Processus communicants : un langage formel et ses modèles : problèmes d'analyse

Juan Manuel Pereira-Fernandez

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE
INFORMATIQUE

par

Juan Manuel PEREIRA–FERNANDEZ

PROCESSUS COMMUNICANTS: un langage formel et ses modèles.
PROBLEMES D'ANALYSE.

Thèse soutenue le 8 juin 1984 devant la Commission d'Examen :

Monsieur	J. MOSSIERE	:	Président
Messieurs	Ch. BOITET P. COUSOT Ph. JORRAND J. SIFAKIS	}	Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier

Monsieur J. Mossière, Professeur à l'Institut National Polytechnique de Grenoble (INPG), pour avoir bien voulu accepter de présider le jury de cette thèse. Je lui en suis reconnaissant.

Monsieur Ch. Boitet, Professeur à l'Université Scientifique et Médicale de Grenoble (USMG), qui a toujours été attentif à mon travail, qui a bien voulu corriger la première version de cette thèse et qui a accepté d'être membre du jury.

Monsieur P. Cousot, Professeur à l'Université de Metz, qui s'est intéressé à ce travail en y apportant des idées pour la suite de mes recherches et qui a bien voulu juger cette thèse.

Monsieur Ph. Jorrand, Directeur de Recherche au CNRS qui m'a proposé mon sujet de thèse, pour la confiance qu'il m'a témoignée en m'accueillant dans son groupe de travail, et pour tout le temps qu'il a consacré à de longues discussions constructives où ses critiques et orientations ont toujours été justifiées. Nombre des résultats exposés ici sont le fruit de cette collaboration avec lui.

Monsieur Sifakis, Chargé de Recherche au CNRS, qui a analysé la première version de ce travail et m'a suggéré des modifications pertinentes, et qui a accepté d'être membre du jury.

Je tiens aussi à remercier Monsieur J. Vidart, Professeur à l'Université "Simón Bolívar" de Caracas (Vénézuela). Il a dirigé mes travaux entre 1977 et 1981 où il m'a appris une certaine façon d'aborder les problèmes scientifiques. C'est lui qui m'a orienté vers l'équipe dans laquelle cette thèse a été réalisée. Il a travaillé pendant son année sabatique avec Mademoiselle M. Cisneros, Monsieur Ph. Jorrand et moi-même dans les premières versions du langage proposé dans cette thèse.

Je ne voudrais pas dissocier de ces remerciements, Messieurs D. Bert, J. Cl. Fernandez, P. Jacquet et P. Lagnier pour leur lecture et commentaires de certaines parties de la thèse.

Tout spécialement, je remercie Monsieur et Madame Marty, je leur dois la "mise en français" de ce travail. Je leur adresse mes plus vifs remerciements.

Enfin, je remercie Mesdemoiselles C. Allosio et C. Maïda pour leur patience dans la dactylographie de cette thèse et tout le personnel du service de reprographie qui ont effectué le tirage.

Cette recherche a été effectuée pendant que j'étais boursier du Centre Régional des Oeuvres Universitaires et Scolaires (CROUS-France) et du "Consejo Nacional de Investigaciones Científicas y Tecnológicas" (CONICIT-Vénézuela) ; sans cette aide financière, ce travail n'aurait pu être accompli.

J. M. Pereira Fernandez

Poincaré disait, avec beaucoup de finesse :

"Il n'y a plus de problèmes résolus et d'autres qui ne le sont pas ; il y a seulement des problèmes plus ou moins résolus."

A Nona y a Marta Cristina

PROCESSUS COMMUNICANTS : UN LANGAGE FORMEL ET SES MODELES.

PROBLEMES D'ANALYSE.

INTRODUCTION

0. INTRODUCTION

Le développement de la technologie des micro-processeurs, dans les dernières années, a orienté la conception des langages de programmation vers les systèmes composés de processus qui fonctionnent en parallèle et qui coopèrent en échangeant des messages.

A l'heure actuelle, la conception, la définition, l'analyse et la vérification de systèmes de processus communicants sont parmi les plus importants sujets de recherche dans la théorie des langages de programmation.

Nous nous proposons, dans une première partie de ce travail (chapitre 1), de contribuer à cette activité. Ainsi, nous proposons un modèle pour la construction de systèmes de processus fondé sur le parallélisme et la communication, qui nous permet la conception et la définition de ces systèmes. Nous pensons que ce modèle est suffisamment général pour permettre, dans l'avenir, l'analyse et la vérification de systèmes de processus communicants.

Les "modèles" de processus communicants que nous proposons comportent un langage formel (niveau syntaxique) pour la description de systèmes de processus communicants et trois interprétations (niveau sémantique) possibles pour ce langage formel.

Ainsi, le langage formel, pour la description de systèmes de processus communicants, peut être interprété comme :

- une spécification équationnelle, comme dans les travaux sur les types abstraits de données [ADJ 78], [Ber 79], [Lis-Zil 77], [Zil 79]. Cela constitue la "sémantique algébrique" du langage, c-à-d que la spécification équationnelle définit une algèbre quotient des termes. Il s'agit donc d'un modèle au sens de la théorie des modèles [Cha-Kei 73], [Rob 63], où le modèle est une interprétation où toutes les équations sont satisfaites (vérifiées).

- un ensemble de règles de calcul ("computation") qui décrivent, ou bien un système de transitions d'états [Kel 76], ou bien un système de transitions de termes. Ces règles de calcul représentent le flot d'événements réalisé par le système de processus, de façon interne (non-visible par le milieu extérieu ou avec le milieu extérieur . Ces événements comportent toute l'information sur les communications qui ont lieu et les messages qui sont transmis ou reçus par le système.

Ces règles de calcul constituent la "sémantique opérationnelle". La première interprétation, c-à-d un système de transitions d'états, est assez classique dans la sémantique opérationnelle. Ce n'est pas le cas de la deuxième interprétation au moyen d'un système de transitions de termes pouvant être considéré comme un système de réécriture de termes avec événements. C'est, à notre connaissance, une idée originale pour exprimer la sémantique opérationnelle de systèmes de processus communicants.

Dans cette sémantique et dans la suivante, on parle de "modèle" au sens d'une "structure mathématique qui représente la réalité".

- un arbre (fini ou infini) utilisé pour décrire le comportement du système de processus du point de vue des connexions avec le milieu extérieur et non du point de vue des messages qui circulent sur les connexions. Cette sémantique est destinée uniquement à l'étude de systèmes de processus où le comportement des suites de connexions ne dépend pas des messages reçus. Ainsi, cette "sémantique arborescente" peut être considérée, en quelque sorte, comme une évaluation symbolique du processus décrit. Cette évaluation est faite par rapport aux messages transmis ou reçus, puisqu'ils seront considérés comme des termes avec variables.

L'une des caractéristiques les plus souhaitables dans les langages de programmation est l'existence de mécanismes d'abstraction. Dans la programmation séquentielle, les types, les classes, les procédures, la récursion, les coroutines, etc., ont permis un degré d'abstraction élevé dans la construction de programmes. Un nouveau style de programmation a été créé pour utiliser, au mieux, ces abstractions.

La programmation de systèmes de processus communicants doit, elle aussi, donner des mécanismes d'abstraction pour la construction de tels systèmes. Les modèles proposés par R. Milner [Mil 80], C.A.R. Hoare [81a], W.C. Rounds - S.D. Brookes [Rou-Bro 81], D. Austry - G. Boudol [Aus-Bou 82] donnent un ensemble d'opérateurs qui permettent la construction de systèmes de processus d'une façon très abstraite. Dans ces modèles, un système de processus est obtenu à partir de processus élémentaires et d'opérateurs.

Nous proposons aussi un ensemble d'opérateurs pour décrire, d'une façon abstraite, la construction des systèmes de processus. Mais, ces opérateurs jouent aussi un autre rôle. Ainsi, l'expression qui décrit un système de processus peut être évaluée pour obtenir la description syntaxique dans notre langage formel. Par conséquent, le sens d'une expression est l'interprétation sémantique de la description syntaxique obtenue par l'évaluation de l'expression dans un modèle.

Nous étudions aussi la puissance d'expression de notre langage formel, autrement dit, ce que nous pouvons calculer.

Du point de vue de l'analyse et de la vérification, nous indiquons les grandes lignes des travaux qu'il serait intéressant d'entreprendre. Certains d'entre eux sont, d'ailleurs, déjà en cours.

Un problème intéressant dans l'analyse et la vérification des systèmes de processus dans notre modèle opérationnel, est celui de l'arrêt d'une règle de transitions de termes avec événements. L'attention, dans cette étude, doit être portée, non seulement sur le fait de savoir quand une règle termine ou non, mais aussi sur la détermination de l'ensemble de termes fermés qui font que la règle ne termine pas. Si cette double caractérisation est possible, alors nous pouvons décider de la divergence dans une large classe de systèmes de processus. Par ailleurs, cette double caractérisation nous permettra aussi d'analyser l'ensemble des valeurs que le milieu extérieur peut introduire dans le système et qui font que le système peut avoir un comportement infini.

C'est précisément l'étude de cette double caractérisation sur les règles de transitions de termes avec événements que nous commençons dans les chapitres 2 et 3.

Ainsi, nous étudions, dans le chapitre 2, la caractérisation des fonctions non-noethériennes définies sur la base finitaire d'un cpo filtré complet. Cette caractérisation doit nous permettre de connaître les conditions nécessaires et suffisantes pour qu'une fonction soit noethérienne, et, dans le cas où elle l'est, de déterminer aussi l'ensemble des valeurs pour lesquelles nous pouvons appliquer la fonction indéfiniment.

Enfin, puisque cette double caractérisation est d'un intérêt plus général que les systèmes de processus communicants, nous étudions les règles de transitions de termes dans le contexte de systèmes de réécriture. Nous prouvons que ces règles peuvent être considérées comme des fonctions sur un cpo filtré complet et nous utilisons les résultats du chapitre 2 pour l'étude de la décidabilité de l'arrêt de ces règles.

C H A P I T R E 1

UN LANGAGE FORMEL ET SES MODELES POUR LA SPECIFICATION DE SYSTEMES DE PROCESSUS COMMUNICANTS

1.1. LES SYSTEMES DE PROCESSUS COMMUNICANTS (SPC). Motivation

Un "processus" peut être considéré comme une boîte (machine, agent) capable de faire des calculs et d'envoyer, et de recevoir des messages à travers certains organes de connexion (ports, canaux, boutons) attachés à la boîte. Ces organes sont appelés "portes" d'entrée et/ou de sortie du processus : ce sont les seuls moyens par lesquels le processus peut transmettre ou recevoir des informations (messages) du milieu dans lequel il agit.

Cette capacité de transmission ou de réception d'informations attribuée aux processus permet de construire des systèmes (réseaux) de processus, tels que :

- les processus échangent entre eux des messages à travers leurs portes : une "communication" est considérée comme le transport instantané d'un message d'une porte de sortie vers une porte d'entrée (non nécessairement du même processus).

- dans ces systèmes, les processus évoluent de façon indépendante, sauf dans les moments dans lesquels ils se communiquent. Cela introduit toutes les possibilités du parallélisme.

L'objectif de ce premier chapitre est de présenter une façon d'exprimer (de dénoter) cette construction de systèmes de processus fondée sur le parallélisme et les communications, qui permette de raisonner sur le fonctionnement du système total. Cet objectif n'est pas nouveau, loin de là puisqu'il bénéficie, actuellement, d'une grande attention parmi les chercheurs. De nombreuses publications sur ce sujet sont parues dernièrement. On peut les classifier selon la spécification formelle du système, de la façon suivante :

- la spécification formelle est donnée dans un langage algorithmique. Voir, par exemple, CSP [Hoa 78], ADA, CCS [Hen-Plo 80].

- la spécification formelle est un système de transitions d'états. Voir, par exemple, les travaux de Ph. Jorrand [Jor 81a,b], R. Milner [Mil 80], R.M. Keller [Kel 76] et J. Sifakis [Sif 80],

- la spécification formelle est donnée dans une algèbre de processus. Voir, par exemple, les travaux de D. Austry - G. Boudol [Aus-Bou 82], R. Milner [Mil 80],

- la spécification formelle est un ensemble de mots infinis. Voir le travail de J. Beauquier - M. Nivat [Bea-Niv 80].

- la spécification formelle est l'ensemble de ses traces. Voir, par exemple, les travaux de C.A.R. Hoare [Hoa 81a], J.R. Abrial [Abr 83].

La plupart de ces publications proposent, ou bien des modèles sémantiques pour l'étude de propriétés du comportement et pour le calcul du système, oubien un langage algorithmique qui permet la construction du système et auquel il faut associer des objets mathématiques qui expriment, d'une façon abstraite, le même comportement que l'algorithme, et sur lesquels on peut raisonner et établir différentes propriétés du système décrit.

L'objectif de la spécification que nous proposons ici est double.

1) - Notre spécification rassemble le niveau de langage algorithmique et le niveau sémantique. Plus précisément, elle propose une description "syntaxique" qui nous permet l'écriture d'un processus. La sémantique de cette écriture peut être définie au moyen de différentes interprétations sémantiques, qui produisent des objets mathématiques (Algèbres, Systèmes de Transitions, Arbres Infinis). Ces objets nous permettront de raisonner et d'analyser différentes propriétés. Par sa structure, cette notation permet aussi une conception algorithmique des processus.

2) - Notre spécification permet de construire un système de processus communicants, comme un cas particulier d'un processus simple. Autrement dit, un SPC est une phrase de la notation "syntaxique" qui ne se différencie pas d'un processus simple.

La construction d'un système de processus devient une manipulation de symboles syntaxiques, aboutissant à une autre phrase syntaxique qui décrit le système.

Cette spécification est fondée sur les idées de Ph. Jorrand [Jor 81a, 81b, 83, 84] et elle a été conçue dans le projet SPARC (Systèmes PARallèles Communicants) pendant la période 1982 - 1983. Une des caractéristiques remarquables de cette spécification est le fait que l'étude des propriétés d'un SPC soit ramenée à celle d'un processus simple.

1.2. DESCRIPTION, CONCEPTION, CONSTRUCTION ET ANALYSE D'UN SPC

De la même façon que la logique peut être considérée selon la formule,

logique = approche syntaxique + approche sémantique

un système de processus communicants peut être décrit par la formule

processus = *description* + *conception*

C'est précisément le développement de plusieurs formules de cette sorte que l'on fait d'abord. Ainsi, nous commençons par préciser la *description* syntaxique formelle d'un processus et nous donnons ensuite plusieurs "sens" formels au mot *conception*. Chaque sens correspond à une formalisation des processus.

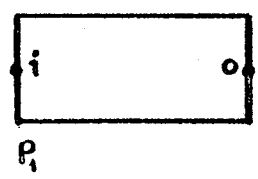
De plus, nous montrons que la *construction* d'un SPC revient à une manipulation syntaxique des descriptions. Cela rejoint notre analogie avec la logique dans le sens où la construction est, en quelque sorte, un système déductif où l'on manipule des phrases (sans considérer de "sens" pour les symbole manipulés) pour déduire des phrases toujours valides.

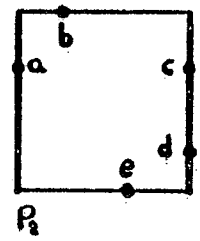
Enfin, nous indiquons quelle sorte d'*analyse* nous pouvons envisager avec chaque formalisation.

1.2.1. Syntaxe et Sémantique Intuitive de Processus

Le vocabulaire d'un langage n'a de signification que si on lui associe une sémantique. Dans les sections suivantes, nous présentons un vocabulaire et ses significations. Cependant, avant de faire cette présentation formelle, nous donnons quelques idées intuitives sur la description des processus.

Comme nous l'avons précisé dans la section 1.1., un processus est une boîte qui calcule et qui utilise ses portes pour communiquer avec le milieu extérieur. Nous supposons que chaque processus possède un nom et un nombre fini de portes et que chaque porte a, elle aussi, un nom qui l'identifie.





Par conséquent, la description d'un processus doit contenir toute l'information sur son nom, ses portes, ses communications et les fonctions qu'il calcule.

Pour arriver à la formalisation de cette description, il faut préciser quelques détails importants sur le modèle de processus que l'on veut décrire. Ainsi, et en tenant toujours compte qu'un processus, pour nous, peut décrire un SPC :

- nous considérons que plusieurs communications peuvent être faites simultanément. Notre modèle décrit donc des ensembles de communications simultanées, qui seront appelées, par la suite "événements".

- nous considérons aussi un processus comme une *"unité isolée"*. Pour l'isoler, nous associons, à chaque porte du processus, une porte fictive dénommée δ, dans laquelle on prend (ou on dépose) les différents messages de (ou pour) le milieu extérieur. De cette façon, nous pouvons étudier le comportement du processus indépendamment du milieu extérieur dans lequel il est utilisé.

- comme conséquence de cet isolement, nous ne pouvons faire aucune supposition sur l'ordre dans lequel le milieu extérieur quelconque va prendre ou déposer des messages dans les portes du processus. Pour cela, on doit considérer la description de toutes les suites possibles d'événements du processus, autrement dit, le processus doit exprimer tous les milieux dans lesquels il peut être utilisé. Ainsi, lorsqu'il sera utilisé dans un milieu donné, une des suites possibles sera son comportement externe.

- enfin, il faut remarquer que la seule restriction que l'on exige pour qu'une communication ait lieu est qu'elle soit synchronisée. Cela veut dire que la communication se produit seulement dans le cas où le transmetteur et le récepteur sont prêts à échanger le message. Cette hypothèse sur les communications a été introduite par C.A.R. Hoare et elle est connue sous l'hypothèse de *"rendez-vous"*.

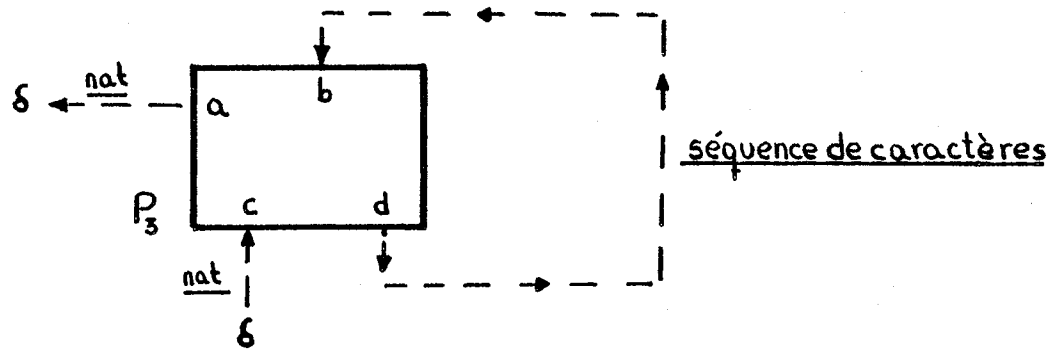
Pour une meilleure compréhension intuitive de la façon de décrire les processus, on divise en deux parties l'information de la description : la structure statique et le comportement dynamique.

1. Structure statique du processus, elle donne toute l'information sur :

i - le nom du processus ; le nombre, les noms et les conditions d'entré et/ou de sortie de ses portes, les connexions entre ses portes ("*connexions internes*") et les connexions avec le milieu extérieur, au moyen de la porte fictive δ ("*connexions externes*").

ii - le type des messages qui peuvent circuler sur chaque connexion.

Autrement dit, la structure statique cerne toute l'information du processus, s'il est considéré comme un diagramme du genre suivant :



Les connexions internes et externes peuvent être formalisées de la façon suivante :

Soit P le nom d'une boîte et Portes = $\{p_1,\ldots,p_n\}$ l'ensemble fini des noms de ses portes.

La description d'une connexion est un élément de

$$K = K_{IN} \cup K_{EX}$$

où

$K_{IN} \subseteq$ Portes x Portes est l'ensemble des connexions interne

$K_{EX} \subseteq K_{en} \cup K_{so}$ est l'ensemble des connexions externe

$K_{en} \subseteq \{\delta\}$ x Portes est l'ensemble de connexions d'entré

$K_{so} \subseteq$ Portes x $\{\delta\}$ est l'ensemble de connexions de sort

Les éléments $\langle x,y \rangle$ de K seront dénotés par x.y, et ils représenteront la connexion entre la porte de sortie x et la porte d'entrée y.

Chacun des éléments de K sert à émettre et/ou à recevoir des messages (un à la fois). Le couple formé d'une connexion et d'un message constitue une "*communication*". Pour exprimer la sorte de messages qui peuvent circuler par une connexion, nous associons un type de données à la connexion. Ainsi, *la description d'une communication* est constituée de deux éléments, une connexion $x.y \in K$ et un "*identificateur de domaine*" qui dénote le type de données en question.

2. Comportement dynamique du processus, il donne les différentes suites "*d'événements*" permis, et les différentes fonctions calculées. Au niveau du diagramme, cela exprime quels sont les différents ordres dans lesquels les messages peuvent circuler sur les organes, et quels sont les valeurs transmises. La clef de cette description est la notion intuitive d'état (d'une machine ou d'un processus). Ainsi, le processus prévoit les différents événements qui peuvent se réaliser dans un état donné, ainsi que les valeurs des messages à émettre ou à recevoir.

Le comportement dynamique du processus est donc décrit par "*un système de règles*" de la forme

$$t_1 \xrightarrow{t_2} t_3$$

où les termes t_1, t_3 sont des caractérisations d'ensembles d'"*états*" et le terme t_2 est la caractérisation de l'ensemble des "*événements*". Chaque événement de cet ensemble a lieu quand on passe d'un état particulier de t_1 à un état particulier de t_3. Cela constitue une des interprétations possibles de ces règles (la vision opérationnelle).

Il est possible que l'événement soit vide. Cela veut dire que, lors du changement d'état, le processus ne manifeste aucune communication avec le milieu extérieur. Dans ce cas, nous dénotons le terme t_2 par le symbole τ. Enfin, puisque toute machine a un ou plusieurs états initiaux, nous devons spécifier, pour le processus, quels sont les "*états initiaux*" possibles du processus.

Avec toutes ces idées intuitives, nous sommes prêts à aborder le langage formel de description de processus et les différentes sémantiques

1.2.2. La description d'un processus : *SYNTAXE FORMELLE*

Un langage $\mathcal{L}$ pour décrire un processus (un SPC) est une paire

$\mathcal{L} = \langle D, L \rangle$

où

- D est un ensemble d'*identificateurs de domaines* ("sort" en anglais)
- L consiste en trois signatures :

  i- un ensemble F de *symboles d'opération* avec une fonction d'*arité*

  $$\rho_F : F \to D^* \times D$$

  ou D^* est l'ensemble des suites finies d'éléments de D, y compris la suite vide λ

  ii- un ensemble $K = K_{IN} \cup K_{EX}$ (voir section 1.2.1.) de *symboles de connexion* avec une fonction d'*arité*

  $$\rho_K : K \to D$$

  iii- en ensemble E de *symboles d'états* avec une fonction d'*arité*

  $$\rho_E : E \to D^*$$

Soit $V(\mathcal{L}) = \{V_d | d \in D\}$ une famille d'ensembles non-vides indexée par D et appelée "variables de $\mathcal{L}$".

Soit $D' = D \cup \{\text{état, évén}\}$ avec état, évén $\notin D$

On définit l'ensemble de termes de $\mathcal{L}$ $T(\mathcal{L}) = \{T_{d'} | d' \in D'\}$ comme la plus petite famille d'ensembles indexées par D, t.q.

i. $V_d \subseteq T_d$ pour $d \in D$

ii. si $f \in F$ et $\rho_F(f) = \langle \lambda, d \rangle$ alors $f \in T_d$ pour $d \in D$

iii. si $f \in F$ et $\rho_F(f) = \langle d, \ldots d_K, d \rangle$ et $t_i \in T_{d_i}$ pour $1 \leq i \leq K$
alors $f\ t_1, \ldots, t_n \in T_d$ pour $d \in D$

iv. si $\emptyset \neq \{K_1, \ldots, K_n\} \subseteq K$ et $\rho_K(K_i) = d_i$ et $t_i \in T_{d_i}$ et pour $1 \leq i \leq n$
alors $\{K_1(t_1), \ldots, K_n(t_n)\} \in T_{\text{évén}}$

v. $\tau \in T_{\text{évén}}$ (τ dénote l'événément vide)

vi. si $e \in E$ et $\rho_E(e) = d_1 \ldots d_n$ et $t_i \in T_{d_i}$ pour $1 \leq i \leq n$
alors $e\ t_1, \ldots, t_n \in T_{\text{état}}$

Pour chaque d de D, les ensembles T_d engendrés par i,ii,iii, sont les ensembles de "termes d'opération" qui donnent, comme résultat, un élément dans le domaine d. On dénote par $T_{\text{opér}} = \{T_d | d \in D\}$ où oper $\notin D'$.

L'ensemble $T_{\text{éven}}$ engendré par iv,v est appelé ensemble des "termes d'événements".

L'ensemble $T_{\text{état}}$ engendré par vi est appelé ensemble des "termes d'états".

Les éléments $t \in T(\mathcal{L})$ peuvent avoir des variables. Nous dénoterons par var(t) la famille de variables qui apparaissent dans t.

Dans le cas où $t \in T_{\text{éven}}$, la famille var(t) est divisée en deux :

$$\text{var}_{\text{en}}(t) = \bigcup_{t_i \in \text{En}(t)} \text{var}(t_i) \text{ où } \text{En}(t) = \{t_i | K_i(t_i) \in t \wedge K_i \in K_{\text{en}}\}$$

$$\text{var}_{\text{so}}(t) = \bigcup_{t_i \in \text{So}(t)} \text{var}(t_i) \text{ où } \text{So}(t) = \{t_i | K_i(t_i) \in t \wedge K_i \in K_{\text{so}} \cup K_{\text{IN}}\}$$

Ce sont les variables de portes d'entrée et les variables de portes de sortie et de connexions internes.

Pour pouvoir exprimer certaines propriétés sémantiques, on définit l'ensemble des règles de $\mathcal{L}$:

$$\mathcal{R}(\mathcal{L},V) = \{t_1 \xrightarrow{t_2} t_3 \mid t_1,t_3 \in T_{état} \wedge t_2 \in T_{éven} \wedge var_{so}(t_2),\ var(t_3) \subseteq var(t_1) \cup var_{en}(t_2)\}$$

Une description d'un processus est un triplet

$$p = <\mathcal{L},\ \mathcal{I},\ \mathcal{R}>$$

où

$\mathcal{L} = <D,L>$ est un langage de processus où L contient F, K, E

$\mathcal{I} \subseteq T_{état} \in T(\mathcal{L})$ tel que : $\mathcal{I} \neq \emptyset$ et pour chaque $t \in \mathcal{I}$ $var(t) = \emptyset$

$\mathcal{R} \subseteq \mathcal{R}(\mathcal{L},V)$ et $\mathcal{R}$ fini

L'ensemble $\mathcal{I}$ contient les états initiaux du processus

Finalement, la notation que l'on emploiera dans les exemples sera la suivante, pour rendre la description plus lisible :

p = processus

. connexions

$K_{1.1}, \ldots, K_{1.r} : d_1$

$\vdots$

$K_{s.1}, \ldots, K_{s.r} : d_s$

opérations

$f_{1.1}, \ldots, f_{1.n} : d'_{1.1} \ldots d'_{1.i} \longrightarrow d'_1$

$\vdots$

$f_{m.1}, \ldots, f_{m.n} : d'_{m.1} \ldots d'_{m.i} \longrightarrow d'_m$

états

$e_{1.1}, \ldots, e_{1.p} : d_{1.1} \ldots d_{1.j}$

$\vdots$

$e_{q.1}, \ldots, e_{q.p} : d_{q.1} \ldots d_{q.j}$

initiaux

. $t_1, \ldots, t_k$

règles

$t_{1.1} \xrightarrow{t_{2.1}} t_{3.1}$

$\vdots$

$t_{1.l} \xrightarrow{t_{2.l}} t_{3.l}$

où on peut déduire, trivialement, les éléments de la description formelle :

$$D = \{d_{1.1}, \ldots, d_{1.i}, d_1, d_{m.1}, \ldots, d_{m.i}, d_m, d_{1.1}, \ldots, d_{1.j}, d_{q.1}, \ldots, d_{q.j}, d_1, \ldots, d_s\}$$

$$K = \{k_{1.1}, \ldots, k_{1.r}, \ldots, k_{s.1}, \ldots, k_{s.r}\} \quad F = \{f_{1.1}, \ldots, f_{1.n}, \ldots, f_{m.1}, \ldots, f_{m.n}\}$$

$$E = \{e_{1.1}, \ldots, e_{1.p}, \ldots, e_{q.1}, \ldots, e_{q.p}\} \quad I = \{t_1, \ldots, t_k\}$$

$$R = \{t_{1.1} \xrightarrow{t_{2.1}} t_{3.1}, \ldots, t_{1.l} \xrightarrow{t_{2.l}} t_{3.l}\}$$

Cette notation a comme avantage, d'avoir les fonctions d'arité d'une façon explicite dans la description . On voit aussi que l'on peut déduire, sans peine, les ensembles K_{IN}, K_{EX}, K_{en}, K_{so}.

Exemple 1.1.

▼

Problème : Ecrire un processus qui reçoit des naturels par sa porte d'entrée et les transmet par sa porte de sortie dans l'ordre FIFO (le dernier entré est le premier qui sort)

Pile$^{\infty}$ = processus

connexions δ.i, o.δ : Nat

opérations

cons : Suite de Nat, Nat ⟶ Suite de Nat

Λ : ⟶ Suite de Nat

états C : Suite de Nat

initial C(Λ)

règles C(w) $\xrightarrow{\{\delta.i(n)\}}$ C(cons(w.n))

C(cons(w.n)) $\xrightarrow{\{o.\delta(n)\}}$ C(w)

▲

Exemple 1.2.

▼

Problème : Le processus doit décrire un pont à une seule voie et tous les comportements possibles de toutes les voitures qui peuvent être sur lui. On suppose que la loi, pour utiliser le pont, est la suivante : s'il y a une voiture qui traverse le pont dans le même sens que moi, je peux prendre le pont, sinon je dois attendre jusqu'à ce qu'il soit vide.

Cet exemple décrit seulement les suites d'évènements, et ne fait aucun calcul. Pour cela, on considère que les messages communiqués sont des signaux (**ZZ**).

Pont = processus

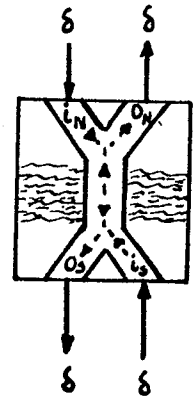
connexions $\delta.i_N$, $o_N.\delta$, $\delta.i_S$, $o_S.\delta$: Signal

opérations

succ : Nat ⟶ Nat

o : ⟶ Nat

ZZ : ⟶ signal



états

A : λ

B,C : Nat

initial A

règles

$A \xrightarrow{\{\delta.i_N(ZZ)\}} B(o)$

$A \xrightarrow{\{\delta.i_S(ZZ)\}} C(o)$

$B(x) \xrightarrow{\{\delta.i_N(ZZ)\}} B(succ(x))$

$B(succ(x)) \xrightarrow{\{o_S.\delta(ZZ)\}} B(x)$

$B(o) \xrightarrow{\{o_S.\delta(ZZ)\}} A$

$C(x) \xrightarrow{\{\delta.i_S(ZZ)\}} C(succ(x))$

$C(succ(x)) \xrightarrow{\{o_N.\delta(ZZ)\}} C(x)$

$C(o) \xrightarrow{\{o_N.\delta(ZZ)\}} A$

▲

Exemple 1.3.

