1, \dots, A_p]$ avec $p \geq 1$ et supposons le résultat démontré pour toute liste de longueur $< p$.

Soit $\{x_1, \dots, x_q\}$ les variables de L et soit $[x \backslash a]$ une assignation des variables de L dans $\mathcal{M}$.

Si $\mathcal{M} \models A_1[x \backslash a]$ alors $\mathcal{M} \models [L]_f^{EF}[x \backslash a]$ et d'après l'hypothèse sur le modèle $\mathcal{M}$, on a $\mathcal{M} \models \neg A_1[x \backslash a]$ et donc $\mathcal{M} \models \neg [L]_f^{SF}[x \backslash a]$.

Sinon, comme $\mathcal{M} \models \forall [L]_f^F$:

$$\mathcal{M} \models A_1[x \backslash a] \text{ et } \mathcal{M} \models \neg A_1[x \backslash a].$$

On en déduit que :

$$\mathcal{M} \models ([L]_f^{EF} \Leftrightarrow [A_2, \dots, A_p]_f^{EF})[x \backslash a]$$

et :

$$\mathcal{M} \models ([L]_f^{SF} \Leftrightarrow [A_2, \dots, A_p]_f^{SF})[x \backslash a]$$

ce qui permet de conclure d'après l'hypothèse de récurrence.

b) Soit $\{x_1, \dots, x_q\}$ l'ensemble des variables figurant dans les termes s_i , t_i et dans la liste L et soit $[x \backslash a]$ une assignation de ces variables dans $\mathcal{M}$.

Si $\mathcal{M} \models (s \neq t)[x \backslash a]$ alors :

$$\mathcal{M} \models C^{EF}(s)[x \backslash a]$$

et

$$\mathcal{M} \models \neg C^{SF}(s)[x \backslash a].$$

Sinon :

$$\mathcal{M} \models C^{EF}(s)[x \backslash a] \Leftrightarrow [L]_f^{EF}[x \backslash a]$$

et

$$\mathcal{M} \models C^{SF}(s)[x \backslash a] \Leftrightarrow [L]_f^{SF}[x \backslash a]$$

et la partie a) permet de conclure.

□

Proposition VI.5 : Soit P un programme Prolog, $\mathcal{M}'$ un pré-modèle de la $CET(L)$ et soit $\mathcal{M} = \text{pppf}(T_f^{\mathcal{M}'})$. Soit L une liste d'atomes. Si tous les arbres SLD de $P \cup \{:- L\}$ sont finis, alors $\mathcal{M} \models \forall [L]_f^F$.

Démonstration : On démontre que si $\mathcal{M} \models \neg \forall [L]_f^F$ alors le but $:- L$ admet au moins un but dérivé $:- L'$ pour lequel $\mathcal{M} \models \neg \forall [L']_f^F$, ce qui prouvera par induction qu'un arbre SLD de $:- L$ est infini.

Soit $L = [A_1, \dots, A_p]$.

Si $\mathcal{M} \models \neg \forall [L]_f^F$ alors il existe $k \in \{1, \dots, p\}$ tel que $\mathcal{M} \models \neg \forall (A_k \vee A'_k)$. Soit $A_k = p(s)$ où $s = (s_1, \dots, s_n)$ et p est un symbole de prédicat n -aire.

Soient $C_1, \dots, C_q$ les q règles de P de prédicat de tête p .

$$\forall (C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s) \Rightarrow A'_k)$$

et

$$\forall (C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \Rightarrow A_k)$$

sont des instances d'axiomes de P_f et donc des conséquences logiques de P_f^1 . Donc :

$$\forall ((C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)) \vee (C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)))) \Rightarrow (A_k \vee A'_k))$$

est une conséquence logique de P_f .

Soit $i \in \{1, \dots, q\}$ et soit $C_i = p(t) :- L_i$.

- Si A_k ne s'unifie pas avec la tête de C_i , alors $\text{CET}(\mathcal{L}) \models \forall C_i^F(s)$ d'après le lemme VI.5 et en particulier $\mathcal{M} \models \forall C_i^F(s)$.

- Sinon, soit θ_i l'upg des n -uplets s et t .

D'après le lemme VI.5, on a :

$$\text{CET}(\mathcal{L}) \models \forall C_i^F(s) \Leftrightarrow \forall [L_i \theta_i]_f^F.$$

Supposons que pour tous les buts L' dérivés de L , on ait : $\mathcal{M} \models \forall [L']_f^F$. Alors, d'après le lemme VI.7 on a aussi : $\mathcal{M} \models \forall [L_i \theta_i]_f^F$.

Pour tout $i \in \{1, \dots, q\}$, on a donc : $\mathcal{M} \models \forall C_i^F(s)$. On en déduit que :

$$\mathcal{M} \models \forall (C_1^F(s) \wedge \dots \wedge C_q^F(s)) \text{ et,}$$

$$\mathcal{M} \models \forall ((C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)) \vee (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s))) \Rightarrow (A_k \vee A'_k))$$

puisque $\mathcal{M}$ est un modèle de P_f .

$$\mathcal{M} \models \forall (C^{EF}(s) \Leftrightarrow \neg C^{SF}(s))$$

d'après le lemme VI.8. En effet, il existe un ordinal α tel que $T_f^{\mathcal{M}'} \uparrow \alpha = \text{pppf}(T_f^{\mathcal{M}'})$ et la proposition IV.3 montre que pour tout ordinal α et tout atome A , on a :

$$T_f^{\mathcal{M}'} \uparrow \alpha \models \neg \exists (A \wedge A').$$

Donc :

$$\mathcal{M} \models \forall (C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s) \Leftrightarrow \neg (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)))$$

et en particulier :

$$\mathcal{M} \models \forall ((C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s)) \vee (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)))$$

donc :

¹Dont la définition générale se trouva au § IV.2.1.

$$\mathcal{M} \models \forall(A_k \vee A'_k)$$

ce qui est contradictoire. $\square$

Proposition VI.6 : Soit P un programme Prolog et $:-L$ un but. Si tous les arbres SLD de $P \cup \{:-L\}$ sont finis, alors $P_f^{CET} \models \forall[L]_f^F$.

Démonstration : Soit $\mathcal{M}$ un modèle de P_f^{CET} de pré-modèle $\mathcal{M}'$ et soit $\mathcal{N} = \text{pppf}(T_f^{\mathcal{M}'})$. On a $\mathcal{N} \models \forall[L]_f^F$ d'après la proposition précédente, et comme $\forall[L]_f^F$ est une formule positive, on a aussi $\mathcal{M} \models \forall[L]_f^F$ puisque $\mathcal{M}$ est une expansion de $\mathcal{N}$. $\square$

Lemme VI.9 : Soit P un programme Prolog et A un atome tel que tous les arbres SLD de $P \cup \{:-A\}$ soient finis et $A_1 = p(s)$. Soient $C_1, \dots, C_q$ les règles de P de prédicat de tête p . Alors : $P_f^{CET} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s))$.

Démonstration : En effet, soit $C_i = p(t) :- L_i$. Le lemme VI.5 montre que si les n -uplets s et t ne sont pas unifiables alors $P_f^{CET} \models \forall C_i^F(s)$. Sinon, soit θ_i leur upg. Le même lemme montre que $P_f^{CET} \models \forall C_i^F(s) \Leftrightarrow \forall[L\theta_i]_f^F$ et la proposition précédente permet alors de conclure. $\square$

Proposition VI.7 : Soit P un programme Prolog et soit $:-L$ un but. Si tous les arbres SLD de $P \cup \{:-L\}$ sont finis et si le but $:-L$ réussit avec la réponse calculée θ alors, $P_f^{CET} \models \forall[L\theta]_f^{SF}$.

Démonstration : Par récurrence sur la profondeur de l'arbre SLD standard de $:-L$. Si la profondeur est nulle, L est la liste vide et le résultat est évident. Sinon, soit $A_1 = p(s)$ le premier atome de L , M la liste telle que $L = [A_1], M$ et soient $C_1, \dots, C_q$ les règles de P de prédicat de tête p . On a :

$$P_f^{CET} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s) \wedge (C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \Rightarrow A_1)$$

mais également, d'après le lemme précédent :

$$P_f^{CET} \models \forall(C_1^F(s) \wedge \dots \wedge C_q^F(s)).$$

On en déduit que :

$$P_f^{CET} \models \forall(C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s) \Rightarrow A_1)$$

et :

$$P_f^{CET} \models \forall((C_1^{SF}(s) \vee \dots \vee C_q^{SF}(s)) \wedge [M]_f^{SF} \Rightarrow [L]_f^{SF}).$$

Soit $C_i = p(t) :- L_i$ une règle qui permet d'obtenir la réponse calculée θ en sélectionnant A_1 . On a :

$$P_f^{CET} \models \forall(C_i^{SF}(s) \wedge [M]_f^{SF} \Rightarrow [L]_f^{SF}).$$

Soit θ_i l'upg des n -uplets s et t et σ une réponse calculée du but $:-L_i\theta_i, M\theta_i$ telle que $L\theta = L\theta_i\sigma$.

Soient $\{y_1, \dots, y_k\}$ les variables figurant dans les termes t_i et la liste L_i et $\{x_1, \dots, x_m\}$ les variables figurant dans les termes s_i (on suppose que ces deux ensembles sont disjoints). On a :

$$P_f^{CET} \models \forall x_1 \dots x_m (\exists y_1 \dots y_k (s = t \wedge [L_i]_f^{SF} \wedge [M]_f^{SF}) \Rightarrow [L]_f^{SF}).$$

Supposons qu'il existe un modèle $\mathcal{M}$ de P_f^{CET} tel que $\mathcal{M} \models \neg \forall [L\theta]_f^{SF}$ et soit $[x \backslash a]$ une assignation des variables x_i satisfaisant l'équation associée à θ et pour laquelle on a :

$$\mathcal{M} \models \neg [L\theta]_f^{SF} [x \backslash a]$$

ou encore (d'après le lemme II.2) :

$$\mathcal{M} \models \neg [L]_f^{SF} [x \backslash a].$$

On a :

$$\mathcal{M} \models \neg \exists y_1 \dots y_k (s = t \wedge [L_i]_f^{SF} \wedge [M]_f^{SF}) [x \backslash a]$$

soit :

$$\mathcal{M} \models \neg \exists y_1 \dots y_k (s = t \wedge [L_i, M]_f^{SF}) [x \backslash a]$$

soit encore :

$$\forall y_1 \dots y_k (s \neq t \vee \neg ([L_i, M]_f^{SF})) [x \backslash a]$$

et d'après le lemme II.2 :

$$\forall y_1 \dots y_k \neg ([L_i \theta_i, M \theta_i]_f^{SF}) [x \backslash a].$$

Mais d'autre part, par hypothèse de récurrence, on a :

$$\mathcal{M} \models \forall [L_i \theta_i \sigma, M \theta_i \sigma]_f^{SF}$$

et en particulier :

$$\forall y_1 \dots y_k [L_i \theta_i \sigma, M \theta_i \sigma]_f^{SF} [x \backslash a]$$

ce qui est contradictoire.

On en déduit que pour tout modèle $\mathcal{M}$ de P_f^{CET} , on a $\mathcal{M} \models \forall [L\theta]_f^{SF}$ soit encore $P_f^{CET} \models \forall [L\theta]_f^{SF}$.

□

Proposition VI.8 : Soit P un programme Prolog et $:-L$ un but. Si tous les arbres SLD de $P \cup \{:-L\}$ sont des arbres d'échecs finis alors, $P_f^{CET} \models \forall [L]_f^{EF}$.

Démonstration : Par récurrence sur la somme N des profondeurs des arbres SLD du but $:-L$ (qui n'a qu'un nombre fini d'arbres SLD).

Si $N = 0$, aucun des atomes de L ne s'unifie avec aucune tête de règle du programme P . Pour tout atome $A = p(s)$ de L , et toute règle de prédicat de tête p on a donc :

$$P_f^{CET} \models \forall C^{EF}(s) \text{ et donc } P_f^{CET} \models \forall A'.$$

On en déduit que $P_f^{CET} \models \forall [L]_f^{EF}$.

Supposons la propriété démontrée pour un entier N et supposons que la somme des profondeurs des arbres SLD de $:-L$ soit égale à $N+1$.

S'il est faux que $P_f^{CET} \models \forall [L]_f^{EF}$, il existe un atome $A = p(s)$ de L tel que $P_f^{CET} \models \forall A'$ soit faux.

Soient $C_1, \dots, C_q$ les q règles de P de prédicat de tête p .

$$P_f^{CET} \models \forall (C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s) \Rightarrow A')$$

donc il est faux que :

$$P_f^{CET} \models \forall (C_1^{EF}(s) \wedge \dots \wedge C_q^{EF}(s))$$

et soit i tel que $P_f^{CET} \models \forall C_i^{EF}(s)$ soit faux. La tête de C_i s'unifie avec A . Soit θ_i l'upg et soit L_i le corps de C_i . D'après le lemme VI.5, il est faux que $P_f^{CET} \models \forall [L_i\theta_i]_f^{EF}$.

La somme des profondeurs des arbres SLD du but L' dérivé de L en sélectionnant l'atome A et en appliquant la règle C_i est inférieure ou égale à N . Donc, d'après l'hypothèse de récurrence, $P_f^{CET} \models \forall [L']_f^{EF}$ et en particulier :

$$P_f^{CET} \models \forall [L_i\theta_i]_f^{EF}$$

ce qui est contradictoire. On en conclut que :

$$P_f^{CET} \models \forall [L]_f^{EF}.$$

□

Les trois propositions précédentes établissent **la correction de la sémantique opérationnelle invariante finie pour la sémantique axiomatique invariante finie dans le cadre de la CET**. Nous résumons ces résultats dans le théorème suivant :

Théorème VI.9 : *Soit P un programme Prolog et $:-L$ un but. Si tous les arbres SLD de $P \cup \{:-L\}$ sont finis, alors :*

- a) $P_f^{CET} \models \forall [L]_f^F$.
- b) *Si ce but échoue finiment, alors $P_f^{CET} \models \forall [L]_f^{EF}$.*
- c) *S'il réussit avec la réponse calculée θ , alors $P_f^{CET} \models \forall [L\theta]_f^{SF}$.*

Nous avons vu que la sémantique invariante finie généralise, d'une certaine manière, la sémantique standard finie en cela qu'elle décrit le noyau commun effectivement calculable par tous les interpréteurs Prolog. Les théorèmes VI.4 et VI.9 montrent que : **le noyau commun effectivement calculable par tous les interpréteurs Prolog admet une sémantique axiomatique : "la traduction invariante finie d'un programme P dans le cadre logique de la Théorie Equationnelle de Clark du langage sous-jacent à P ".**

Le tableau ci-dessous récapitule l'essentiel des résultats obtenus dans les chapitres III, IV et VI.