▼

Problème : Ecrire un processus qui reçoit des fichiers par sa porte d'entrée et les transmet par sa porte de sortie dans l'ordre où ils sont entrés.

Notation : fl = fichier de lignes, sfl = suite de fichier de lignes

Buffer$^{\infty}$ = processus

connexions $\delta.i, o.\delta$: fl

opérations

Λ : $\longrightarrow$ sfl

cons : sfl, fl $\longrightarrow$ slf

états

A : sfl

B : sfl, sfl, sfl

C : sfl, sfl

initial A(Λ)

règles

(* peut toujours recevoir *)	A(x)	$\xrightarrow{\{\delta.i(a)\}}$	B(x,Λ, cons(Λ,a))
(* queue ...> pile *)	B(cons(x,b),z,y)	$\xrightarrow{\tau}$	B(x, cons(z,b),y)
(* queue vide *)	B(Λ,z,y)	$\xrightarrow{\tau}$	C(z,y)
(* pile ...> queue avec a dans le fond *)	C(cons(z,b),y)	$\xrightarrow{\tau}$	C(z, cons(y,b))
(* pile vide *)	C(Λ,y)	$\xrightarrow{\tau}$	A(y)
(* si le buffer n'est pas vide, on peut sortir la valeur correspondante *)	A(cons(x,b))	$\xrightarrow{\{o.\delta(b)\}}$	A(x)

\

Cet exemple montre une utilité du terme d'évènement τ.

▲

Exemple 1.4.

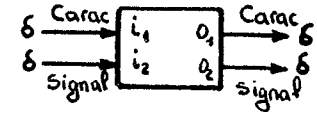
▼

Problème : Ecrire un processus qui copie les caractères lus d'un processus dans un autre. Chaque fois qu'il reçoit un caractère, il reçoit aussi, en même temps, un signal par une autre porte, qu'il doit aussi copier en même temps.

copier = processus

connexion $\delta.i_1, O_1.\delta$: Caractère

$\delta.i_2, O_2.\delta$: Signal



états A : λ

initial A

règles A $\xrightarrow{\{\delta.i_1(x),\ \delta.i_2(v),\ O_1.\delta(x),\ O_2.\delta(v)\}}$ A

▲

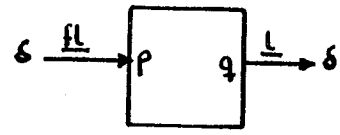
Exemple 1.5.

▼

Problème : Ecrire un processus qui prend un fichier de lignes et le transmet, ligne par ligne, par sa porte de sortie.

Notation : l = ligne fl = fichier de lignes

Imprimante = processus



connexions $\delta.p$: fl

$q.\delta$: l

opérations Λ : $\longrightarrow$ fl

cons : fl, l $\longrightarrow$ fl

états P : fl

initial $P(\Lambda)$

règles $P(\Lambda) \xrightarrow{\{\delta.p(w)\}} P(w)$

$P(cons(w,s)) \xrightarrow{\{q.\delta(s)\}} P(w)$

▲

Exemple 1.6.

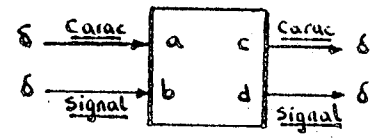
▼

Problème : Lire des cartes jusqu'à ce qu'un signal indique la fin du fichier de cartes. Donner en sortie (caractère par caractère) tous les caractères contenus dans le fichier. Un espace blanc (ƀ) doit être inséré à la fin de chaque carte. Le processus doit aussi transmettre le signal de fin de fichier.

Notation : l'identificateur de domaine Signal identifie deux valeurs

ff : fin de fichier et $\overline{ff}$: non fin de fichier

Deballer = processus



connexions

δ.a : Carte

δ.b, d.δ : Signal

c.δ : Caractère

opérations

cons : Fichier de Cartes, Carte ⟶ Fichier de Cartes

$\wedge$: ⟶ Fichier de Cartes

ff, $\overline{ff}$: ⟶ Signal

états

F : λ

C : Fichier de Cartes, Signal

initial

F

règles

$F \xrightarrow{\{\delta.a(car),\delta.b(s)\}} C(car,s)$

$C(cons(y,K),s) \xrightarrow{\{c.\delta(K)\}} C(y,s)$

$C(\wedge,\overline{ff}) \xrightarrow{\{c.\delta(ƀ)\}} F$

$C(\wedge,ff) \xrightarrow{\{c.\delta(ƀ),d.\delta(ff)\}} F$

▲

Exemple 1.7.

▼

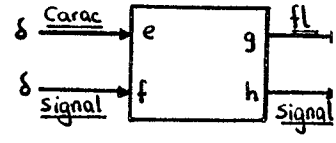
Problème : Lire une entrée, caractère par caractère, jusqu'à ce qu'un signal indique la fin du fichier des caractères. Donner comme sortie (ligne à ligne) tout le fichier de lignes. Chaque ligne doit avoir 125 caractères. La dernière ligne doit être complétée par des espaces (b), si nécessaire. Le processus transmet aussi le signal de fin de fichier.

Emballer = processus

connexions $\delta.e$: Caractère

$\delta.f, h.\delta$: Signal

$g.\delta$: fl



opérations

cons : fl, l → fl

$\wedge$: → fl

ff, $\overline{ff}$: → Signal

0 : → Nat

Succ : Nat → Nat

états L : fl, Signal, Nat

initial $L(\wedge, \overline{ff}, succ^{125}(0))$

règles

$L(x,\overline{ff},succ(z)) \xrightarrow{\{\delta.e(j),\delta.f(t)\}} L(cons(x,j),t,z)$

$L(x,\overline{ff},0) \xrightarrow{\{g.\delta(x),h.\delta(\overline{ff})\}} L(\wedge,\overline{ff},succ^{125}(0))$

$L(x,ff,succ(z)) \xrightarrow{\tau} L(cons(x,b),ff,z)$

$L(x,ff,0) \xrightarrow{\{g.\delta(x),h.\delta(ff)\}} L(\wedge,\overline{ff},succ^{125}(0))$

▲

1.2.3. La Conception d'un Processus : *SEMANTIQUE FORMELLE*

Nous montrons ici comment la description d'un processus peut être utilisée pour concevoir les processus.

Ainsi, nous donnons trois versions possibles d'interprétation de la description, à savoir :

- une structure algébrique,
- un système de transitions d'états,
- un arbre (fini ou infini).

Chacune de ces trois conceptions donnent une sémantique différente qui permet d'étudier, d'une façon plus commode, différentes propriétés des processus.

1.2.3.1. La Description d'un Processus = La Présentation d'une Variété d'Algèbres : *SEMANTIQUE ALGEBRIQUE*

Avant d'entrer directement dans la sémantique algébrique des processus, nous présentons un résumé des notions fondamentales de la Théorie des Modèles, et nous montrons deux exemples particulièrement intéressants de l'utilisation de cette théorie dans l'informatique.

1.2.3.1.1. Avant-propos : Théorie des Modèles et Présentations

La théorie des modèles est une branche de la logique mathématique dans laquelle on établit la relation entre un langage formel et des structures mathématiques. Puisque nous sommes intéressés par la sémantique algébrique, nous travaillons seulement sur les langages formels en relation avec les structures algébriques. Ces structures algébriques sont hétérogènes puisqu'elles comportent plusieurs types d'objets (ou de valeurs).

Pour ce résumé, nous nous sommes inspirés de [Rob 63], [Cha-Kei 73], [ADJ 78], [Zil 79], [Bur-San 81].

Un langage d'algèbres (ou de structures algébriques) $\mathcal{L}$ consiste en :

- un ensemble D d'*identificateurs de domaines*
- en ensemble $\mathcal{F}$ de *symboles d'opérations* avec une fonction d'arité

$$\rho_{\mathcal{F}} : \mathcal{F} \longrightarrow D^* \times D$$

Une $\mathcal{L}$-algèbre (ou $\mathcal{L}$-structure algébrique) est une paire $\mathfrak{U}(\mathcal{L}) = \langle A, \mathbf{F} \rangle$ où

- $A = \{A_d \mid d \in D\}$ est une famille d'ensembles non-vides indicée par D et appelés *supports de l'algèbre*. La famille A est appelée *univers de* $\mathfrak{U}(\mathcal{L})$.

- $\mathbf{F}$ est un ensemble de fonctions fondamentales $\mathbf{f} : A_{d_1} \times \ldots \times A_{d_n} \to A_d$ une pour chaque $f \in \mathcal{F}$ où $\rho_{\mathcal{F}}(f) = \langle d_1 \ldots d_n, d \rangle$

La correspondance entre $\mathcal{L}$ et $\mathfrak{U}(\mathcal{L})$, implicite dans les définitions, est définie par une fonction $\mathcal{I}_{\mathcal{L},A}$ appelée interprétation qui associe, à chaque symbole de $\mathcal{L}$, la fonction fondamentale sur A.

Notation : Dans le cas ou $\mathcal{L}$ est fini, c-à-d $D = \{d_1, \ldots, d_n\}$, $\mathcal{F} = \{f_1, \ldots, f_m\}$, on écrit

$$\mathcal{L} = \langle d_1, \ldots, d_n \; ; \; f_1 : d_1^1 \ldots d_{n_1}^1 \to d^1 ; \ldots ; f_m : d_1^m \ldots d_{n_m}^m \to d^m \rangle$$

ainsi, l'arité est explicitée, et :

$$\mathfrak{U}(\mathcal{L}) = \langle A_{d_1}, \ldots, A_{d_n} \; ; \; \mathbf{f}_1, \ldots, \mathbf{f}_m \rangle$$

en entendant que $\mathcal{I}_{\mathcal{L},A}(f_j) = \mathbf{f}_j$ pour chaque $f_i \in \mathcal{F}$

Exemple 1.8.

▼

Soit $\mathcal{L}_1 = \langle \underline{d} \; ; \; o : \underline{d}\,\underline{d} \to \underline{d} \; ; \; 1 : \to \underline{d} \rangle$

Nous pouvons dire que la structure algébrique $\langle D_{\underline{d}}, \mathbf{o}, \mathbf{1} \rangle$ du monoïde sur un domaine identifié par $\underline{d}$ est une $\mathcal{L}_1$-structure.

▲

A travers cet exemple, nous pouvons voir qu'il manque quelque chose pour bien établir la correspondance entre le langage $\mathcal{L}$ et la $\mathcal{L}$-structure. Ainsi, par exemple, on sait que les "*conditions*" (ou "*lois*") du monoïde sont les suivantes :

$$a \bullet (b \bullet c) = (a \bullet b) \bullet c \qquad \text{(associativité)}$$

$$\left.\begin{array}{l} a \bullet 1 = a \\ 1 \bullet a = a \end{array}\right\} \text{(élément neutre)}$$

pour chaque a,b,c dans $D_{\underline{d}}$

Donc, pour exprimer ces conditions et ainsi pouvoir bien caractériser une structure précise, on "*axiomatise*" le langage $\mathcal{L}$. Cette axiomatisation est réalisée par des "*équations*" qui vont avoir une valeur logique (vrai ou faux) dans la $\mathcal{L}$-structure. Si la valeur d'une équation φ dans la $\mathcal{L}$-structure $\mathcal{U}(\mathcal{L})$ est vraie, nous dirons que "φ *est vraie dans* $\mathcal{U}(\mathcal{L})$" et aussi que "$\mathcal{U}(\mathcal{L})$ *est un modèle de* φ". De plus, et puisqu'en général, on a un ensemble E d'équations, nous disons que "$\mathcal{U}(\mathcal{L})$ *est un modèle de* E" ssi $\mathcal{U}$ est un modèle pour chaque équation dans E.

L'axiomatisation d'un langage $\mathcal{L}$ exprime les conditions ou lois que "*satisfont*" une $\mathcal{L}$-algèbre.

Formellement, pour l'axiomatisation d'un langage d'algèbres $\mathcal{L} = \langle D,F\rangle$ on a besoin de :

- une famille $V(\mathcal{L}) = \{V_d \mid d \in D\}$ d'ensembles non-vide indicée par D et appelée "<u>variables de $\mathcal{L}$</u>" et avec laquelle nous définissons une autre famille $T(\mathcal{L}) = \{T_d \mid d \in D\}$ d'ensembles, indicée par D, appelée "termes de $\mathcal{L}$", comme la plus petite famille d'ensembles tels que :

i. $V_d \subseteq T_d$ pour chaque $d \in D$

ii. si $f \in F \wedge \rho_F(f) = \langle \lambda, d\rangle$ alors $f \in T_d$

iii. si $f \in F \wedge \rho_F(f) = \langle d_1 \ldots, d_n, d\rangle$ et $t_i \in T_{d_i}$ alors $f(t_1 \ldots t_n) \in T_d$

On utilise $t(v_0 \ldots v_n)$ pour dénoter un terme t dont les variables sont un sous-ensemble de $\{v_0,\ldots,v_n\}$.

- un symbole "$\equiv$" tel que $\equiv \notin D \cup \mathcal{F} \cup V(\mathcal{L})$ avec lequel nous définissons "les équations de $\mathcal{L}$" comme

$$E(\mathcal{L}) = \{t \equiv t' \mid t,t' \in T(\mathcal{L})\}$$

Donc, les éléments d'un langage formel d'algèbres sont $\mathcal{L}$ et $E(\mathcal{L})$. Nous dénoterons, par la suite, un langage formel d'algèbres par $<\mathcal{L},E>$ Un sous-ensemble fini $E \subseteq E(\mathcal{L})$ est appelé système d'équations.

Pour établir le lien entre un langage formel d'algèbres $<\mathcal{L},E>$ et les $\mathcal{L}$-structures,

- on considère une interprétation $\mathcal{I}_{\mathcal{L},A}$ à travers laquelle on obtient la $\mathcal{L}$-structure $\mathfrak{U}(\mathcal{L})$ implicite.

- et on donne un "*sens*" à chaque équation φ dans E. Ce "*sens*" est une valeur de vérité (vrai ou faux) que l'on associe à chaque équation φ pour indiquer qu'elle est "*satisfaite*" par $\mathfrak{U}(\mathcal{L})$.

Ainsi,

Soit $<\mathcal{L},E>$ un langage formel d'algèbres avec $\mathcal{L} = <D,\mathcal{F}>$,
$\mathfrak{U}(\mathcal{L}) = <A,\mathcal{F}>$ une $\mathcal{L}$-structure algébrique avec $\mathcal{I}_{\mathcal{L},A}$ l'interprétation.
$Y \subseteq V$ une famille de variables indicée par D (c-à-d $Y_d \subseteq V_d$ pour chaque $d \in D$)

Une fonction $\theta : Y \to A$ (c-à-d $<\theta_d : Y_d \to A_d>$) qui associe des valeurs des domaines d dans A à des variables du même domaine d dans Y, est appelée une substitution.

La valeur d'un terme $t(v_0 \ldots v_n)$ pour une substitution $\theta : \{v_0,\ldots,v_n\} \to A$ est dénotée par $\theta^{\#}(t)$ et définie de la façon suivante :

i. si $t = v \wedge v \in \{v_0,\ldots,v_n\}$ alors $\theta^{\#}(t) = \theta(v)$

ii. si $t = c \wedge \rho_{\mathcal{F}}(c) = <\lambda,d>$ alors $\theta^{\#}(c) = \mathcal{I}_{\mathcal{L},A}(c)$

iii. si $t = f(t_1,\ldots,t_m) \wedge \rho_{\mathcal{F}}(f) = <d_1 \ldots d_m,d>$
alors $\theta^{\#}(t) = f(\theta^{\#}(t_1) \ldots \theta^{\#}(t_m))$ où $f = \mathcal{I}_{\mathcal{L},A}(f)$

Soit $\varphi = t_1 \equiv t_2$ une équation dans E

$\text{var}\varphi = \text{var}(t_1) \cup \text{var}(t_2)$

L'équation $\varphi = t_1 \equiv t_2$ est <u>satisfaite</u> par une substitution $\theta : \text{var}(\varphi) \to A$ ssi :

$$\theta^{\#}(t_1) = \theta^{\#}(t_2)$$

Nous dirons, dans ce cas, que l'équation <u>φ est satisfaite dans $\mathfrak{U}(\mathcal{L})$ par θ</u> puisque A est l'univers de $\mathfrak{U}(\mathcal{L})$, et nous le noterons par $\mathfrak{U}(\mathcal{L}) \models \varphi[\theta]$.

Enfin, une équation <u>φ est vraie dans $\mathfrak{U}(\mathcal{L})$</u> ssi, pour chaque substitution θ :

$$\mathfrak{U}(\mathcal{L}) \models \varphi\,[\theta]$$

Dire que l'équation φ est vraie dans $\mathfrak{U}(\mathcal{L})$ équivaut à dire :

<u>$\mathfrak{U}(\mathcal{L})$ satisfait φ</u>

<u>φ est satisfaite dans $\mathfrak{U}(\mathcal{L})$</u>

<u>$\mathfrak{U}(\mathcal{L})$ est un modèle pour φ</u>

et on le note par $\mathfrak{U}(\mathcal{L}) \models \varphi$

Soit E un ensemble d'équations de $E(\mathcal{L})$.

On dit que <u>$\mathfrak{U}(\mathcal{L})$ est un modèle de E</u> ssi $\mathfrak{U}(\mathcal{L}) \models \varphi$ pour chaque $\varphi \in E$ et on note $\mathfrak{U}(\mathcal{L}) \models E$.

Par le développement précédent, on a établi qu'en connaissant un langage d'algèbres $\mathcal{L} = \langle S, F \rangle$ et un système E d'équations dans $E(\mathcal{L})$, on peut déterminer si une $\mathcal{L}$-structure $\mathfrak{U}(\mathcal{L})$ est un modèle de E.

Exemple 1.3. (suite)

▼

Soit $\mathcal{L}_1$ et $V = \{V_d\}$ où $V_d = \{x,y,z\}$

et $E_1 = \{x \circ (y \circ z) \equiv (x \circ y) \circ z,\ x \circ 1 \equiv x,\ 1 \circ x \equiv x\}$

alors $\mathfrak{U}(\mathcal{L}_1) \models E_1$ c-à-d que la structure $\mathfrak{U}(\mathcal{L}_1)$ est un modèle de E_1

▲

Mais la définition de $\mathfrak{U}(\mathcal{L}) \models E$ est plus générale et utile. Ainsi, elle nous permet de considérer une "*variété*" de structures algébriques.

Exemple 1.8 (suite)

▼

Soit $\mathcal{L}_1$ et E_1 dont la définition a été donnée ci-dessus

$\mathcal{U}(\mathcal{L}_1^1) = \langle \Sigma^*, \text{concaténation}, \lambda \rangle$ avec $\Sigma^* = \{\Sigma^* \text{ chaînes sur } \Sigma\}$

$\mathcal{U}(\mathcal{L}_1^2) = \langle \mathcal{P}(D), \cup, \emptyset \rangle$ avec $\mathcal{P}(D) = \{\mathcal{P}(D)$ ensemble de parties de $D\}$

$\mathcal{U}(\mathcal{L}_1^3) = \langle \mathcal{P}(D), \cap, D \rangle$ " "

$\mathcal{U}(\mathcal{L}_1^4) = \langle \mathbb{N}, +, 0 \rangle$ avec $\mathbb{N} = \{\mathbb{N}_{nat}\}$

▲

Dans toutes ces structures, avec leurs lois habituelles, on a $\mathcal{U}(\mathcal{L}_1^i) \models E_1$ $i = 1 \ldots 4$, c-à-d, elles sont toutes modèles pour E_1.

Formellement, soit $\mathcal{L}$ un langage algébrique et $E \subseteq E(\mathcal{L})$, un système d'équations.

Un ensemble $\mathcal{V}_{\mathcal{L},E}$ d'algèbres (ou de structures algébriques) est appelé "variété" si $\mathcal{V}_{\mathcal{L},E}$ comprend toutes les $\mathcal{L}$-algèbres qui sont modèles de l'ensemble d'équations E, c-à-d :

$$\mathcal{V}_{\mathcal{L},E} = \{\mathcal{U}(\mathcal{L}) \mid \mathcal{U}(\mathcal{L}) \models E\}$$

Pour décrire une variété $\mathcal{V}_{\mathcal{L},E}$, on utilise un triplet $p = \langle D, \mathcal{F}, E \rangle$ appelé "présentation" en considérant $\mathcal{L} = \langle D, \mathcal{F} \rangle$.

Il est parfois utile d'avoir une $\mathcal{L}$-structure algébrique qui sert de "prototype" pour la variété décrite par une présentation. Dans la prochaine section, nous montrons comment construire cette algèbre "prototype".

L'algèbre quotient de l'univers de Herbrand d'une présentation

Parmi les algèbres d'une variété $\mathcal{V}_{\mathcal{L},E}$ définie par une présentation $p = \langle D, \mathcal{F}, E \rangle$, on s'intéresse, par la suite, à une algèbre quotient de l'univers de Herbrand engendrée par $\mathcal{L} = \langle D, \mathcal{F} \rangle$. Cette algèbre a, comme univers, la famille des classes d'équivalence de termes de Herbrand qui peuvent être déduites de l'ensemble d'équations (c-à-d, la congruence définie par les équations). L'algèbre sera notée $\mathcal{T}(\mathcal{L})$ et construite de façon telle que :

$$\mathcal{T}(\mathcal{L}) \models E$$

c-à-d, qu'elle est un modèle de E.

Formellement,

Soit $\mathcal{L} = \langle D, F \rangle$ un langage d'algèbres.

L'univers de Herbrand engendré par $\mathcal{L}$ est la famille

$$H_{\mathcal{L}} = \{H_d | d \in D\}$$

telle que

$$H_d = \{c | c \in T_d(\mathcal{L}) \wedge var(c) = \emptyset\}$$

Les éléments de $H_{\mathcal{L}}$ sont appelés termes de Herbrand

Sur cet univers $H_{\mathcal{L}}$, on définit une relation $\sim_E \subseteq H_{\mathcal{L}} \times H_{\mathcal{L}}$ qui relie les termes de Herbrand engendrés par un ensemble d'équations $E \subseteq E(\mathcal{L})$. Cette relation $\sim_E$ est appelée congruence définie par les équations E et est définie de la façon suivante :

$c \sim_E c'$ s'il existe une suite finie d'équations dans E
$\varphi_1 = t_1 \equiv t'_1, \ldots, \varphi_n = t_n \equiv t'_n$
et une séquence associée de substitutions $\theta_i : var(\varphi_i) \to H_{\mathcal{L}}$
$i = 1 \ldots n$
telles que l'on peut déduire c' à partir de c, en utilisant les remplacements

$$\theta_1 t_1 | \theta_1 t'_1, \ldots, \theta_n t_n | \theta_n t'_n$$

(Note : $\theta_i t_i | \theta_i t'_i$ cela veut dire, ou bien θ t est remplacé par θ t' ou bien $\theta_i t_i$ est remplacé par $\theta_i t'_i$)

La famille $T_{\langle D,F,E \rangle} = \{T_{\langle d,F,E \rangle} | d \in D\}$ indicée par D (où $T_{\langle d,F,E \rangle} = \{c | \exists c'. c \sim_E c'\}$ est appelée univers de Herbrand de la présentation $\langle D,F,E \rangle$. Elle contient tous les termes de Herbrand dans $H_{\mathcal{L}} = \langle D,F \rangle$ qui peuvent être engendrés par substitutions à partir des membres des équations $E \subseteq E(\mathcal{L})$.

Proposition 1.1.

$\sim_E$ est une relation d'équivalence, c-à-d $\sim_E$ est

i. réflexive : $\forall c \in T_{\langle D,F,E \rangle}$ $c \sim_E c$
ii. symétrique : $\forall c,c' \in T_{\langle D,F,E \rangle}$ $c \sim_E c' \Rightarrow c' \sim_E c$
iii. transitive : $\forall c,c',c'' \in T_{\langle D,F,E \rangle}$ $c \sim_E c' \wedge c' \sim_E c'' \Rightarrow c \sim_E c''$

de plus $\sim_E$ est

iv. stable par composition :

$$c_1 \sim_E c'_1, \ldots, c_n \sim_E c'_n \Rightarrow f(c_1, \ldots, c_n) \sim_E f(c'_1, \ldots, c'_n)$$

Preuve :

■

i. $\sim_E$ réflexive trivial avec la séquence vide.

ii. $\sim_E$ symétrique il suffit de prendre les remplacements à l'envers

iii. $\sim_E$ transitive il suffit de prendre la concaténation de séquences et substitutions

iv. $\sim_E$ stable par composition il suffit de prendre les mêmes séquences de

$c_1 \sim_E c_1', \ldots, c_n \sim_E c_n'$

■ c.q.f.d.

Finalement, on définit la $\mathcal{L}$-algèbre quotient de l'univers de Herbrand d'une présentation $\langle D,\mathcal{F},E\rangle$ comme :

$$\mathcal{T}(\mathcal{L}) = \langle T_{\langle D,\mathcal{F},E\rangle}|_{\sim_E}, F\rangle$$

où

- $\mathcal{L} = \langle D,\mathcal{F}\rangle$ est un langage d'algèbres
- $T\langle D,\mathcal{F},E\rangle|_{\sim_E} = \{[c]_{\sim_E} | c \in T\langle D,\mathcal{F},E\rangle\}$ est l'ensemble quotient de l'univers de Herbrand de la présentation $\langle D,\mathcal{F},E\rangle$. La notation $[c]_{\sim_E}$ représente la classe d'équivalence de c par $\sim_E$, c-à-d $[c]_{\sim_E} = \{x | x \sim_E c\}$.
- F est l'ensemble des fonctions

$$f : T_{\langle d_1,\mathcal{F},E\rangle}|_{\sim_E} \times \ldots \times T_{\langle d_n,\mathcal{F},E\rangle}|_{\sim_E} \rightarrow T_{\langle d,\mathcal{F},E\rangle}|_{\sim_E}$$

une pour chaque f avec $\mathcal{F}(f) = \langle d_1 \ldots d_n, d\rangle$ tel que

$$f([c_{d_1}]_{\sim_E}, \ldots, [c_{d_n}]_{\sim_E}) = [f(c_{d_1}, \ldots, c_{d_n})]_{\sim_E}$$

Proposition 1.2.

Soit une présentation $\langle D,\mathcal{F},E\rangle$

alors $\mathcal{T}(\mathcal{L}) \models E$ avec $\mathcal{L} = \langle D,\mathcal{F}\rangle$

Preuve :

■

Il faut prouver que $\forall \varphi \in E.\ \mathcal{T}(\mathcal{L}) \models \varphi$

et, pour cela, il faut prouver $\forall \theta : var(\varphi) \to T_{<D,F,E>}|_{\sim E}.\ \mathcal{T}(\mathcal{L}) \models \varphi[\theta]$

Soit $\varphi = t_1 \equiv t_1'$ et θ une substitution quelconque.

Soit $\theta' : var(\varphi) \to T<D,F,E>$ t.q. $\theta'(x) = c$ ssi $\theta(x) = [c]$

$\theta^{\#}(t_1) = [\theta' t_1]$

$\theta^{\#}(t_2) = [\theta' t_2]$

Mais $\theta' t_1 \sim_E \theta' t_2$ puisque la suite $t_1 \equiv t_2$ avec θ' font $\theta' t_1 = \theta' t_2$

Donc $\theta^{\#}(t_1) = [\theta' t_1, \theta' t_2] = \theta^{\#}(t_2)$. Par suite $\mathcal{T}(\mathcal{L}) \models \varphi[\theta]$

■ c.q.f.d.

Deux exemples : Les types abstraits et les types abstraits génériques

Les types abstraits de données

Informellement, D. Scott [Sco 76] dit:"un type abstrait de données est une collection de données qui ont été groupées pour des raisons de 'similitude' ou peut-être par 'pure convenance'".

Dans la formalisation des types abstraits de données, comme des algèbres, [ADJ 78], [Lis-Zil 77], [Zil 79], cette similitude ou pure convenance est exprimée au moyen d'équations dans une présentation. Ainsi, un type abstrait de données peut être considéré comme l'algèbre quotient de l'univers de Herbrand d'une présentation.

Exemple 1.9.

▼

Entiers-naturels = < Nat — D_1

o : → Nat

succ : Nat → Nat

pred : Nat → Nat

\+ : Nat x Nat → Nat

\- : Nat x Nat → Nat — F_1

pred(o) ≡ o

pred(succ(x)) ≡ x

x + o ≡ o

x + succ(y) ≡ succ(x+y)

x - o ≡ o — E_1

Cela définit l'algèbre quotient de Herbrand des entiers naturels membres de la variété $\mathcal{V}_{\mathcal{L}_1,E_1}$ où $\mathcal{L}_1 = \langle D_1, \mathcal{F}_1 \rangle$.

▲

Exemple 1.10.

▼

File-d'entiers-naturels = <file de Nat, Nat, Bool>

D_2

creerf : → file de Nat
ajouterf : file de Nat, Nat → Nat
enleverf : file de Nat → file de Nat
sommet : file de Nat → Nat
vide : file de Nat → Bool
errornat : → Nat

$\mathcal{F}$

vide (creerf) ≡ vrai
vide (ajouterf(x,i)) ≡ faux
enleverf (creerf) ≡ creerf
enleverf (ajouterf(x,i)) ≡ si vide (x) alors creerf
sinon ajouterf (enleverf(x,i))
sommet (creerf) ≡ errornat
sommet (ajouterf(x,i)) ≡ si vide (x) alors i
sinon sommet (g)

E

Cet exemple définit l'algèbre quotient dans l'univers d'Herbrand de la variété $\mathcal{V}_{\mathcal{L}_2,E_2}$ où $\mathcal{L}_2 = \langle D_2, \mathcal{F}_2 \rangle$ de toutes les algèbres de files des entiers naturels.

Ici, nous avons utilisé, dans la spécification de $\mathcal{F}_2$, les types abstraits Nat, Bool et aussi vrai, faux comme opérations sur le type Bool dans les équations de la présentation. L'expression conditionnelle "si ... alors ... sinon" est considérée, en général, comme un opérateur défini implicitement sur le type abstrait File-d'entiers-naturels (voir [ADJ 78]).

▲

Enfin, dans les types abstraits de données, l'algèbre quotient de l'univers de Herbrand d'une présentation $\langle D, \mathcal{F}, E \rangle$, est un modèle de E.

Les types abstraits génériques

Dans la programmation générique [Ber 79, 82a, 82b], les propriétés sont un outil très puissant. Une propriété est aussi une présentation <D,F,E> où certains éléments dans D sont "distingués".

Exemple 1.11. *(voir l'exemple 1.8. et sa suite)*

▼

Monoïde = < d ;

0 : d, d → d } F

1 : → d } F

x o (y o z) ≡ (x o y) o z } E

x o 1 ≡ x } E

1 o x ≡ x > } E

où d est considéré distingué.