TABLEAU RÉCAPITULATIF

Interpréteur Prolog	Sémantique Opérationnelle	Traduction axiomatique	Résultats dans le cadre des modèles de Herbrand	Résultats dans le cadre de la Théorie équationnelle de Clark
Idéal - largeur d'abord - arbres équitables	Idéale : $SS \cup FF'$	P_{SI}, P_{SSI}	$A \in SS \text{ ssi } P_{SI}^{Her} \models A \text{ ssi } P_{SSI}^{Her} \models A$	$A \in SS \text{ ssi } P_{SSI}^{CET} \models A$
			$A \in FF \text{ ssi } A \in H \setminus T_{SI} \downarrow \omega$	$A \in FF \text{ ssi } P_{SSI}^{CET} \models \neg A$
		P_{SSId}	$A \in SS \text{ ssi } P_{SSId}^{Her} \models A$	$A \in SS \text{ ssi } P_{SSId}^{CET} \models A$
			$A \in FF \text{ ssi } A' \in T_{SSId} \uparrow \omega$	$A \in FF \text{ ssi } P_{SSId}^{CET} \models A'$
Standard - profond d'abord - arbres non équitables	Standard : $SS_{st} \cup FF'_{st}$	P_{st}	Si P vérifie l'hypothèse H_3 : $A \in SS_{st} \text{ ssi } T_{st} \uparrow \omega \models A$ $A \in FF_{st} \text{ ssi } T_{st} \uparrow \omega \models A'$	Si P vérifie l'hypothèse H_3 : $A \in SS_{st} \text{ ssi } P_{st}^{CET} \models A$ $A \in FF_{st} \text{ ssi } P_{st}^{CET} \models A'$
			$A \in SS_{fst} \text{ ssi } T_{fst} \uparrow \omega \models A$	$A \in SS_{fst} \text{ ssi } P_{fst}^{CET} \models A$
	Standard finie : $SS_{fst} \cup FF'_{fst}$	P_{fst}	$A \in FF_{st} \text{ ssi } T_{fst} \uparrow \omega \models A'$	$A \in FF_{st} \text{ ssi } P_{fst}^{CET} \models A'$
			$A \in SS_f \text{ ssi } T_f \uparrow \omega \models A$	$A \in SS_f \text{ ssi } P_f^{CET} \models A$
Tout interpréteur Prolog	Invariante finie :	P_f	$A \in FF_f \text{ ssi } P_f^{Her} \models A'$	$A \in FF_f \text{ ssi } P_f^{CET} \models A'$
	$SS_f \cup FF_f$		Cas général	Prog. sans variable locale ou sans symbole de fonction

Chapitre VII :

Logique trivaluée et dédoublement des symboles de prédicats

Nous avons choisi de mener la présente étude des sémantiques axiomatiques de Prolog en utilisant une technique de dédoublement (et même de "détriplement" dans l'étude de la sémantique standard) des symboles de prédicats du langage sous-jacent au programme. Cette technique présente de multiples avantages :

- elle permet de rester dans le cadre de la logique classique
- les traductions axiomatiques que nous avons définies sont constituées de programmes généraux positifs et nous avons vu au chapitre V que ces systèmes d'axiomes ont des propriétés particulièrement intéressantes : monotonie de l'opérateur T de conséquence immédiate, caractérisation logique de l'itéré T^ω , ...
- de plus, cette technique s'est révélée suffisamment fine pour permettre d'exprimer les propriétés opérationnelles que nous avons choisies d'étudier.

Ainsi que nous l'avons mentionné au § II.5.5, une autre approche possible de la sémantique des programmes logiques consiste à utiliser la logique trivaluée¹ [Fit 85, Del 87b, Thi 90, Kun 87a, 89]. Les résultats les plus généraux sur la complétude de la résolution SLDNF par rapport à un système d'axiomes ont été obtenus pour des axiomatiques trivaluées [Kun 89]². Cela n'est pas étonnant dans la mesure où, dans le cas général, les mécanismes de calcul utilisés en programmation logique ne permettent pas de déduire qu'une formule de la forme $A \vee \neg A$ est un théorème : ce qui est également à la base de la logique trivaluée.

Ces deux approches sont évidemment très liées. En effet, un atome clos A peut recevoir en logique trivaluée trois valeurs de vérité relativement à une interprétation (trivaluée) du langage L : A est Vrai, A est Faux, A est Indéterminé. Ceci correspond aux cas suivants pour une interprétation (dédoublee) I de L_d : A appartient à I , A' appartient à I , ni A , ni A' n'appartiennent à I .

¹On utilise parfois aussi le terme de "logique partielle" : ceci exprime peut-être le souci de ne pas définir un nouveau formalisme "qui pourrait décourager le lecteur non spécialiste" [Bid 89].

²S. Cerrito a obtenu depuis des résultats comparables en utilisant la logique linéaire [Cer 90].

Nous étudions systématiquement dans ce paragraphe, le lien entre ces deux approches. Cela nous permettra en particulier d'énoncer un théorème de complétude en logique trivaluée qui généralise le résultat de [Kun 87a]. Nous verrons également (dans un cas particulier important : celui des différents complétés d'un programme trivalué à têtes positives) que l'on peut identifier la "logique dédoublée" à une restriction de la logique trivaluée : la logique trivaluée monotone, c'est-à-dire ne prenant en compte que les formules conséquences n'ayant pas d'autres occurrences de connecteurs que ceux de la logique bivaluée classique.

Nous faisons de fréquentes références dans ce paragraphe aux travaux de Fitting [Fit 85] (qui a été un des premiers à envisager une sémantique en logique trivaluée pour les programmes logiques), à ceux de J.-P. Delahaye et de V. Thibau [Del 87a, 87b, 87c, 88b, 90c, 90d, Thi 90] (dans lesquels on trouvera plus de précision sur les notions redéfinies ici) et à ceux de Kunen : le résultat principal que nous présentons ici est une généralisation du théorème de complétude démontré dans [Kun 87a].

VII.1 Traduction dédoublée - trivaluée

Comme notre préoccupation majeure dans ce paragraphe est d'examiner les liens entre les approches dédoublées et trivaluées pour l'étude des sémantiques de Prolog et comme nous avons vu dans les chapitres précédents que l'on obtient les meilleurs résultats en se plaçant dans le cadre de la Théorie Equationnelle de Clark, nous n'envisagerons que des modèles du langage $\mathbf{L}$ qui sont aussi des modèles de la $\text{CET}(\mathbf{L})$.

Soit $\mathbf{L}$ un langage du premier ordre et $\mathcal{M}$ un pré-modèle de la $\text{CET}(\mathbf{L})$.

Nous rappelons que $H^{\mathcal{M}}$ est l'ensemble des $(p; \alpha_1, \dots, \alpha_n)$ où p est un symbole de prédicat n -aire de $\mathbf{L}$ et $(\alpha_1, \dots, \alpha_n)$ est un n -uplet d'éléments du domaine $U^{\mathcal{M}}$ de $\mathcal{M}$.

Notation : Soit I une interprétation de $\mathbf{L}$ sur $\mathcal{M}$ (c'est-à-dire une partie de $H^{\mathcal{M}}$).

On note :

$$\neg I = \{(\neg p; \alpha_1, \dots, \alpha_n) ; (p; \alpha_1, \dots, \alpha_n) \in I\} \text{ et } \\ I' = \{(p'; \alpha_1, \dots, \alpha_n); (p; \alpha_1, \dots, \alpha_n) \in I\}.$$

Définition VII.1 : Soit $\mathcal{M}$ un pré-modèle de la $\text{CET}(\mathbf{L})$. On appelle **interprétation trivaluée (ou partielle) de $\mathbf{L}$ sur $\mathcal{M}$** , toute partie I de $H^{\mathcal{M}} \cup \neg H^{\mathcal{M}}$ tq :

$$\{(p; \alpha_1, \dots, \alpha_n) ; (p; \alpha_1, \dots, \alpha_n) \in I \text{ et } (\neg p; \alpha_1, \dots, \alpha_n) \in I\} = \emptyset.$$

Notation : On note $\text{It}^{\mathcal{M}}$ l'ensemble des interprétations trivaluées de $\mathbf{L}$ sur $\mathcal{M}$.

Définition VII.2 : Soit $\mathcal{M}$ un pré-modèle de la $\text{CET}(\mathbf{L})$. On appelle **interprétation dédoublée de $\mathbf{L}$ sur $\mathcal{M}$** , toute partie de $H^{\mathcal{M}} \cup H^{\mathcal{M}'}$ tq :

$$\{(p; \alpha_1, \dots, \alpha_n) ; (p; \alpha_1, \dots, \alpha_n) \in I \text{ et } (p'; \alpha_1, \dots, \alpha_n) \in I\} = \emptyset.$$

Notation : On note $\text{Id}^{\mathcal{M}}$ l'ensemble des interprétations dédoublées de $\mathbf{L}$ sur $\mathcal{M}$.

Remarque : Ni $It^{\mathcal{M}}$, ni $Id^{\mathcal{M}}$ ne sont des treillis complets (lorsqu'on les munit de l'inclusion). Par exemple, $\{\{A\}, \{A'\}\}$ n'a pas de majorant.

On introduit, comme dans [Del 87b, 88b, 90c, Thi 90], un nouveau symbole 'Contra' (pour désigner la contradiction) et on pose :

$$It_{\infty}^{\mathcal{M}} = It^{\mathcal{M}} \cup \{\text{Contra}\} \quad \text{et} \quad Id_{\infty}^{\mathcal{M}} = Id^{\mathcal{M}} \cup \{\text{Contra}\}$$

la relation d'inclusion étant prolongée par :

$$\forall I \in It_{\infty}^{\mathcal{M}}, I \subset \text{Contra} \quad \text{et} \quad \forall I \in Id_{\infty}^{\mathcal{M}}, I \subset \text{Contra}.$$

$It_{\infty}^{\mathcal{M}}$ et $Id_{\infty}^{\mathcal{M}}$ munis de l'inclusion sont alors des **treillis complets**.

Notation : On note d l'application :

$$\begin{aligned} d : \quad It^{\mathcal{M}} &\longrightarrow Id^{\mathcal{M}} \\ I &\longrightarrow d(I) = \{(p; \alpha_1, \dots, \alpha_n) ; (p; \alpha_1, \dots, \alpha_n) \in H^{\mathcal{M}} \cap I\} \cup \\ &\quad \{(p'; \alpha_1, \dots, \alpha_n) ; \neg(p; \alpha_1, \dots, \alpha_n) \in I\}. \end{aligned}$$

d est bijective.

Cette application peut être prolongée en une bijection de $It_{\infty}^{\mathcal{M}}$ sur $Id_{\infty}^{\mathcal{M}}$ en posant :

$$d(\text{Contra}) = \text{Contra}.$$

d est alors un **isomorphisme** pour la structure de treillis.

Notations : On note $\text{For}(\mathbf{L})$ (resp. $\text{For}^+(\mathbf{L})$) l'ensemble des formules (resp. formules positives¹) du langage $\mathbf{L}$.

Soit F une formule de $\mathbf{L}$. On suppose que les connecteurs figurant dans F sont parmi : $\neg$, $\vee$ et $\wedge$.

Notation : On note F^+ la formule de $\text{For}^+(\mathbf{L}_d)$ obtenue à partir de F par application itérée (et saturée) des règles suivantes :

- 1- remplacer $\neg(P \vee Q)$ par $\neg P \wedge \neg Q$, $\neg(P \wedge Q)$ par $\neg P \vee \neg Q$ et $\neg\neg P$ par P .
- 2- remplacer $\neg \exists P$ par $\forall \neg P$ et $\neg \forall P$ par $\exists \neg P$
- 3- remplacer $\neg A$ par A'

où P et Q sont des formules et A un atome de $\mathbf{L}$ de prédicat différent de '='.

Remarque : On peut montrer que ce procédé définit bien une formule unique de $\mathbf{L}_d$.

Exemple VII.1 :

Soit $F(x) = \neg(x = 0 \vee \exists y(x = s(y) \wedge p(y)))$.

$F^+(x) = x \neq 0 \wedge \forall y(x \neq s(y) \vee p'(y))$.

¹Définition 1.23

On définit ainsi une application bijective de $\text{For}(\mathbf{L})$ sur $\text{For}^+(\mathbf{L}_d)$ ([Del 87b et 88a, Ble 90, Gil 89]). On peut étendre cette application aux ensembles de formules de $\mathbf{L}$:

$$\text{si } Ax \subset \text{For}(\mathbf{L}), Ax^+ = \{F^+ / F \in Ax\} \subset \text{For}^+(\mathbf{L}_d)$$

Remarques :

- 1- Si θ est une substitution des variables libres de la formule F telle qu'aucune variable de $\text{Im}(\theta)$ ne soit liée dans F , alors $(F\theta)^+ = (F^+)\theta$.
- 2- Pour toute interprétation trivaluée I de $\text{It}^{\mathbf{M}}$, $d(I) = \{L^+ / L \in I\}$.

La notion de valeur de vérité d'une formule F relativement à une interprétation trivaluée I de $\mathbf{L}$ sur un pré-modèle $\mathbf{M}$ et à une assignation $(x_1 \backslash \alpha_1, \dots, x_m \backslash \alpha_m)$ des variables de F dans le domaine de $\mathbf{M}$ est définie dans [Thi 90].

Notation : Nous la noterons $\text{vv}_I(F[\alpha_1, \dots, \alpha_m])$.

vv_I est une application à valeurs dans $\{V, F, I\}$ où V désigne Vrai, F désigne Faux et I désigne Indéterminé.