▲

Cette "distinction"

- est, en quelque sorte, similaire aux variables libres du calcul du premier ordre. Ainsi, si l'on considère l'algèbre quotient de l'univers de Herbrand, toutes les variables de domaines distingués (x,y,z) sont interprétés comme constantes.

- définit une variété, c-à-d, une algèbre quelconque dans la variété qui va être une sous-algèbre (au renommage des opérations près) de n'importe quelle algèbre définie par un type abstrait qui satisfait la propriété.

Exemple 1.12.

▼

Une déclaration : Entiers-naturels :: Monoïde (∘ = +, 1 = 0) indique que l'algèbre définie par la présentation des Entiers-naturels vérifie les propriétés du monoïde avec ses opérations +, 0.

▲

En résumé, un modèle pour un type abstrait qui satisfait une propriété est une algèbre qui vérifie :

- toutes les équations du type abstrait
- toutes les équations de la propriété avec les opérations correspondantes du type abstrait.

1.2.3.1.2. La Présentation d'un Processus

La description d'un processus (présentée dans la section 2.2.) a quelques ressemblances avec la présentation d'une variété, mais aussi quelques différences.

Le tableau suivant montre ce fait :

	description $p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R} \rangle$	présentation $p = \langle D, \mathcal{F}, \mathcal{E} \rangle$	
langage de processus	$\mathcal{L} = \langle D, L \rangle$	$\mathcal{L} = \langle D; \mathcal{F} \rangle$	langage d'algèbres
identificateurs de domaines	D	D	
symboles du langage	L contient F,K,E		
opérations	F	$\mathcal{F}$	opérations
connexions	K		
états	E		
initiaux	$\mathcal{I}$		
règles	$\mathcal{R} \subseteq \mathcal{R}(\mathcal{L}, V)$		
		$\mathcal{E} \subseteq \mathcal{E}(\mathcal{L})$	équations
variables de $\mathcal{L}$	$V(\mathcal{L}) = \{V_d \mid d \in D\}$	$V(\mathcal{L}) = \{V_d \mid d \in D\}$	Variables de $\mathcal{L}$

L'objectif de cette partie est d'établir une correspondance entre les descriptions et les présentations, au moyen de deux interprétations particulières :

1. Les éléments de E et de $\mathcal{P}(K)$ dans p seront interprétés comme des symboles d'opérations.

2. Les éléments dans $\mathcal{I}$, $\mathcal{R}$ seront interprétés comme des équations.

L'interprétation de E et $\mathcal{P}(K)$

On définit l'ensemble des opérations-états par $\overline{E} = E$ avec la fonction d'arité

$\overline{\rho}_E : E \to D^* \times \{p\}$ où p est le nom de la description ($p \notin D$)
et $\overline{\rho}_E(e) = \langle d_1 \ldots d_n p \rangle$ ssi $\rho_E(e) = d_1 \ldots d_n$

Soit $\pi(s)$ une fonction arbitraire qui ordonne linéairement les éléments d'un ensemble $s \quad T_{\text{évén}}$.

exemple : $\pi(\{K_1(v_1), K_3(v_3), K_2(v_2)\}) = K_1(v_1)K_2(v_2)K_3(v_3)$

Soit $\pi_1(s)$ la suite des éléments de K extraits, dans l'ordre, dans $\pi(s)$

exemple : $K_1K_2K_3$

On définit l'ensemble des opérations-évènements par

$$\overline{\mathcal{P}}(K) = \{\pi_1(t_2) \mid t_1 \xrightarrow{t_2} t_3 \in R\}$$

avec fonction d'arité $\overline{\rho}_{\overline{\mathcal{P}}(K)} = \overline{\mathcal{P}}(K) \longrightarrow (D^* \cup \{p\}) \times \{p\}$

où p comme ci-dessus

et $\overline{\rho}_{\overline{\mathcal{P}}(K)}(\tau) = \langle p, p \rangle$ ssi $\tau \in \overline{\mathcal{P}}(K)$

$\overline{\rho}_{\overline{\mathcal{P}}(K)}(K_1 \ldots K_n) = \langle d_1 \ldots d_n p, p \rangle$ ssi $K_1 \ldots K_n \in \overline{\mathcal{P}}(K)$
$\wedge \quad \rho_K(K_i) = d_i \qquad i = 1 \ldots n$

D'après cette interprétation, tous les symboles de la description sont des opérations dans le sens des présentations. Ainsi,

$$F_L = F \cup \overline{\mathcal{P}}_{(K)} \cup \overline{E}$$

Interprétation de $\mathcal{I}$, $\mathcal{R}$ comme équations

Soit σ un symbole qui n'apparaît pas dans la description. Ce symbole sera interprété comme l'évènement initial et avec arité $\langle\lambda,p\rangle$.

On définit l'ensemble des équations-initiales par

$$E_{\mathcal{I}} = \{\sigma \equiv t \mid t \in \mathcal{I}\}$$

Soit $\pi_2(s)$ la suite des éléments de T_d ($d \in D$), séparés par des virgules, extraits dans l'ordre, dans $\pi(s)$

exemple : v_1, v_2, v_3

On définit l'ensemble des équations-règles par

$$E_{\mathcal{R}} = \{\pi_1(t_2)(\pi_2(t_2),t_1) \equiv t_3 \mid t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}\}$$

La présentation d'un processus

La description d'un processus peut donc être traduite sous la forme d'une présentation.

Soit la description $p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R}\rangle$ où $\mathcal{L} = \langle D, L\rangle$

sa traduction est

$$p = \langle D \cup \{p\}, F_L \cup \{\sigma :\rightarrow p\}, E_{\mathcal{I}} \cup E_{\mathcal{R}}\rangle$$

Exemple 1.1. (suite)

▼

La description de $Pile^{\infty}$ devient la présentation

$$
\begin{array}{lll}
Pile^{\infty} = \langle Nat,\ Suite\ de\ Nat,\ Pile^{\infty} & \} & D \cup \{p\} \\
\quad cons : Suite\ de\ Nat,\ Nat \to Nat & \} & F \\
\quad \Lambda : \to Suite\ de\ Nat & & \\
\quad \delta.i : Nat \times Pile^{\infty} \to Pile^{\infty} & \} & P(K) \\
\quad o.\delta : Nat \times Pile^{\infty} \to Pile^{\infty} & & \\
\quad A : Suite\ de\ Nat \to Pile^{\infty} & \} & \overline{E} \\
\quad \sigma : \to Pile^{\infty} & & \\
\quad \sigma \equiv A(\Lambda) & \} & E_I \\
\quad \delta.i(n.A(x)) \equiv A(cons(x.n)) & \} & E_R \\
\quad o.\delta(n.A(cons(x.n)) \equiv A(x) & &
\end{array}
\qquad \} F_L
$$

En considérant $\mathcal{L} = \langle D \cup \{p\},\ F_L \cup \{\sigma\} \rangle$ et en utilisant la notation habituelle pour les naturels, on aura :

$o.\delta(3,A(cons(cons(y,1),3)))$

$o.\delta(3,\delta.i(3,A(cons(y,1))))$

$o.\delta(3,\delta.i(3,o.\delta(2,A(cons(cons(y,1)),2))))$

$o.\delta(3,\delta.i(3,o.\delta(2,\delta.i(2,A(cons(y,1))))))$

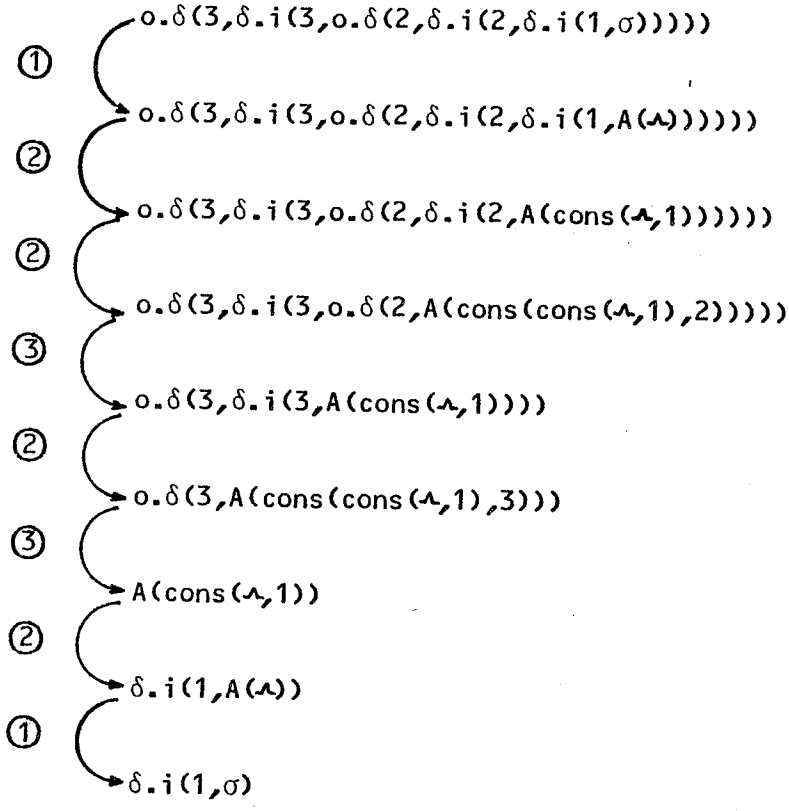
$o.\delta(3,\delta.i(3,o.\delta(2,\delta.i(2,\delta.i(1,A(y))))))$

$o.\delta(3,\delta.i(3,o.\delta(2,\delta.i(2,\delta.i(1,A(\Lambda))))))$

$o.\delta(3,\delta.i(3,o.\delta(2,\delta.i(2,\delta.i(1,\sigma))))))$

qui sont tous des termes du langage d'algèbres du processus $Pile^{\infty}$, c-à-d, qui ils appartiennent à $T(\mathcal{L})$.

Dans tous ces termes, si l'on substitue la variable y par le terme λ, alors, ils sont des termes dans l'univers de Herbrand de la présentation $Pile^{\infty}$. De plus, ils appartiennent tous à la même congruence $\sim_{E_I \cup E_R}$ dans l'algèbre $\mathcal{T}(\mathcal{L})$. Cela est prouvé par la suivante déduction :



où ① implique l'utilisation de l'équation $\sigma \equiv A(\lambda)$

② " " " " $\delta.i(n,A(x)) \equiv A(cons(x,n))$

③ " " " " $o.\delta(n,A(cons(x,n))) \equiv A(x)$

▲

1.2.3.2. Description d'un Processus = Système de Transitions : *SEMANTIQUE OPERATIONNELLE.*

Nous présentons ici la sémantique opérationnelle de la description d'un processus. Pour cela, on introduit :

- un Système de Transitions d'Etats,
- un Système de Transitions de Termes.

La première interprétation de la description donne un automate, en général non-déterministe et infini. La deuxième interprétation donne une sorte de système de réécriture de termes avec événements. On peut dire que la première est le modèle conceptuel et la deuxième est un interprétation finie de l'automate infini.

1.2.3.2.1. Système de Transitions d'Etats

Un système de transitions d'états est défini par un triplet $S = (Q, T, Rel)$ où :

- Q est un ensemble (non nécessairement fini) d'états,
- T est un ensemble (non nécessairement fini) de transitions (dans notre cas d'événements)
- Rel est un sous-ensemble de $Q \times T \times Q$

Nous allons construire un système de transitions d'états à partir d'une description de processus :

Soient $p = \langle \mathcal{L}, I, R \rangle$ (où $\mathcal{L} = \langle D, L \rangle$ avec L contenant F,K,E) une description de processus, et

$H_F(p) = \{c \mid var(c) \neq \emptyset \wedge c \in T_d \in T(\mathcal{L})$ pour chaque $d \in D\}$
l'univers de Herbrand des opérations dans p

La saturation dans $H_F(p)$ d'une règle $t_1 \xrightarrow{t_2} t_3 \in R$ est l'ensemble

$$Sat_{H_F(p)}\ (t_1 \xrightarrow{t_2} t_3) = \{e_1 \xrightarrow{e_2} e_3 \mid e_1 = \theta t_1 \wedge e_2 = \theta t_2 \wedge e_3 = \theta t_3 \wedge \theta : var(t_1) \cup var(t_2) \longrightarrow H_F(p)\}$$

Il s'agit donc de l'ensemble de toutes les règles $e_1 \xrightarrow{e_2} e_3$ qui sont obtenues quand on substitue les variables dans $t_1 \xrightarrow{t_2} t_3$, par les termes de $H_F(p)$. Les occurrences de la même variable dans la règle sont substituées par le même terme. Il faut remarquer que tous les termes d'état ou d'événement qui apparaissent dans cet ensemble sont des termes de Herbrand.

Soient Univers-de-règles $= \bigcup_{\varphi \in R} Sat_{H_{F(P)}}(\varphi)$ où $\varphi = t_1 \xrightarrow{t_2} t_3$

$$\text{Etats} = \{e_1, e_3 \mid e_1 \xrightarrow{e_2} e_3 \in \text{Univers-de-règles}\}$$

$$\text{Even} = \{e_2 \mid e_1 \xrightarrow{e_2} e_3 \in \text{Univers-de-règles}\}$$

$$\text{Initiaux} = \bigcup_{e \in I} Sat_{H_F(p)}(e) \text{ où } Sat_{H_F(p)}(e) = \{e' \mid \exists \theta : var(e) \to H_F(p)$$

$$\text{t.q. } e' = \theta e \wedge var(e') = \emptyset\}$$

L'évolution de la description p est une relation $\rightsquigarrow_n \subseteq$ Etats x Even x Etats, définie par (on écrit $a \overset{b}{\rightsquigarrow}_n c$, au lieu de $\langle a,b,c\rangle \in \rightsquigarrow_n$) :

1. $e_1 \overset{e_2}{\underset{o}{\rightsquigarrow}} e_3$ ssi $e_1 \xrightarrow{e_2} e_3 \in$ Univers-de-règles et $e_1 \in$ Initiaux

2. $e_1 \overset{e_2}{\underset{n}{\rightsquigarrow}} e_3$ ssi $e_1 \xrightarrow{e_2} e_3 \in$ Univers-de-règles

et

$\exists\, e'_1 \overset{e'_2}{\underset{n-1}{\rightsquigarrow}} e'_3$ t.q. $e'_3 = e_1$

Donc, le système de transitions d'états d'une description est défini par :

$S = (\text{Etats}, \text{Even}, \rightsquigarrow)$ où $\rightsquigarrow = \bigcup_{n \in \mathbb{N}} \rightsquigarrow_n$

Ce système de transitions d'états donné par une description de processus définit un automate (en général non-déterministe) qui peut avoir un nombre infini d'états et d'événements.

Cet automate peut être représenté par un graphe, où :

- Etats est l'ensemble des sommets,
- Even est l'ensemble d'étiquettes sur les arcs,
- Il y a un arc de c à c" étiqueté par c' ssi $c \overset{c'}{\underset{n}{\rightsquigarrow}} c''$ pour $n \in \mathbb{N}$.

Ce graphe est donc, en général, infini.

Exemple 1.1. (suite)

▼

La description de $Pile^{\infty}$ devient le système de transitions d'états

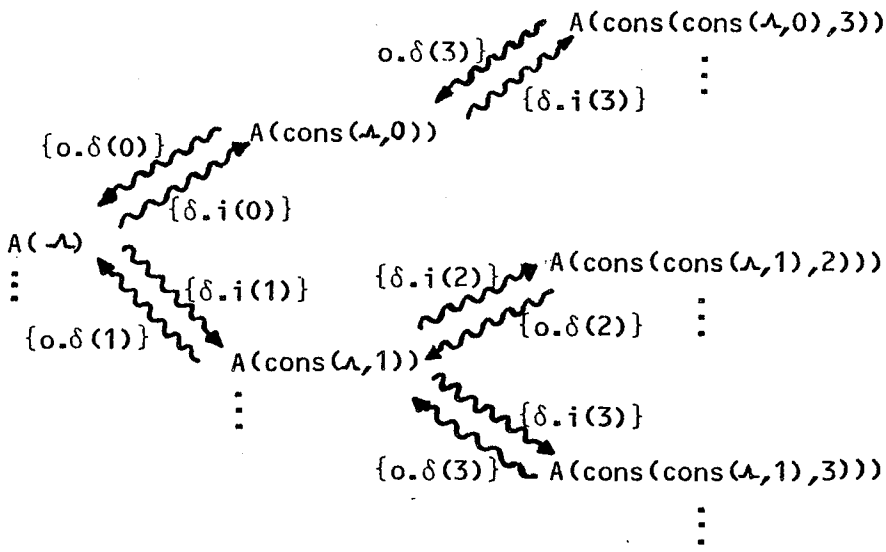
$$S_{Pile^{\infty}} = (Etats_{Pile^{\infty}}, Even_{Pile^{\infty}}, \rightsquigarrow)$$

Voici quelques détails :

$$\left.\begin{array}{l} A(\Lambda) \xrightarrow{\{\delta.i(1)\}} A(cons(\Lambda,1)) \\ A(cons(\Lambda,1)) \xrightarrow{\{\delta.i(2)\}} A(cons(cons(\Lambda,1),2)) \\ A(cons(\Lambda,1)) \xrightarrow{\{\delta.i(3)\}} A(cons(cons(\Lambda,1),3) \end{array}\right\} \in Sat_{H_p(F)}(A(x) \xrightarrow{\{\delta.i(n)\}} A(cons(x,n))$$

$$\left.\begin{array}{l} A(cons(cons(\Lambda,1),2)) \xrightarrow{\{o.\delta(2)\}} A(cons(\Lambda,1)) \\ A(cons(cons(\Lambda,1),3)) \xrightarrow{\{o.\delta(3)\}} A(cons(\Lambda,1)) \end{array}\right\} \in Sat_{H_p(F)}(A(cons(x,n) \xrightarrow{\{o.\delta(n)\}} A(x))$$

Voici une partie de la relation $\rightsquigarrow$



▲

1.2.3.2.2. Un Système de Transition de Termes

L'étude de certaines propriétés sur les processus est plus facilement réalisable si l'on a une conception plus "finie" de cette machine infinie que l'on vient de formaliser. Ainsi, les réseaux de Petri ou les "Programmes Parallèles [Kel 76]" sont des formalismes opérationnels pour avoir une représentation finie de ces relations infinies.

Dans notre cas, la représentation finie est la description d'un processus, si on l'interprète comme un système de transitions de termes. Ce système de transition est à la base du calcul de la relation $\rightsquigarrow_n$. Ce calcul peut, en effet, être fait directement à partir de la description d'un processus, en utilisant les règles comme un système de réécriture de termes, auquel on ajoute des événements. L'idée est que, dans chaque règle $t_1 \xrightarrow{t_2} t_3 \in R$ d'une description, les termes t_1, t_3 représentent une quantité infinie d'états et le terme t_2, une quantité infinie d'événements. Donc, une règle sera applicable ssi l'état courant de la machine appartient aux états dénotés par t_1

Le résultat d'une telle application est que

- l'évènement "correspondant" dans t_2 se produit,
- l'état "correspondant" dans t_3 devient le nouvel état de la machine.

Cette correspondance est formalisée au moyen de la substitution effectuée sur les variables dans t_1 et t_2.

Formellement :

Soient $H_F(p)$ l'univers de Herbrand des opérations dans $p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R} \rangle$

$t \in T(\mathcal{L})$

$$Sat_{H_F(p)}(t) = \{e \mid \theta t = e \quad \wedge \quad \theta : var(t) \rightarrow H_F(p)\}$$

Dans le cas où $t \in T_{état}$, l'ensemble $Sat_{H_F(p)}(t)$ contient tous les états dénotés par t.

Soit $$Etats = \bigcup_t \{Sat_{H_F(p)}(t) \mid t \in T_{état} \in T(\mathcal{L})\}$$

Une règle $t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}$ est dite <u>applicable</u> à partir d'un état $e \in Etats$, s'il existe une substitution $\theta : var(t_1) \cup var(t_2) \rightarrow H_F(p)$ t.q. $\theta t_1 = e$.

Lorsque cette règle est appliquée :

- <u>l'événement qui se produit</u> est θt_2, et
- <u>l'état résultat</u> est θt_3

Tout cela justifie que l'on considère l'ensemble $\mathcal{R}$ dans p comme <u>un système de transitions de termes avec événements</u>

Comme dans les systèmes de réécriture de termes, le calcul de la substitution est fait par l'algorithme d'unification [Rob 65, 71].

Il faut noter que la différence avec un système de réécriture normal (voir par exemple [Der 82]) n'est pas seulement l'existence d'un terme d'événement dans chaque règle, mais aussi que les variables qui n'apparaissent pas à gauche, mais dans les portes d'entrée de l'événement, peuvent apparaître à droite.

Il y a aussi une autre différence qui a des conséquences très importantes dans l'analyse d'un processus, comme on le verra dans la section 1.2.5. du chapitre 3 : il s'agit du remplacement total du terme quand la règle est appliquée lors d'une transition.

Enfin,

$$\text{Soit} \quad \text{Even} = \bigcup_{t} \text{Sat}_{H_F(p)}\ (t) \quad \text{t.q.} \quad t \in T_{\text{évèn}} \in T(\mathcal{L}).$$

<u>La relation calculée par la description d'un processus</u> $p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R} \rangle$ est

$$\vdash = \bigcup_{n \in \mathbb{N}} \vdash_n \subseteq \text{Etat} \times \text{Evén} \times \text{Etat}$$

où

$e_1 \vdash_0^{e_2} e_3$ ssi il existe $t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}$ et $e_1 \in \mathcal{I}$ t.q. la règle est applicable à partir de l'état e_1

c-à-d, $\exists\ \theta : \text{var}(t_1) \cup \text{var}(t_2) \to H_F(p)$
t.q. $\theta t_1 = e$ et $\theta t_2 = e_2$ et $\theta t_3 = e_3$

$e_1 \vdash_n^{e_2} e_3$ ssi il existe $t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}$ et $e'_1 \vdash_{n-1}^{e'_2} e_1$ t.q. la règle est applicable à partir de l'état e_1

On peut prouver sans peine que $\vdash_n = \rightsquigarrow_n$.

Comme dans le cas de systèmes de transition d'états, on associe un graphe à la description. Mais cette fois, le graphe est fini.
Dans ce graphe :

- l'ensemble des sommets est l'ensemble $\mathcal{R}$ de règles et l'ensemble d'états initiaux.

- les arcs entre les sommets existent chaque fois que l'on peut passer d'un état du côté droit de la règle (ou d'un état initial) à un état du côté gauche d'une autre règle.

Formellement,

Soit $p = \langle \mathcal{L}, \mathcal{I}', \mathcal{R} \rangle$

Le graphe associé à la description p est défini par

$N = \mathcal{R} \cup \mathcal{I}$

$A : N \rightarrow \mathcal{P}(N)$

t.q. $t_1' \xrightarrow{t_2'} t_3' \in A(t_1 \xrightarrow{t_2} t_3)$ ssi $Sat_{H_F(p)}(t_3) \cap Sat_{H_F(p)}(t_1') \neq \emptyset$

et $t_1' \xrightarrow{t_2'} t_3' \in A(t_1)$ ssi $Sat_{H_F(p)}(t_1) \cap Sat_{H_F(p)}(t_1') \neq \emptyset$

Sur ce graphe, il est utile de faire la distinction entre les arcs (origine → destination) qui sont :

- "toujours possibles" (pour n'importe quel état résultat dans le sommet origine, on peut appliquer la règle du sommet destination),

- "parfois possibles" (il y a certains états résultatsdans le sommet origine qui ne permettent pas d'appliquer la règle du sommet destination).

Formellement,

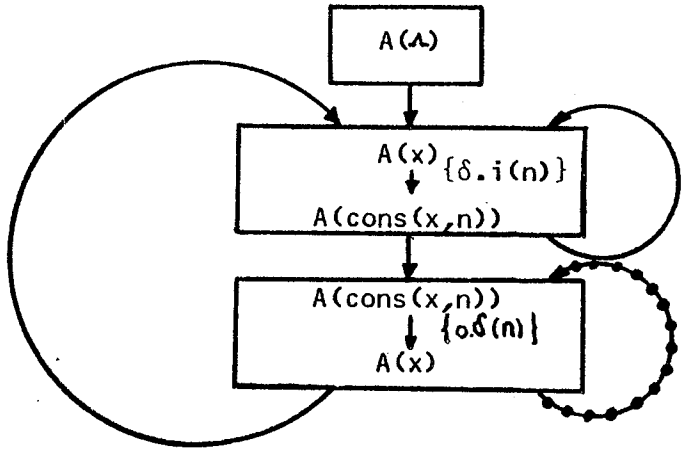
Nous disons que l'arc entre le sommet origine $t_1 \xrightarrow{t_2} t_3$ et le sommet destination $t_1' \xrightarrow{t_2'} t_3'$ (où $t_1' \xrightarrow{t_2'} t_3' \in A(t_1 \xrightarrow{t_2} t_3)$) est <u>toujours possible</u> ssi

$$Sat_{H_F(p)}(t_1') \subseteq Sat_{H_F(p)}(t_3)$$

Les arcs parfois possibles sont représentés dans l'exemple par ⟿

Cette distinction entre les arcs doit être considérée et exploitée dans l'analyse des processus dans l'avenir.

Exemple 1.1. (suite)



1.2.3.3. Description d'un Processus = Arbre (fini ou infini) : *SEMANTIQUE ARBORESCENTE*

Dans cette section, nous présentons une interprétation arborescente pour une famille de descriptions de processus.

L'intérêt, dans cette sémantique, est centré sur l'étude des comportements infinis d'un processus. Ces comportements sont étudiés du point de vue des événements et non sur les valeurs transmises ou reçues.

Les descriptions auxquelles cette sémantique est orientée sont les processus dont le comportement des suites d'événements ne dépend pas des valeurs reçues.

Cette sémantique rejoint, en quelque sorte, "les arbres de synchronisation" de R. Milner [Mil 80], c-à-d les arbres à branchement finis où les arcs sont étiquetés avec les éléments d'un ensemble fixe L d'actions observales, ou avec τ (l'action non-observale).

1.2.3.3.1. Avant propos : L'Espace Ultramétrique des Arbres Infinis

Soit n un entier

$[n] = \{1, \ldots, n\}$ représente l'ensemble des n premiers entiers, et $[n]^*$ l'ensemble des mots finis sur un alphabet [n] (ε dénote le mot vide)

Le domaine d'une fonction partielle $f : A \rightarrow B$ sera noté dom(f)

Soit X un ensemble quelconque, et n un entier

Un arbre, dont le degré est au plus n, sur X est une fonction partielle de la forme

$$t : [n]^* \to X$$

telle que dom(t) a les propriétés suivantes :

i) les "ancêtres" sont dans l'arbre, c-à-d en notant $\leqslant$ l'opération "préfixe de "

$\forall \mu \in dom(t), \nu \in [n]^*. (\nu \leqslant \mu \Rightarrow \nu \in dom(t)$

cette propriété est connue aussi sous le nom de "*préfixe fermé*"

ii) les "frères cadets" sont dans l'arbres, c-à-d

$\forall p,p' \in [n]. (\mu p \in dom(t) \wedge p' < p \Rightarrow \mu p' \in dom(t))$

La différence entre cette définition et celle utilisée habituellement (voir [Cou 81], [Arn-Niv 78]) est que, pour nous, l'arbre n'est pas défini sur un alphabet gradué. Avec les arbres définis sur les alphabets gradués, A. Arnold et M. Nivat ont fait un développement de la théorie des arbres infinis dans le contexte de la topologie ; plus précisément, ils ont défini l'espace ultramétrique des arbres infinis. Cependant, cette différence n'introduit aucun problème pour faire un développement parallèle à ce qu'ils ont fait.

Ainsi,

nous pouvons définir, inductivement, $A_n^i(X)$, l'ensemble des arbres sur X dont le degré est au plus n et la profondeur i, par

pour $i = o$ $\quad X = A_n^o(X)$

pour $i > o$ $\quad \forall x \in X, \forall\ o < j \leqslant n.$ si $t_1,\ldots,t_j \in A_n^{i-1}$

alors $x(t_1,\ldots,t_j) \in A_n^i(X)$

Donc, $M_n(X) = \bigcup_{o \leqslant i < w} A_n^i$ est l'ensemble des arbres finis dont le degré est au plus n.

Pour pouvoir définir la notion de "*proximité*" entre deux arbres, il nous faut une fonction qui calcule l'arité d'un sommet quelconque dans un arbre. Nous pouvons définir cette fonction de la façon suivante :

<u>l'arité</u> du sommet μ d'un arbre $t \in M_n(x)$ est un entier "arité(μ)" défini par

$$\text{arité} : \text{dom}(t) \to \mathbb{N}$$

avec

$$\text{arité}(\mu) = \begin{cases} 0 & \text{si } \nexists\, p \in [n] \text{ t.q. } \mu p \in \text{dom}(t) \\ p & \text{si } p = \max\{i \mid \mu i \in \text{dom}(t)\} \end{cases}$$

Avec cette fonction, on peut donc définir

<u>la troncature à profondeur m</u> d'un arbre t comme l'image de t par l'application

$$\alpha_m : M_n(X) \to M_n(X \cup \{\Omega\})$$

où Ω est un nouveau symbole $\Omega \in X$, qui exprime que la branche d'un arbre a été éliminée dans la troncature.

L'application α_m est définie comme

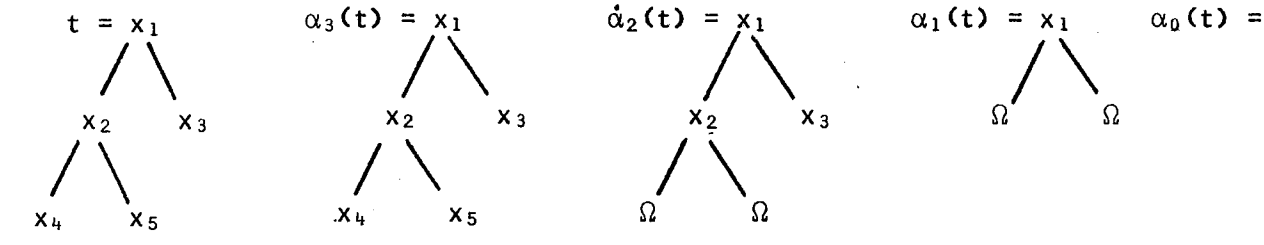
$$\alpha_o(t) = \Omega \qquad \text{pour chaque } t$$

$$\alpha_{m+1}(t) = \begin{cases} t & \text{si arité}(\varepsilon) = 0 \\ x(\alpha_m(t_1), \ldots, \alpha_m(t_p)) & \text{si arité}(\varepsilon) = p \neq 0 \text{ et } t = x(t_1, \ldots, t_p) \end{cases}$$

c-à-d, on remplace les sommets des branches suivantes au niveau m par Ω

Exemple

▼



▲

Avec ces définitions, nous pouvons suivre le même chemin que A. Arnold - M. Nivat, à savoir :

- la notion de "proximité" utilisée est

  deux arbres finis sont <u>proches</u> si la profondeur minimum à partir de laquelle on obtient des arbres différents par troncation, est suffisamment grande.

Pour mesurer cette <u>proximité</u>, on peut maintenant définir

$$\bar{\alpha}(t_1,t_2) = \min\{m \mid \alpha_m(t_1) \neq \alpha_m(t_2)\}$$

Nous voulons utiliser cette notion de proximité pour définir un "espace" d'arbres infinis, à la façon de la complétion de Cantor.

Pour utiliser la méthode de Cantor, il faut avoir une fonction à valeurs réelles avec les mêmes propriétés que la valeur absolue pour pouvoir définir une notion de "convergence" dans l'espace et construire un "*nouvel espace*". Cette fonction est <u>la distance</u> entre deux arbres t_1 et t_2 définie comme une application :

$$\text{distance} : M_n(X) \times M_n(X) \to \mathbb{R}_+$$

par

$$\text{distance}(t_1,t_2) = \begin{cases} 0 & \text{si } t_1 = t_2 \\ 2^{-\bar{\alpha}(t_1,t_2)} & \text{si } t_1 \neq t_2 \end{cases}$$

plus la distance entre t_1 et t_2 est petite, plus t_1 et t_2 seront proches.

Cette fonction a les propriétés de la valeur absolue, mais aussi une condition plus forte que l'inégalité triangulaire. Cette propriété est la suivante :

$$\text{distance}(t_1,t_3) \leq \max(\text{distance}(t_1,t_2), \text{distance}(t_2,t_3))$$

et elle est connue comme <u>la propriété ultramétrique</u>.

On considère alors l'espace ultramétrique des arbres finis comme l'espace $(M_n(X), \text{distance})$. Et on dit qu'une suite $(t_n)_{n \in \mathbb{N}}$ dans $M_n(X)$ est <u>convergente</u> ss'il existe $\hat{t} \in M_n(x)$ tel que

$$\forall \varepsilon > 0. \exists n \in \mathbb{N}. \forall p \geq n \quad \text{distance}(t_p, \hat{t}) < \varepsilon$$

Cet élément $\hat{t}$, s'il existe, est appelé <u>la limite de la suite</u> $(t_n)_{n \in \mathbb{N}}$ et dénoté par $\lim_{n \to \infty} (t_n)$.

Donc, une suite $(t_n)_{n \in \mathbb{N}}$ est <u>une suite de Cauchy</u> si :

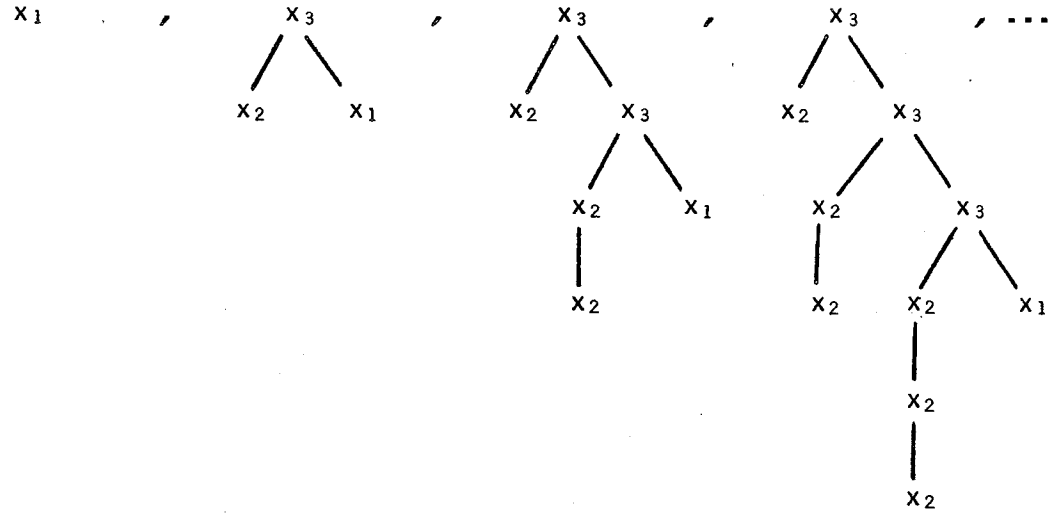
$$\forall \varepsilon > 0. \exists n \in \mathbb{N}. \forall p \geqslant n. \forall q \geqslant n \quad \text{distance } (t_p, t_q) < \varepsilon$$

et du fait que la distance est ultramétrique, on a le théorème suivant :

<u>Théorème 1.1.</u>

Une suite $(t_n)_{n \in \mathbb{N}}$ est une suite de Cauchy ssi ssi $\lim_{n \to \infty}$ distance $(t_n, t_{n+1}) = 0$

Finalement, puisqu'il y a des suites de Cauchy qui ne convergent pas dans l'espace,



on effectue <u>la complétion de Cantor</u> de l'espace $(M_n(X)$, distance), c-à-d, plonger l'espace ultramétrique dans un nouvel espace ultramétrique $(M_n^{\infty}(X)$, distance) dans lequel :

- $M_n^{\infty}(X)$ est <u>complète</u>, c-à-d, toute suite de Cauchy d'éléments de $M_n^{\infty}(X)$ converge dans $M_n^{\infty}(X)$.

- $M_n^\infty(X)$ est l'ensemble de classes d'équivalences de suites de Cauchy d'éléments de $M_n(X)$ avec l'équivalence suivante :

$$t = (t_n)_{n \in \mathbb{N}} \approx t' = (t'_n)_{n \in \mathbb{N}} \text{ ssi } \lim_{n \to \infty} \text{distance } (t_n, t'_n) = 0$$

- $M_n(X)$ est <u>dense</u> dans $M_n^\infty(X)$, c-à-d, tout élément dans $M_n^\infty(X)$ est limite d'une suite convergente d'éléments de $M_n(X)$.

Tout cela est assuré par le théorème :

<u>Théorème de complétion</u> (voir, par exemple [Dug 66], p. 304)

Tout espace métrique (Y,d) peut être isométriquement plongé dans un espace métrique complet $(\hat{Y},\hat{d})$. Cet espace métrique est unique (à une isométrie près) et Y est dense dans $\hat{Y}$.

On peut faire l'extension de α_m sur $M_n^\infty(X)$ de la manière suivante :

Soit $T \in M_n^\infty(X)$

Soient $t,t' \in B_\varepsilon^o(T) = \{t'' \in M_n(X) \mid \widehat{\text{distance}}\ (T,t'') < \varepsilon = 2^{-m}$
c-à-d, t,t' sont deux arbres finis égaux au moins jusqu'au niveau m.
Comme $\alpha_m(t) = \alpha_m(t')$, donc α_m posseds une valeur commune dans $B_\varepsilon^o(T)$.
Alors, nous définissons $\alpha_m(T)$ comme cette valeur commune.

1.2.3.3.2. <u>La Description d'un Processus</u> = <u>Une suite de Cauchy</u>

Comme nous l'avons signalé au début de cette section (1.2.3.3.), nous considérons ici uniquement les descriptions de processus dont le comportement de ses suites d'événement est indépendant des valeurs reçues par le processus. Cette limitation a pour objectif d'éliminer la possibilité d'un branchement infini dans l'arbre associé à la description. C'est pour cela que l'on considère que, au moment ou se produit un événement, les valeurs dans les portes d'entrée sont des termes avec variables qui représentent l'ensemble des valeurs possibles qui peuvent être transmises ou reçues.

L'application d'une règle introduit des variables nouvelles par rapport au terme d'état qui permet l'application de la règle.

Ainsi, cette sémantique arborescente peut être considérée, en quelque sorte, comme une évaluation symbolique du processus décrit. Cette évaluation symbolique est, par rapport au messages transmis ou reçus, puisqu'ils sont considérés comme des termes avec variables.

Soient $\mathcal{L} = \langle D, L \rangle$ un langage de processus

$V(\mathcal{L})$, $T(\mathcal{L}) = \{T_d | d \in D\}$, $T_{état}$ et $T_{évén}$ comme dans 1.2.2.

Soient $X = T_{état} \times T_{évén}$

$p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R} \rangle$ une description de processus où son comportement de suites d'événements ne dépend pas des valeurs reçues.

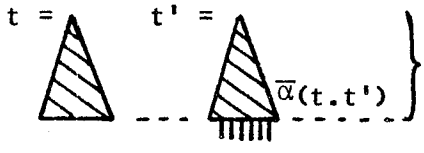
$n = card(\mathcal{R})$. Le branchement dans $M_n(X)$ est toujours fini puisque $n < \omega$.

Nous considérons l'ensemble $\mathcal{R}$ comme un ensemble itératif de règles et nous définissons la fonction à appliquer dans chaque itération de $\mathcal{R}$, comme suit :

$$I_{\mathcal{R}} : M_n(X) \rightarrow M_n(X)$$

où $I_{\mathcal{R}}(t) = t'$

tel que



1. $t : dom(t') - \{\mu \mid |\mu| = \overline{\alpha}(t.t') + 1\} \rightarrow X$
et $\forall \alpha \in dom(t) \; . \; t(\alpha) = t'(\alpha)$

2. $\forall t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}$
$\forall \mu \in dom(t)$ t.q. $|\mu| = \overline{\alpha}(t.t')$
$\wedge \; t(\mu) = (q, e)$
pour chaque q et e
s'il existe $\theta : var(t_1) \cup var(t_2) \rightarrow \{T_d | d \in D\}$
t.q. $q = \theta t_1$

c-à-d, la fonction I_R prend, comme argument, un arbre t et construit un autre arbre t' en ajoutant un nouveau niveau à l'arbre t. Ce nouveau niveau est formé par tous les événements et les états possibles correspondants obtenus par l'application de l'ensemble des règles R à chaque élément dans la frontière de t.

Donc, les itérations successives finies d'une description de processus $p = \langle \mathcal{L}, I, R \rangle$ sont représentées par la suite

$$(I^n_{R'}(t_0))_{n \in \mathbb{N}}$$

où

$$R' = R \cup \{\sigma \xrightarrow{\tau} e \mid e \in I\} \quad \text{où} \quad \sigma \notin T(\mathcal{L})$$
$$t_0 : \{\varepsilon\} \to X \cup \{\sigma\} \quad \text{t.q.} \quad t_0(\varepsilon) = \langle \sigma, \tau \rangle$$

Proposition 1.3.

la suite $(I^n_{R'}(t_0))_{n \to \infty}$ est une suite de Cauchy

Preuve :

■

Par le théorème 1.2., prouver cela revient à prouver

$$\lim_{n \to \infty} \text{distance } (I^n_{R'}(t_0),\ I^{n+1}_{R'}(t_0)) = 0$$

Pour prouver cela, on prend deux cas :

Cas 1 : il existe un $n \in \mathbb{N}$ t.q. $I^n_{R'}(t_0) = I^{n+1}_{R'}(t_0)$

donc, par la définition de I_R on a $I^n_{R'}(t_0) = I^j_{R'}(t_0)$ pour tout $n \geq j$

d'où, en utilisant la définition de distance, on a

$$\lim_{n \to \infty} \text{distance } (I^n_{R'}(t_0),\ I^{n+1}_{R'}(t_0)) = 0$$

Cas 2 : il n'existe pas un $n \in \mathbb{N}$ t.q. $I^n_{R'}(t_0) = I^{n+1}_{R'}(t_0)$

soit $m = \overline{\alpha}(I^n_{R'}(t_0), I^{n+1}_{R'}(t_0))$

donc, par la définition de $I_{R'}$, on peut prouver que

$$\overline{\alpha}(I^{n+1}_{R'}(t_0), I^{n+2}_{R'}(t_0) = m+1$$

d'où on peut écrire

$$\lim_{n\to\infty} \text{distance } (I^n_{R'}(t_0), I^{n+1}_{R'}(t_0)) = \lim_{n\to\infty} 2^{-m}$$

et par conséquent, égal à zéro.

■ c.q.f.d.

Par une meilleur approximation aux règles, nous préférons représenter, sur les arcs d'arbre , l'événement qui a eu lieu au passage d'un sommet à un de ses fils. Cette transformation dans l'arbre $\lim(I^n_{R'}(t_0))_{n\to\infty}$ est définie par l'arbre sur $T_{état}$ étiqueté par $T_{évén}$ de la manière suivante :

$$\lim \overline{(I^n_{R'}(t_0))}_{n\to\infty}$$

où

- dom $\overline{I^n_{R'}(t_0)}$ = dom $I^n_{R'}(t_0)$ pour chaque n
- les étiquettes sur les arcs de $\overline{I^n_{R'}(t_0)}$ sont définies par

  <u>étiquette</u> $(\mu,\mu i) = \text{projection}_2\ (I^n_{R'}(t_0)(\mu i))$ $\mu,\mu i \in \text{dom } \overline{I^n_{R'}(t_0)}$
- l'information dans les sommets de $I^n_{R'}(t_0)$ est définie par

  $\overline{I^n_{R'}(t_0)}(\mu) = \text{projection}_1\ (I^n_{R'}(t_0)\mu)$ $\mu \in \text{dom } \overline{I^n_{R'}(t_0)}$

où

projection_1 comme l'application classique $(q,e) \to q$, et
projection_2 " " " $(q,e) \to e$.

Exemple 1.1. (suite)

▼

$$R = \{C(w) \xrightarrow{\{\delta.i(n)\}} C(\mathrm{cons}(w,n)),\ C(\mathrm{cons}(w,n)) \xrightarrow{\{o.\delta(n)\}} C(w)\}$$

$$R' = R \cup \{\sigma \xrightarrow{\tau} C(\Lambda)\}$$

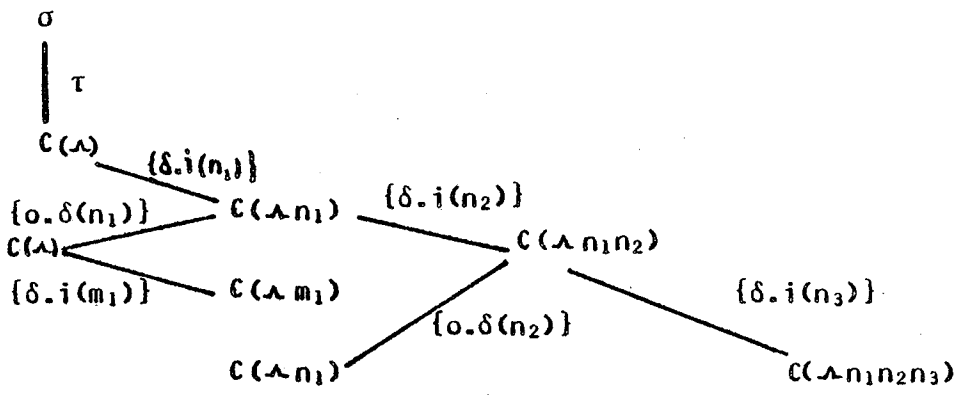
Pour faire la notation plus courte, on dénote $\mathrm{cons}(\mathrm{cons}(\ldots(\mathrm{cons}(\Lambda,n_1),\ldots),n_{i-1}),n_i)$ par $\Lambda n_1 n_2 \ldots n_{i-1} n_i$

$t_0 = \sigma$

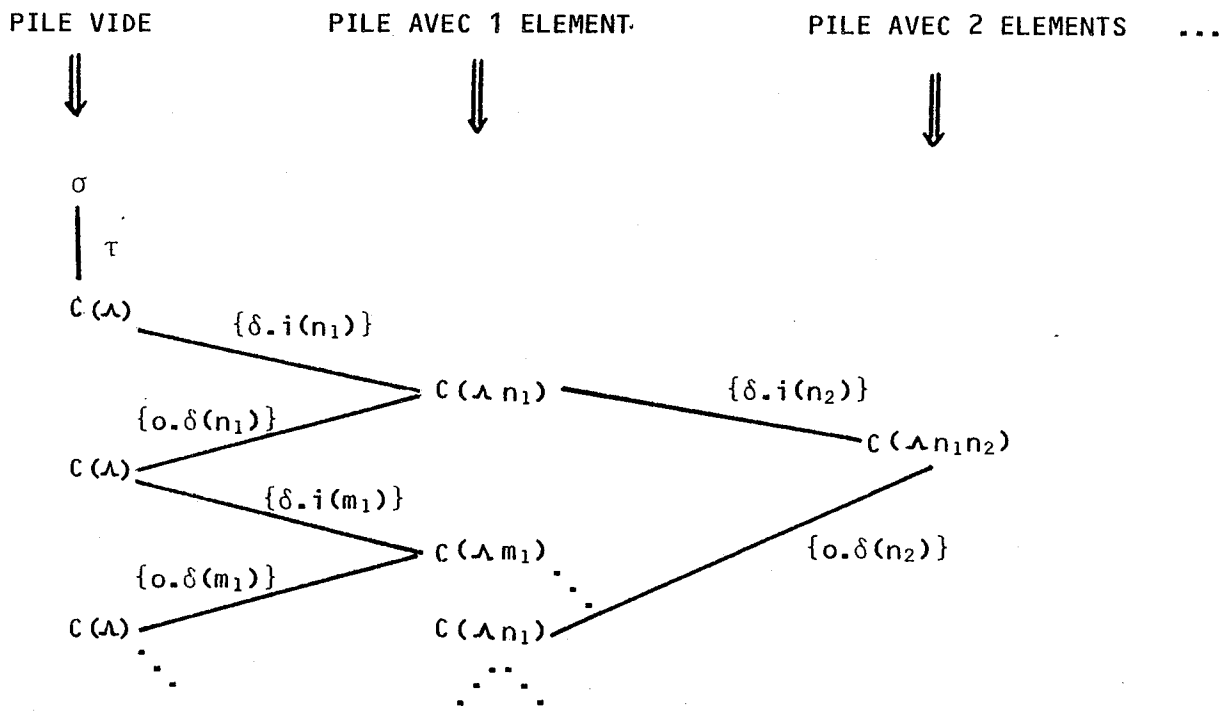
$I^1_{R'}(t_0)$ = σ —τ→ $C(\Lambda)$

$I^2_{R'}(t_0)$ = σ —τ→ $C(\Lambda)$ —$\{\delta.i(n_1)\}$→ $C(\Lambda,n_1)$

$I^3_{R'}(t_0)$ = σ —τ→ $C(\Lambda)$ —$\{\delta.i(n_1)\}$→ $C(\Lambda,n_1)$ —$\{o.\delta(n_1)\}$→ $C(\Lambda)$, —$\{\delta.i(n_2)\}$→ $C(\Lambda n_1 n_2)$



et $\lim_{n\to\infty} (I^n_{\mathcal{R}'}(t_0))$ peut être représenté par l'arbre



▲

Ces arbres de comportements de suites d'événements peuvent être étendus aux "arbres de communications" de R. Milner (voir [Mil 80]) dans le cas où $\lim_{n\to\infty} (I^n(t_0)) \in M_n(X)$, autrement dit, pour les comportements finis. Cette généralisation est similaire à celle réalisée par R. Milner dans le chapitre 6 de [Mil 80].

Les travaux de R. Milner [Mil 80], H.C.B. Hennessy - R. Milner [Hen-Mil 80], Ph. Darandeau [Dar 82] proposent des relations d'équivalence observationnelle pour les arbres de comportements (synchronisations et communications) finis. Nous pouvons utiliser ces résultats dans notre modèle.

1.2.4. La construction d'un SPC

La spécification de systèmes de processus communicants suppose la possibilité de "combiner" d'autres processus déjà spécifiés. Cela permet un degré d'abstraction qui facilite la "construction" de systèmes très complexes.

Pour définir cette combinaison, C.A.R. Hoare [Hoa 81a], R. Milner [Mil 81], W.C. Rounds - S.D. Brookes [Rou-Bro 81] utilisent des algèbres de processus. Une expression dans ces algèbres :

- a comme opérandes, des expressions de l'algèbre,
- décrit un système de processus, c-à-d, le "sens" d'une telle expression qui est dans [Hoa 81a] un ensemble de traces, dans [Mil 81] un arbre de synchronisation ou de communication, dans [Rou-Bro 81] un ensemble de traces et un arbre.

Dans le modèle que l'on a présenté, nous définissons aussi une algèbre d'expressions (calcul de Jorrand [Jor 81, 84]), pour avoir ce degré d'abstraction dans la construction de systèmes de processus communicants. Mais, contrairement aux autres auteurs cités, cette algèbre n'est pas une algèbre de processus dans le sens mentionné plus haut. En effet, dans cette algèbre, une expression

- a aussi comme opérandes des expressions dans l'algèbre,
- a pour résultat une description de processus.

Cela veut dire que cette algèbre est simplement une manipulation symbolique des descriptions aux opérandes des expressions. Pour connaître le "sens", il faut interpréter la description résultat par une des sémantiques données dans la section 1.2.3.

De cette façon, tout système de processus se réduit dans un processus simple.

1.2.4.1. L'Algèbre des Descriptions de Processus : Le Calcul de Jorrand

Puisque l'algèbre que l'on va définir est une manipulation formelle de descriptions, nous utilisons trois fonctions Ψ, ϕ, π qui nous permettent de faire certaines combinaisons de symboles et ainsi, définir les ensembles "produit d'états" et "produit de termes" d'états nécessaires pour la définition des opérateurs dans l'algèbre. Ces fonctions et ensembles sont les suivants :

Soient Ψ une combinaison arbitraire de suites finies (la concaténation de suites, par exemple).

E_1, E_2 deux ensembles de symboles d'états avec fonctions d'arité ρ_{E_1}, ρ_{E_2}

Le produit d'états $E_1 \times E_2$ est un ensemble de symboles d'états $\hat{E}$ avec fonction d'arité $\rho_{\hat{E}}$ t.q. il existe une fonction $\phi : E_1 \times E_2 \longleftrightarrow \hat{E}$ où, si $\phi(e_1, e_2) = e$, alors $\rho_{\hat{E}}(e) = \Psi(\rho_{E_1}(e_1), \rho_{E_2}(e_2))$

Exemple : on représente, entre crochets, les éléments d'un ensemble d'états et son arité

▼

Soient Ψ la comcaténation de suites

$$\text{Etat}_{\text{Buffer}^\infty} = [\ A : \text{sfl}\ ;\ B : \text{sfl,sfl,sfl}\ ;\ C : \text{sfl}\ ]$$

$$\text{Etat}_{\text{Impremante}} = [\ P : \text{fl}\ ]$$

Donc, le produit d'états

$$\text{Etat}_{\text{Buffer}^\infty} \times \text{Etat}_{\text{Impremante}} = \langle\ AP : \text{sfl,fl}\ ;\ BP : \text{sfl,sfl,sfl,sfl}\ CP : \text{sfl,sfl,fl}\ \rangle$$

où $\phi(A,P) = AP$, $\phi(B,P) = BP$, $\phi(C,P) = CP$

▲

Le produit de termes d'états $T_{Etat_{E_1}} \times T_{Etat_{E_2}}$ est un ensemble de termes d'états définis par la fonction,

$$\pi_{E_1 \times E_2} : T_{Etat_{E_1}} \times T_{Etat_{E_2}} \longrightarrow T_{Etat_{E_1 \times E_2}}$$

t.q.

$$\pi_{E_1 \times E_2}((e\ t_1,\ldots,t_n),(e'\ t'_1,\ldots,t'_m)) = e_1(\Psi(\langle t_1,\ldots,t_n\rangle,\langle t'_1\ldots t'_m\rangle))$$

où $\phi(e.e') = e_1$

Exemple : *(suite)*

$\pi(B(cons(x,b),Z,Y),P(\lambda)) = BP(cons(x,b),Z,Y,\lambda)$ où $\phi(B,P) = BP$

$\pi(C(\lambda,y),P(cons(w,s))) = CP(\lambda,y,cons(w,s))$ où $\phi(CP) = CP$

Nous définissons maintenant l'algèbre des descriptions de processus (le calcul de Jorrand).

Soit l'univers des descriptions de processus :

$$Descriptions = \{p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R}\rangle \mid \mathcal{L} \text{ est un langage de processus, } \mathcal{I} \subseteq T_{etat} \in T(\mathcal{L}) \quad \mathcal{R} \subseteq \mathcal{R}(\mathcal{L},V)\}$$

L'algèbre des descriptions de processus est la structure algébrique

$$< Descriptions, \|, +, - >$$

où les fonctions (opérateurs) fondamentales sont définies de la façon suivante :

L'opérateur d'union = $\|$: A x A → A

L'objectif de l'opérateur d'union ($\|$) est de mettre ensemble les processus opérandes dénotés par les descriptions de processus p_1 et p_2. Cela veut dire que le processus dénoté par la description résultat de l'expression

$p_1 \| p_2$ où $p_1 = \langle \mathcal{L}_{p_1}, \mathcal{I}_{p_1}, \mathcal{R}_{p_1} \rangle$ où $\mathcal{L}_{p_1} = \langle \mathcal{D}_{p_1}, L_{p_1} \rangle$
avec L_{p_1} composé de F_{p_1}, K_{p_1}, E_{p_1}

et $p_2 = \langle \mathcal{L}_{p_2}, \mathcal{I}_{p_2}, \mathcal{R}_{p_2} \rangle$ où $\mathcal{L}_{p_2} = \langle \mathcal{D}_{p_2}, L_{p_2} \rangle$
avec L_{p_2} composé de F_{p_2}, K_{p_2}, E_{p_2}

peut se comporter à chaque étape de son évolution :

1. soit, comme le processus p_1 évoluant en laissant le processus p_2 dans le même état.

2. soit, comme le processus p_2 évoluant en laissant le processus p_1 dans le même état.

3. Soit, comme les deux processus p_1 et p_2 évoluant simultanément.

Pour obtenir ce résultat, on définit les trois ensembles suivants de règles qui correspondent aux trois points ci-dessus :

$$\hat{\mathcal{R}}_1 = \{\pi(t_1,t'_1) \xrightarrow{t_2} \pi(t_3,t'_1) \mid t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}_{p_1}$$
$$(t'_1 = s(x_1,\ldots,x_n)$$
$$\wedge\ x_1,\ldots,x_n \notin var(t_1 \xrightarrow{t_2} t_3)$$
$$\wp E_{p_2}(s) = d_1 \ldots d_n)$$
$$\text{pour chaque } s \in E_{p_2} \quad \}$$

$$\hat{R}_2 = \{\pi(t_1,t_1') \xrightarrow{t_2'} \pi(t_1,t_3') \mid t_1' \xrightarrow{t_2'} t_3' \in R_{p_2} \wedge (t_1 = s(x_1,\ldots,x_n) \wedge x_1,\ldots,x_n \notin var(t_1' \xrightarrow{t_2'} t_3')) \wedge (E_{p_1}(s) = d_1\ldots d_n) \text{ pour chaque } s \in E_{p_1}\}$$

$$\hat{R}_3 = \{\pi(t_1,t_1') \xrightarrow{t_2 \cup t_2'} \pi(t_3,t_3') \mid t_1 \xrightarrow{t_2} t_3 \in R_{p_1} \wedge t_1' \xrightarrow{t_2'} t_3 \in R_{p_2}\}$$

Donc, la description définie par $p_1 \,||\, p_2$ est la suivante :

$$p_1 \,||\, p_2 = \langle \hat{\mathcal{L}}, \hat{\mathcal{I}}, \hat{\mathcal{R}} \rangle$$

où $\hat{\mathcal{L}} = \langle \mathcal{D}_{p_1} \cup \mathcal{D}_{p_2}, \hat{L} \rangle$ où $\hat{L}$ est composé de $Fp_1 \cup Fp_2$, $K_{p_1} \cup K_{p_2}$, $E_1 \times E_2$

$$\hat{\mathcal{I}} = \{\pi(e_1,e_2) \mid e_1 \in \mathcal{I}_1 \wedge e_2 \in \mathcal{I}_2\}$$

$$\hat{\mathcal{R}} = \hat{\mathcal{R}}_1 \cup \hat{\mathcal{R}}_2 \cup \hat{\mathcal{R}}_3$$

Exemple 1.13a : (nous considérons Ψ comme la concaténation de suites finies et ϕ comme dans les exemples précédents)

La description obtenue pour

$$Buffer^{\omega} \,||\, Imprimante$$ (voir exemples 1.3. et 1.5.)

est la suivante :

Buffer$^{\omega}$ || Imprimante = } $p_1 \,||\, p_2$

processus

connexions $\delta.i,o.\delta,\delta.p : fl$
$q.\delta : l$ } $K_{p_1} \cup K_{p_2}$

opérations

$\Lambda : \rightarrow sfl$
$cons : sfl,fl \rightarrow sfl$
$\Lambda : \rightarrow fl$
$cons : \rightarrow fl,l \rightarrow fl$ } $Fp_1 \cup Fp_2$

états

AP : sfl, fl

BP : sfl,sfl,sfl,fl

CP : sfl,sfl,fl

$E_{p_1} \times E_{p_2}$

$\pi(A(\Lambda), P(\Lambda$

initial

$AP(\Lambda, \Lambda)$

règles

$AP(x,v) \xrightarrow{\{\delta.i(a)\}} BP(x,\Lambda,cons(\Lambda,a),v)$

$BP(cons(x,b),z,v) \xrightarrow{\tau} BP(x,cons(z,b),v)$

$BP(\Lambda,z,y,v) \xrightarrow{\tau} CP(z,y,v)$

$CP(cons(z,b),y,v) \xrightarrow{\tau} CP(z,cons(y,b),v)$

$CP(\Lambda,y,v) \xrightarrow{\tau} AP(y,v)$

$AP(cons(x,b,v) \xrightarrow{\{o.\delta(b)\}} AP(x,v)$

$\hat{R}_1$

$AP(x,\Lambda) \xrightarrow{\{\delta.p(w)\}} AP(x,w)$

$AP(x,cons(w,s) \xrightarrow{\{q.\delta(s)\}} AP(x,w)$

$BP(x,z,y,\Lambda) \xrightarrow{\{\delta.p(w)\}} BP(x,z,y,w)$

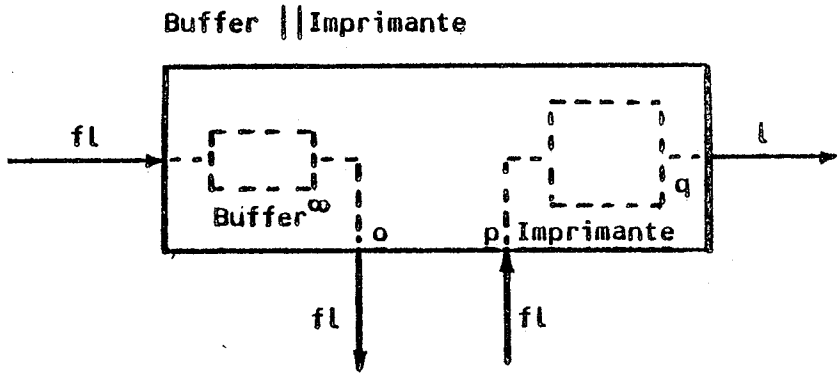
$BP(x,z,y,cons(w,s)) \xrightarrow{\{q.\delta(s)\}} BP(x,z,y,w)$

$CP(z,y,\Lambda) \xrightarrow{\{\delta.p(w)\}} CP(z,y,w)$

$CP(z,y,cons(w,s)) \xrightarrow{\{q.\delta(s)\}} CP(z,y,w)$

$\hat{R}_2$

$$
\begin{array}{lll}
AP(x,\Lambda) & \xrightarrow{\{\delta.i(a),\delta.p(w)\}} & BP(x,\Lambda,cons(\Lambda,a),w) \\
BP(cons(x,b),z,y,\Lambda) & \xrightarrow{\{\delta.p(w)\}} & BP(x,cons(z,b),y,w) \\
BP(\Lambda,z,y,\Lambda) & \xrightarrow{\{\delta.p(w)\}} & CP(z,y,w) \\
CP(cons(z,b),y,\ ) & \xrightarrow{\{\delta.p(w)\}} & CP(z,cons(y,b),w) \\
CP(\Lambda,y,\Lambda) & \xrightarrow{\{\delta.p(w)\}} & AP(y,w) \\
* \quad AP(cons(x,b),\Lambda) & \xrightarrow{\{o.\delta(b),\delta.p(w)\}} & AP(x,w) \\
AP(x,cons(w,s)) & \xrightarrow{\{\delta.i(a),q.\delta(s)\}} & BP(x,\Lambda,cons(\Lambda,a),w) \\
BP(cons(x,b),x,y,cons(w,s)) & \xrightarrow{\{q.\delta(s)\}} & BP(x,cons(z,b),y,w) \\
BP(\Lambda,x,y,cons(w,s)) & \xrightarrow{\{q.\delta(s)\}} & CP(z,y,w) \\
CP(cons(z,b),y,cons(w,s)) & \xrightarrow{\{q.\delta(s)\}} & CP(z,cons,(y,b),w) \\
CP(\Lambda,y,cons(w,s)) & \xrightarrow{\{q.\delta(s)\}} & AP(y,w) \\
AP(cons(x,b),cons(w,s)) & \xrightarrow{\{o.\delta(b),q.\delta(s)\}} & AP(x,w)
\end{array}
\qquad \hat{R}_3
$$



▲

L'opérateur de connexion = $+ : A \times K_{IN} \rightarrow A$

Soit $p = \langle \mathcal{L}, \mathcal{I}, \mathcal{R} \rangle$ où $\mathcal{L} = \langle D, L \rangle$ et L composé de F,K,E. L'objectif de l'opérateur de connexion est de construire un nouveau processus dans lequel deux portes (définies sur le même domaine), une d'entrée $\delta.i$ et l'autre de sortie $o.\delta$ sont connectées. Le nouveau processus permet ainsi toutes les communications de l'ancien processus, plus les communications sur $o.i$ dans les états où $\delta.i$ et $o.\delta$ sont possible simultanément.