Rappelons succinctement cette définition.

- Si F est une formule équationnelle alors :

$$\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = \begin{cases} V & \text{si } \mathbf{M} \models F[\alpha_1, \dots, \alpha_m] \\ F & \text{sinon} \end{cases}$$

- Si $F(y_1, \dots, y_m) = p(t_1(y_1, \dots, y_m), \dots, t_q(y_1, \dots, y_m))$ est une formule atomique alors,

$$\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = \begin{cases} V & \text{si } (p; t_1(\alpha_1, \dots, \alpha_m), \dots, t_q(\alpha_1, \dots, \alpha_m)) \in I \\ F & \text{si } (\neg p; t_1(\alpha_1, \dots, \alpha_m), \dots, t_q(\alpha_1, \dots, \alpha_m)) \in I \\ I & \text{sinon.} \end{cases}$$

On définit ensuite la valeur de vérité d'une formule quelconque par induction sur la complexité de F . Le passage des quantificateurs se fait de manière standard. Le passage des connecteurs $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ s'effectue selon les tables de vérité suivantes :

$\vee$	V	F	I
V	V	V	V
F	V	F	I
I	V	I	I

$\wedge$	V	F	I
V	V	F	I
F	F	F	F
I	I	F	I

$\Rightarrow$	V	F	I
V	V	F	I
F	V	V	V
I	V	I	I

$\Leftrightarrow$	V	F	I
V	V	F	I
F	F	V	I
I	I	I	I

	$\neg$
V	F
F	V
I	I

Nous ne définissons ici que les valeurs de vérité des formules de $\text{For}(\mathbf{L})$, c'est-à-dire ne contenant que des occurrences de connecteurs bivalués.

Proposition VII.1 : *Pour toute interprétation trivaluée I de $\mathbf{L}$ sur le pré-modèle $\mathfrak{M}$ de la CET($\mathbf{L}$), toute formule $F(x_1, \dots, x_m)$ de $\mathbf{L}$ et tous $\alpha_1, \dots, \alpha_m \in U^{\mathfrak{M}}$:*

- $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = V$ ssi $d(I) \models F^+[\alpha_1, \dots, \alpha_m]$
- $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = F$ ssi $d(I) \models (\neg F)^+[\alpha_1, \dots, \alpha_m]$
- $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = I$ ssi $d(I) \models (\neg F^+ \wedge \neg(\neg F)^+)[\alpha_1, \dots, \alpha_m]$

Démonstration : par récurrence sur la complexité de F .

- Si F est une formule équationnelle,

$$\begin{aligned} \text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = V & \quad \text{ssi} \quad \mathfrak{M} \models F[\alpha_1, \dots, \alpha_m] \\ & \quad \text{ssi} \quad \mathfrak{M} \models F^+[\alpha_1, \dots, \alpha_m] \text{ puisque } F = F^+ \\ & \quad \text{ssi} \quad d(I) \models F^+[\alpha_1, \dots, \alpha_m] \end{aligned}$$

De même si $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = F$.

- Si $F(y_1, \dots, y_m) = p(t_1(y_1, \dots, y_m), \dots, t_q(y_1, \dots, y_m))$ est une formule atomique,

$$\begin{aligned} \text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = V & \quad \text{ssi} \quad (p; t_1(\alpha_1, \dots, \alpha_m), \dots, t_q(\alpha_1, \dots, \alpha_m)) \in I \\ & \quad \text{ssi} \quad (p; t_1(\alpha_1, \dots, \alpha_m), \dots, t_q(\alpha_1, \dots, \alpha_m)) \in d(I) \\ & \quad \text{ssi} \quad d(I) \models F^+[\alpha_1, \dots, \alpha_m]. \end{aligned}$$

De même si $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = F$ ou I .

- Si $F = \neg G$,

$$\begin{aligned} \text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = I & \quad \text{ssi} \quad \text{vv}_I(G[\alpha_1, \dots, \alpha_m]) = I \\ & \quad \text{ssi} \quad d(I) \models (\neg G^+ \wedge \neg(\neg G)^+)[\alpha_1, \dots, \alpha_m] \\ & \quad \text{ssi} \quad d(I) \models (\neg(\neg F)^+ \wedge \neg F^+)[\alpha_1, \dots, \alpha_m]. \end{aligned}$$

De même si $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = V$ ou F .

- Si $F = G \vee H$,

$$\begin{aligned} \text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = F & \quad \text{ssi} \quad \text{vv}_I(G[\alpha_1, \dots, \alpha_m]) = F \text{ et } \text{vv}_I(H[\alpha_1, \dots, \alpha_m]) = F \\ & \quad \text{ssi} \quad d(I) \models (\neg G)^+[\alpha_1, \dots, \alpha_m] \text{ et } d(I) \models (\neg H)^+[\alpha_1, \dots, \alpha_m] \\ & \quad \text{ssi} \quad d(I) \models ((\neg G)^+ \wedge (\neg H)^+)[\alpha_1, \dots, \alpha_m] \\ & \quad \text{ssi} \quad d(I) \models ((\neg G \wedge \neg H)^+)[\alpha_1, \dots, \alpha_m] \end{aligned}$$

$$\begin{aligned} \text{ssi} \quad d(I) &\models (\neg(G \vee H))^+[\alpha_1, \dots, \alpha_m] \\ \text{ssi} \quad d(I) &\models (\neg F)^+[\alpha_1, \dots, \alpha_m]. \end{aligned}$$

Les autres cas se démontrent de manière analogue. □

Remarque : On peut également considérer cette proposition comme définissant la valeur de vérité d'une formule de $\mathbf{L}$ relativement à une interprétation trivaluée. Il suffit pour cela de s'assurer que F^+ et $(\neg F)^+$ ne peuvent pas être simultanément vraies dans une interprétation dédoublée.

Proposition VII.2 : *Pour tout pré-modèle $\mathfrak{M}$ de la $\text{CET}(\mathbf{L})$, toute formule $F(x_1, \dots, x_m)$ de $\mathbf{L}$ et tous $\alpha_1, \dots, \alpha_m \in U^{\mathfrak{M}}$, l'application*

$$\begin{aligned} vv : \text{It}^{\mathfrak{M}} &\longrightarrow \{V, F, I\} \\ I &\longrightarrow vv_I(F[\alpha_1, \dots, \alpha_m]) \end{aligned}$$

(où $\{V, F, I\}$ est muni de l'ordre : $I < V$ et $I < F$) est monotone.

Démonstration : Soient I et J deux interprétations trivaluées sur $\mathfrak{M}$ telles que $I \subset J$. On a alors : $d(I) \subset d(J)$.

- Si $vv_I(F[\alpha_1, \dots, \alpha_m]) = V$, alors $d(I) \models F^+[\alpha_1, \dots, \alpha_m]$.

Comme F^+ est positive, $d(J) \models F^+[\alpha_1, \dots, \alpha_m]$ et donc $vv_J(F[\alpha_1, \dots, \alpha_m]) = V$.

- Démonstration analogue si $vv_I(F[\alpha_1, \dots, \alpha_m]) = F$. □

Définition VII.3 : *On appelle modèle trivalué d'une formule close F de $\mathbf{L}$, toute interprétation trivaluée I de $\mathbf{L}$ telle que $vv_I(F) = \text{Vrai}$.*

Définition VII.4 : *On appelle modèle dédoublé d'une formule close F de $\mathbf{L}$, toute interprétation dédoublée de $\mathbf{L}$ qui est un modèle de F^+ (dans $\mathbf{L}_d$).*

Notation : On note CONS l'ensemble d'axiomes $\{\forall \neg(A \wedge A')\}$ où A est un atome de $\mathbf{L}$.

Proposition VII.3 : *Soit I une interprétation de $\mathbf{L}_d$ et soit F une formule close de $\mathbf{L}$. Alors, I est un modèle dédoublé de F ssi I est un modèle (dans $\mathbf{L}_d$) de $F^+ \cup \text{CONS}$.*

Démonstration : En effet, une interprétation I de $\mathbf{L}_d$ est une interprétation dédoublée de $\mathbf{L}$ ssi I est un modèle de CONS . □

Proposition VII.4 : *Soit F une formule close de $\mathbf{L}$. La restriction de l'application d à l'ensemble des modèles trivalués de F est une bijection sur l'ensemble des modèles dédoublés de F^+ (qui est égal à l'ensemble des modèles de $F^+ \cup \text{CONS}$).*

Démonstration : Immédiat d'après les définitions précédentes. □

Définition VII.5 : Une formule close F de $\mathcal{L}$ est une **conséquence logique trivaluée** d'un ensemble de formules Ax de $\mathcal{L}$ si tout modèle trivalué de Ax est un modèle trivalué de F .

Notation : $Ax \models_1 F$.

Proposition VII.5 : Une formule close F de $\mathcal{L}$ est une **conséquence logique trivaluée** d'un ensemble de formules Ax de $\mathcal{L}$ ssi F^+ est conséquence logique de $Ax^+ \cup \text{CONS}$.

Démonstration : Les modèles de $Ax^+ \cup \text{CONS}$ sont les modèles dédoublés de Ax^+ qui sont eux-mêmes les images par l'application d des modèles trivalués de Ax . La conclusion est donc une conséquence directe des propositions VII.3 et VII.4. □

VII.2 Programmes trivalués

Définition VII.6 : [Del 87b, 88b, 90c, Thi 90] Le **connecteur trivalué** $\leftarrow$ est défini, pour toute interprétation trivaluée I , par la table de vérité suivante :

$\rightarrow$	V	F	I
V	V	F	F
F	V	V	V
I	V	V	V

Définition VII.7 : On appelle **programme trivalué** tout système d'axiomes P de la forme $\forall(A \leftarrow F)$ ou $\forall(\neg A \leftarrow F)$ où A est un atome de $\mathcal{L}$ formé à partir d'un prédicat différent de '=' et F une formule de $\mathcal{L}$.

Remarque : Le quantificateur ' $\forall$ ' est souvent omis lorsqu'un programme trivalué est représenté sous forme de système.

Comme dans le cas des programmes généraux (bivalués) (Cf II.1.5), on peut montrer que l'on peut supposer, sans perte de généralité, qu'un programme trivalué est normalisé, c'est-à-dire que :

- toute tête d'axiome de P est de la forme $p(x_1, \dots, x_n)$ ou $\neg p(x_1, \dots, x_n)$ où p est un symbole de prédicat (distinct de '=') et où $x_1, \dots, x_n$ sont des variables de $\mathcal{L}$
- pour tout symbole de prédicat p (distinct de '='), le programme P contient **exactement** un axiome de tête p et un axiome de tête $\neg p$.

Définition VII.8 : Etant donnés m formules $H_1, \dots, H_m$ où chaque H_i est de la forme $p(t_{1,i}, \dots, t_{n,i}) \leftarrow F_i$ (resp. $\neg p(t_{1,i}, \dots, t_{n,i}) \leftarrow F_i$) et p est un symbole de prédicat n -aire, on appelle **forme normale** de $\{H_1, \dots, H_m\}$ la formule suivante :

$$p(x_1, \dots, x_n) \Leftarrow G_1 \vee \dots \vee G_m \quad (\text{resp. } \neg p(x_1, \dots, x_n) \Leftarrow G_1 \vee \dots \vee G_m)$$

où G_i est la formule :

$$\exists y_1 \dots \exists y_{p_i} x_1 = t_{1,i} \wedge \dots \wedge x_n = t_{n,i} \wedge F_i$$

où les y_i sont les variables libres figurant dans les termes $t_{j,i}$ et dans la formule F_i et où les x_i sont de nouvelles variables.

Proposition VII.6 : Si H_{Norm} est la forme normale des formules $H_1, \dots, H_m$ alors H_{Norm} et $\{H_1, \dots, H_m\}$ sont équivalentes pour la CET($\mathbf{L}$) dans un sens trivalué, c'est-à-dire que pour tout pré-modèle $\mathcal{M}$ de la CET($\mathbf{L}$) et toute interprétation I de $\mathbf{L}$ sur $\mathcal{M}$:

$$vv_I(\forall H_{\text{Norm}}) = vv_I(\forall H_1 \wedge \dots \wedge \forall H_m).$$

Démonstration : Soit $\mathcal{M}$ un pré-modèle de la CET($\mathbf{L}$) et I une interprétation de $\mathbf{L}$ sur $\mathcal{M}$.

- Supposons que $vv_I(\forall H_{\text{Norm}}) = V$.

Soit $i \in \{1, \dots, m\}$ et supposons que $vv_I(F_i[\beta_1, \dots, \beta_{p_i}]) = V$ où $\beta_1, \dots, \beta_{p_i}$ est une assignation des variables $y_1, \dots, y_{p_i}$ dans $\mathcal{M}$.

On a alors : $vv_I(G_i[\alpha_1, \dots, \alpha_n]) = V$ où $\alpha_j = t_{j,i}[\beta_1, \dots, \beta_{p_i}]$ et comme $vv_I(\forall H_{\text{Norm}}) = V$, on a $vv_I(p[\alpha_1, \dots, \alpha_n]) = V$, soit $vv_I(p(t_{1,i}, \dots, t_{n,i})[\beta_1, \dots, \beta_{p_i}]) = V$.

On en déduit donc que $vv_I(\forall H_i) = V$ et par conséquent, $vv_I(\forall H_1 \wedge \dots \wedge \forall H_m) = V$.

- Supposons que $vv_I(\forall H_{\text{Norm}}) = F$.

Il existe une assignation $\alpha_1, \dots, \alpha_n$ des variables $x_1, \dots, x_n$ telle que $vv_I(p[\alpha_1, \dots, \alpha_n]) \neq V$ et un entier i tel que, $vv_I(\exists y_1 \dots \exists y_{p_i} \alpha_1 = t_{1,i} \wedge \dots \wedge \alpha_n = t_{n,i} \wedge F_i) = V$.

Soit $\beta_1, \dots, \beta_{p_i}$ une assignation des variables $y_1, \dots, y_{p_i}$ dans $\mathcal{M}$ pour laquelle on a :

$$vv_I(\alpha_1 = t_{1,i}[\beta_1, \dots, \beta_{p_i}] \wedge \dots \wedge \alpha_n = t_{n,i}[\beta_1, \dots, \beta_{p_i}] \wedge F_i[\beta_1, \dots, \beta_{p_i}]) = V.$$

On a donc $vv_I(p(t_{1,i}, \dots, t_{n,i})[\beta_1, \dots, \beta_{p_i}]) \neq V$ et $vv_I(F_i[\beta_1, \dots, \beta_{p_i}]) = V$.

On en déduit que $vv_I(\forall H_i) = F$ et donc $vv_I(\forall H_1 \wedge \dots \wedge \forall H_m) = F$.

- Ni $\forall H_{\text{Norm}}$, ni $\forall H_1 \wedge \dots \wedge \forall H_m$ ne peuvent prendre la valeur de vérité I .

□

Définition VII.9 : Soit P un programme trivalué. Soit P_{Norm} le programme général formé à partir de P :

- en remplaçant, pour tout symbole de prédicat p , tous les sous ensembles d'axiomes de P dont la tête est de la forme : $p(t_1, \dots, t_n)$ (resp. $\neg p(t_1, \dots, t_n)$) par leur forme normale

- en introduisant, pour tout symbole de prédicat p , l'axiome :

$$\forall (p(x_1, \dots, x_n)) \leftarrow \text{Faux} \text{ (resp. } \forall (\neg p(x_1, \dots, x_n)) \leftarrow \text{Faux})$$

si p (resp. $\neg p$) ne figure en tête d'aucun axiome de P .

P_{Norm} est le programme normalisé associé à P .

Proposition VII.7 : *Tout programme trivalué P est équivalent au programme normalisé associé P_{Norm} pour la CET($\mathbf{L}$), c'est-à-dire que tout pré-modèle $\mathcal{M}$ de la CET($\mathbf{L}$) et toute interprétation I de $\mathbf{L}$ sur $\mathcal{M}$,*

$$vv_I(P) = vv_I(P_{\text{Norm}}).$$

Démonstration : Les axiomes de la forme $\forall (p(x_1, \dots, x_n)) \leftarrow \text{Faux}$ et $\forall (\neg p(x_1, \dots, x_n)) \leftarrow \text{Faux}$ sont vraies dans toute interprétation. Cette proposition est donc une conséquence immédiate de la proposition précédente. □

Exemple :

Soit P le programme trivalué :

$$\begin{cases} \text{pair}(0) \\ \text{pair}(s(x)) \leftarrow \neg \text{pair}(x) \\ \neg \text{pair}(s(x)) \leftarrow \text{pair}(x) \\ \text{impair}(x) \leftarrow \neg \text{pair}(x) \end{cases}$$

P_{Norm} est égal à la clôture universelle de :

$$\begin{cases} \text{pair}(x) \leftarrow x = 0 \vee \exists y (x = s(y) \wedge \neg \text{pair}(y)) \\ \neg \text{pair}(x) \leftarrow \exists y (x = s(y) \wedge \text{pair}(y)) \\ \text{impair}(x) \leftarrow \neg \text{pair}(x) \\ \neg \text{impair}(x) \leftarrow \text{Faux} \end{cases}$$

Définition VII.10 : *On appelle programme trivalué à têtes positives, tout système d'axiomes P de la forme $p(x_1, \dots, x_m) \leftarrow F(x_1, \dots, x_m)$ où p est un symbole de prédicat de $\mathbf{L}$ (différent de '=') et F une formule de $\mathbf{L}$. On suppose de plus que P est normalisé c.-à-d. que pour tout symbole de prédicat p de $\mathbf{L}$, P contient exactement un axiome de tête p .*

Notation : Pour tout programme trivalué P , on note P^+ le système d'axiomes de $\mathbf{L}$ défini par :

$$(Li \leftarrow F) \in P \text{ ssi } (Li^+ \leftarrow F^+) \in P^+.$$

Remarque : On définit ainsi une bijection de l'ensemble des programmes trivalués de $\mathbf{L}$ sur l'ensemble des programmes généraux positifs de $\mathbf{L}_d$.

Notations : Soit P un programme trivalué.

1- On note, comme dans les chapitres précédents, $P^{CET} = P \cup CET(L)$ et $(P^+)^{CET} = P^+ \cup CET(L_d)$. Comme dans les paragraphes précédents, on se limite aux interprétations de P^{CET} (resp. $(P^+)^{CET}$) qui sont des extensions de l'interprétation standard $\mathcal{U}$ de L (resp. de L_d).

2- Si $\mathcal{M}$ est un pré-modèle de la $CET(L)$, on note $T_{l,p}^{\mathcal{M}}$ l'opérateur de conséquence immédiate trivalué associé à P défini sur les interprétations trivaluées de L sur $\mathcal{M}$ (l'exposant $\mathcal{M}$ est omis lorsqu'il s'agit de l'interprétation standard $\mathcal{U}$) et à valeurs dans $It_{\infty}^{\mathcal{M}}$.

Plus précisément :

$$\begin{aligned} T_{l,p}^{\mathcal{M}}(I) &= \{ (Li; \alpha_1, \dots, \alpha_m) ; Li(x_1, \dots, x_m) \leftarrow F(x_1, \dots, x_m) \text{ est un} \\ &\quad \text{axiome de } P \text{ et } \forall v_I(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai} \} \text{ si cet} \\ &\quad \text{ensemble est une interprétation trivaluée de } L \\ &= \text{Contra sinon.} \end{aligned}$$

3- Si $\mathcal{M}$ est un pré-modèle de la $CET(L)$, on note $T_{d,p}^{\mathcal{M}}$ l'opérateur de conséquence immédiate dédoublé associé à P défini sur les interprétations dédoublées de L sur $\mathcal{M}$ (l'exposant $\mathcal{M}$ est omis lorsqu'il s'agit de l'interprétation standard $\mathcal{U}$) et à valeurs dans $Id_{\infty}^{\mathcal{M}}$.

Plus précisément :

$$\begin{aligned} T_{d,p}^{\mathcal{M}}(I) &= \{ (Li^+; \alpha_1, \dots, \alpha_m) ; Li(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m) \text{ est un} \\ &\quad \text{axiome de } P \text{ et } I \models F^+[\alpha_1, \dots, \alpha_m] \} \text{ si cet} \\ &\quad \text{ensemble est une interprétation dédoublée de } L \\ &= \text{Contra sinon.} \end{aligned}$$

Exemple :

Soit P le programme trivalué :

$$\begin{cases} \text{pair}(x) \leftarrow x = 0 \vee \exists y(x = s(y) \wedge \neg \text{pair}(y)) \\ \neg \text{pair}(x) \leftarrow \exists y(x = s(y) \wedge \text{pair}(y)) \end{cases}$$

P^+ est égal à :

$$\begin{cases} \text{pair}(x) \leftarrow x = 0 \vee \exists y(x = s(y) \wedge \text{pair}'(y)) \\ \text{pair}'(x) \leftarrow \exists y(x = s(y) \wedge \text{pair}(y)) \end{cases}$$

On a :

$$T_{l,p} \uparrow 1 = T_{l,p}(\emptyset) = \{\text{pair}(0)\} \quad \text{et} \quad T_{l,p} \uparrow 2 = \{\text{pair}(0), \neg \text{pair}(s(0))\}$$

$$T_{d,P}\uparrow 1 = T_{d,P}(\emptyset) = \{\text{pair}(0)\} \quad \text{et} \quad T_{d,P}\uparrow 2 = \{\text{pair}(0), \text{pair}'(s(0))\}$$

$$T_{i,P}(\{\text{pair}(0), \text{pair}(s(0))\}) = \text{Contra}$$

car le deuxième axiome permet de déduire $\neg \text{pair}(s(0))$ et $\{\text{pair}(0), \text{pair}(s(0)), \neg \text{pair}(s(0))\}$ n'est pas une interprétation trivaluée.

De même :

$$T_{d,P}(\{\text{pair}(0), \text{pair}(s(0))\}) = \text{Contra}$$

Par contre :

$$T_{P^+}(\{\text{pair}(0), \text{pair}(s(0))\}) = \{\text{pair}(0), \text{pair}(s(0)), \text{pair}'(s(0)), \text{pair}'(s(s(0)))\}$$

Remarque : Pour toute interprétation dédoublée I de $\mathbf{L}$, on a :

$$\begin{aligned} T_{d,P}^{\mathcal{M}}(I) &= T_{P^+}^{\mathcal{M}}(I) \text{ si } T_{P^+}^{\mathcal{M}}(I) \in \text{Id}^{\mathcal{M}} \\ &= \text{Contra} \quad \text{sinon} \end{aligned}$$

où $T_{P^+}^{\mathcal{M}}$ est l'opérateur de conséquence immédiate bivalué (classique) associé à P^+ .

Proposition VII.8 : Soit P un programme trivalué. Les opérateurs $T_{i,P}^{\mathcal{M}}$ et $T_{d,P}^{\mathcal{M}}$ sont monotones.

Démonstration :

a) Soient I et J deux interprétations trivaluées telles que : $I \subset J$.

- Si $T_{i,P}^{\mathcal{M}}(I) = \text{Contra}$, il existe deux axiomes de P : $p(x_1, \dots, x_m) \leftarrow F(x_1, \dots, x_m)$ et $\neg p(x_1, \dots, x_m) \leftarrow G(x_1, \dots, x_m)$ et $\alpha_1, \dots, \alpha_m \in \mathcal{M}$ tq $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$ et $\text{vv}_I(G[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$. D'après la proposition VII.2, on a également : $\text{vv}_J(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$ et $\text{vv}_J(G[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$.

Donc, $T_{i,P}^{\mathcal{M}}(I) = \text{Contra}$.

- Si $T_{i,P}^{\mathcal{M}}(I) \neq \text{Contra}$, soit $(Li; \alpha_1, \dots, \alpha_m) \in T_{i,P}^{\mathcal{M}}(I)$. Il existe un axiome de P dont le corps F vérifie $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$. D'après la proposition VII.2, on a également : $\text{vv}_J(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$. Donc, $(Li; \alpha_1, \dots, \alpha_m) \in T_{i,P}^{\mathcal{M}}(J)$ et $T_{i,P}^{\mathcal{M}}(I) \subset T_{i,P}^{\mathcal{M}}(J)$.

b) Démonstration analogue pour $T_{d,P}^{\mathcal{M}}$.

□

Proposition VII.9 : Soit P un programme trivalué et I une interprétation trivaluée de $\mathbf{L}$ sur $\mathcal{M}$. Alors,

$$d(T_{i,P}^{\mathcal{M}}(I)) = T_{d,P}^{\mathcal{M}}(d(I)).$$

Démonstration :

- 1- $T_{d,P}^{\mathcal{M}}(d(I)) = \text{Contra}$
 ssi il existe deux axiomes de $P : A(x_1, \dots, x_m) \Leftarrow F(x_1, \dots, x_m)$ et
 $\neg A(x_1, \dots, x_m) \Leftarrow G(x_1, \dots, x_m)$ et $\alpha_1, \dots, \alpha_m \in \mathcal{M}$ tq
 $\text{vv}_I(F[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$ et $\text{vv}_I(G[\alpha_1, \dots, \alpha_m]) = \text{Vrai}$
 ssi $T_{L,P}^{\mathcal{M}}(I) = \text{Contra}$
 ssi $d(T_{L,P}^{\mathcal{M}}(I)) = \text{Contra}$

2- $T_{d,P}^{\mathcal{M}}(d(I)) \neq \text{Contra}$.

Soit A un littéral de $\mathcal{L}$ et soit $Li \Leftarrow F$ un axiome de P .

Pour toute substitution θ , on a :

$$\begin{aligned} A = Li\theta & \text{ ssi } A^+ = (Li\theta)^+, \text{ soit} \\ A = Li\theta & \text{ ssi } A^+ = Li^+\theta \text{ et} \\ \text{vv}_I(F\theta) & = \text{Vrai ssi } \text{vv}_{d(I)}((F\theta)^+) = \text{Vrai.} \end{aligned}$$

On en déduit que :

$$\begin{aligned} A \in T_{L,P}^{\mathcal{M}}(I) & \text{ ssi } A^+ \in T_{d,P}^{\mathcal{M}}(d(I)) \text{ soit} \\ A^+ \in d(T_{L,P}^{\mathcal{M}}(I)) & \text{ ssi } A^+ \in T_{d,P}^{\mathcal{M}}(d(I)). \end{aligned}$$

□

Proposition VII.10 : Soit P un programme trivalué. Pour tout ordinal α , on a :

$$d(T_{L,P}^{\mathcal{M}} \uparrow \alpha) = T_{d,P}^{\mathcal{M}} \uparrow \alpha.$$

Démonstration : Par induction sur α .

- Si $\alpha = 0$, $d(T_{L,P}^{\mathcal{M}} \uparrow 0) = d(\emptyset) = \emptyset = T_{d,P}^{\mathcal{M}} \uparrow 0$.

Supposons la relation vérifiée pour tout ordinal $\beta < \alpha$.