Donc,

$p + o.i = \langle \hat{\mathcal{L}}, \hat{\mathcal{I}}, \hat{\mathcal{R}} \rangle$

où

- $\hat{\mathcal{L}} = \langle \hat{D}, \hat{L} \rangle$ et $\hat{L}$ contient $F, \hat{K} = K \cup \{o.i\}$, E
  où $\rho_{\hat{K}}(k) = \rho_K(k)$ pour $k \in K$
  et
  $\rho_{\hat{K}}(o.i) = \rho_K(i) = \rho_K(o)$

- $\hat{\mathcal{I}} = \mathcal{I}$

- $\hat{\mathcal{R}} = \mathcal{R} \cup \{\theta t_1 \xrightarrow{\theta t_2'} \theta t_3 \mid t_1 \xrightarrow{t_2} t_3 \in \mathcal{R}$
  $\exists t, u.\ o.\delta(t) \in t_2 \wedge \delta.i(u) \in t_2$
  $\wedge\ t_2' = t_2 - \{o.\delta(t), \delta.i(u)\} \cup \{i.o(\theta t)\}$
  $\wedge\ \theta = \{pgu(t,u)\}$

p.g.u(t,u) dénote le plus grand unificateur de t et u, c-à-d, une substitution $\theta : var(t) \cup var(u) \rightarrow T_{ope}(\hat{\mathcal{L}})$ t.q. $\theta t = \theta u$ est appelée l'unificateur de t et u. Si l'on a deux unificateurs θ' et θ'', θ'' est plus général que θ', s'il existe une substitution θ t.q. $\theta' = \theta \circ \theta''$. (le p.g.u (t,u) est calculé par l'algorithme d'unification [Rob 65, 71]).

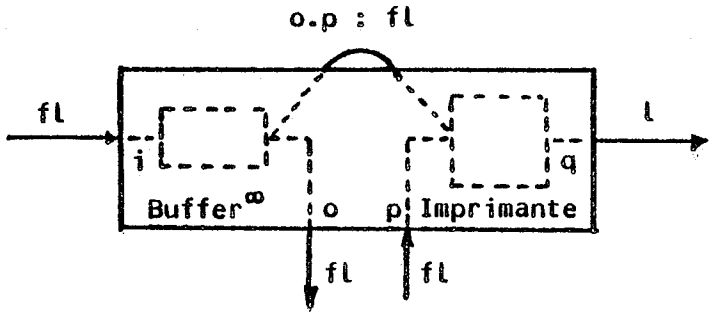
Exemple 1.13b (suite)

La description que l'on obtient pour (Buffer$^{\infty}$||Imprimante) + o.p est la même que dans *1.13a*, en ajoutant :

dans les connexions : o.p : fl

dans les règles : $AP(cons(x,b),\wedge) \xrightarrow{\{o.p(b)\}} AP(x,b)$

(voir la règle * dans $\hat{R}_3$ de l'*exemple 3.13a*)



L'opérateur de resctriction = - : A x K → A

Soit p = <£, I, R> où £ = <D,L> et L est composé de F,K,E L'objectif de l'opérateur de restriction est de construire un nouveau processus dans lequel la connexion K, (argument de l'opérateur), est rendue invisible. Cela veut dire, dans le cas où :

1. $k \in K_{EX}$, la porte est murée, c-à-d que le nouveau processus ne permet aucune communication sur cette porte.
2. $k \in K_{IN}$, le nouveau processus peut effectuer les communications (les échanges des messages à travers la connexion k), mais elles sont inobservables par le milieu extérieur.

Donc

$p - k = \langle \hat{\mathcal{L}}, \hat{\mathcal{I}}, \hat{\mathcal{R}} \rangle$

où

- $\hat{\mathcal{L}} = \langle D, \hat{L} \rangle$ où $\hat{L}$ est composé de $F, \hat{K} = K - \{k\}, E$

- $\hat{\mathcal{I}} = \mathcal{I}$

- $\hat{\mathcal{R}} = (\mathcal{R} - \mathcal{R}') \cup \mathcal{R}''$

avec

$$\mathcal{R}' = \{t_1 \xrightarrow{t_2} t_3 \mid t_1 \xrightarrow{t_2} t_3 \in \mathcal{R} \wedge \exists u \text{ t.q. } K(u) \in t_2\}$$

$$\mathcal{R}'' = \{t_1 \xrightarrow{t_2 - \{k(u)\}} t_3 \mid t_1 \xrightarrow{t_2} t_3 \in \mathcal{R} \wedge k \in K_{IN} \wedge k(u) \in t_2\}$$

Exemple 1.13c (suite)

▼

La description que l'on obtient pour $((Buffer^{\infty} || Imprimante)+o.p)-o.\delta$ es la même que dans *1.13b*, en

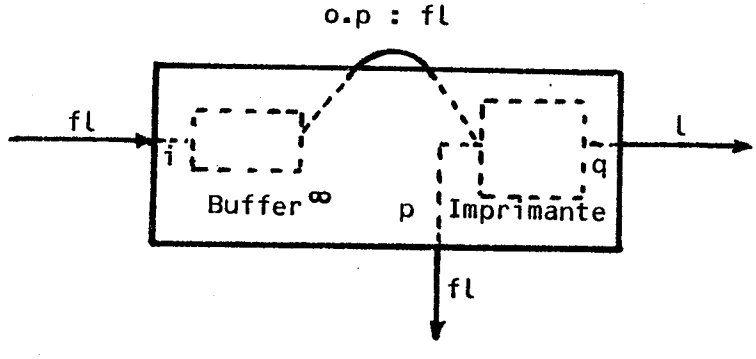
- éliminant les règles

$$AP(cons(x,b),v) \xrightarrow{\{o.\delta(b)\}} AP(x,v)$$

$$AP(cons(x,b),\Lambda) \xrightarrow{\{o.\delta(b),\delta.p(w)\}} AP(x,w)$$

$$AP(cons(x,b),cons(w,s)) \xrightarrow{\{o.\delta(b),q.\delta(s)\}} AP(x,w)$$

- éliminant dans les connexions $o.\delta$: fl



La description que l'on obtient pour ((Buffer$^{\infty}$||Imprimante)+o.p)-o.p est la même que dans *1.13b*, en

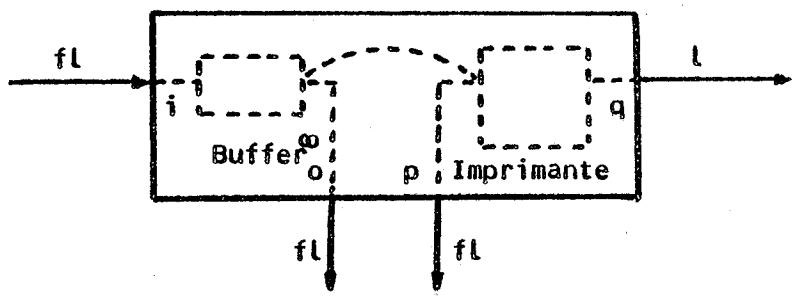
- éliminant la règles

$$AP(cons(x,b),\Lambda) \xrightarrow{\{o.p(b)\}} AP(x,b)$$

- éliminant, dans les connexions a.p : fl

- ajoutant la règle

$$AP(cons(x,b),\Lambda) \xrightarrow{\tau} AP(x,b)$$



Enfin,

la description que l'on obtient pour ((Buffer$^{\infty}$||Imprimante)+o.p)-o.δ -δ.p-o.p est la suivante

processus

connexions δ.i : fl
q.δ : l

opérations

Λ : → sfl
cons : sfl,l → sfl
Λ :→fl
cons : fl,l → fl

états AP : sfl,fl
BP : sfl,sfl,sfl,fl
CP : sfl,sfl,fl

initial AP(Λ,Λ)

règles

AP(x,v)	$\xrightarrow{\{\delta.i(a)\}}$	BP(x,Λ,cons(Λ,a),v)
BP(cons(x,b),z,v)	$\xrightarrow{\tau}$	BP(x,cons(z,b),v)
BP(Λ,z,y,v)	$\xrightarrow{\tau}$	CP(x,y,v)
CP(cons(z,b),y,v)	$\xrightarrow{\tau}$	CP(z,cons(y,b),v)
CP(Λ,y,v)	$\xrightarrow{\tau}$	AP(y,v)
AP(x,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	AP(x,w)
BP(x,z,y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	BP(x,z,y,w)
CP(z,y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	CP(z,y,w)
AP(x,cons(w,s))	$\xrightarrow{\{\delta.i(a),q.\delta(s)\}}$	BP(x,Λ,cons(Λ,a),w)
BP(cons(x,b),x,y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	BP(x,cons(z,b),y,w)
BP(Λ,x,y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	CP(z,y,w)
CP(cons(z,b),y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	CP(z,cons(y,b),w)
CP(Λ,y,cons(w,s))	$\xrightarrow{\{q.\delta(s)\}}$	AP(y,w)
AP(cons(x,b),Λ)	$\xrightarrow{\tau}$	AP(x,b)

▲

Finalement, il faut noter que l'évaluation des expressions dans l'algèbre de descriptions peut être optimisée. Un algorithme est proposé dans [Ark 84] pour faire l'évaluation optimisée du calcul.

2.4.2. La puissance de l'algèbre des descriptions

L'algèbre de descriptions $<A,||,+,->$ que l'on vient de définir a un intérêt multiple. Ainsi :

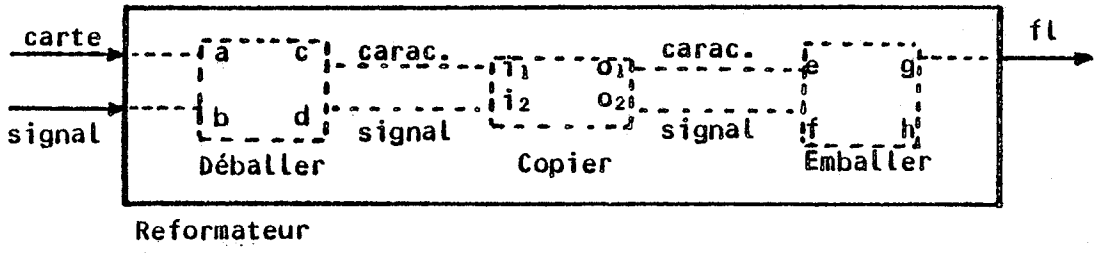
1. cette algèbre ramène l'étude d'un SPC à l'étude d'une description. Cela est fondamental, car étudier et raisonner sur les SPC est équivalent à étudier et raisonner sur toutes les descriptions que l'on peut écrire.

2. la conception d'un SPC est simple et claire : on écrit la description des éléments (processus) simples et, en utilisant les opérations, nous obtenons des systèmes complexes, sans avoir besoin d'écrire leur description.

Par exemple :

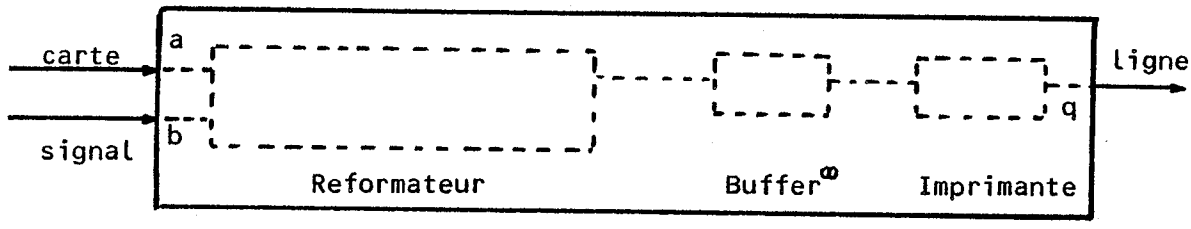
$$\text{Déballer}||\text{Copier}||\text{Emballer} + c.i_1+d.i_2+o_1.e+o_2.f - c.\delta-d.\delta-\delta.i_1-\delta.i_2-o_1.\delta-o_2.\delta - \delta.e-\delta.f-c.i_1-d.i_2-o_1.e-o_2.f-h.\delta$$

} Reformateur



décrit le problème du reformateur : passer des fichiers de cartes de 80 caractères chacune, à des fichiers de lignes de 125 caractères chacune. Chaque carte est suivie d'un espace (b) et la dernière ligne est complétée par des espaces, s'il le faut.

Nous pouvons voir aussi, dans cet exemple, que le niveau graphique est une grande aide à la conception d'un SPC. Le dessin ci-dessous suffit pour comprendre le comportement de la boîte englobante.



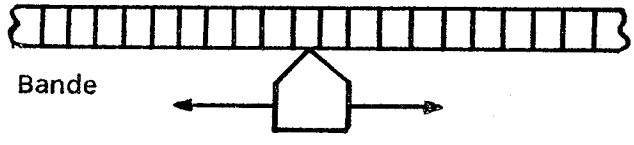
Une thèse réalisée par E. Arkaxhiu [Ark 84] est en cours sur la conception de processus au moyen d'expressions graphiques.

3. Dans [Jor 84], il est démontré qu'avec les bases de cette algèbre, nous pouvons décrire les opérateurs du calcul de R. Milner [Mil 80], C.A.R. Hoare - S.D. Brookes - A.W. Roscoe [Hoa-Bro-Ros 81], de D. Austry - G. Boudol [Aus-Bou 82] et de W.C. Rounds - S.D. Brookes [Rou-Bro 81].

4. Enfin, nous pouvons exprimer toutes les fonctions calculables, puisqu'on peut exprimer n'importe quelle machine de Turing. En effet, une machine de Turing M est formée par :

- une bande infinie (ou mémoire) divisée en cellules, contenant chacune un symbole. Toutes ces cellules, sauf un nombre fini, contiennent initialement un symbole distingué appelé "blanc" (b). L'ensemble des symboles qui peuvent apparaître dans la bande est dénoté par Z.

- une unité de contrôle à nombre fini d'états (Q) reliée à la bande par une tête de lecture-écriture. Cette tête est située, à tout instant, sur une cellule et elle a trois fonctions à réaliser dans chaque opération de l'unité de contrôle : lire (l) le symbole de la cellule désignée par la tête, écrire (e) un symbole dans la cellule désignée par la tête et gérer les déplacements de la tête, soit sur la cellule adjacente à droite (d), soit sur la cellule adjacente à gauche (g), soit rester (-) en face de la même cellule.



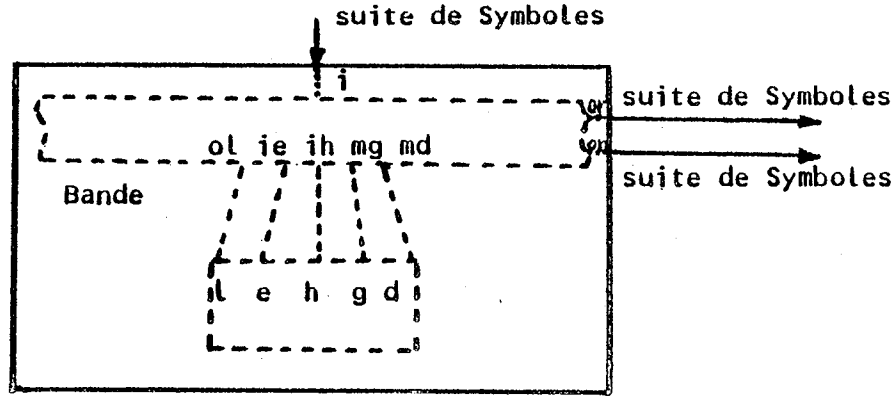
On exprime, d'habitude, le comportement d'une telle machine au moyen d'ensemble fini δ de quadruplets, d'une des trois formes suivantes :

	état courant	lire (l)	écrire (e)	gérer (d,g,-)	état résultat
1.	q_i	s_j	s_k	d	q_n
2.	q_i	s_j	s_k	g	q_n
3.	q_i	s_j	s_k	-	q_n

L'interprétation de telles règles est la suivante : la machine est dans l'état q_i, lit le symbole s_j dans la cellule désignée par la tête, écrit dans cette cellule le symbole s_k et déplace la tête à droite (d), ou à gauche (g), ou reste dans la même cellule. La machine passe donc dans l'état q_n.

Pour décrire le calcul effectué par une machine de Turing, il faut donner l'état initial de la machine, la suite de symboles à placer initialement sur les cellules qui forment la bande (on donne ici seulement une quantité finie de symboles différents du "blanc" (b)) et la cellule initialement désignée par la tête (qui est habituellement la première cellule à gauche contenant un symbole non blanc). Enfin, le résultat d'un tel calcul est constitué par le contenu de la bande quand la machine de Turing s'arrête, c-à-d la machine atteint un état dans lequel aucune règle n'est applicable. Nous réservons le symbole H pour désigner cet état d'arrêt.

Pour décrire une machine de Turing M dans notre langage, on propose la construction du SPC suivant :



Machine de Turing M

figure 1.1

où le processus "unité de contrôle" est défini par :

Unité de Contrôle M =

processus

connexions $\delta.l, e.\delta$: Symbole

$h.\delta, g.\delta, d.\delta$: Signal

opérations

$x : \longrightarrow$ Symbole pour chaque $x \in Z$

$zz : \longrightarrow$ Signal

états $q_o, \ldots, q_m, H : \lambda$ où $\{q_o, \ldots, q_m, H\} = Q$

initial q_o

règles

$$q_i \xrightarrow{\{\delta.l(s_j), e.\delta(s_k), d.\delta(zz)\}} q_n$$

pour chaque quadruplet $q_i\ s_j\ s_k\ d\ q_n$ dans δ

$$q_i \xrightarrow{\{\delta.l(s_j), e.\delta(s_k), g.\delta(zz)\}} q_n$$

pour chaque quadruplet $q_i\ s_j\ s_k\ g\ q_n$ dans δ

$$q_i \xrightarrow{\{\delta.l(s_j), e.\delta(s_k)\}} q_n$$

pour chaque quadruplet $q_i\ s_j\ s_k\ -\ q_n$ dans δ

$$H \xrightarrow{\{h.\delta(zz)\}} q_o$$

La dernière règle sert à indiquer que le calcul est fini et que le résultat est contenu dans la bande. Elle laisse aussi la machine prête pour un autre calcul.

Le processus "Bande" est le même pour n'importe quelle Unité de Contrôle, et il est défini comme suit :

(on suppose que la tête désigne toujours le premier symbole du deuxième paramètre des symboles d'état A)

Bande =

processus

connexions δ.i,on.δ,or.δ : suite de Symboles
ol.δ,δ.ie : Symbole
δ.ih,δ.mg,δ.md : Signal

opérations cons : suite de Symboles, Symbole → suite de Symboles
Λ : → suite de Symboles
zz : → Signal

états b : → Symbole
I : λ
A : suite de Symboles, suite de Symboles

initial I

règles

I —{δ.i(x)}→ A(Λ,x) } entrée initiale de symboles

A(y,cons(x,b)) —{ol.δ(b),δ.ie(c)}→ A(y,cons(x,c)) } la tête reste sur la même cellule
A(y,Λ) —{ol.δ(b),δ.ie(c)}→ A(y,cons(Λ,c))

A(y,cons(x,b)) —{ol.δ(b),δ.ie(c),δ.md(zz)}→ A(cons(y,c),x) } la tête se déplace à droite
A(y,Λ) —{ol.δ(b̄),δ.ie(c),δ.md(zz)}→ A(cons(y,c),Λ)

A(cons(y,a),cons(x,b)) —{ol.δ(b),δ.ie(c),δ.mg(zz)}→ A(y,cons(cons(x,c),a)) } la tête se déplace à gauche
A(Λ, cons(x,b)) —{ol.δ(b),δ.ie(c),δ.mg(zz)}→ A(Λ,cons(cons(x,c),b̄))
A(Λ,Λ) —{ol.δ(b̄),δ.ie(c),δ.mg(zz)}→ A(Λ,cons(cons(Λ,c),b̄))
A(cons(y,a),Λ) —{ol.δ(b̄),δ.ie(c),δ.mg(zz)}→ A(y,cons(cons(Λ,c),a))

A(y,x) —{δ.ih(zz),or.δ(y),on.δ(x)}→ I } sortie

Ainsi, le SPC de la figure 1.1. donne, comme résultat, le processus Machine de Turing pour une unité de contrôle donné. Autrement dit, l'expression :

Bande||Unité de Contrôle M + ol.l+e.ie+h.ih+g.mg+d.md
- δ.l -δ.ie-δ.ih-δ.mg-δ.md
- ol.δ-e.δ -h.δ -g.δ -d.δ
- ol.l-e.ie-h.ih-g.mg-d.md

dénote la machine de Turing M.

Un exemple de cette construction est le suivant :

Soit la machine de Turing M1 donnée par les quadruplets

q_0	0	ƀ	d	q_0		q_1	0	ƀ	d	q_1
q_0	1	ƀ	d	q_1		q_1	1	ƀ	d	q_0
q_0	ƀ	0	-	H		q_1	ƀ	1	-	H

Cette machine de Turing est supposée recevoir, en entrée, une suite de symboles construits sur $\{0,1\}^*$. Lorsque la machine atteint l'état d'arrêt H, elle a laissé sur la bande un "0" si le nombre de symboles "1", en entrée, était pair, et un "1" dans le cas contraire.

Le processus "unité de contrôle" pour cette machine de Turing M1 est le suivant :

Unité de Contrôle M1 =

processus

connexions δ.l,e.δ : Symbole
h.δ,g.δ,d.δ : Signal

opérations

0,1,ƀ: → Symbole
zz : → Signal

états q_0, q_1, H : λ

initial q_0

règles

$q_0 \xrightarrow{\{\delta.l(0), e.\delta(\not b), d.\delta(zz)\}} q_0$

$q_0 \xrightarrow{\{\delta.l(1), e.\delta(\not b), d.\delta(zz)\}} q_1$

$q_0 \xrightarrow{\{\delta.l(\not b), e.\delta(0)\}} H$

$q_1 \xrightarrow{\{\delta.l(0), e.\delta(\not b), d.\delta(zz)\}} q_1$

$q_1 \xrightarrow{\{\delta.l(1), e.\delta(\not b), d.\delta(zz)\}} q_0$

$q_1 \xrightarrow{\{\delta.l(\not b), e.\delta(1)\}} H$

$H \xrightarrow{\{h.\delta(zz)\}} q_0$

et le processus Machine de Turing M1 résultat de l'expression

Bande||Unité de Contrôle M1 + ol.l+e.ie+h.ih+g.mg+d.md
- δ.l -δ.ie-δ.ih-δ.mg-δ.md
- ol.δ-e.δ -h.δ -g.δ -d.δ
- ol.l-e.ie-h.ih-g.mg-d.md

est le suivant :

Machine de Turing M1 =

processus

connexions δ.i,or.δ,on.δ : suite de Symboles

opérations

cons : suite de Symboles, Symbole → suite de Symboles

Λ : → suite de Symboles

0, 1, $\not b$: → Symbole

zz : → Signal

états Iq_0 : λ

Aq_0, Aq_1, AH : suite de Symboles, suite de symboles

initial Iq_0

règles

$Iq_0 \xrightarrow{\{\delta.i(x)\}} Aq_0(\Lambda,x)$

$Aq_0(y,cons(x,0)) \xrightarrow{\tau} Aq_0(cons(y,b),x)$

$Aq_0(y,cons(x,1)) \xrightarrow{\tau} Aq_1(cons(y,b),x)$

$Aq_0(y,\Lambda) \xrightarrow{\tau} AH(y,cons(\ ,0))$

$Aq_1(y,cons(x,0)) \xrightarrow{\tau} Aq_1(cons(y,b),x)$

$Aq_1(y,cons(x,1)) \xrightarrow{\tau} Aq_0(cons(y,b),x)$

$Aq_1(y,\Lambda) \xrightarrow{\tau} AH(y,cons(\ ,1))$

$AH(y,x) \xrightarrow{\{or.\delta(y).on.\delta(x)\}} Iq_0$

Le calcul de ce processus pour l'entrée 1011011 peut être décrit de la façon suivante :

$Iq_0 \xrightarrow{\{\delta.i(\Lambda 1101101)\}} Aq_0(\Lambda,\Lambda 1101101)$

$\xrightarrow{\tau} Aq_1(\Lambda b,\Lambda 110110)$

$\xrightarrow{\tau} Aq_1(\Lambda bb,\Lambda 11011)$

$\xrightarrow{\tau} Aq_0(\Lambda bbb,\Lambda 1101)$

$\xrightarrow{\tau} Aq_1(\Lambda bbbb,\Lambda 110)$

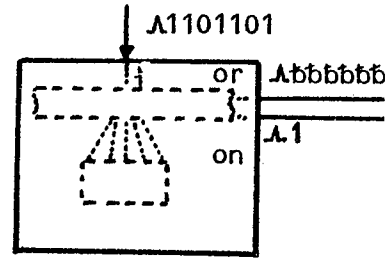
$\xrightarrow{\tau} Aq_1(\Lambda bbbbb,\Lambda 11)$

$\xrightarrow{\tau} Aq_0(\Lambda bbbbbb,\Lambda 1)$

$\xrightarrow{\tau} Aq_1(\Lambda bbbbbbb,\Lambda)$

$\xrightarrow{\tau} AH(\Lambda bbbbbbb,\Lambda 1)$

$\xrightarrow{\{or.\delta(\Lambda bbbbbbb),on.\delta(\Lambda 1)\}} Iq_0$



1.2.5. L'Analyse de Systèmes de Processus

Pour finir ce chapitre, nous allons signaler brièvement, dans cette section, quelles sont les sortes d'analyses que nous pouvons envisager avec ce langage formel de Système de Processus Communicants. Quelques travaux sont déjà en cours et les autres seront abordés dans l'avenir.

Les trois sens ("sémantiques") d'un même objet syntaxique que nous avons donné à notre langage formel ont un intérêt dans l'analyse et la vérification des systèmes de processus communicants. Ainsi,

- la sémantique algébrique permet de considérer le langage formel de descriptions comme un langage de spécification de processus. Donc, nous pouvons envisager une approche algébrique de la même sorte que dans les types abstraits de données. Ainsi, par exemple, dans ce domaine, on considère la spécification comme une description mathématique d'un concept implémenté par un programme. Donc, prouver la correction d'un programme revient à prouver l'équivalence entre la spécification et le programme [Lis-Zil 77], [Ber-Gog 80].

En outre, une étude sur la théorie de catégories appliquée aux systèmes de processus communicants peut être réalisée. Cette étude permettra, peut-être, d'avoir une définition précise de l'équivalence entre les systèmes de processus, en utilisant les homomorphismes et isomorphismes entre les descriptions.

Par ailleurs, cette sémantique donne les bases nécessaires pour introduire le modèle algébrique proposé ici dans un langage qui serait une extension de LPG (Langage de Programmation Générique), [Ber 79,82a,82b]. Une thèse dans cette voie est en train d'être réalisée par M. Cisneros (voir pour les détails [Jor 84b]).

- la sémantique opérationnelle propose une autre façon d'exprimer les systèmes de transitions d'états. Cette nouvelle façon est représentée par un ensemble de règles de calcul qui décrivent un système de transitions de termes avec événement.

Cette sémantique permet de considérer le côté gauche de chaque règle comme un prédicat qui caractérise l'ensemble des états qui permettent l'application de la règle. La même sorte d'interprétation peut être réalisée pour l'événement et le côté droit de chaque règle. Nous pensons que cette interprétation permet (au moyen de transformations du graphe fini associé à la description) une approche opérationnelle, c-à-d, l'étude de propriétés telles que : invariants, trajectoires, "liveness", terminaison, "deadlock", "livelock", ... dans le style de recherches réalisées par J. Sifakis [Sif 80]. Une thèse dans cette voie est en train d'être réalisée par P. Lagnier.

- la sémantique arborescente que nous avons définie sur les descriptions, a été orientée vers l'étude des comportements infinis. Nous avons présenté ces comportements infinis comme des arbres infinis limites de certaines suites de Cauchy.

Or, un grand travail de recherche sur les processus communicants, à l'heure actuelle, est fondé sur l'étude de l'observationnalité [Mil 80], [Dar 82] des processus à comportement fini. Les objets étudiés dans ce contexte sont "arbres de comportements". Ces arbres représentent tous les comportements possibles d'un processus. Ils sont utilisés pour l'étude de l'équivalence et la congruence de processus vis-à-vis d'un milieu extérieur quelconque qui observe les communications que le processus réalise.