- Si α est un ordinal successeur, soit β tel que $\alpha = \beta + 1$.

$$\begin{aligned} d(T_{L,P}^{\mathcal{M}} \uparrow \alpha) & = d(T_{L,P}^{\mathcal{M}}(T_{L,P}^{\mathcal{M}} \uparrow \beta)) \\ & = T_{d,P}^{\mathcal{M}}(d(T_{L,P}^{\mathcal{M}} \uparrow \beta)) \text{ d'après la proposition VII.9} \\ & = T_{d,P}^{\mathcal{M}}(T_{d,P}^{\mathcal{M}} \uparrow \beta) \text{ d'après l'hypothèse de récurrence} \\ & = T_{d,P}^{\mathcal{M}} \uparrow \alpha. \end{aligned}$$

- Si α est un ordinal limite,

$$\begin{aligned} d(T_{L,P}^{\mathcal{M}} \uparrow \alpha) & = d\left(\bigcup_{\beta < \alpha} T_{L,P}^{\mathcal{M}} \uparrow \beta\right) \\ & = \bigcup_{\beta < \alpha} d(T_{L,P}^{\mathcal{M}} \uparrow \beta) \text{ d'après la définition de l'application } d \\ & = \bigcup_{\beta < \alpha} T_{d,P}^{\mathcal{M}} \uparrow \beta \text{ d'après l'hypothèse de récurrence} \\ & = T_{d,P}^{\mathcal{M}} \uparrow \alpha. \end{aligned}$$

□

Proposition VII.11 : Soit P un programme trivalué et I une interprétation trivaluée de L sur le pré-modèle $\mathcal{M}$ de la $CET(L)$. Alors,

$$\begin{aligned}
 I \text{ est un modèle trivalué de } P^{CET} & \quad ssi \quad T_{l,p}^{\mathcal{M}}(I) \subset I \\
 & \quad ssi \quad T_{d,p}^{\mathcal{M}}(d(I)) \subset d(I) \\
 & \quad ssi \quad d(I) \text{ est un modèle dédoublé de } (P^+)^{CET}. \\
 & \quad ssi \quad d(I) \text{ est un modèle de } (P^+)^{CET} \cup CONS.
 \end{aligned}$$

Démonstration : On peut supposer P normalisé.

Soit $p(x_1, \dots, x_n) \leftarrow F(x_1, \dots, x_n)$ un axiome de P .

I est un modèle de cet axiome ssi pour tous $\alpha_1, \dots, \alpha_n \in U^{\mathcal{M}}$,

$\text{vv}_I(F[\alpha_1, \dots, \alpha_n]) = \text{Vrai}$ implique $\text{vv}_I(p[\alpha_1, \dots, \alpha_n]) = \text{Vrai}$ c.-à-d. $(p; \alpha_1, \dots, \alpha_n) \in I$.

Et comme $\text{vv}_I(F[\alpha_1, \dots, \alpha_n]) = \text{Vrai}$ ssi $(p; \alpha_1, \dots, \alpha_n) \in T_{l,p}^{\mathcal{M}}(I)$, on en déduit que I est un modèle de cet axiome ssi $T_{l,p}^{\mathcal{M}}(I) \subset I$.

La deuxième équivalence est immédiate d'après la proposition VII.9.

Les deux dernières équivalences proviennent de la proposition V.4.

□

Remarque : Les points fixes des opérateurs de conséquence immédiate sont des modèles ssi ils sont différents de Contra.

Définition VII.11 : Un programme trivalué P est **consistant relativement à un pré-modèle $\mathcal{M}$** de la $CET(L)$ si P^{CET} admet un modèle trivalué de pré-modèle $\mathcal{M}$.

Remarque : Il se peut qu'un programme soit consistant relativement à un pré-modèle et inconsistant relativement à un autre, comme le montre l'exemple suivant :

Soit P le programme :
$$\begin{cases} A \leftarrow \exists x (x \neq a) \\ \neg A \leftarrow a = a. \end{cases}$$

Si $\mathcal{M}$ est le pré-modèle standard d'univers $\{a\}$, P est consistant et a pour plus petit modèle $\{\neg A\}$

Si $\mathcal{M}$ est le pré-modèle standard d'univers $\{a, b\}$, P est inconsistant.

On définira donc la consistance d'un programme de la manière suivante :

Définition VII.12 : Un programme trivalué P est **consistant** s'il est consistant relativement à tous les pré-modèles de la $CET(L)$.

Remarque : Cette définition est très exigeante. Il y a-t-il moyen de l'affaiblir ? Est-il possible de caractériser simplement les programmes consistants ? Peut-on trouver une condition sur le langage L afin que la consistance d'un programme relativement au pré-modèle standard implique sa consistance ?

Nous pensons que si le langage sous-jacent est un langage de Kunen, la consistance relativement au pré-modèle standard implique la consistance; mais nous ne sommes pas parvenu à démontrer cette conjecture.

En fait, dans tous les cas que nous avons considéré dans cette étude, les programmes généraux positifs normalisés P^+ considérés vérifient la propriété suivante : si P est le programme trivalué associé à P^+ et si F (resp. G) est le corps de l'axiome qui définit $p(x_1, \dots, x_n)$ (resp. $\neg p(x_1, \dots, x_n)$) dans P alors la formule $\neg \exists (F \wedge G)$ est universellement valide (d'un point de vue bivalué classique). C'est ce que prouvent, dans un autre formalisme, les propositions III.1, IV.1 et IV.3 pour les traductions axiomatiques standard, standard finie et invariante finie d'un programme Prolog. Cette propriété implique la consistance du programme trivalué P .

Proposition VII.12 :

Soit P un programme trivalué consistant et $\mathcal{M}$ un pré-modèle de la $CET(\mathcal{L})$. Alors,

a- Le plus petit point fixe de $T_{\mathcal{L},P}^{\mathcal{M}}$ est aussi le plus petit modèle de P^{CET} sur $\mathcal{M}$.

b- Le plus petit point fixe de $T_{d,P}^{\mathcal{M}}$ est égal au plus petit point fixe de $T_{P^+}^{\mathcal{M}}$. C'est le plus petit modèle dédoublé de $(P^+)^{CET}$ sur $\mathcal{M}$ et également le plus petit modèle de $(P^+)^{CET} \cup \text{CONS}$ sur $\mathcal{M}$.

Démonstration :

a- Le plus petit point fixe de $T_{\mathcal{L},P}^{\mathcal{M}}$ est contenu dans tous les modèles de P sur $\mathcal{M}$ et il en existe puisque P est consistant. Ce plus petit point fixe est donc distinct de Contra : c'est le plus petit modèle.

b- Immédiat d'après ce qui précède.

□

Proposition VII.13 : Soit P un programme trivalué consistant. Alors, une formule close F de $\mathcal{L}$ est conséquence trivaluée de P^{CET} ssi F^+ est conséquence logique de $(P^+)^{CET}$. Autrement dit,

$$P^{CET} \models_{\mathcal{L}} F \text{ ssi } (P^+)^{CET} \models F^+.$$

Démonstration : La proposition VII.5 montre que :

$$P^{CET} \models_{\mathcal{L}} F \text{ ssi } (P^+)^{CET} \cup \text{CONS} \models F^+.$$

Si $(P^+)^{CET} \models F^+$ alors $(P^+)^{CET} \cup \text{CONS} \models F^+$ puisque tout modèle de $(P^+)^{CET} \cup \text{CONS}$ est un modèle de $(P^+)^{CET}$.

Réciproquement, supposons que $(P^+)^{CET} \cup \text{CONS} \models F^+$. Soit I un modèle de $(P^+)^{CET}$ de pré-modèle $\mathcal{M}$ et soit J le plus petit point fixe de $T_{d,P}^{\mathcal{M}}$. C'est un modèle de $(P^+)^{CET} \cup \text{CONS}$ d'après la proposition précédente et $J \subset I$. Donc, $J \models F^+$ et comme F^+ est positive, $I \models F^+$.

□

VII.3 Théorèmes de complétude en logique trivaluée

La traduction "dédoublée-trivaluée" que nous avons définie précédemment nous permet maintenant d'énoncer des théorèmes de complétude en logique trivaluée comparables à ceux du chapitre V.

Théorème VII.1 : *Soit P un programme trivalué consistant et F une formule close de $\mathcal{L}$. Alors, si F est une conséquence logique trivaluée de $\mathcal{P}^{\text{CET}}$, il existe un entier n tq :*

$$T_{\mathcal{L},P} \uparrow n \models_i F.$$

Démonstration :

Comme F est conséquence logique trivaluée de $\mathcal{P}^{\text{CET}}$, la proposition précédente montre que F^+ est une conséquence logique de $(P^+)^{\text{CET}}$.

D'après le théorème principal du paragraphe précédent, il existe un entier n pour lequel $T_{P^+} \uparrow n \models F^+$ soit $T_{\mathcal{L},P} \uparrow n \models F^+$ ou encore $d(T_{\mathcal{L},P} \uparrow n) \models F^+$ (prop. VII.10) soit, d'après la proposition VII.4, $T_{\mathcal{L},P} \uparrow n \models_i F$.

□

Remarque : Le fait que F soit une formule de $\mathcal{L}$ (c'est-à-dire ne comporte que des connecteurs bivalués : $\neg, \wedge, \vee, \dots$) est indispensable (pour les mêmes raisons qui rendent indispensable le fait que F soit une formule positive dans le théorème V.2).

Exemple : Soit P le programme trivalué :

$$\begin{cases} p(x) \leftarrow \exists y (x=s(y) \wedge p(y)) \vee x = a \\ \neg p(x) \leftarrow \text{Faux} \end{cases}$$

et soit $F = \forall x [p(x) \leftarrow \exists y (x=s(y) \wedge p(y)) \vee x = a]$.

Pour tout entier n et tout pré-modèle $\mathcal{M}$ de la $\text{CET}(\mathcal{L})$, $T_{\mathcal{L},P}^{\mathcal{M}} \uparrow n = \{p(s^m(a)) ; 0 \leq m \leq n-1\}$.

P est consistant puisque $T_{\mathcal{L},P}^{\mathcal{M}} \uparrow \omega$ est un modèle trivalué de $\mathcal{P}^{\text{CET}}$.

La formule F est une conséquence logique de $\mathcal{P}^{\text{CET}}$ puisque c'est un axiome de P et F est fausse dans $T_{\mathcal{L},P}^{\mathcal{M}} \uparrow n$ pour tout entier n.

Corollaire VII.1 : *Soit P un programme trivalué consistant. Pour tout littéral clos L, si L est une conséquence logique trivaluée de $\mathcal{P}^{\text{CET}}$, alors $L \in T_{\mathcal{L},P} \uparrow \omega$.*

Démonstration : C'est un cas particulier du théorème précédent. En effet, comme L est clos, s'il existe un entier n tel que $T_{\mathcal{L},P} \uparrow n \models_i L$, alors $L \in T_{\mathcal{L},P} \uparrow \omega$.

□

Théorème VII.2 : Soit P un programme trivalué consistant et F une formule close de $\mathbf{L}_K$ (où $\mathbf{L}_K$ est un langage de Kunen). Alors, F est une conséquence logique trivaluée de $\mathbf{P}^{\text{CET}}$ ssi il existe un entier n tq :

$$T_{L,P} \uparrow n \models_i F.$$

Démonstration :

Soit n tq : $T_{L,P} \uparrow n \models_i F$. On a $d(T_{L,P} \uparrow n) \models F^+$ d'après la proposition VII.4, soit $T_{d,P} \uparrow n \models F^+$ (prop. VII.9) et encore $T_{P^+} \uparrow n \models F^+$. Comme $\mathbf{L}_{K_d}$ est un langage de Kunen, on a d'après le théorème VII.5, $(P^+)^{\text{CET}} \models F^+$ et donc a fortiori, $(P^+)^{\text{CET}} \cup \text{CONS} \models F^+$. La proposition VII.5 permet alors de conclure.

□

Corollaire VII.2 : Soit P un programme trivalué consistant sur un langage de Kunen $\mathbf{L}_K$. Pour tout littéral clos L , L est une conséquence logique trivaluée de $\mathbf{P}^{\text{CET}}$ ssi il existe un entier n tq : $L \in T_{L,P} \uparrow n$. Autrement dit :

$$T_{L,P} \uparrow \omega = \{L ; L \text{ est un littéral clos et } \mathbf{P}^{\text{CET}} \models_i L\}.$$

Démonstration : C'est un cas particulier du théorème précédent.

□

Ces théorèmes de complétude suggèrent l'idée de concevoir des interpréteurs "trivalués" s'appliquant à des programmes plus généraux que ceux envisagés habituellement, et qui seraient corrects et complets par rapport à leur sémantique trivaluée.

Définition VII.13 : *Fitting* a introduit en programmation logique dans [Fit 85] le connecteur d'équivalence trivaluée faible de Kleene que l'on note ici ' $\leftrightarrow$ ', défini par la table de vérité suivante :

$\leftrightarrow$	V	F	I
V	V	F	F
F	F	V	F
I	F	F	V

Définition VII.14 : [Del 87b, 88b, 90c, Fit 85, Thi 90]

Soit P un programme trivalué.

1- On définit le **complété** - $\neg$ de P par :

$$\text{comp}(\neg, P) = P \cup \{ \neg L \leftarrow \neg F ; L \leftarrow F \in P \}$$

2- On définit le complété - $\leftrightarrow$ de P par :

$$\text{comp}(\leftrightarrow, P) = \{L \leftrightarrow F ; L \leftarrow F \in P\}$$

Ce dernier complété est le complété de Fitting de P .

Exemple :

Soit P le programme trivalué $\begin{cases} A \leftarrow B \\ B \leftarrow B \end{cases}$

$$\text{comp}(\neg, P) = \begin{cases} A \leftarrow B \\ \neg A \leftarrow \neg B \\ B \leftarrow B \\ \neg B \leftarrow \neg B \end{cases} \quad \text{et} \quad \text{comp}(\leftrightarrow, P) = \begin{cases} A \leftrightarrow B \\ B \leftrightarrow B \end{cases}$$

On peut remarquer que ces complétés ne sont pas équivalents d'un point de vue trivalué. L'interprétation trivaluée $\{A\}$ est un modèle de $\text{comp}(\neg, P)$ puisque aucun corps d'axiome n'est vrai dans cette interprétation. Et $\{A\}$ n'est pas un modèle du complété de Fitting puisque A et B n'ont pas la même valeur de vérité.

Par contre, tout modèle de $\text{comp}(\leftrightarrow, P)$ est un modèle de $\text{comp}(\neg, P)$ puisque si A et B ont les mêmes valeurs de vérité, $A \leftarrow B$, $\neg A \leftarrow \neg B$ sont vraies.

Il n'y avait pas lieu d'introduire ces deux types de complétés en logique classique (bivaluée) en raison de l'équivalence :

$$A \leftrightarrow B \quad \text{ssi} \quad ((A \Rightarrow B) \wedge (\neg A \Rightarrow \neg B))$$

Définition VII.15 : Soit P un programme trivalué à têtes positives normalisé. Pour tout pré-modèle $\mathcal{M}$ de la CET(L), on définit l'opérateur de conséquence immédiate au sens de Fitting, noté $F_{l,p}^{\mathcal{M}}$ et défini sur les interprétations trivaluées de L sur $\mathcal{M}$ par :

$$F_{l,p}^{\mathcal{M}}(I) = \begin{aligned} &\{(L; \alpha_1, \dots, \alpha_n) ; \quad L(x_1, \dots, x_n) \leftarrow F(x_1, \dots, x_n) \text{ est un axiome de } P \text{ et} \\ &\quad \text{vv}_I(F[\alpha_1, \dots, \alpha_n]) = \text{Vrai}\} \cup \\ &\{(\neg L; \alpha_1, \dots, \alpha_n) ; \quad L(x_1, \dots, x_n) \leftarrow F(x_1, \dots, x_n) \text{ est un axiome de } P \text{ et} \\ &\quad \text{vv}_I(F[\alpha_1, \dots, \alpha_n]) = \text{Faux}\} \end{aligned}$$

Remarque : Pour les programmes trivalués à têtes positives "destinés" à être complétés, on peut adopter une convention de normalisation légèrement différente de celle de la définition VII.9 en ne conservant que les axiomes à tête positive; les autres n'apportent jamais aucune information.

Exemple : On peut considérer que le programme normalisé de :

$$\{A \leftarrow B \quad \text{est} \quad \begin{cases} A \leftarrow B \\ B \leftarrow \text{Faux} \end{cases}$$

Proposition VII.14 [Del 87b, 88b, 90c, Thi 90] : Si P est un programme trivalué à têtes positives normalisé alors, pour tout pré-modèle $\mathcal{M}$ de la $CET(\mathcal{L})$:

$$T_{\mathcal{L}, \text{comp}(\neg, P)}^{\mathcal{M}} = F_{\mathcal{L}, P}^{\mathcal{M}}.$$

Démonstration : Soit I une interprétation trivaluée sur $\mathcal{M}$ et soit A un atome clos.

- Si $A \in T_{\mathcal{L}, \text{comp}(\neg, P)}^{\mathcal{M}}(I)$, il existe une instance d'axiome de P de corps F telle que $vv_I(F) = V$. Donc, $A \in F_{\mathcal{L}, P}^{\mathcal{M}}(I)$.

- Si $\neg A \in T_{\mathcal{L}, \text{comp}(\neg, P)}^{\mathcal{M}}(I)$, il existe une instance d'axiome de P de corps F telle que $vv_I(\neg F) = V$, soit $vv_I(F) = F$. Donc, $\neg A \in F_{\mathcal{L}, P}^{\mathcal{M}}(I)$.

L'inclusion inverse se montre de manière analogue. □

Remarque : Si P est un programme à tête positive, $F_{\mathcal{L}, P}$ est une application de $It^{\mathcal{M}}$ vers $It^{\mathcal{M}}$. Elle ne peut pas prendre la valeur Contra puisqu'il faudrait pour cela qu'il existe une interprétation I et une formule F pour laquelle $vv_I(F) = F$ et $vv_I(F) = V$.

Proposition VII.15 : Si P est un programme trivalué à têtes positives, le complété $\text{comp}(\neg, P)$ est un programme trivalué consistant.

Démonstration : En effet, pour tout pré-modèle $\mathcal{M}$ de la $CET(\mathcal{L})$, $T_{\mathcal{L}, \text{comp}(\neg, P)}^{\mathcal{M}} = F_{\mathcal{L}, P}^{\mathcal{M}}$ et d'après la remarque précédente, le plus petit point fixe de $F_{\mathcal{L}, P}^{\mathcal{M}}$ est un modèle de $\text{comp}(\neg, P)$ sur $\mathcal{M}$. □

Proposition VII.16 [Del 87b, 88b, 90c, Thi 90] : Soit P un programme trivalué à têtes positives et $\mathcal{M}$ un pré-modèle de la $CET(\mathcal{L})$. Alors, pour toute interprétation I de $\mathcal{L}$ sur $\mathcal{M}$:

a- I est un modèle trivalué de $\text{comp}(\neg, P)^{CET}$ ssi $F_{\mathcal{L}, P}^{\mathcal{M}}(I) \subset I$.

b- I est un modèle trivalué de $\text{comp}(\leftrightarrow, P)^{CET}$ ssi $F_{\mathcal{L}, P}^{\mathcal{M}}(I) = I$.

Démonstration :

a- I est un modèle trivalué de $\text{comp}(\neg, P)$ si et seulement si pour tout axiome de P de la forme $p(x_1, \dots, x_m) \leftarrow F(x_1, \dots, x_m)$ et tout n-uplet $\alpha_1, \dots, \alpha_m$ de $\mathcal{M}$:

$vv_I(F[\alpha_1, \dots, \alpha_m]) = V$ implique que $(p; \alpha_1, \dots, \alpha_m) \in I$ et

$vv_I(\neg F[\alpha_1, \dots, \alpha_m]) = V$ implique que $(\neg p; \alpha_1, \dots, \alpha_m) \in I$.

Cela revient exactement à dire que $F_{\mathcal{L}, P}^{\mathcal{M}}(I) \subset I$.

b- Démonstration analogue. □

Proposition VII.17 : Soit P un programme trivalué à têtes positives et $\mathcal{M}$ un pré-modèle de la $CET(\mathcal{L})$. Alors, pour toute formule F de $For(\mathcal{L})$:

$$\begin{aligned} \text{comp}(\leftrightarrow, P)^{CET} \models F & \text{ ssi } \text{comp}(\neg, P)^{CET} \models F \\ & \text{ssi } \text{comp}(\neg, P)^{+CET} \models F^+. \end{aligned}$$

Démonstration :

Tout modèle de $\text{comp}(\leftrightarrow, P)^{CET}$ est un modèle de $\text{comp}(\neg, P)^{CET}$ donc

$$\text{comp}(\neg, P)^{CET} \models F \quad \text{implique} \quad \text{comp}(\leftrightarrow, P)^{CET} \models F.$$

Supposons maintenant que $\text{comp}(\leftrightarrow, P) \models F$.

Soit I un modèle de $\text{comp}(\neg, P)$ sur $\mathcal{M}$.

Comme $F_{\mathcal{L},P}^{\mathcal{M}}(I) \subset I$, on en déduit que la suite $[F_{\mathcal{L},P}^{\mathcal{M}}]^\alpha(I)$ où α décrit les ordinaux est une suite décroissante d'interprétations de $\mathcal{L}$ sur $\mathcal{M}$. Elle admet donc un point fixe J qui est un modèle de $\text{comp}(\leftrightarrow, P)$.

On a donc $J \models F$.

On déduit donc de la monotonie de l'application vv (Prop. VII.2) que I est un modèle de F . Donc,

$$\text{comp}(\leftrightarrow, P)^{CET} \models F \quad \text{implique} \quad \text{comp}(\neg, P)^{CET} \models F.$$

La dernière équivalence est une application immédiate des propositions VII.14 et VII.15. □

Remarque : La proposition précédente montre que, si les deux complétés $\text{comp}(\leftrightarrow, P)$ et $\text{comp}(\neg, P)$ sont différents d'un point de vue trivalué, par contre, si l'on se limite aux conséquences (au sens trivalué) des formules **n'utilisant que des connecteurs bivalués**, ces deux systèmes d'axiomes sont équivalents.

Nous pensons développer cette remarque ultérieurement. Il nous semble que le cadre logique qui rend le mieux compte de la sémantique de Prolog peut être défini en "quotientant" la logique trivaluée par la relation d'équivalence : "deux formules F et G sont équivalentes ssi les formules *monotones*¹ conséquences (au sens trivalué) de l'une sont aussi conséquences de l'autre". Dans cette logique (trivaluée monotone ?), les complétés $\text{comp}(\leftrightarrow, P)$ et $\text{comp}(\neg, P)$ d'un programme trivalué à têtes positives P seraient égaux.

On peut maintenant retrouver le résultat de Kunen [Kun 87a].

Théorème VII.3 : Soit P un programme trivalué à têtes positives. Alors, pour toute formule F de $For(\mathcal{L})$:

$$\text{si } \text{comp}(\leftrightarrow, P)^{CET} \models F \text{ alors il existe un entier } n \text{ tel que } F_{\mathcal{L},P} \uparrow n \models F.$$

Démonstration : Si $\text{comp}(\leftrightarrow, P)^{CET} \models F$ alors $\text{comp}(\neg, P)^{CET} \models F$.

Le théorème précédent montre alors qu'il existe un entier n pour lequel :

$$T_{\mathcal{L}, \text{comp}(\neg, P)} \uparrow n \models F \text{ soit } F_{\mathcal{L},P} \uparrow n \models F. \quad \square$$

¹C.-à-d. pour lesquelles l'application vv est monotone.

Théorème VII.4 : Si P un programme trivalué à têtes positives et si L_K est un langage de Kunen alors, pour toute formule F de $\text{For}(L_K)$:

$$\text{comp}(\leftrightarrow, P)^{\text{CET}} \models F \text{ ssi il existe un entier } n \text{ tel que } F_{L_P} \uparrow n \models F.$$

Démonstration : Ce résultat résulte directement du théorème VII.2. □

VII.4 Complétude de la résolution SLDNF

Kunen a démontré dans [Kun 89] un résultat de complétude de la résolution SLDNF pour les programmes et les buts normaux autorisés.

Rappelons quelques définitions avant d'énoncer ce résultat.

Définitions VII.13 : Soit L un langage du premier ordre.

Un programme **normal** construit sur L est une suite finie de règles de la forme :

$$A :- L_1, \dots, L_n \quad (n \geq 0)$$

où A est un atome de L et $L_1, \dots, L_n$ sont des littéraux (positifs ou négatifs) de L .

Un but **normal** est représenté par :

$$:- L_1, \dots, L_n \quad (n \geq 1)$$

où $L_1, \dots, L_n$ sont des littéraux de L .

Un programme normal est **autorisé** (allowed) si pour chaque règle $R = A :- L_1, \dots, L_n$ de P , toute variable figurant dans R apparaît dans au moins un littéral positif L_i de cette règle.

Un but normal $G = :- L_1, \dots, L_n$ est **autorisé** si toute variable figurant dans G apparaît dans au moins un littéral positif L_i de ce but.

Remarque : Si P est un programme autorisé, les règles unités de P sont toutes de la forme $A :-$ où A est un atome clos.

Exemple :

$\begin{cases} A(y) :- B(x), \neg C(x) \\ C(a) :- \end{cases}$ n'est pas un programme normal autorisé.

$\begin{cases} A(x) :- B(x), \neg C(x) \\ C(a) :- \end{cases}$ par contre, en est un.

Ne nous servant pas explicitement de la définition de la SLDNF, nous renvoyons à [Llo 87] pour cette définition (ou [Kun 89] pour une présentation particulièrement succincte).

A tout programme normal P , on peut associer un programme trivalué P_t de la manière suivante :

$$A :- L_1, \dots, L_n \in P \text{ ssi } \forall (A \leftarrow L_1 \wedge \dots \wedge L_n) \in P_t.$$

On peut également associer à tout programme normal P normalisé un programme dédoublé. Cette traduction a été définie dans [Gil 89, Ble 90] où elle est notée $\tau(P^*)$. Nous préférons la noter P_{SSId} par cohérence avec nos notations précédentes. Nous la définissons dans notre formalisme de la manière suivante :

$$P_{SSId} = \text{comp}(\neg, P_t)^+.$$

Exemple : Soit P le programme

$$\begin{cases} A(x) :- B(x), \neg C(x) \\ C(a) :- \end{cases}$$

$$P_t = \begin{cases} A(x) \leftarrow B(x) \wedge \neg C(x) \\ C(a) \end{cases}$$

$$P_{SSId} = \begin{cases} A(x) \Leftarrow B(x) \wedge C'(x) \\ A'(x) \Leftarrow B'(x) \vee C(x) \\ C(x) \Leftarrow x = a \\ C'(x) \Leftarrow x \neq a \end{cases}$$

Le résultat de complétude de [Kun 89] est le suivant :

Théorème VII.5 : Soit P un programme normal autorisé, $G = :- L_1, \dots, L_n$ un but normal autorisé et σ une substitution des variables de G .

- si $\text{comp}(\leftrightarrow, P_t)^{CET} \models (L_1 \wedge \dots \wedge L_n)\sigma$ alors σ est une réponse calculée par la SLDNF pour $P \cup \{G\}$

- si $\text{comp}(\leftrightarrow, P_t)^{CET} \models \neg \exists (L_1 \wedge \dots \wedge L_n)$ alors $P \cup \{G\}$ a un arbre SLDNF d'échecs finis.

Remarque : Les hypothèses sur le programme implique que dans le premier cas, σ soit une substitution close.

Nous pouvons déduire de ce théorème et de l'ensemble des résultats de ce chapitre un théorème de complétude de la SLDNF pour la traduction P_{SSId} d'un programme normal autorisé.

Théorème VII.6 : Soit P un programme normal autorisé, $G = :- L_1, \dots, L_n$ un but normal autorisé et σ une substitution des variables de G .

- si $P_{SSId}^{CET} \models (L_1 \wedge \dots \wedge L_n)^+\sigma$ alors σ est une réponse calculée par la SLDNF pour $P \cup \{G\}$

- si $P_{SSId}^{CET} \models \neg \exists (L_1 \wedge \dots \wedge L_n)^+$ alors $P \cup \{G\}$ a un arbre SLDNF d'échecs finis.

Démonstration :

En effet, si F est une formule close de $\mathbf{L}$, on a :

$\text{comp}(\leftrightarrow, P_i)^{\text{CET}} \models F$ ssi $\text{comp}(\neg, P_i)^{\text{CET}} \models F^+$ d'après la proposition VII.17, c'est-à-dire :

$$\text{comp}(\leftrightarrow, P_i)^{\text{CET}} \models F \quad \text{ssi} \quad P_{\text{SSId}}^{\text{CET}} \models F^+.$$

On peut donc appliquer le théorème VII.5.

En conclusion : La technique de dédoublement des symboles de prédicats présente, en plus des avantages mentionnés dans l'introduction (rester dans le cadre de la logique classique, permettre des traductions axiomatiques sous forme de programmes généraux positifs ayant de bonnes propriétés, bonne expressivité) celui de rendre compte des résultats les plus intéressants obtenus en programmation logique dans le cadre de la logique trivaluée.