La sémantique arborescente que nous avons donnée est un commencement pour l'étude de cette observationnalité sur les comportements infinis. Nous pensons que les résultats obtenus par R. Milner, M. Hennessy - R. Milner et Ph. Darandeau pour les processus finis, peuvent être étendus aux comportements infinis, en utilisant les résultats de G. Comyn sur les espaces métriques récursifs et les suites de Cauchy effectives.

Une propriété intéressante à étudier sur les processus est la divergence. Autrement dit, la possibilité, pour le processus, de pouvoir calculer, de façon indéfinie, sans communiquer avec le milieu extérieur. Du point de vue de la sémantique arborescente, cela exprime qu'il y au moins un chemin infini dans l'arbre où il n'y a pas une seule communication visible par le milieu extérieur, c-à-d, le chemin ne comporte que des τ.

Cette propriété, dans le langage formel que nous avons proposé, est indécidable. Ainsi, la construction de la machine de Turing que nous avons réalisée (1.2.4.2.), a été enrichie par les portes $\delta.i$, $on.\delta$, $or.\delta$ non courantes dans les machines de Turing. Cet enrichissement nous permet de ramener l'indécidabilité de la divergence à l'indécidabilité du problème de l'arrêt ("Halting-Problem"). Ainsi, puisque le problème de l'arrêt est indécidable, nous ne pouvons pas décider quand il aura une communication par les portes $on.\delta$ et $or.\delta$, car ces communications sont réalisées dans le cas où le processus Unité de Contrôle a fini son calcul. Par conséquent, la divergence est aussi indécidable.

Néanmoins, nous pensons qu'il y a une grande famille de processus dans laquelle cette propriété est décidable. Ainsi, si nous pouvons décider l'arrêt d'une règle de transitions de termes avec l'événement τ, alors nous pouvons affirmer qu'un système de transitions de termes ne diverge pas si ses cycles de τ s'arrêtent toujours. Plus encore, si nous pouvons décider l'ensemble d'états qui font qu'une règle de transitions de termes avec τ ne termine pas, nous pouvons analyser d'où provient la divergence. Une des caractéristiques des règles de transitions de termes $t_1 \xrightarrow{t_2} t_3$ est que $var(t_3) \subseteq var(t_1)$.

Par ailleurs, dans le cas où la règle de transitions de termes $t_1 \xrightarrow{t_2} t_3$ a un événement différent de τ, il est intéressant, pour l'analyse générale du processus, de savoir si cette règle peut être appliquée d'une façon indéfinie. Or, puisque dans ce cas, on n'a pas forcément $var(t_3) \subseteq var(t_1)$, alors la question revient à savoir quel est l'ensemble des valeurs qui peut envoyer le milieu extérieur dans les communications dans t_2, pour que cette règle soit appliquée d'une façon indéfinie.

C'est précisément l'étude de ces deux questions que nous réalisons dans le reste de cette thèse. Nous remarquons notre intérêt dans l'étude de la décidabilité de la propriété non-noethérienne puisqu'elle nous permettra de répondre aux deux questions en même temps. Ces questions peuvent être résumées par :

- Est-il décidable si une règle de transitions de termes avec événement s'arrête ou non ?

- Quel est l'ensemble de valeurs pour lesquelles la règle ne termine pas ?

CHAPITRE 2

FONCTIONS NOETHERIENNES FILTRANTES INFERIEUREMENT SUR CPO'S FILTRES COMPLETS

CHAPITRE 2

PRESENTATION

L'objectif de ce chapitre est de définir ce qu'est une fonction noethérienne et étudier sa caractérisation dans le cas où la fonction vérifie certaines propriétés.

Ces propriétés sont exprimées par les termes "filtrante inférieurement sur un cpo filtré complet". L'étude est orientée uniquement sur les éléments finitaires sur lesquels la fonction est définie. L'intérêt d'une telle étude est fondée (comme nous le verrons dans le chapitre 3) dans le fait que toute règle $t_1 \xrightarrow{t_2} t_3$ peut être vu comme une fonction qui vérifie ces propriétés. Les propositions données dans ce chapitre vont permettre de connaître "les conditions" qu'il faut vérifier pour déterminer si la fonction associée à la règle $t_1 \xrightarrow{t_2} t_3$ vérifie la propriété d'être "noethérienne" ou non.

2.1. COMPOSITION ITEREE

Intuitivement, une paire ordonnée sont deux objets donnés dans un ordre fixe. Ces paires ordonnées sont dénotées par $\langle x,y \rangle$ où x est le premier élément de la paire et y le deuxième.

Une relation binaire R est alors définie comme un ensemble de paires ordonnées.

Une fonction f est une relation binaire telle que
$\langle x,y \rangle \in f \wedge \langle x,z \rangle \in f \Rightarrow y = z$.

On dit que la fonction f est définie par l'objet x, et on écrira $f(x)$ est définie s'il existe un objet y tel que $\langle x,y \rangle \in f$. On sait, par la définition d'une fonction, que cet objet est unique, s'il existe ; il sera dénoté par $f(x)$.

Le domaine d'une fonction f est défini par :
$\text{dom } f = \{x \mid f(x) \text{ est définie}\}$ et le codomaine par :
$\text{codom } f = \{y \mid \exists x \in \text{dom } f.\ f(x) = y\}$.

La fonction restriction d'une fonction f à un ensemble A est définie par :

$$f|_A = \{\langle x,f(x) \rangle \ / \ x \in A \wedge f(x) \text{ est définie}\}$$

La fonction composition d'une fonction g avec une fonction f est définie par :

$$f \circ g = \{\langle x,f(g(x)) \rangle \ / \ g(x) \text{ est définie} \wedge f(g(x)) \text{ est définie}\}$$

On peut voir qu'il est possible dans certains cas de composer une fonction f avec elle-même un certain nombre successif de fois. Cette composition itérée de f avec elle-même est définie inductivement de la façon suivante :

1. (base) $f^0(x) = x$ ssi $x \in \text{dom } f$
2. (pas d'induction) $f^{n+1}(x) = f(f^n(x))$ ssi $f^n(x)$ est définie et $f(f^n(x))$ est définie

La fonction f^n obtenue par la seule application de 1 et 2 est appelée la puissance n-ième de f, et elle en fait la fonction $f \circ \overset{n \text{ fois}}{\ldots\ldots} \circ f$.

Parmi les multiples propriétés que l'on peut déduire sur les compositions itérées, on en retient deux qui seront utilisées par la suite.

Lemme 2.1.

Soit f une fonction telle que $\forall n.f^n(x)$ est définie et $f(f^n(x))$ est définie :
$\forall n.\text{dom } f^{n+1} \subseteq \text{dom } f^n \wedge \text{codom } f^{n+1} \subseteq \text{codom } f^n$

Preuve :
■

Par l'absurde, on prouve $\forall n.\text{dom } f^{n+1} \subseteq \text{dom } f^n$.
Supposons que $\exists n$ t.q. $\text{dom } f^{n+1} \not\subseteq \text{dom } f^n$,
alors, pour ce n il existe $x \in \text{dom } f^{n+1}$ t.q. $x \notin \text{dom } f^n$.
Néanmoins, $x \in \text{dom } f^{n+1} \Rightarrow f^n(x)$ est définie et $f(f^n(x))$ est définie,
mais $f^n(x)$ est définie $\Rightarrow x \in \text{dom } f^n (\Rightarrow \Leftarrow)$

De la même manière, on peut prouver $\forall n.\text{codom } f^{n+1} \subseteq \text{codom } f^n$

■ c.q.f.d.

Lemme 2.2.

Soit f une fonction,
$\forall n>1.\text{dom } f^n = \{x \mid f(x) \in \text{dom } f^{n-1}\}$

Preuve :
■

Par induction sur n
base n = 2

$x \in \text{dom } f^2 \Leftrightarrow f^2(x)$ est défini $\Leftrightarrow f(x)$ est défini $\wedge f(f(x))$ est défini $\Leftrightarrow f(x) \in \text{dom } f$
donc $\text{dom } f^2 = \{x \mid f(x) \in \text{dom } f\}$

induction : supposons que, pour un $n>2$ donné,

$$\text{dom } f^n = \{x \mid f(x) \in \text{dom } f^{n-1}\}$$

par définition de domaine et de composition itérée,

$$\text{dom } f^{n+1} = \{x \mid f^{n+1}(x) \text{ est défini}\} =$$
$$= \{x \mid f(f^n(x) \text{ est défini et } f^n(x) \text{ est défini}\}$$

mais, par associativité de la composition et définition de domaine

$= \{x \mid f^n(f(x) \text{ est défini} \wedge x \in \text{dom } f^n\} = \{x \mid f(x) \in \text{dom } f^n \wedge x \in \text{dom } f$

donc, par hypothèse inductive sur $x \in \text{dom } f^n$, on a

$= \{x \mid f(x) \in \text{dom } f^n \wedge f(x) \in \text{dom } f^{n-1}\}$

et par le lemme 2.1., on obtient

$= \{x \mid f(x) \in \text{dom } f^n\}$ c.q.f.d.

■

2.2. FONCTIONS NOETHERIENNES

Soit $\mathcal{F}$ la classe de toutes les fonctions, et ω le premier ordinal infini.

On dit que $f \in \mathcal{F}$ est une <u>fonction noethérienne</u> ssi

$$\forall x \in \text{dom } f. \exists n<\omega. f^n(x) \text{ n'est pas définie}$$

c-à-d, il n'existe pas un $x \in \text{dom } f$ t.q. $x, f(x), f^2(x), \ldots$ soit une suite infinie ou stationnaire.

De même, $f \in \mathcal{F}$ est une <u>fonction non-noethérienne</u> ssi

$$\exists x \in \text{dom } f. \forall n<\omega. f^n(x) \text{ est définie}$$

On dira, dans ce cas, que f <u>ne termine pas</u> en x

On peut voir, par cette définition, que la fonction Ω telle que dom Ω = codom Ω = $\emptyset$ est une fonction noethérienne. La fonction Ω reçoit le nom de fonction vide.

On est intéressé par l'étude de cette propriété dans un sous-ensemble $\mathcal{U}$ (les fonctions filtrant inférieurement dans la base finitaire d'un cpo filtré complet) de $\mathcal{F}$ que l'on verra ci-après. Néanmoins, on peut dès maintenant établir une condition suffisante de la propriété "non-noethérienne" qui se vérifie pour n'importe quelle fonction.

Lemme 2.3.

Soit $f \in \mathcal{F}$

codom $f \subseteq$ dom $f \neq \emptyset \Rightarrow f$ non-noethérienne

Preuve :

■

Prouver que f est non-noethérienne est montrer qu'il existe un $x \in$ dom f t.q. $\forall n$ $f^n(x)$ soit définie.
Puisque dom $f \neq \emptyset$, il existe au moins un x dans dom f.

On va prouver, par induction, que $\forall n$ $f^n(x)$ est définie
base $n = 0$ $f^0(x) = x$ par définition de f^0
induction supposons que $f^n(x)$ est définie

donc, par hypothèse, $f^n(x) \in$ codom $f \Rightarrow f^n(x) \in$ dom f
alors $f(f^n(x))$ est définie ①
donc, par ① et hypothèse inductive, on a $f^{n+1}(x)$ est définie

c.q.f.d.

■

2.3. ORDRE SUR L'ENSEMBLE $\mathcal{P}(f)$

La définition de fonction noethérienne que l'on vient de donner conduit à étudier l'ensemble des compositions itérées d'une fonction $f \in \mathcal{F}$. Dans ce paragraphe, on définit une relation qui permet de bien ordonner cet ensemble.

Rappel sur les ordres

Avant d'entrer dans ce propos, on va rappeler certaines définitions sur les ordres :

Les paires $\langle x,y \rangle$ d'une relation binaire R sont dénotées par xRy

Une relation binaire R dans l'ensemble D est une relation binaire R telle que $xRy \Rightarrow x,y \in D$

La structure $(D,\sqsubseteq)$ est dite "partiellement ordonnée" ssi D est un ensemble non-vide et $\sqsubseteq$ est une relation binaire sur l'ensemble D, qui satisfait les conditions suivantes :

i) réflexive : $x \sqsubseteq x$ pour chaque $x,y \in D$
ii) antisymétrique : $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$ pour chaque $x,y \in D$
iii) transitive : $x \sqsubseteq y \quad y \sqsubseteq z \Rightarrow x \sqsubseteq z$ pour chaque $x,y,z \in$

Un élément $x \in D$ est appelé minimum de D ssi $\forall y \in D.x \sqsubseteq y$.
On sait que, par l'antisymétrie de la relation , le minimum, s'il existe, est unique. Une structure partiellement ordonnée $(D,\sqsubseteq)$ est dite bien-ordonnée ssi, pour chaque $A \subseteq D \wedge A \neq \emptyset$, le minimum de A existe.

La relation d'ordre sur $\mathcal{P}(f)$

Soit $f \in \mathcal{F}$. On peut définir $\mathcal{P}(f)$, l'ensemble de puissances finie de f inductivement par :

1. (base) $f \in \mathcal{P}(f)$
2. (pas d'induction) $g \in \mathcal{P}(f) \Rightarrow g \circ f \in \mathcal{P}(f)$
3. $\mathcal{P}(f)$ est le plus petit ensemble vérifiant 1. et 2.

On définit la relation binaire $\leqslant_f$ dans $\mathcal{P}(f)$ de la façon suivante :

$g_1 \leqslant_f g_2$ ssi $g_1 = f^{n_1} \wedge g_2 = f^{n_2} \wedge n_1 \leqslant n_2$

Le symbole $\leqslant_f$ peut être lu comme "est une puissance de f plus petite ou égale".

Lemme 2.4.

Soit $f \in \mathcal{F}$

La structure $(\mathcal{P}(f), \leqslant_f)$ est partiellement ordonnée

Preuve :

■

$\leqslant_f$ réflexive $(g \leqslant_f g)$: trivial

$\leqslant_f$ antisymétrique $(g_1 \leqslant_f g_2 \wedge g_2 \leqslant_f g_1 \Rightarrow g_1 = g_2)$

puisque $g_1, g_2 \in \mathcal{P}(f) \Rightarrow \exists n_1, n_2$ t.q. $g_1 = f^{n_1} \wedge g_2 = f^{n_2}$

on a

$g_1 \leqslant_f g_2 \Rightarrow n_1 \leqslant n_2$

$g_2 \leqslant_f g_1 \Rightarrow n_2 \leqslant n_1$

} alors $n_1 = n_2$ par conséquent $g_1 = f^{n_1} = f^{n_2} = g_2$

$\leqslant_f$ transitive $(g_1 \leqslant_f g_2 \wedge g_2 \leqslant_f g_3 \Rightarrow g_1 \leqslant_f g_3)$

puisque $g_1, g_2, g_3 \in \mathcal{P}(f) \Rightarrow \exists n_1, n_2, n_3$ t.q. $g_i = f^{n_i}$ $1 \leqslant i \leqslant 3$

on a

$g_1 \leqslant_f g_2 \Rightarrow n_1 \leqslant n_2$

$g_2 \leqslant_f g_3 \Rightarrow n_2 \leqslant n_3$

} alors $n_1 \leqslant n_3$ par conséquent $g_1 = f^{n_1} \leqslant_f g_3 = f^{n_3}$

■

c.q.f.d.

Proposition 2.5.

Soit $f \in \mathcal{F}$

La structure $(\mathcal{P}(f), \leqslant_f)$ est bien ordonnée

Preuve
■

Par le lemme 2.4. $(\mathcal{P}(f), \leqslant_f)$ est partiellement ordonnée.
Il reste à prouver que, pour chaque $D \subseteq \mathcal{P}(f) \wedge D \neq \emptyset$, il existe un minimum de D.
On va le faire par l'absurde :

Supposons que $D \subseteq \mathcal{P}(f) \wedge D \neq \emptyset$ t.q. Il n'existe pas un minimum de D. Si l'on prouve que $D = \emptyset$, alors on a une contradiction (=><=) et la preuve est terminée.

Pour prouver que $D = \emptyset$, on va prouver par induction que chaque élément dans D est au moins aussi grand que n'importe quelle fonction g dans $\mathcal{P}(f)$, c-à-d :

$$\forall g \forall h \, [g \in \mathcal{P}(f) \wedge h \in D \Rightarrow g \leqslant_f h]$$

Puisqu'aucune fonction dans $\mathcal{P}(f)$ n'est plus grande ou égale à chaque g dans $\mathcal{P}(f)$,
alors $D \neq \emptyset$ c.q.f.d.

La preuve par induction de $\forall g \forall h \, [g \in \mathcal{P}(f) \wedge h \in D \Rightarrow g \leqslant_f h]$ est la suivante :

base	$\forall h \, [h \in D \Rightarrow f \leqslant_f h]$ trivialement
induction	supposons que $\forall h \, [h \in D \Rightarrow g \leqslant_f h]$ est vrai pour un g arbitraire
	alors, $g \notin D$ puisqu'on a supposé que D n'a pas de minimum
	donc, $\forall h \, [h \in D \Rightarrow g \leqslant_f h \wedge g \neq_f h]$ est vrai
	alors $\forall h \, [h \in D \Rightarrow g \circ f \leqslant_f h]$ est vrai

■ c.q.f.d.

De plus, cet ensemble $\mathcal{P}(f)$ est dénombrable (fini ou énumérable) et on peut lui associer une fonction φ_f d'<u>énumération</u> ($\mathrm{dom}\, \varphi_f = \mathbb{N}$) définie comme suit :

$$\varphi^0_f = \{\langle x.x \rangle \mid x \in \mathrm{dom}\, f\}$$

$$\varphi^n_f = \varphi^{n-1}_f \circ f$$

2.4. LA CLASSE $\Pi(B,\sqsubseteq,\sqcap B)$ = FONCTIONS FILTRANTES INFERIEUREMENT SUR UNE BASE FINITAIRE

Dans cette section, on définit formellement la classe des fonctions dans laquelle on caractérise la propriété d'être noethérienne. Pour cela, on rappelle d'abord quelques faits et définitions sur les ordres partiels complets (aussi connus par cpo's ou ensembles inductifs) et on définit les cpo's filtrés complets et les fonctions filtrantes inférieurement.

2.4.1. Cpo's filtrés complets

Soit $(D,\sqsubseteq)$ un ensemble partiellement ordonné. L'ensemble des "majorants" de $A \subseteq D$ est défini par :

$$\uparrow A = \{d \in D \mid \forall a \in A,\ a \sqsubseteq d\}$$

Un élément de D, s'il existe, est appelé le "supremum" de $A \subseteq D$ ssi $\forall a \in \uparrow A . d \sqsubseteq a$; on dénote cet élément, s'il existe par $\sqcup A$. On sait, de plus, par l'antisymétrie de la relation $\sqsubseteq$, que le supremum, s'il existe, est unique. Dualement, on peut définir minorants, $\downarrow A$, infimum, $\sqcap A$.

Un sous-ensemble $\Delta \subseteq D$ est appelé "dirigé" ssi :

$$\forall d_1 . d_2 \in \Delta . \exists d_3 \in \Delta \ .\ d_1 \sqsubseteq d_3 \wedge d_2 \sqsubseteq d_3$$

c-à-d toute paire d'éléments de Δ a un majorant dans Δ. De proche en proche, on peut vérifier que cette condition est équivalente à

$$\forall A \subseteq \Delta \ ,\ A \text{ fini}, \exists y \in \Delta \cap \uparrow A$$

c-à-d toute partie finie de Δ est majorée dans Δ.

Soit $(D, \sqsubseteq, \bot)$ un ensemble partiellement ordonné avec un élément minimum $\bot$. Cet ensemble est appelé "cpo" (ordre partiel complet ou ensemble inductif) ssi

$$\forall \Delta \subseteq D,\ \Delta \text{ dirigé},\ \bigsqcup \Delta \in D$$

c-à-d tout sous-ensemble dirigé admet un supremum dans D.

Soit $(D, \sqsubseteq, \bot)$ un cpo. Un sous-ensemble $B \subseteq D$ est appelé base de ssi

$$\forall x \in D.\ \exists \Delta \subseteq B,\ \Delta \text{ dirigé t.q. } x = \bigsqcup \Delta$$

c-à-d tout les éléments de D sont supremums de dirigés dans B

Il est connu [Com 82] [Mar 76] qu'un ensemble B partiellement ordonné avec un plus petit élément peut être plongé dans un cpo $(B^{\infty}, \sqsubseteq, \bot)$ de façon telle que :

1. B est isomorphe à un sous-ensemble de B^{∞}
2. tout élément du complet B^{∞} est le supremum d'une partie dirigée de B.

La complétion par idéaux est une méthode pour faire cette complétion. Elle est résumée par les définitions et théorèmes suivants :

Un sous-ensemble $I \subseteq B$ est appelé idéal de B ssi

i. I est dirigé

ii. $x \in I \ \wedge\ y \leqslant x \Rightarrow y \in I$

Un idéal Δ^a est principal s'il a comme élément maximal a, autrement dit :

$\Delta^a = \{x \mid x \in B \ \wedge\ x \leqslant a\}$ est l'idéal principal engendré par a

L'ensemble des idéaux principaux sur B est isomorphe à B.

Théorème 2.6. [Com 82]

Soit $(B, \sqsubseteq, \perp)$ un ensemble partiellement ordonné avec élément minimu

L'ensemble des idéaux de B noté I(B) est un cpo et l'application injective $j : B \to I(B)$ qui associe à chaque élément $x \in B$ son idéal principal Δ^x est telle que :

i. $\forall x.y \in B,\ x \leqslant y$ ssi $j(x) \leqslant j(y)$ dans I(B)

ii. Pour tout $x \in I(B)$, l'ensemble $J_x = \{j(a) \mid a \in B \wedge j(a) \leqslant x\}$ est dirigé et l'on a : $x = \bigsqcup J_x$

L'ensemble B est considéré comme la base du cpo B^∞

Soit $(D, \sqsubseteq, \perp)$ un cpo. Un élément $d \in D$ est dit <u>finitaire</u> ssi
$\forall\ \Delta \subseteq D,\ \Delta$ dirigé, $d \sqsubseteq \bigsqcup \Delta \Rightarrow \exists c \in \Delta,\ d \sqsubseteq c$

Une base B est finitaire ssi $\forall b \in B$, b est finitaire.

Soit $(D, \sqsubseteq, \perp)$ un cpo avec une base finitaire B. Un élément $d \in D$ est appelé <u>purement infinitaire</u> ssi il est le supremum d'une suite infinie strictement croissante des éléments de B. On dénote par B^ω l'ensemble des éléments purement infinitaires.

Le théorème suivant établit qu'aucun élément finitaire ne peut être supremum d'une suite infinie strictement croissante d'éléments de B.

Théorème 2.7. [Com 82]

Soit $(D, \sqsubseteq, \perp)$ un cpo et B une base finitaire de D
$B^\omega = B^\infty - B$

Notre étude portera sur l'existence d'éléments finitaires qui font qu'une fonction ne termine pas. On caractérise donc la propriété noethérienne dans une sous-classe de fonctions qui ont comme domaine la base finitaire d'un cpo.

Soit $(B, \sqsubseteq)$ un ensemble partiellement ordonné dénombrable. B est un inf-demi-treillis-complet ssi $\forall A \subseteq B, A \neq \emptyset, \sqcap A \in B$. On note ces structures par $(B, \sqsubseteq, \sqcap B)$ et, puisque le plus petit élément de B est $\sqcap B$, le cpo que l'on obtient par complétion par idéaux de B sera noté $(B^{\infty}, \sqsubseteq, \sqcap B)$. Il faut signaler que cette structure $(B^{\infty}, \sqsubseteq, \sqcap B)$ est un treillis complet puisque B a élément minimum, et par conséquent, I(B) est un treillis complet (voir [Gra 78] pp. 24 cor. 15). La structure $(B^{\infty}, \sqsubseteq, \sqcap B)$ sera appelée cpo filtré complet en raison de:l'usage que nous ferons de la notion de fonction filtrante inférieurement et aussi parce que un cpo filtré (voir Comyn [82]) poseds comme base finitaire un inf-demi-treillis.

2.4.2. Fonctions Filtrantes Inférieurement sur cpo's filtrés complets

Soit B la base finitaire dénombrable d'un cpo filtré complet. Pour chaque élément fixe $b \in B$, l'ensemble $\nabla_b = \{x \in B | b \sqsubseteq x\}$ est appelé le filtré principal engendré par b

Lemme 2.8.

Soit $(B^{\infty}, \sqsubseteq, \sqcap B)$ un cpo filtré complet

Soient $\nabla_a, \nabla_{a'} \subseteq B^{\infty}$ deux filtrés principaux engendrés par a et a' respectivement

$$\nabla_a \cap \nabla_{a'} \neq \emptyset \Rightarrow \nabla_a \cap \nabla_{a'} = \nabla_{a \sqcup a'}$$

Preuve : trivial

Une fonction f est dite filtrante inférieurement sur la base finitaire B d'un cpo filtré-complet ssi :

$$\forall n \ (\text{dom } f^n \neq \emptyset \Rightarrow \text{dom } f^n = \nabla_t \wedge \text{codom } f^n = \nabla_{t'})$$

où $\nabla_t, \nabla_{t'}$ sont filtres principaux dans $(B, \sqsubseteq, \sqcap B)$

Enfin, on caractérise la propriété noethérienne pour les fonctions filtrantes inférieurement.

On définit donc la classe $\mathbb{H}(B, \sqsubseteq, \sqcap B)$ des fonctions filtrantes inférieurement comme suit :

$$\mathbb{H}(B, \sqsubseteq, \sqcap B) = \{h \in \mathcal{F} \mid h \text{ est filtrantes inférieurement sur } (B, \sqsubseteq, \sqcap B), \text{ base finitaire d'un cpo filtre complet}\}$$

On peut déduire dans peine que :

1. $\Omega \in \mathbb{H}(B, \sqsubseteq, \sqcap B)$ pour tout $(B, \sqsubseteq, \sqcap B)$
2. $h \in \mathbb{H}(B, \sqsubseteq, \sqcap B) \Rightarrow \mathcal{P}(h) \subset \mathbb{H}(B, \sqsubseteq, \sqcap B)$ pour tout $(B, \sqsubseteq, \sqcap B)$ *

et établir le lemme suivant qui sera utilisé par la suite :

Lemme 2.9.

Soit $h \in \mathbb{H}(B, \sqsubseteq, \sqcap B)$

$\forall\ h^n, h^m \in \mathcal{P}(h).\ (h^n \leqslant_h h^m \neq \Omega \Rightarrow \sqcap \mathrm{dom}\ h^n \sqsubseteq \sqcap \mathrm{dom}\ h^m)$

Preuve ■

Soit $h^n, h^m \in \mathcal{P}(h)$ t.q. $h^n \leqslant_h h^m \neq \Omega$

Puisque $h \in \mathbb{H}(B, \sqsubseteq, \sqcap B)$ par * on a $h^m, h^n \in \mathbb{H}(B, \sqsubseteq, \sqcap B)$.

De plus, puisque $h^m \neq \Omega$, on sait que $h^n \neq \Omega$ et aussi que $\mathrm{dom}\ h^m \neq \emptyset$ et $\mathrm{dom}\ h^n \neq \emptyset$. Donc, par définition des fonctions filtrantes inférieurement, on a :

$\mathrm{dom}\ h^n = \nabla_{c_1}$, et $\mathrm{dom}\ h^m = \nabla_{c_2}$ où $\nabla_{c_1}, \nabla_{c_2}$ sont deux filtres principaux dans $(B, \sqsubseteq, \sqcap)$.

De plus, c_1 = minimum de $\mathrm{dom}\ h^n = \sqcap \mathrm{dom}\ h^n$ et

c_2 = minimum de $\mathrm{dom}\ h^m = \sqcap \mathrm{dom}\ h^m$

et du lemme 2.1. $(\mathrm{dom}\ h^m \sqsubseteq \mathrm{dom}\ h^n)$, on déduit $\nabla_{c_2} \sqsubseteq \nabla_{c_1}$

donc $\sqcap \mathrm{dom}\ h^n \sqsubseteq \sqcap \mathrm{dom}\ h^m$

c.q.f.d.

■

2.5. LES FONCTIONS NON-NOETHERIENNES DANS $\mathcal{H}(B, \sqsubseteq, \sqcap B)$

On montre ici quatre propositions qui caractérisent la propriété des fonctions non-noethériennes dans une classe $\mathcal{H}(B, \sqsubseteq, \sqcap B)$ quelconque.

Soit $h \in \mathcal{H}(B, \sqsubseteq, \sqcap B)$

On définit l'ensemble $\mathcal{C}^{\bullet}h = \{x \in \text{dom } h \mid \forall n. h^n(x) \text{ est définie}\}$; c-à-d, l'ensemble $\mathcal{C}^{\bullet}h$ est composé des éléments du domaine de h qui sont tels que l'application itérée de cette fonction ne termine pas.

On peut alors établir les faits suivants :

Lemme 2.10

Soit $h \in \mathcal{H}(B, \sqsubseteq, \sqcap B)$
h non-noethérienne $\Rightarrow \exists\, n < \omega.\ \sqcap \text{dom } h^n = \sqcap \mathcal{C}^{\bullet}h$

Preuve
■

Puisque h est non-noethérienne $\mathcal{C}^{\bullet}h \neq \emptyset$
Puisque h est définie sur un inf-demi-treilli-complet $\sqcap \mathcal{C}^{\bullet}h \in B$
Par ailleurs, la suite $\varphi_h^0 \leqslant_h \varphi_h^1 \leqslant_h \ldots$ peut être stationnaire ($\mathcal{P}(h)$ fini) ou non stationnaire ($\mathcal{P}(h)$ énumérable).

Donc, par le lemme 2.9., $\sqcap \text{dom } \varphi_h^0 \sqsubseteq \sqcap \text{dom } \varphi_h^1 \sqsubseteq \ldots \sqsubseteq \sqcap \text{dom } \varphi_h^n \sqsubseteq \ldots$ peut être aussi stationnaire ou non.

Mais, cette dernière suite est majorée par un élément de B, puisque

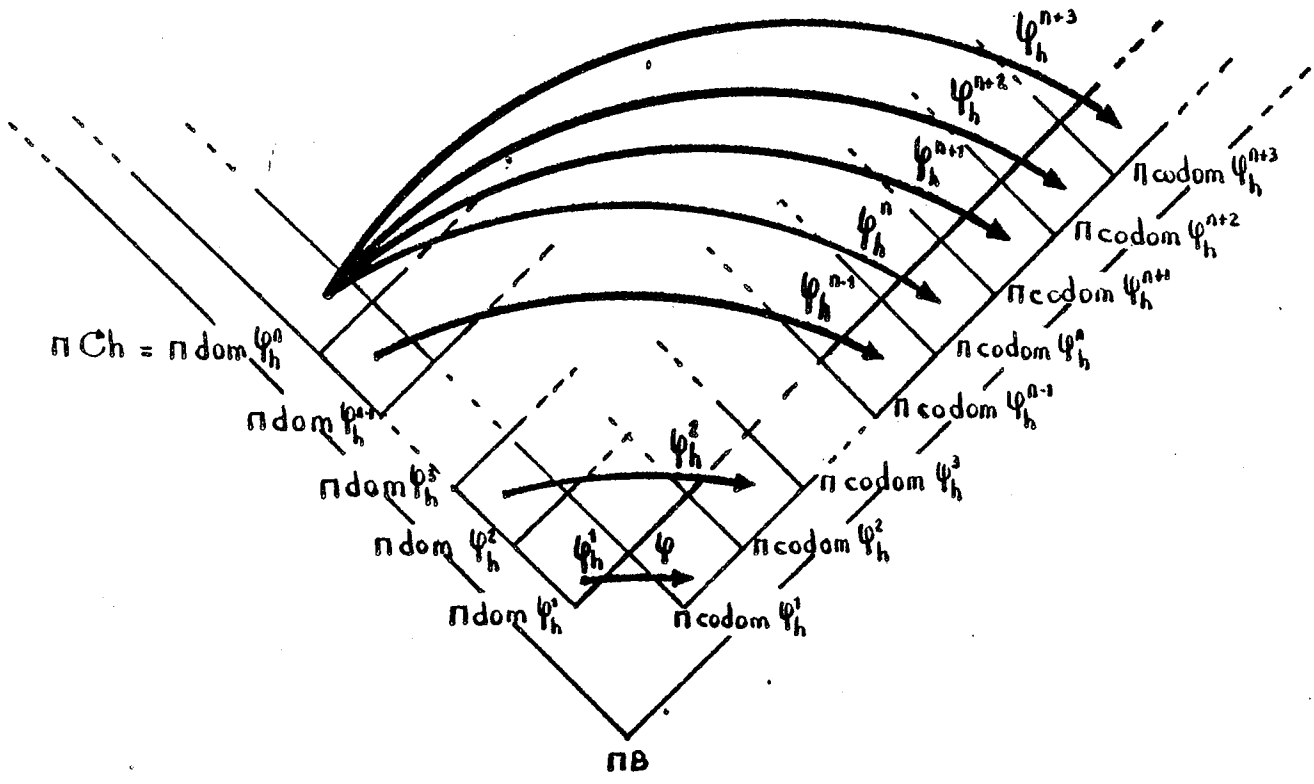
1. $\forall i \quad \mathcal{C}^{\bullet}h \subseteq \text{dom } h^i$, donc $\text{dom } h^i \neq \emptyset$
2. puisque h est filtrante inférieurement, $\text{dom } h^i = \nabla_b$, $b = \sqcap \text{dom } h^i \in B$, donc $\forall i \quad \sqcap \text{dom } h^i \sqsubseteq \sqcap \mathcal{C}^{\bullet}h \in B$

et puisque aucun élément de B ne peut être supremum d'une suite infinie strictement croissante d'éléments de B par le théorème 2.7. alors

$\sqcap \text{dom } \varphi_h^0 \sqsubset \sqcap \text{dom } \varphi_h^1 \sqsubset \ldots \sqsubset \sqcap \text{dom } \varphi_n^m \sqsubset \ldots \sqsubseteq \mathcal{C}^{\bullet}h$ est finie
donc, il existe forcément un $n < \omega$ t.q. $\sqcap \text{dom } \varphi_h^n = \mathcal{C}^{\bullet}h$

■ c.q.f.d

Il faut retenir de cette preuve, qu'en général, (h noethérienne ou non), la suite $\varphi_h^0 \leqslant_h \varphi_h^1 \leqslant_h \ldots$ peut être non-stationnaire. Mais, par contre, dans le cas où h est non-noethérienne, la suite $\Pi\mathrm{dom}\,\varphi_h^0 \subset \Pi\mathrm{dom}\,\varphi_h^1 \subset \ldots \subset \Pi\mathrm{dom}\,\varphi_h^n = \Pi\,\mathcal{C}h$ avec $\mathcal{C}h \neq \emptyset$ est toujours stationnaire. Ce fait peut être illustré par le dessin suivant (avec h non-noethérienne).



La première proposition de caractérisation de la propriété "non-noethérienne" peut être formulée dans les termes suivants :

Proposition 2.11.

Soit $h \in \mathcal{H}(B, \sqsubseteq, \sqcap)$

h non-noethérienne $\iff \exists\, n < \omega . \mathrm{dom}\, h^n = \mathrm{dom}\, h^{n+1} \neq \emptyset$

Preuve :

■

$\Rightarrow$) Par le lemme 2.10. et puisque h est filtrante inférieurement, on sait que h non-noeth $\Rightarrow \exists\, n < \omega . \mathrm{dom}\, h^n = \nabla_{\sqcap \mathbb{C}h}$

donc $\mathrm{dom}\, h^n \neq \emptyset$ puisque $\sqcap \mathbb{C}h \in \mathrm{dom}\, h^n$.

De plus, h non-noeth $\Rightarrow \emptyset \neq \mathbb{C}h \subseteq \mathrm{dom}\, h^{n+1}$ et par le lemme 2.1. $\emptyset \neq \mathbb{C}h \subseteq \mathrm{dom}\, h^{n+1} \subseteq \mathrm{dom}\, h^n$ ①

Maintenant, raisonnons par l'absurde :

Supposons que $\mathrm{dom}\, h^{n+1} \neq \mathrm{dom}\, h^n$. En conséquence de ①

$\mathrm{dom}\, h^{n+1} \subseteq \{x \in B \mid \sqcap \mathbb{C}h \sqsubset x\}$

Mais, puisque h est filtrante inférieurement et $\mathrm{dom}\, h^{n+1} \neq \emptyset$, on a : $\mathrm{dom}\, h^{n+1} = \nabla_{\sqcap \mathrm{dom}\, h^{n+1}}$

et ceci est contradictoire avec le fait que l'infimum de $\mathbb{C}h$ soit $\sqcap \mathbb{C}h$ (=><=)

$\Rightarrow$) $\exists n < \omega . \mathrm{dom}\, h^n = \mathrm{dom}\, h^{n+1} \neq \emptyset$

donc, par le lemme 2.2. $\mathrm{dom}\, h^{n+1} = \{x \mid h(x) \in \mathrm{dom}\, h^n\} = \mathrm{dom}\, h^n \neq \emptyset$

si l'on prouve que $\forall m . \mathrm{dom}\, h^n = \mathrm{dom}\, h^{n+m} \neq \emptyset$

alors, puisque $\forall x \in \mathrm{dom}\, h^n . h^{n+m}(x) \in \mathrm{dom}\, h$

pour chaque m et, puisque ce x existe parce que $\mathrm{dom}\, h^n \neq \emptyset$, alors h est non-noethérienne.

La preuve par induction de $\forall m$. $\mathrm{dom}\, h^n = \mathrm{dom}\, h^{n+m} \neq \emptyset$ est la suivante :

base $m = 0$ trivial

induction supposons que pour un m donné

$\mathrm{dom}\, h^n = \mathrm{dom}\, h^{n+m} = \emptyset$

donc, par le lemme 2.2. et par l'hypothèse inductive,

$\mathrm{dom}\, h^{n+m+1} = \{x \mid h(x) \in \mathrm{dom}\, h^{n+m}\} = \{x \mid h(x) \in \mathrm{dom}\, h^n\}$

alors par ②

Corollaire 2.12

Soit $h \in \Pi(B, \leqq, \Pi B)$

h non-noethérienne $\Rightarrow \exists\, n < \omega.\ \mathrm{dom}\ h^n = \mathbb{C}^{\bullet} h$

autrement dit h non-noethérienne implique que $\mathbb{C}^{\bullet} h$ est un filtre principal

Preuve :

■

Par la proposition 2.11. h non-noeth $\Rightarrow \exists\, k < \omega.\ \mathrm{dom}\ h^k = \mathrm{dom}\ h^{k+1} \neq \emptyset$

On a prouvé aussi dans la preuve $\Rightarrow$) de la proposition 2.11. que

$$\forall m.\ \mathrm{dom}\ h^k = \mathrm{dom}\ h^{k+m} \neq \emptyset$$

Mais, puisque $\forall m.\ \mathbb{C}^{\bullet} h \subseteq \mathrm{dom}\ h^{k+m}$ et h est filtrante inférieurement, on a

$$\forall m.\ \Pi\, \mathbb{C}^{\bullet} h \in \mathrm{dom}\ h^{k+m}$$

alors h^k ne termine pas avec $\Pi\, \mathbb{C}^{\bullet} h$, et par conséquence, h aussi, c-à-d

$$\Pi\, \mathbb{C}^{\bullet} h \in \mathbb{C}^{\bullet} h \qquad (1)$$

De plus, par le lemme 2.10., h non-noeth $\Rightarrow \exists\, n < \omega.\ \mathrm{dom}\ h^n = \nabla_{\Pi\, \mathbb{C}^{\bullet} h}$

donc, par (1) .

$$h \text{ non noeth} \Rightarrow \exists\, n < \omega.\ \mathrm{dom}\ h^n = \nabla_{\Pi\, \mathbb{C}^{\bullet} h} = \mathbb{C}^{\bullet} h$$

■

La proposition 2.11. fait la caractérisation de la propriété "non-noéthérienne" dans Π, en considérant uniquement les domaines des puissances successives de la fonction à caractériser. On va donner une autre caractérisation qui se déduit de cette proposition et du lemme 2.3., et qui prend en compte les domaines et les codomaines de puissances successives de la fonction.

Proposition 2.13.

Soit $h \in \Pi(B, \leqq, \Pi B)$

h non noethérienne $\Leftrightarrow \exists\, n < \omega.$ [(codom h^n et dom h^n sont filtres principaux)

$\wedge$ codom $h^n \subseteq$ dom h^n]

Preuve :

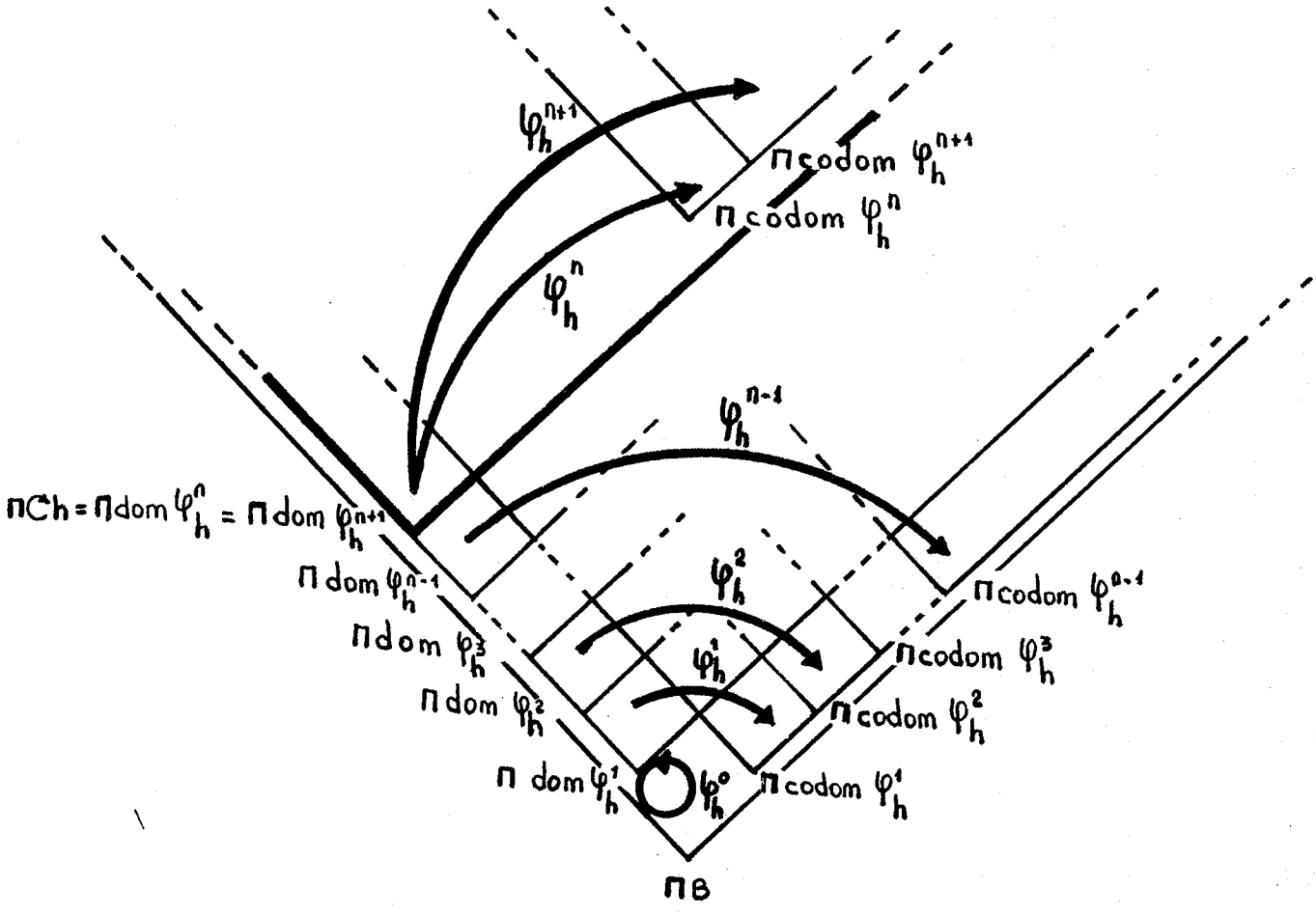
■

=>) par le corollaire 2.12. $\exists\ n < \omega . \mathrm{dom}\ h^n = \complement h$
c-à-d h^n ne termine pas pour chaque $x \in \mathrm{dom}\ h^n$
alors, forcément, codom h^n est formé par des éléments qui font aussi que la fonction ne termine pas, donc codom $h^n \subseteq$ dom h^n.
De plus, puisque dom $h^n = \nabla_{\Pi \complement h}$ on a dom $h^n \neq \emptyset$,
par conséquent, codom $h^n \neq \emptyset$ et puisque h est filtrante inférieurement, codom h^n est un filtre principal

<=) comme une conséquence du lemme 2.3., puisque h^n vérifie la condition.

■ c.q.f.d.

Le dessin suivant donne une vision graphique des propositions 2.11. et 2.13.



Les deux autres propositions suivantes de caractérisation de la propriété "non-noethérienne" rapportent cette caractérisation à certaines fonctions qui interviennent dans le calcul d'une fonction itérée h^n.

Ainsi, la composition itérée d'une fonction f quelconque définit, pour chaque n, une fonction f^n. Cette fonction f^n peut être exprimée comme la composition de n fonctions :

$$f^n = f_{n.n} \circ \ldots \circ f_{n.1}$$

où les $f_{n.i}$, $1 \leqslant i \leqslant n$, sont fonctions restrictions de f à certains sous-ensembles du domaine de h. Ainsi, par exemple, $f_{n.1} = f|_{\text{dom } f^n}$.

Les prochines propositions 2.14. et 2.15. sont les homologues des propositions 2.12. et 2.13., mais établissant la caractérisation de la propriété "non-noethérienne" uniquement à travers des fonctions $h_{n.1}$ des puissances successives de h. Ainsi,

Proposition 2.14.

Soit $h \in \Pi(B, \leqq, \Omega)$ et $h^n = h_{n.n} \circ \ldots \circ h_{n.1}$ pour chaque puissance n de h

h non-noethérienne $\iff \exists n < \omega . h_{n.1} \neq h_{n+1.1} \neq \Omega$

Preuve :

■

$\Rightarrow$) par la proposition 2.12, h non-noeth $\Rightarrow \exists$ n < ω.dom h^n = dom h^{n+1} = ∅

donc

$$\forall x \in \text{dom } h_{n.1} = \text{dom } h^n = \text{dom } h^{n+1} = \text{dom } h_{n+1.1} = \emptyset$$

alors

$$h_{n.1} \neq \Omega \wedge h_{n+1.1} \neq \Omega$$

De plus, puisque $h_{n.1} = h|_{\text{dom } h^n} \wedge h_{n+1.1} = h|_{\text{dom } h^{n+1}} \wedge$ dom h^n = dom h^{n+1}

on a $h_{n.1} = h_{n+1.1}$

$\Leftarrow) \exists n < \omega . h_{n.1} = h_{n+1,n} \neq \Omega \Rightarrow \text{dom } h_{n.1} = \text{dom } h^n = \text{dom } h_{n+1.1} = \text{dom } h^{n+1} \neq \emptyset$
donc, par la proposition 2.12., h est non-noethérienne

■

c.q.f.d.

Cette dernière proposition en suggère une autre plus générale

h non-noethérienne $\Leftrightarrow \exists n < \omega . \forall p \ 1 \leqslant p \leqslant n, \ h_{n.p} = h_{n+1.p}$

Cependant, pour nos besoins, la proposition 2.14. est suffisante.

Avant de donner la dernière proposition de ce chapitre, il nous faut, pour sa démonstration, établir le lemme suivant :

Lemme 2.15.

Soit f une fonction quelconque

$\forall n . \text{codom } f_{n.1} = \text{dom } f^{n.1} \cap \text{codom } f$

Preuve :

■

Soit un n quelconque, $f_{n.1}$ la fonction comme ci-dessus

$x \in \text{codom } f_{n.1} \Leftrightarrow \exists y . (f|_{\text{dom } f^n})(y) = x$

$\Leftrightarrow f(y) = x \wedge y \in \{z | f(z) \in \text{dom } f^{n-1}\}$

$\Leftrightarrow f(y) = x \in \text{dom } f^{n-1}$

$\Leftrightarrow x \in \text{codom } f \wedge x \in \text{dom } f^{n-1}$

■

c.q.f.d.

Proposition 2.16.

Soit $h \in \mathbb{H}(B, \subseteq, \sqcap B)$ et $h^n = h_{n.n} \text{ o } \ldots \text{ o } h_{n.1}$ pour chaque puissance n de h

h non noethérienne $\Leftrightarrow \exists n < \omega . [(\text{dom } h_{n.1}$ et $\text{codom } h_{n.1}$ sont filtres pricipaux) $\wedge$

$\text{codom } h_{n.1} \subseteq \text{dom } h_{n.1}]$

Preuve :

■

=>) par la proposition 2.11.,

h non noeth $\Rightarrow \exists n < \omega . \text{dom } h^n = \text{dom } h^{n+1} \neq \emptyset$

donc $\exists n < \omega . \text{dom } h_{n.1} = \text{dom } h^n = \text{dom } h^{n+1} = \text{dom } h_{h+1.1} \neq \emptyset$ ①

alors $\text{codom } h^h \neq \emptyset$

De plus, puisque h est filtrante inférieurement, on a :

$\text{dom } h^n = \text{dom } h_{n.1}$, codom h et $\text{dom } h^{n-1}$ sont filtres principaux

alors, par le lemme 2.15.,

$\text{codom } h_{n.1} = \text{dom } h^{n-1} \cap \text{codom } h$

et par le lemme 2.8., donc $\text{codom } h_{n.1}$ est un filtre principal.

D'ailleurs, par le lemme 2.2. et ①

$\text{dom } h^{n+1} = \text{dom } h_{n+1.1} = \{x / h(x) \in \text{dom } h^n\} = \text{dom } h^n = \text{dom } h_{n.1}$ ②

Soit $y \in \text{codom } h_{n.1}$, donc $\exists x \in \text{dom } h_{n.1} = \text{dom } h^n . h(x) = y$

et par ②, puisque $x \in \text{dom } h^n$, on a $h(x) \in \text{dom } h^n$

donc, $\text{codom } h_{n.1} \subseteq \text{dom } h^n = \text{dom } h_{n.1}$

<=) puisque $\text{dom } h_{n.1}$ et $\text{codom } h_{n.1}$ sont filtres principaux,

alors, $\text{codom } h_{n.1} \neq \emptyset$

donc, par le lemme 2.3., $h_{n.1}$ est non-noethérienne

et, puisque $h_{n.1}$ est une restriction de h, alors h est non-noethérienne.

■ c.q.f.d.

Corollaire 2.17.

Soit $h \in \mathbb{H}(B, \sqsubseteq, \sqcap B)$

h non-noethérienne $\Leftrightarrow \exists n < \omega . (\text{dom } h_{n.1} \neq \emptyset \wedge \sqcap \text{dom } h_{n.1} \sqsubseteq \sqcap \text{codom } h_{n.1})$

L'intérêt d'avoir présenté quatre propositions pour la caractérisation des fonction noethériennes filtrantes inférieurement sur cpo's filtrés complet réside dans les faits suivants :

- les propositions 2.11. et 2.13. sont les plus générales et elles permettent l'étude de la propriété, en utilisant les puissances successives de la fonction.

- les propositions 2.14. et 2.16. permettent un calcul plus simple de cette étude en permettant de réduire l'étude sur les fonctions $h_{n.1}$.

- comme on le verra dans le prochain chapitre, la propositon 2.14. permet la décision de la terminaison d'une règle de transition de termes. Par ailleurs, la proposition 2.16. et le corollaire 2.17. permettent de connaître l'ensemble des valeurs pour lesquelles une règle ne termine pas.

CHAPITRE 3

REGLES DE TRANSITIONS DE TERMES,

FONCTION NOETHERIENNE

ET DECIDABILITE D'ARRET

CHAPITRE 3

PRESENTATION

L'objectif de ce chapitre est d'étudier la terminaison d'une "règle de transitions de termes" ($t_1 \xrightarrow{t_2} t_3$) présentée dans la section 1.2.3.2. Les résultats présentés dans ce chapitre utilisent les propositions de caractérisation des fonctions noethériennes présentées dans le chapitre précédent. Ces résultats sont d'une utilité plus générale que pour les SPC du chapitre 1. Ainsi, nous pouvons les utiliser sur "les règles de réécriture ($t \rightarrow t'$) dans lesquelles la réécriture est le remplacement total d'un terme par un autre terme". Pour cela, nous reprenons ici une définition du type de règles à étudier mais, cette fois-ci, du point de vue général des systèmes de réécriture. Ces règles seront aussi appelées règles de transitions de termes. Elles sont formées par des termes externes qui correspondent aux termes d'états définis dans la section 1.2.3.2. Les termes d'opérations sont appelés, ici, termes internes. La nouveauté, par rapport aux travaux de terminaison de systèmes de réécriture [Der 82], [Der-Man 79], [Jou-Les-Rei 81], [Jou-Kir 81] de la méthode employée

- permet la semi-décidabilité de la propriété non-noethérienne sur une règle de transitions de termes ($g \rightarrow d$) où $var(d) \subseteq var(g)$. Cette restriction sur les variables est habituelle dans les systèmes de réécriture.

- permet, dans le cas de non-noethérienneté, de connaître l'ensemble de termes fermés pour lesquels la réécriture ne termine pas.

- permettra peut-être, dans de futures investigations, une généralisation à une règle de réécriture quelconque où $var(d) \subseteq var(g)$.

Par ailleurs, la méthode employée

- nous fait penser que la propriété non-noethérienne est décidable, autrement dit, que nous pouvons borner le calcul de la propriété. Nous donnons certaines idées qui justifient cette intuition.

- nous permettra aussi de considérer le cas de règles de transitions de termes ($g \rightarrow d$) sans restrictions, c-à-d $var(d) \subseteq var(g)$. Cela nous permettra de connaître quels sont les termes fermés que les nouvelles variables peuvent prendre et pour lesquels la réécriture ne termine pas. Nous donnons certaines idées dans cette voie.

3.1. REGLES DE TRANSITIONS DE TERMES

Soient

$\mathbb{V}$ un ensemble dénombrable d'éléments, appelés variables et parcourus par $x,y,z,\ldots$

$\mathbb{F}$, $\mathbb{S}$ sont deux ensembles finis, t.q. $\mathbb{F} \cap \mathbb{S} = \emptyset \wedge (\mathbb{F} \cup \mathbb{S}) \cap \mathbb{V} = \emptyset$, rangés par une fonction d'arité $\rho : \mathbb{F} \cup \mathbb{S} \to \mathbb{N}$. Les éléments de $\mathbb{F}$ sont appelés symboles de fonctions internes et parcourus par f, h, ... Les éléments de $\mathbb{S}$ sont appelés symboles de fonctions externes et parcourus par $\mathbb{S}$.

Un terme interne sur $\mathbb{F}$ est défini inductivement par

- $x \in \mathbb{V}$ est un terme interne
- si $f \in \mathbb{F}$ et $i_1, \ldots, i_{\rho(f)}$ sont des termes internes alors $fi_1, \ldots, i_{\rho(f)}$ est un terme interne

L'ensemble des termes internes sera dénoté par $H_{\mathbb{F},\mathbb{V}}$ et parcouru par $i, i_1, i'_1, \ldots$

Un terme externe sur $\mathbb{S}$ est défini par

- si $S \in \mathbb{S}$ et $i_1,\ldots,i_{\rho(S)}$ sont des termes internes

  alors $S\, i_1,\ldots,i_{\rho(S)} \in \mathbb{S}$

L'ensemble des termes externes sera dénoté par $H_{\mathbb{S},\mathbb{F},\mathbb{V}}$ est parcouru par $e,e_1,e', \ldots$

L'ensemble des variables qui apparaissent dans un terme $t \in H_{\mathbb{S},\mathbb{F},\mathbb{V}} \cup H_{\mathbb{F},\mathbb{V}}$ est dénoté par Var(t) et défini inductivement par

1. (base) $Var(x) = \{x\}$ si $x \in \mathbb{V}$
2. (induction) $Var\,(u(i_1,\ldots,i_{\rho(u)})) = \bigcup_{j=1}^{\rho(u)} Var(i_j)$ pour chaque $u \in \mathbb{S} \cup \mathbb{F}$

L'ensemble des termes internes fermés est l'ensemble

$$H_{\mathbb{F}} = \{c \in H_{\mathbb{F},\mathbb{V}} \;/\; Var(c) = \emptyset\}$$

L'ensemble des termes externes fermés est l'ensemble

$$H_{\Sigma,F} = \{c \in H_{\Sigma,F,V} \;/\; Var(c) = \emptyset\}$$

Les deux ensembles vont être parcourus par c, c', c_1, c'_1, ... (constantes, sans variables) et le contexte dira à quel ensemble on fait référence .

Une substitution est une fonction $\sigma : V \to H_{\Sigma,F,V} \cup H_{F,V}$ t.q. $\sigma(x) = x$ presque partout.
Les substitutions sont dénotées par $\sigma, \sigma_1, \ldots, \rho, \rho_1, \ldots, \theta, \theta_1, \ldots$
L'ensemble fini $Dom(\sigma) = \{x \in V \;/\; \sigma(x) \neq x\}$ est appelé le domaine de σ.
L'ensemble fini $Codom(\sigma) = \{i \in H_{F,V} \;/\; \sigma(x) = i, \; x \in Dom(\sigma)\}$ est appelé le codomaine de σ . L'ensemble des substitutions est dénoté par Subs.

Une substitution est fermée ssi $Codom(\sigma) \subseteq H_F$. L'ensemble des substitutions fermées est dénoté par Subs.fermées $\subset$ Subs.

Enfin, on appelle les règles de la forme $e \to e'$ où $e, e' \in H_{\Sigma,F,V}$ règles de transitions de termes.

3.2. DECIDABILITE D'ARRET POUR LES REGLES DE TRANSITIONS DE TERMES ($g \to d$) OU $var(d) \subseteq var(g)$.

Dans les systèmes de réécriture les règles $(g \to d)$ ont en général la restriction $var(d) \subseteq var(g)$. Cette restriction permet de définir la relation de réécriture de termes comme un sous-ensemble de $H_{\Sigma,F} \times H_{\Sigma,F}$. Formellement la relation de réécriture $\vdash_R \subseteq H_{\Sigma,F} \times H_{\Sigma,F}$ engendrée par une règle de transitions de termes $R = e \to e'$ est définie par

$$c \vdash_R c' \iff \exists\, \sigma \in \text{Subs.fermées} \;\text{ t.q. }\; c = \sigma e \;\wedge\; c' = \sigma e'$$

Nous disons alors qu'une règle de transitions de termes $R = e \to e'$ termine pour les termes externes fermés ssi la relation $\vdash_R$ est noéthérienne c-à-d s'il n'existe pas une séquence infinie d'éléments de $H_{\Sigma,F}$ telles que

$$c_0 \vdash_R c_1 \vdash_R c_2 \vdash_R \cdots \vdash_R c_n \vdash \cdots \qquad c_0, c_1, \ldots, c_n \in H_{\Sigma,F}$$

Par ailleurs, à notre connaissance, la décidabilité de la terminaison d'un système de réécriture à une seule règle est un problème ouvert. Des recherches récentes [Jou-Kir 81] ont donné une condition suffisante de terminaison d'une règle de réécriture. Le but de cette partie est de trouver des conditions nécessaires et suffisantes pour la terminaison d'une règle de transitions de termes qui soient calculables à l'aide si possible d'un calcul borne. Le premier paragraphe (3.2.1.) présente les règles de transitions de termes comme des fonctions sur la base finitaire d'un cpo filtré complet. Le deuxième paragraphe (3.2.2) montre comment nous pouvons "semi-décider" la propriété non-noethérienne sur de telles fonctions (donc sur de telles règles). Enfin, le troisième paragraphe (3.2.3) donne les idées intuitives qui nous font penser que cette propriété non-noéthérienne est "décidable".

3.2.1. Règles de transitions de termes et leurs fonctions associées.

3.2.1.1. La Règle de Transitions de Termes R comme une Fonction f_R

Les règles de transitions de termes que l'on a défini ont deux caractéristiques particulières :

1. Pour qu'un terme externe fermé soit réécrit, il faut qu'il soit unifiable avec tout le terme gauche de la règle. Donc, le non-déterminisme sur l'application d'une règle à un sous-terme du terme externe fermé donné a été éliminé.

2. On peut donc voir la règle ($g \rightarrow d$) comme une fonction. Ainsi, une règle peut être appliquée seulement aux termes externes fermés filtrables avec la partie gauche de la règle, et on obtient comme résultat d'une telle application un élément unique de l'ensemble des termes externes fermés filtrables avec la partie droite de la règle, puisque $var(d) \subseteq var(g)$.

Formellement :

Soit $R = e \rightarrow e'$ où $e, e' \in H_{\Sigma,F,V}$

La fonction f_R définie par la règle R est la suivante

$$f_R = \{\langle \sigma e, \sigma e' \rangle \ / \sigma \in \text{Subs.fermés}\}$$

Il est clair que f_R est une fonction, mais en plus $f_R = \vdash_R$ comme le montre la proposition suivante :

Proposition 3.1.

$$\forall\, c,c' \qquad c \vdash_R c' \iff f_R(c) = c'$$

Preuve : (trivial)

■ $c \vdash_R c' \iff \exists\, \sigma \in$ Subs-fermés t.q $c = \sigma e \wedge c' = \sigma e' \iff f_R(c) = \sigma e' \iff f_R(c) = c'$ c.q.f.d.

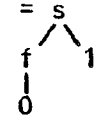
Donc, le problème de savoir si $\vdash_R$ termine est équivalent au problème de savoir si f_R est noethérienne. Comme on a vu dans le chapitre précédent, on connaît une caractérisation de cette propriété dans la classe $\mathcal{H}(B, \sqsubseteq, \Pi B)$ des fonctions h qui vérifient les conditions.