Elle offre en particulier un outil permettant d'obtenir des **caractérisations purement logiques** de propriétés opérationnelles (et pas seulement des sémantiques de types ensemblistes) sans adopter pour autant une logique non classique nécessitant un investissement théorique important.

Néanmoins, la logique trivaluée nous semble avoir une "sémantique apparente" plus satisfaisante que celle fournie par la technique de dédoublement : $\neg A$ représente mieux et plus naturellement la négation de A que A' . La logique trivaluée peut être alors vue comme un cadre logique général dans lequel peut s'exprimer de manière élégante et uniforme tous les résultats obtenus par dédoublement des symboles de prédicats (sous réserve de propriétés de consistance). Nous avons déjà noté que d'autres logiques non classiques (la logique modale [Bal 90] ou la logique linéaire [Cer 90] par exemple) permettent également ce type d'approche.

Perspectives

Nous nous sommes proposé, au début de ce travail, d'étendre aux interpréteurs standards, les résultats sur la sémantique axiomatique des programmes Prolog pur évalués par des interpréteurs idéaux. Nous avons pour cela défini les sémantiques opérationnelles **standard** et **standard finie** d'un programme Prolog et décrit pour chacune d'elles une sémantique axiomatique associée. Nous avons montré que cette caractérisation axiomatique de la sémantique **standard finie** est correcte et complète. Ceci montre qu'il est possible de tenir compte des stratégies retenues lors de l'implémentation des interpréteurs Prolog standards sans pour autant quitter l'environnement théorique dans lequel les résultats classiques sur les interpréteurs idéaux ont été obtenus. Nous avons également défini la sémantique opérationnelle **invariante finie** d'un programme Prolog qui décrit un "noyau" calculable par tout interpréteur Prolog et nous en avons aussi donné une caractérisation axiomatique correcte et complète.

Parmi les questions qui restent posées :

- est-il possible d'obtenir une caractérisation axiomatique "simple", correcte et complète de la sémantique opérationnelle **standard** ?
- plus généralement, étant donné un interpréteur Prolog quelconque défini par certaines stratégies, est-il possible de lui associer de manière simple et constructive une description axiomatique correcte et complète ?

Une méthode générale associant une traduction axiomatique à un interpréteur permettrait d'unifier les démonstrations des chapitres III et IV.

Les programmes Prolog que nous avons considérés sont **purs** en ce sens que les corps des clauses qui les composent ne contiennent ni littéraux négatifs, ni prédicats non logiques tels que CUT, VAR, ASSERT, etc. La question de savoir si les traductions axiomatiques que nous avons définies peuvent être étendues à des programmes plus généraux se pose naturellement. Elle est dans la lignée de la question initiale : prendre en compte les implémentations **réelles** des interpréteurs Prolog.

Les sémantiques opérationnelles que nous avons envisagées sont décrites par des ensembles d'atomes clos. On peut détailler plus finement le comportement des interpréteurs Prolog à l'aide de sémantiques représentables par des ensembles d'atomes quelconques [Der 87, Fer 86, 87, Fal 89a, 89b]. Est-il possible d'obtenir des résultats analogues à ceux de [Fal 89b] en tenant compte des implémentations particulières des interpréteurs Prolog ?

Le théorème de complétude du chapitre V (et du chapitre VII sous sa forme trivaluee) a une portée qui dépasse largement les applications que nous en avons tirées.

C'est un résultat qui relie les notions de **formules calculables** et de **formules conséquences** pour des classes générales de formules et de programmes dans le cadre de la Théorie Equationnelle de Clark (qui constitue le cadre logique minimal qui autorise la procédure d'unification). Ce théorème montre que tout procédé de calcul consistant à effectuer un chaînage avant sur un programme général positif (resp. trivalué consistant) a une caractérisation logique (resp. en logique trivaluée) immédiate. On peut donc concevoir d'autres systèmes de programmation logique que Prolog, par exemple : un interpréteur trivalué fonctionnant en chaînage avant sur une partie d'un programme et utilisant la SLDNF sur un sous-programme autorisé ... Des extensions de la SLDNF ont été étudiées dans [Bar 88, Lug 88].

D'autre part, une des conclusions de notre travail (et plus particulièrement du théorème de complétude) est que la Théorie Equationnelle de Clark représente le cadre logique le plus performant pour étudier la sémantique des programmes Prolog. Cette théorie a déjà été étudiée de manière très approfondie dans [Cla 78, Com 88, Kun 87a, 87b, Mal 61, Mah 88] et il nous semble possible que les techniques utilisées dans ces travaux permettent d'unifier (et donc de simplifier) certaines démonstrations du chapitre VI. Une autre question se pose naturellement : peut-on généraliser ce théorème de complétude à d'autres classes de théories équationnelles (et à d'autres procédés de calculs que ceux basés sur l'unification) ?

Les logiques trivaluées, linéaires, modales rendent bien compte des problèmes rencontrés en programmation logique. Elles ont toutes par ailleurs une portée beaucoup plus générale. Nous avons établi au chapitre VII un lexique entre "technique de dédoublement des symboles de prédicats" et logique trivaluée et montré que pour ce qui est de l'application de la logique trivaluée à l'étude de la sémantique de Prolog, les deux approches sont équivalentes. Peut-on également "plonger" cette technique de dédoublement des symboles de prédicats dans les logiques linéaires et modales ? Ces logiques apportent-elles quelque chose de plus que cette technique (pour la sémantique de Prolog) ? Peut-on identifier la logique la mieux adaptée à la sémantique de Prolog ?

Toutes ces questions définissent autant de prolongements possibles de notre travail.

Bibliographie

- [Apt 82] K. APT and M. VAN EMDEN, *Contribution to the Theory of Logic Programming*, Journal of the Association for Computing Machinery, 29.3, 1982, pp. 841-862.
- [Apt 88] K. APT, H. BLAIR, A. WALKER, *Towards a Theory of Declarative Knowledge* in : J. Minker, ed., *Fundation of Deductive Databases and Logic Programming* (Morgan Kaufman, Los Altos, (1988), pp. 89-142.
- [Arb 87] B. ARBAB and D. M. BERRY, *Operational and denotational semantics of Prolog*, J. Logic Programming 1987;4/309-329.
- [Bal 90] P. BALBIANI, *Une caractérisation modale de la sémantique des programmes logiques avec négation*, Thèse de l'Université de Toulouse, 1990.
- [Bar 88] R. BARBUTI, P. MANCARELLA, D. PEDRESHI, F. TURINI, *A Transformational approach to Negation in Logic Programming*, J. Logic Programming, 1990 : 8 : 201-228.
- [Bek 86] Y. BEKKERS, B. CARRET, O. RIDOUX, O. UNGARO : MALI, *A memory with Real-Time Garbage Collector for Implementing Logic Programming Languages*, Proc. of the 3rd Symposium on Logic Programming, IEEE, Salt Lake City, USA 86.
- [Bid 89] N. BIDOIT, *Negation in Rule-Based Database Language : A Survey*, Thèse d'Etat, Université d'Orsay, 1989.
- [Bil 88] M. BILLAUD, *Simple Operational and Denotational Semantics for Prolog with cut*, Informatica 88, 1988.
- [Ble 90] N. BLEUZEN-GUERNALEC, B. GIL, *A semantical approach of an Internal Rendering of the Negation in General Logic Programs*, Department of Mathematics and Computer Science, University of Aix-Marseille II, Rapport de recherche.
- [Bör 90] E. BÖRGER, *A logical operational semantics of full Prolog*, MFCS 90, Lecture Notes in Computer Science, 452, Springer Verlag.
- [Cer 89] S. CERRITO, *Negation as Failure : A Linear Axiomatization*, Rapport Interne, LRI n° 434, Université Paris X.
- [Cer 90] S. CERRITO, *A linear Semantics for Allowed Logic Programs*, IEEE pp. 219-227, 1990.
- [Cha 73] C. C. CHANG, H.J. KEISLER, *Model Theory*, North-Holland, Amsterdam, 1973.
- [Cla 78] K. L. CLARK, *Negation as Failure. Logic and Data Bases*, H. GALLAIRE and J. MINKER ed., Plenum Press, New-York, 1978, pp. 293-324.

- [Col 73] A. COLMERAUER, H. KANOUI, P. ROUSSEL et R. PASERO, *Un système de communication Homme-Machine en Français*, Groupe de Recherche en Intelligence Artificielle, Université d'Aix-Marseille, 1973.
- [Com 88] H. COMON, *Unification et disunification. Théorie et applications*. Thèse de doctorat de l'Institut National Polytechnique de Grenoble (1988).
- [Dav 60] M. DAVIS, H. PUTNAM, *A Computing Procedure for Quantification Theory*, J. ACM 7 (1960), pp. 201-215.
- [Deb 87] S.K. DEBRAY, P. MISHRA, *Denotational and Operational Semantics for Prolog*, Formal Description for Programming Concept III, IFIP, 1987.
- [Del 86] J.-P. DELAHAYE, P. PARADINAS, *Définitions de stratégies équitables en programmation logique*, Actes du séminaire de programmation logique de Trégastel, 1986, pp. 7-24.
- [Del 87a] J.-P. DELAHAYE, *Sémantique et complétude du chaînage avant en calcul propositionnel*, Publication interne, IT n° 92, Université des Sciences et Techniques de Lille, Flandres, Artois, Janvier 1987.
- [Del 87b] J.-P. DELAHAYE, *Programmation en logique trivaluée*, Publication interne, IT n° 115, Université des Sciences et Techniques de Lille, Flandres, Artois, Octobre 1987.
- [Del 87c] J.-P. DELAHAYE, *Chaînage avant et calculs de modèles en logique bivaluée et trivaluée*, 7^e journées sur les systèmes experts, Avignon, 1987.
- [Del 87d] J.-P. DELAHAYE, *Outils logiques pour l'intelligence artificielle*, Editions Eyrolles, 1987.
- [Del 88a] J.-P. DELAHAYE, *Sémantique Logique et Dénotationnelle des Interpréteurs Prolog*, Informatique théorique et Applications, vol 22, n°1, 1988, pp. 3 - 42.
- [Del 88b] J.-P. DELAHAYE, V. THIBAU, *Logic programming and trivalued logic*, European Workshop on Logical Methods in Artificial Intelligence. JELIA 88, Roscoff 27-30 Juin 1988, pp. 23-27.
- [Del 90a] J.-P. DELAHAYE, F. DENIS, *Operational Semantics of Standard Prolog : an axiomatic approach*, Actes du Séminaire de Programmation Logique de Trégastel, 1990.
- [Del 90b] J.-P. DELAHAYE, V. THIBAU, *Optimal fixed points and logic programming*, Actes du séminaire de Programmation Logique du CNET 1990.
- [Del 90c] J.-P. DELAHAYE, V. THIBAU, *Programming in three-valued logic*, Theoretical Computer Science, 1990, à paraître.
- [Den 89] F. DENIS, *Approximations de la sémantique opérationnelle d'un interpréteur Prolog Standard par des sémantiques axiomatiques*, Rapport de recherche, septembre 1989, LIFL, Bât. M3, USTL, 59655 Villeneuve d'Ascq, CEDEX.
- [Den 90a] F. DENIS, J.-P. DELAHAYE, *Is there an axiomatic semantics for Standard pure Prolog ?*, Rapport de recherche, mai 1990, LIFL, Bât. M3, USTL, 59655 Villeneuve d'Ascq, CEDEX à paraître dans Theoretical Computer Science.

[Den 90b] F. DENIS, J.-P. DELAHAYE, *Unfolding, Procedural and Fixpoint Semantics of Logic Programs*, Rapport de Recherche, septembre 1990, IT n° 192, LIFL, Bât. M3, USTL, 59655 Villeneuve d'Ascq, CEDEX (ou Proceeding of STACS 91).

[Den 90c] F. DENIS, J.-P. DELAHAYE, *Un théorème de complétude. Applications à la logique trivaluée et à la sémantique des langages à base de règles*, Rapport de Recherche, septembre 1990, IT n° 191, LIFL, Bât. M3, USTL, 59655 Villeneuve d'Ascq, CEDEX

[Der 87] P. DERANSART, G. FERRAND, *Programmation en Logique avec Négation : Présentation Formelle*, INRIA-Université d'Orléans, Faculté des Sciences, BP 6759, 45067 ORLEANS Cédex 2.

[Emd 76] M. H. VAN EMDEN et R. A. KOWALSKI, *The Semantics of Predicate Logic as a Programming Language*, Journal of the Association for Computing Machinery, vol. 23-4, 1976, pp. 733-742.

[Fal 89a] M. FALASCHI, G. LEVI, M. MARTELLI, C. PALAMIDESI, *Declarative modeling of the operational behaviour of logic languages*, Theoretical Computer Science 69, (1989), pp 289-318, North-Holland.

[Fal 89b] M. FALASCHI, G. LEVI, M. MARTELLI, C. PALAMIDESI, *A model-Theoretic Reconstruction of the Operational semantics of Logic Programs*, Università degli studi di Pisa, Dipartimento di informatica, TR-32/89.

[Fef 82] S. FEFERMAN, *Toward Useful Type-Free Theories*, Journal of Symbolic Logic, 1982.

[Fer 86] G. FERRAND, *A Reconstruction of Logic Programming with Negation*, Laboratoire d'Informatique de la Faculté des Sciences d'Orléans (1986).

[Fer 87] G. FERRAND, *Error diagnosis in Logic Programming, an Adaptation of E. Y. Shapiro's method*, J. Logic Programming 1987:4:177-198.

[Fit 85] M. FITTING, *A deterministic Prolog fixpoint semantics*, The Journal of Logic Programming, 1985 : 2, pp. 111-118.

[Fit 85] M. FITTING, *A Kripke-Kleene semantics for logic programs*, J. Logic Programming, 4 (1985) pp. 295-312.

[Fit 86] M. FITTING, *Notes on the Mathematical Aspects of Kripke's Theory of Truth*, Notre Dame Journal of Formal Logic, Volume 27, Number 1, January 1986.

[Gil 89] B. GIL, *Complete extension of General Logic Programs*, Proceeding of the 8th Symposium on Mathematics and Computer Science, LNCS, Springer-Verlag.

[Gilm 60] P.C. GILMORE, *A Proof Method for Quantification Theory*, IBM J. Res. Develop. 4 (1960), pp. 28-35.

[Gilm 74] P.C. GILMORE, *The consistency of partial set theory without extensionality*, Axiomatic Set Theory, Proc. symposia Pure Maths, Part II, Amer. Math. Soc. 147-153.

[Gue 88] I. GUESSARIAN, *Some fixpoint techniques in algebraic structures and applications to computer science*, Rapport LITP, Université Paris VII, 1988.

- [Her 30] J. HERBRAND, *Investigations in Proof Theory*, in From Frege to Gödel : A Source Book in Mathematical Logic, 1879, 1931, van Heijenoort, J. (ed.), Harvard University Press, Cambridge, Mass., 1967, pp. 525-581.
- [Hil 74] R. HILL, *Lush-resolution and its completeness*, Depart. of Artificial Intelligence, Univ. of Edinburgh, 1974.
- [Jaf 83] J. JAFFAR, J.L. LASSEZ, L. LLOYD, *Completeness of the negation as failure rule*, IJCAI-83, Karlsruhe, 1983, pp.500-506.
- [Jan 90] S. JANOT, S. LE HUITOUZE, *Definition of fair Prolog Interpreters and implementation with MALI*, Rapport de recherche IT n° 195, octobre 1990, LIFL, Bât. M3, USTL, 59655 Villeneuve d'Ascq, CEDEX.
- [Jon 84] N.D. JONES, A. MYCROFT, *Stepwise Development of Operational and Denotational Semantics for Prolog*, International Symposium on Logic Programming, 1984.
- [Jou 90] J.-P. JOUANNAUD, *Syntactic Theories*, MFCS 90, in Lectures Notes in Computer Science 452, Springer Verlag.
- [Kan 87] T. KANAMORI, K. HORIUCHI, *Construction of Logic programs Based on Generalized Unfold/Fold Rules*, Proc. 4th Int. Conf. on Logic Programming, The MIT Press, Series in Logic Programming (1988).
- [Kan 89] K. KANCHANASUT, P. STUCKEY, *Eliminating negation from normal logic programs*, Department of Computer Science, University of Melbourne, Parkville 3052, Australia.
- [Kna 28] B. KNASTER, *Un théorème sur les fonctions d'ensembles*, Annales de la Société Polonaise de Mathématiques, vol. 6 (1928), pp. 133-134.
- [Kow 71] R. A. KOWALSKI, *Predicate Logic as a Programming Language*, Information Processing 74, Stockholm, North Holland, 1974, pp. 569-574.
- [Kun 87a] K. KUNEN, *Negation in Logic Programming*, Journal of Logic Programming, 4, 1987, pp. 289-308.
- [Kun 87b] K. KUNEN, *Answer sets and Negation as Failure*, Proc. 4th Int. Conf. on Logic Programming, Melbourne, 1987.
- [Kun 89] K. KUNEN, *Signed Data Dependencies in Logic Programs*, Journal of Logic Programming, 7, 1989, pp. 231-245.
- [Lan 90] T. LANGHOLM, *What is a Horn Clause in Partial Logic*, draft paper, 1990.
- [Las 84] J.L. LASSEZ, M.J. MAHER, *Closures and Fairness in Semantics of Programming Logic*, T.C.S., 29, 1984, pp. 167-184.
- [Lev 90] G. LEVI, M. MARTELLI, C. PALAMIDESSI, *Failure and Success made Symmetric*, Università degli studi di Pisa, Dipartimento di informatica.
- [Llo 87] J. LLOYD, *Foundation of Logic Programming*, Springer-Verlag, New-York, Second Edition 1987.

- [Llo 87] J.W. LLOYD, J.C. SHEPHERDSON, *Partial evaluation in Logic Programming*. Technical report CS-87-09, Dept. of Computer Science, University of Bristol (1987).
- [Lug 89] D. LUGIEZ, *A Deduction Procedure for First Order Programs*, in ICLP 1989, edited by G. LEVI and M. MARTELLI, The MIT Press, Cambridge, Massachusetts, pp. 585-599.
- [Mah 87] M.J. MAHER, *Equivalences of Logic Programs*, in : J. Minker, ed., *Foundation of Deductive Databases and Logic Programming* (Morgan Kaufman, Los Altos, (1988), pp. 627-658.
- [Mah 88] M.J. MAHER, *Complete axiomatizations of the Algebras of Finite, Rational and Infinite Trees*, 3th LICS, Edinburgh, 1988.
- [Mal 71] A. MAL'CEV, *Axiomatisable Classes of Locally Free Algebras of Various Types*, in : *The metamathematics of Algebraic Systems : Collected Papers* chapter 23, 262-281, 1971.
- [Mar 89] MAREK, *The relationship between logic program semantics and non monotonic reasoning*, in ICLP 1989, edited by G. LEVI and M. MARTELLI, The MIT Press, Cambridge, Massachusetts, pp. 600-620.
- [Myc 84] A. MYCROFT, *Logic programs and many-valued logic*, Proc. first STACS Conf., (Eds. M. Fontet and K. Mehlhorn), *Lecture Notes in Computer Science* 166, 1984, pp. 274-286.
- [Pab 76] J.-F. PABION, *Logique Mathématique*, 1976, Hermann, Paris.
- [Pra 60] D. PRAWITZ, *An Improved Proof Procedure*, *Theoria* 26 (1960), pp. 102-139.
- [Prz 89] T. C. PRZYMUNSINSKI, *Non-monotonic formalisms and logic programming*, in ICLP 1989, edited by G. LEVI and M. MARTELLI, The MIT Press, Cambridge, Massachusetts, pp. 655-674.
- [Reg 88] L. REGNIER, *Logique linéaire et Prolog*, 7^{ème} séminaire de Programmation Logique de Trégastel, 1988.
- [Ric 89] G. RICHARD, *Contribution à la réalisation d'une spécification formelle pour Prolog*, Thèse à l'Université d'Orléans, 1989.
- [Rob 65] J. A. ROBINSON, *A Machine Oriented Logic Based on the Resolution Principle*, *J. of A.C.M.*, vol. 12, 1965.
- [Rou 75] P. ROUSSEL, *PROLOG : Manuel de référence et d'utilisation*, Groupe d'Intelligence Artificielle, Aix-Marseille, 1975.
- [She 84] J.C. SHEPHERDSON, *Negation as Failure I*, *J. of Logic Programming*, 1984, 1, pp. 51-79.
- [She 85] J.C. SHEPHERDSON, *Negation as Failure II*, *J. of Logic Programming*, 1985, 1, pp. 185-202.
- [She 87] J. C. SHEPHERDSON, *Negation in logic programming*, J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 19-88, Morgan Kaufmann, Los Altos, CA., 1987.

- [Tar 55] A. TARSKI, *A lattice-theoretical fixpoint Theorem and its applications*, Pacific Journal of Mathematics, vol. 5 (1955), pp. 285-309.
- [Thi 90] V. THIBAU, *Une logique trivaluée appliquée à la programmation logique*, Thèse de l'Université des Sciences et Techniques de Lille, Flandres, Artois, 1990.
- [Yas 71] A. YASUHARA, *Theory of recursive functions*, Academic Press, New York, 1971.

Index

$\mathbf{L}, \mathbf{L}_d$	12	P_{fst}	77
$\models$	12	SS_{fst}^n, FF_{st}^n (pour la	80
$T \uparrow \alpha, T \downarrow \alpha$	13	profondeur standard finie)	
pppf, pgpf	14	$[L]_f^{SF}, [L]_f^{EF}, [L]_f^F$	88
U_L, U	14	C^S, C^{EF}, C^F (pour P_f)	88
H_L, H	14	P_f	89
$Her(L), Ax^{Her}$	16	SS_f^n, FF_f^n (pour la	91
$CET(L), Ax^{CET}$	17	profondeur maximale)	
$\mathcal{U}_L, \mathcal{U}$	17	$\mathbf{L}_Y, \mathcal{U}_Y$	103
U^M, H^M	18	f_D	104
L_k	19	$\Pi_{DA_i}, \Pi_D \mathcal{U}_i$	104
$\mathcal{M}_i$	20	ν_n, ν_ω	110
T_P^M, T^M, T	23	It^M, Id^M	140
$\mathcal{M}_n$	23	Contra	141
$Dom(\theta), Im(\theta)$	27	It_∞^M, Id_∞^M	141
upg	28	$For(L), For^+(L), F^+$	141
$s=t, s \neq t, [x \backslash a]$	29	Ax^+	142
$G \xrightarrow[\mathcal{C}]{\theta} G'$	31	$\forall \mathcal{V}_I$	142
SS, FF	34	CONS	144
SS_{st}, FF_{st}	35	$\models_i$	145
SS_{fst}, SS_f, FF_f	36	$\leftarrow$	145
P_{SI}	41	$T_{i,p}^M, T_{d,p}^M$	148
P_{SSI}	42	$\leftrightarrow$	154
P_{SSId}	43	$comp(\neg, P), comp(\leftrightarrow, P)$	154
P_θ	45	$F_{i,p}^M$	155
$[L]_\theta^S, [L]_\theta^{EF}$	45	arbre d'échecs finis	32
C^S, C^{EF} (pour P_θ)	45	arbre de dérivation SLD	32
H_1	51	base de Herbrand	14
H_2	53	but	27
H_3	54	but autorisé	158
L_i, H_i, E''	58	but défini	16
$[L]_{st}^S, [L]_{st}^{EF}, [L]_{st}^F$	58	but normal	158
C^S, C^{EF}, C^F (pour P_{st})	59	clause définie	16
P_{st}	59	clause unité	16
$SS_{st}^n, SS_{fst}^n, FF_{st}^n$ (pour la	63	clause vide	32
profondeur standard)		complété de Fitting	155
H_3	71	complétude ($T \uparrow \omega$)	40
$[L]_{fst}^{SF}, [L]_{fst}^{EF}, [L]_{fst}^F$	76	conséquence logique trivaluée	145
C^S, C^{EF}, C^F (pour P_{fst})	77	consistant	103

continu	14	réponse calculée	31
correction ($T \uparrow \omega$)	40	règle Prolog	26
dérivable	31	règle unité	26
dérivation SLD	31	saturé ($-\omega$)	105
dirigée (partie)	13	substitution	27
échec invariant fini	36	substitution de renommage	28
échec standard fini	35	succès invariant fini	36
équitable	32	succès standard	35
filtre	104	succès standard fini	36
filtre non clos par intersection		théorie équationnelle de Clark	17
dénombrable	104	treillis complet	13
formule équationnelle	17	ultraproduit	104
formule négative	24	ultrapuissance	104
formule positive	24	unificateur	28
formules dépliées	106	univers de Herbrand	14
interprétation de Herbrand	14	variable locale	27
interprétation dédoublée	140	variantes	28
interprétation sur un pré-modèle	19		
interprétation trivaluée	140		
interpréteur idéal	32		
interpréteur standard	32		
langage de Kunen	19		
littéral	16		
modèle de Herbrand	15		
modèle dédoublé	144		
modèle trivalué	144		
monotone	13		
opérateur de conséquence			
immédiate	23		
point fixe	14		
pré-modèle	17		
pré-modèle standard	17		
produit réduit	104		
profondeur maximale	90		
profondeur standard	62		
profondeur standard finie	80		
programme autorisé	158		
programme à têtes positives	147		
programme défini	16		
programme général	20		
programme général normalisé	22		
programme général positif	24		
programme normal	158		
programme Prolog	26		
programme trivalué	145		
programme trivalué consistant	151		
programme trivalué normalisé	146		
réfutation	31		



TOWARDS THE STUDY OF AXIOMATIC SEMANTICS WITHIN LOGIC PROGRAMMING

The operational semantics of definite Prolog programs usually studied is suitable for "ideal" Prolog systems, that is which only generate fair SLD-trees and are provided with a breadth-first search rule. Hence, the very complete theoretical results obtained until now are not adapted to the usual standard Prolog Systems. We define new operational semantics adapted to standard Prolog systems or, more generally, which describe what can be computed whatever Prolog system is used. Next, we show that these semantics can be characterized by means of systems of axioms. These results constitute a more realistic version of classical results. In this work, we use a "predicates symbols doubling technique". We study the relationship between this technique and the three-valued logic. We prove completeness theorems in classical and three-valued logics for general classes of logic programs (including numerous axiomatic translations which occur in logic programming) in the framework of the Clark's Equational Theory.



Résumé :

La sémantique opérationnelle des programmes Prolog (définis) la plus fréquemment étudiée fait implicitement l'hypothèse que ces programmes sont évalués par des interpréteurs idéaux (en largeur d'abord et équitables). En conséquence, les résultats théoriques très complets qui ont été obtenus jusqu'à présent ne s'appliquent pas aux interpréteurs standard usuels. Nous définissons de nouvelles sémantiques opérationnelles, adaptées aux interpréteurs standards ou plus généralement, décrivant un noyau calculable quel que soit l'interpréteur envisagé. Nous montrons ensuite que ces sémantiques peuvent être caractérisées axiomatiquement. Ces résultats constituent donc une version plus réaliste et plus générale des résultats classiques. Nous avons utilisé, au cours de cette étude, une technique de dédoublement des symboles de prédicats dont nous avons étudié systématiquement les rapports avec la logique trivaluée. Nous avons également démontré des théorèmes de complétude en logiques bivaluée et trivaluée s'appliquant à des classes générales de programmes logiques (comprenant bon nombre des traductions axiomatiques rencontrées en programmation logique) dans le cadre de la Théorie Equationnelle de Clark.

Mots clés :

COMPLETUE
INTERPRETEUR PROLOG STANDARD
LOGIQUE TRIVALUEE
PROGRAMMATION LOGIQUE

PROGRAMME DEFINI
PROLOG
SEMANTIQUE AXIOMATIQUE
THEORIE EQUATIONNELLE DE
CLARK