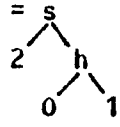
1. dom h et codom h sont filtres principaux dans la base finitaire $(B, \sqsubseteq, \Pi B)$ d'un cpo filtré complet
2. h est filtrante inférieurement

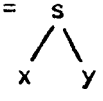
On sait que l'infimum d'un sous-ensemble $A \subseteq H_{F,W}$ existe [voir 5,16 Huet], donc dans le cas où A est un sous-ensemble de $H_{\subseteq,F}$ il existe aussi. Mais cet infimum prend, en quelque sorte, l'aspect d'un terme non-fermé. Voyons ceci à travers les deux exemples suivants.

Exemple 3.1.

▼

Soit e = 

e' = 

donc $e \wedge e'$ = 

où x,y sont des variables définies par une bijection $\Phi : H_F \times H_F \to W$ t.q.

$\Phi\,(f(0),2) = x$

$\Phi\,(1,h(0,1)) = y$

▲

Exemple 3.2.

▼

Soit $e_1 = s \mid f \mid o$ $\quad e_2 = s \mid f \mid f \mid o$ $\quad e_3 = s \mid f \mid f \mid f \mid o$ ...

$\Pi \{e_i\} = s \mid x$ où x est une variable définie par une bijection

$$\Phi : H_{IF} \longrightarrow \mathbb{V} \quad \text{t.q.}$$

$$\Phi (f^i(o)) = x$$

▲

On voit donc que l'infimum correspond à une variable dont l'ensemble de valeurs est déterminé par la bijection Φ.

Pour simplifier la méthode à suivre nous préférons définir une autre fonction g_R appelée la fonction associée à la règle R, telle que :

condition 1) f_R soit plongé dans g_R

condition 2) $g_R \in \mathbb{H}$ (dom g_R, $\sqsubseteq$, Πdom g_R)

condition 3) on peut décider la propriété noethérienne de f_R en utilisant g_R

Avant de donner la définition de cette fonction et d'étudier ces conditions, on énumère certaines propriétés de $H_{\sqsubseteq,F,\mathbb{V}}$.

3.2.1.2. Les Inf-Demi-Treillis-Bien-Fondés (ITF) et les Termes Externes

Soit $(D,\sqsubseteq)$ un ensemble partiellement ordonné.

Une suite d'éléments de D indexée par les entiers naturels et strictement décroissante

$$d_1 \sqsupset d_2 \sqsupset d_3 \sqsupset \ldots d_n \sqsupset \ldots$$

s'appelle : chaîne descendante

Une propriété que certaines structures partiellement ordonnées vérifient est la condition de chaîne descendante (aussi appelée la condition d'ordre bien fondé ou la condition minimale) qui peut être exprimée par un des termes de l'équivalence :

toute chaîne descendante dans D est finie <=>

$\forall A \subseteq D, A \neq \emptyset, \exists$ l'élément minimal de A

On note les structures qui vérifient cette propriété par ITF(D,$\leq$,$\sqcap$) et on remarque que :

Théorème 3.2.

Tout ITF(D,$\leq$,$\sqcap$) est un Inf-demi-treilli-complet (c-à-d $\forall A \subseteq D$, $A \neq \emptyset$, $\sqcap A \in D$)

Preuve

■

On va prouver $\forall A \subseteq D$, $A \neq \emptyset$, $\exists$ l'élément minimal de A $\Rightarrow \forall B \subseteq D$, $B \neq \emptyset$, $\sqcap B \in D$

Soit $B \subseteq D$, $B \neq \emptyset$

Soit $\bar{B}$ le plus petit sous-ensemble de D t.q.

i. $\forall a_1, a_2 \in \bar{B}$, $\sqcap\{a_1, a_2\} \in A$ c-à-d fermé par $\sqcap$

ii. contient B

Par hypothèse, il existe au moins un minimal de B et par la fermeture (i) l'élément minimal de B = $\sqcap B$ (le minimum de B) et puisque $B \subset \bar{B}$ (ii) on a $\forall b \in B. \sqcap \bar{B} \leq b$ donc B est minoré.
De plus on sait par définition de $\bar{B}$ que

$\forall d \in D$, $d \in \downarrow B$ (minorant de B) => $d \in \downarrow \bar{B}$

donc $\sqcap B$ = minimum de $\bar{B}$ = minimal de $\bar{B}$

■ c.q.f.d.

Dans le cas de l'ensemble $H_{S,F,W}$ on peut établir un préordre

Soient e, e' $\in H_{S,F,W}$

L'ordre de filtrage est un préordre $\trianglelefteq \subseteq H_{S,F,W} \times H_{S,F,W}$ défini par la relation binaire suivante

$e \trianglelefteq e'$ ssi $\exists \sigma \in$ Subs t.q. $\sigma e = e'$

On définit l'équivalence de filtrage par la relation $\equiv$ définie par

$$e \equiv e' \text{ ssi } e \sqsubseteq e' \wedge e' \sqsubseteq e$$

et si l'on prend l'ensemble quotient de $H_{\Sigma,F,V}$ par l'équivalence $\equiv$, alors $\sqsubseteq$ définit un ordre partiel sur ce quotient que nous noterons aussi par $\sqsubseteq$, de plus

Lemme 3.3.

$(H_{\Sigma,F,V}|_{\equiv}, \sqsubseteq, \sqcap)$ est un ITF

Preuve :

■ Comme conséquence directe du théorème 13 de Huet [Huet 76] qui dit $(H_{F,V}|_{\equiv}, \sqsubseteq, \sqcap)$ est un ITF où $\sqsubseteq, \equiv$ sont définis par $H_{F,V}$ ■

Corollaire 3.4.

$(H_{\Sigma,F,V}|_{\equiv}, \sqsubseteq, \sqcap)$ est un inf-demi-treilli-complet

Preuve :

■ par le théorème 3.2. et le lemme 3.3. ■

On dénote par $[e]$ la classe d'équivalence de e et par $\nabla_{[e]}$ le filtre principal engendré par la classe d'équivalence $[e]$ dans la structure $(H_{\Sigma,F,V}|_{\equiv}, \sqsubseteq, \sqcap)$.

3.2.1.3. La Règle de Transitions de Termes R considérée comme Fonction g_R

Soit $R = e \rightarrow e'$

La fonction associée à une règle R est la suivante

$$g_R = \{ \langle [e_1], [\sigma e'] \rangle \ / \ [e_1] \in H_{\Sigma,F,V}|_{\equiv} \wedge \exists \sigma \in \text{Subs} \text{ t.q. } e_1 = \sigma e \}$$

Cette fonction g_R vérifie les trois conditions mentionnées dans 3.1.1.2. Ainsi,

la condition 1 : exprime le plongement de f_R dans g_R. La proposition suivante établit ce plongement comme résultat de la définition de g_R.

Proposition 3.5.

Soit $R = e \rightarrow e'$

$\forall\, c \in \text{dom } f_R \quad [f_R(c)] \sqsubseteq g_R([c])$

Preuve

■ $c \in \text{dom } f_R . \quad [f_R(c)] = [\sigma e'] \wedge \sigma \in \text{Subs.fermés} \subseteq \text{Subs} \wedge c = \sigma e \iff [\sigma e'] = g_R([c])$

■ c.q.f.d.

Si l'on considère que g_R est définie sur les termes plutôt que sur la classe d'équivalence de termes, alors cette proposition 3.5. exprime que

$f_R = g_R \mid_{\text{dom } f_R}$

La condition 2 : on sait que la structure $(H_{\Sigma,F,V}|_{\equiv}, \subseteq, \sqcap)$ est un inf-demi-treilli-complet qui forme la base finitaire du cpo obtenu par la complétion des idéaux $(H^{\infty}_{\Sigma,F,V}|_{\equiv}, \subseteq, \sqcap)$.

La proposition suivante établit que g_R est une fonction filtrante inférieurement sur la base finitaire d'un cpo filtré complet.

Proposition 3.6.

Soit $R = e \rightarrow e'$

$g_R \in \Pi(H_{\Sigma,F,V}|_{\equiv}, \subseteq, \sqcap)$

Preuve

■

1) g_R est définie sur un inf-demi-treilli-complet, base finitaire du cpo filtre complet $(H^{\infty}_{\Sigma,F,V}|_{\equiv}, \subseteq, \sqcap)$

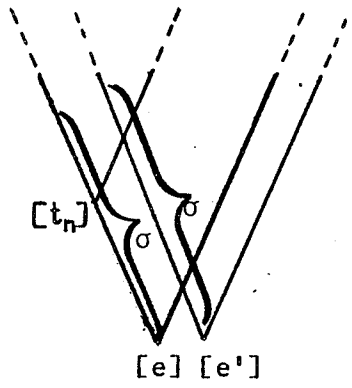
2) Pour prouver que g_R est filtrante inférieurement, il faut prouver $\forall\, n \ (\text{dom } g_R^n \neq \emptyset \Rightarrow \text{dom } g_R^n = \bigtriangledown_t \wedge \text{codom } g_R^n = \bigtriangledown_t)$

Ici, on donne la preuve que pour le domaine :

Par induction sur n

base $n = 1$ $\quad \mathrm{dom}\, g_R^1 \neq \emptyset$ puisque $[e] \in \mathrm{dom}\, g_R^1$ donc par corollaire 1 $\sqcap \mathrm{dom}\, g_R$ existe et trivialement $\sqcap \mathrm{dom}\, g_R = [e]$, alors par définition de g_R on a $\mathrm{dom}\, g_R = \bigtriangledown_{[e]}$

Induction supposons que pour un n donné $\mathrm{dom}\, g_R^n \neq \emptyset \Rightarrow \mathrm{dom}\, g_R^n = \bigtriangledown_{[t_n]}$



si $\mathrm{dom}_R^{n+1} \neq \emptyset$ alors $\mathrm{codom}\, g_{n+1,1} \neq \emptyset$ et par le lemme 2.15

$\mathrm{codom}\, g_{n+1,1} = \mathrm{dom}\, g_R \cap \mathrm{codom}\, g_R$ où $\mathrm{dom}\, g_R^n = \bigtriangledown_{[t_n]}$ et

$\mathrm{codom}\, g_R = \bigtriangledown_{[e']}$

mais par le lemme 2.8. on a

$$\mathrm{codom}\, g_{n+1,1} = \bigtriangledown_{[t_n] \cup [e']}$$

Soit $\sigma \in \mathrm{Subs}$ t.q. $t_n \cup e' = \sigma e'$, il est donc clair que $[\sigma e] \in \mathrm{dom}\, g_R^{n+1}$ puisque $g_R([\sigma e]) = [\sigma e'] \in \mathrm{codom}\, g_{n+1,1}$ (1)

Si l'on prouve que $\forall x \in \mathrm{dom}\, g_R^{n+1}.\ [\sigma e] \sqsubseteq x$ on a prouvé avec (1) que $[\sigma e]$ est le minimum de $\mathrm{dom}\, g_R^{n+1}$ c-à-d $\sqcap \mathrm{dom}\, g_R^{n+1} = \sigma e$ donc

$$\mathrm{dom}\, g_R^{n+1} = \bigtriangledown_{\sigma e}$$

La preuve de $\forall x \in \mathrm{dom}\, g_R^{n+1}.\ [\sigma e] \sqsubseteq x$ est la suivante

soit $x \in \mathrm{dom}\, g_R^{n+1}$ donc $g_R(x) \in \mathrm{codom}\, g_{n+1}$ alors $g(x) = [\Theta\sigma\, e']$

où $x = [\Theta\sigma e]$ donc $[\sigma e] \sqsubseteq x$

■ c.q.f.d.

La condition 3 : la proposition suivante exprime dans quelle mesure on peut décider la propriété "noethérienne" de f_R , en étudiant la même propriété sur g_R . Il faut remarquer que dans le cas où il n'existe aucun symbole de fonction (externe ou interne) le domaine de f_R est vide et donc f_R est noethérienne.

Proposition 3.7.

Soit $R = e \rightarrow e'$

S'il existe $a \in \mathbb{F} \cup \mathbb{E}$ t.q. $\rho(a) = 0$

alors (f_R est non-noethérienne <=> g_R est non-noethérienne)

sinon f_R est noethérienne ($f_R = \Omega$)

Preuve

■

Supposons que $\forall a \in \mathbb{F} \cup \mathbb{E}$ $\rho(a) \neq 0$ donc $H_{\mathbb{F},\mathbb{E}} = \emptyset$ et par conséquent dom $f_R = \emptyset$ alors $f_R = \Omega$ c-à-d f_R est noethérienne

Par contre, s'il existe $a \in \mathbb{F} \cup \mathbb{E}$ $\rho(a) = 0$ alors

=>) f_R non-noethérienne veut dire qu'il existe $c \in$ dom f_R .
$\forall n$ $f_R^n(c) \in$ dom f_R , donc en utilisant la proposition 1 on peut prouver sans difficulté que g_R boucle avec [c]

<=) g_R non-noetherienne veut dire qu'il existe $[e_1] \in$ dom f_R .
$\forall n$ $g_R^n([e_1]) \in$ dom g_R . Si l'on considère l'élément σe_1 où

$\forall x \in Var(e_1)$ et $\sigma(x) = a$. Il est clair que g_R boucle avec $[\sigma e_1]$, donc en utilisant la proposition 1 on peut aussi prouver sans difficulté que f_R boucle avec σe_1

■

3.2.2. Le Calcul de la Propriété Non-Noethérienne

D'après le paragraphe précédent (Prop 3.7.) nous connaissons la liaison entre la non-noethériennité de g_R et f_R. Plus précisément, on sait que dans le cas où on a au moins une classe d'équivalence dans $H_{\mathbb{E},F,V}|_{\equiv}$

formée par un terme fermé, la non-noethériennité de g_R est équivalente à la non-noethériennité de f_R, dans le cas contraire, évidemment, f_R est noethérienne. Tout cela nous conduit à étudier le calcul de la propriété non-noethérienne sur g_R comme un "synonyme" de la même propriété sur f_R et en définitive sur l'arrêt (terminaison) de la règle de transitions de termes $R = e \rightarrow e'$.

Dans ce paragraphe nous démontrons que la propriété "non-noethérienne" pour les fonctions g_R est calculable. Cela nous donne la semi-décidabilité de cette propriété. Cette semi-décidabilité nous permet aussi de caractériser l'ensemble de termes fermés (constants, sans variables) finis tels que la règle ne termine pas. Ce fait est à notre connaissance une information nouvelle par rapport aux études sur les systèmes de réécriture.

3.2.2.1. Calculabilité de $g_R n,1$ et $\varphi^n_{g_R}$

Dans le chapitre 2 (2.3.) nous avons défini la fonction d'énumération φ^n_f pour l'ensemble des puissances finies $f \in \mathcal{F}$. Dans ce chapitre 2 nous avons aussi prouvé (prop. 2.14) que la propriété non-noethérienne (dans le cas où $f \in \mathcal{H}(B, \equiv, \sqcap B)$) peut être caractérisée en étudiant uniquement la suite des fonctions :

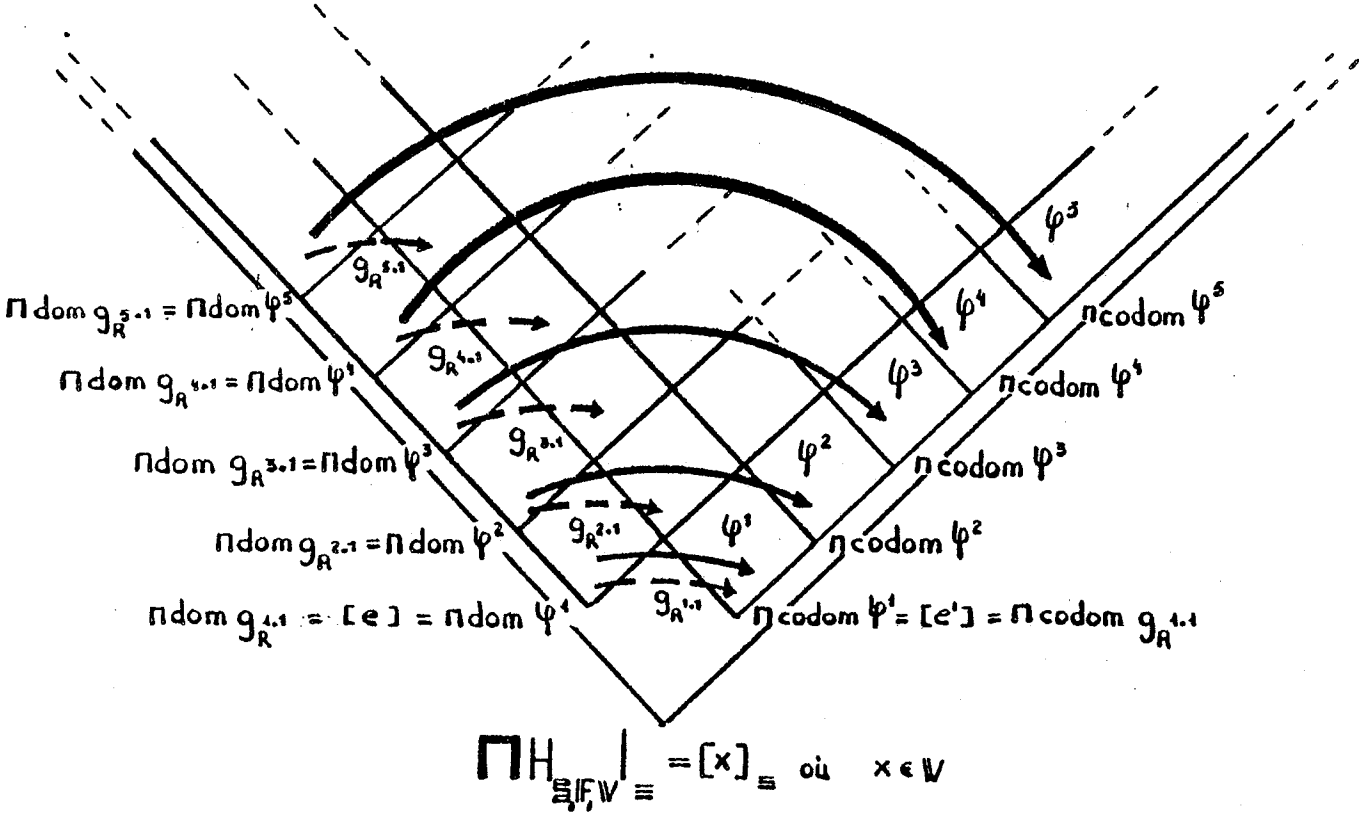
$$f = f_{1.1}, f_{2.1}, \ldots, f_{n.1}, \ldots$$

où $f^n = f_{n.n} \circ \ldots \circ f_{n.1}$ pour chaque n

L'intérêt de travailler sur cette suite plutôt que sur $\varphi^1_f, \varphi^2_f, \ldots, \varphi^n_f$, est une optimisation du calcul. Ainsi, nous n'avons pas besoin de composer n fois la fonction f pour connaître φ^n_f. Donc, pour les fonction qui nous intéressent maintenant (g_R), nous calculons la propriété non-noethérienne pour la suite

$$g_R = g_{R^{1 \cdot 1}}, g_{R^{2 \cdot 1}}, \ldots, g_{R^{n \cdot 1}}, \ldots$$

Par ailleurs, et aussi d'après le chapitre précédent, nous savons (lemme 2.9.) que les fonctions filtrantes inférieurement sont telles que l'infimum des domaines de ses puissances φ_f^n forment une suite croissante. Il en est de même pour les codomaines et nous pouvons vérifier que cela est vrai aussi pour les codomaines de la suite $f_{1\cdot1}$, $f_{2\cdot1}$, ... , $f_{n.n}$, ... Cela se vérifie alors pour g_R à cause de la proposition 3.6. Le dessin suivant illustre les fonctions $\varphi^1 g_R$, ... ; φg_R^n , ... et $g_{R\,1\cdot1}$, ... $g_{R\,n\cdot1}$, ..., et les suites des infimums pour les domaines et codomaines (avec $R = e \to e'$ et en notant $\varphi^n_{g_R}$ par φn)

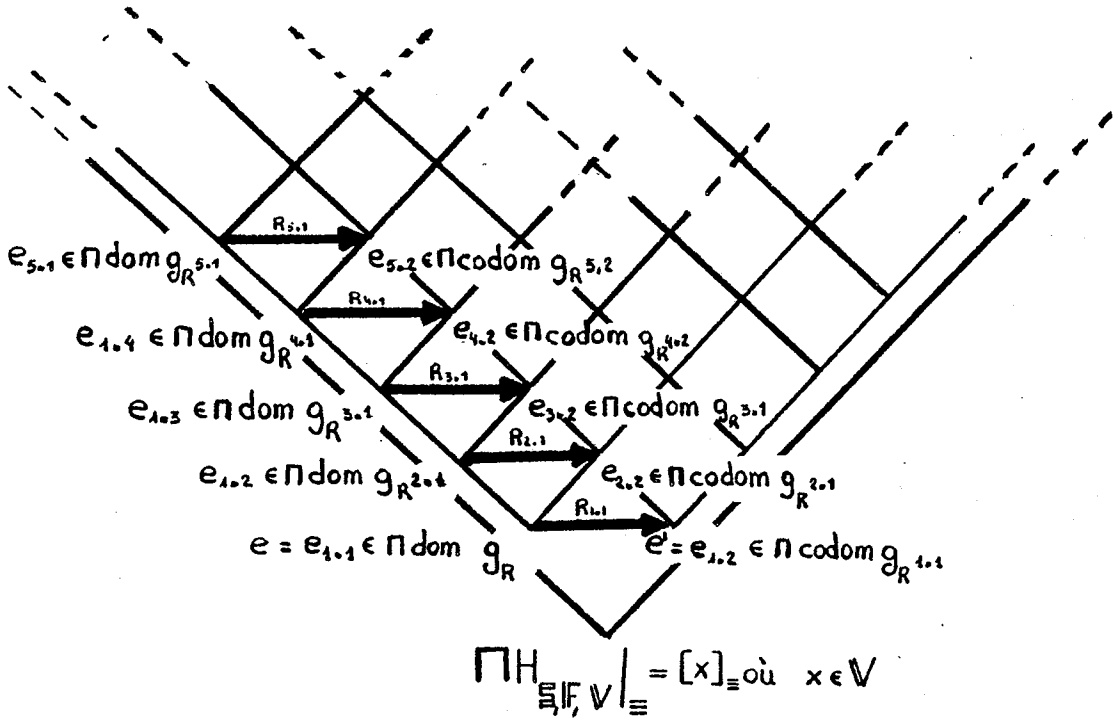


$$\sqcap H_{g,F,V}\Big|_{\equiv} = [x]_{\equiv} \quad \text{où} \quad x \in V$$

Finalement, la fonction g_R est déterminée à partir d'une règle R et elle est filtrante inférieurement , donc les fonctions $g_R{}^{1.1}$, $g_R{}^{2.1}$,..., $g_R{}^{n.1}$, ... non-vides ($\neq \Omega$) vont elles aussi être déterminées par une suite de règles de transitions de termes $R_{1.1}$, ... , $R_{n.1}$, ... t.q.

$$R_{n.1} = e_{n.1} \rightarrow e_{n.2}$$

où $e_{n.1} \in \prod \text{dom}_1 g_R\ n.1$, $e_{n.2} \in \prod \text{codom}\ g_R\ n.1$



La proposition suivante montre que la fonction $g_R{}^{n+1.1}$ est calculable à partir de g_R et $g_R{}^{n.1}$
La preuve de cette proposition donne une méthode pour obtenir la règle $R_{n+1.1}$ à partir de $R_{n.1}$.

Proposition 3.8.

Soient $R = R_{1.1} = e_{1.1} \rightarrow e_{1.2}$

g_R sa fonction associée

$g_R{}^{n.1}$ comme ci-dessous

Alors $g_R{}^{n+1.1}$ est calculable

Preuve :

■

par le lemme 2.15 codom $g_R^{n+1.1}$ = dom g_R^n ∩ codom g_R

de plus dom g_R^n = dom $g_R^{n.1}$

Par ailleurs, la proposition 3.6. dit que g_R est filtrante inférieurement donc

. ou bien dom $g_R^{n\cdot 1}$ = ∅ et dans ce cas $g_R^{n+1\cdot 1}$ = Ω

. ou bien dom $g_R^{n\cdot 1}$ et codom g_R sont filtres principaux. Dans ce cas, et par le lemme 2.8. l'intersection des filtres principaux est un filtre principal. Par conséquent, l'infimum de cette intersection est le supremum des infimums de filtres à intersecter. Ce supremum peut être calculé au moyen de l'unification [Hue 76] .

Ainsi, soient $e_{n\cdot 1}$ ∈ ∩dom $g_R^{n\cdot 1}$ et $e_{1.2}$ ∈ ∩codom g_R

soit ζ_n un renommage sur les variables de R t.q.

$$\text{var}(e_{n\cdot 1}) \cap ((\text{var}\ \zeta_n e_{1.2}) \cup \text{var}(\zeta_n e_{1.1})) = \emptyset$$

alors

- s'il n'existe pas $\sigma_n, \sigma'_n \in$ Subs telles que $\sigma_n\ e_{n\cdot 1} = \sigma'_n\ \zeta_n e_{1.2}$

  alors

  dom $g_R^{n.1}$ ∩ codom g_R = ∅

  donc

  g_R^{n+1} = Ω puisque codom $g_R^{n+1.1}$ = ∅

- s'il existe $\sigma_n, \sigma'_n \in$ Subs telles que $\sigma_n\ e_{n\cdot 1} = \sigma'_n \zeta_n e_{1.2}$ alors

$$\sqcup\{e_{n\cdot 1}, \zeta_n e_{1.2}\} = \sigma_n e_{n\cdot 1} = \sigma'_n\ \zeta_n e_{1.2}$$ (voir Huet 5.5.1 [Huet 76])

donc d'après le lemme 2.8. codom $g_R^{n+1.1} = \nabla\left[\sigma_n e_{n.1}, \sigma'_n\ \zeta_n e_{1.2}\right]$.

Il reste à déterminer dans ce cas ∩dom $g_R^{n+1.1}$. Or codom $g_R^{n+1.1}$ est l'ensemble des valeurs qui peuvent être réécrites au moins n fois après avoir été réécrit une fois. Donc la substitution σ'_n a caractérisé les plus petits

termes qui peuvent être associés aux variables dans $\zeta e_{1.2}$ pour que $\sigma_n \zeta_n e_{1.2} \in$ codom $g_R{}^{n+1.1}$

Par conséquent

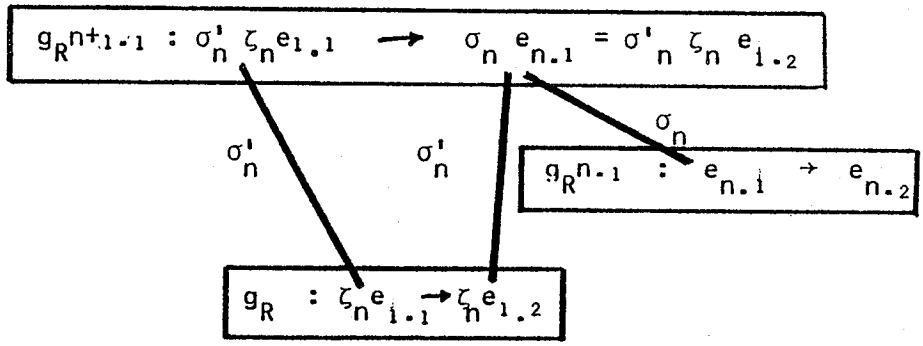
$\sigma'_n \zeta e_{1.1} \in$ dom $g_R{}^{n+1.1}$ et la règle

$R_{n+1.1} = \sigma'_n \zeta_n e_{1.1} \rightarrow \sigma'_n \zeta_n e_{1.2} = \sigma_n e_{n.1}$

caractérise la fonction $g_R{}^{n+1.1} = g_{R_{n+1.1}}$ c.q.f.d.

■

La preuve de la proposition précédente peut être illustrée par



et résumée dans l'algorithme suivant

Algorithme 1

Entrée $g_R : e_{1.1} \rightarrow e_{1.2}$, $g_R{}^{n.1}$ sous forme de:

règle $g_R{}^{n.1} : e_{n.1} \rightarrow e_{n.2}$ ou $g_R{}^{n.1} : \Omega$

Sortie $g_R{}^{n+1.1}$ sous forme de: règle $g_R{}^{n+1.1} : e_{n+1.1} \rightarrow e_{n+1.2}$ ou

$g_R{}^{n+1.1} : \Omega$

Méthode

si $g_R{}^{n.1} : \Omega$

alors $g_R{}^{n+1.1} : \Omega$

Sinon

$g_R{}^{n.1}$ est sous la forme $g_R{}^{n.1} : e_{n.1} \rightarrow e_{n.2}$

Soit ζ_n un renommage de $R = e_{1.1} \rightarrow e_{1.2}$ t.q.

$var(e_{n.1}) \cap (var(\zeta_n e_{1.1}) \cup var(\zeta_n e_{1.2})) = \emptyset$

S ' il n'existe pas σ_n, σ'_n t.q. $\sigma_n e_{n.1} = \sigma'_n \zeta_n e_{1.2}$

alors $g_R{}^{n+1.1} : \Omega$

sinon $g_R{}^{n+1.1} : e_{n+1} = \sigma'_n \zeta_n e_{1.1} \rightarrow e_{n+1.2} = \sigma_n e_{n.1}$

$= \sigma'_n \zeta_n e_{1.1}$

fsi

fsi

Exemple 3.9. :

Soit $R = R_{1.1} = (e_{1.1} = s(x,x,y) \rightarrow e_{1.2} = s(f(x),y,x))$

donc

$g_R{}^{1.1} = g_{R_{1.1}} : s(x,x,y) \rightarrow s(f(x),y,x)$

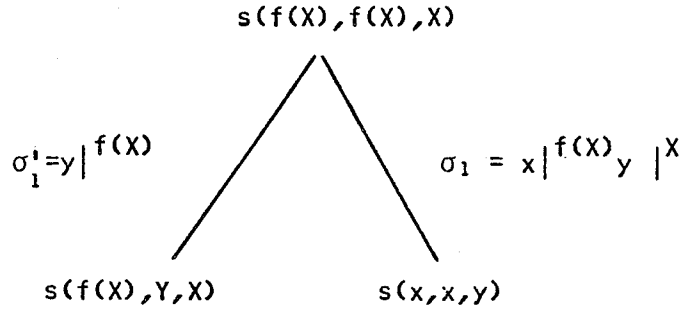
calcul de $g_R{}^{2.1}$

1. soit $\zeta x = X$ $\zeta y = Y$ donc $g_R : s(X,X,Y) \rightarrow s(f(X),Y,X)$

et $g_R{}^{1.1} : s(x,x,y) \rightarrow s(f(x),y,x)$

2. on calcule σ'_1 et σ_1 par unification entre

$s(f(X),Y,X)$ et $s(x,x,y)$



3. alors

$$g_{R^{2.1}} : e_{2.1} = s(X,X,f(X)) \rightarrow e_{2.2} = (f(X),f(X),X)$$

<u>calcul de</u> $g_{R^{3.1}}$

1. soit $\zeta = \Omega$ donc $g_{R^{2.1}} : s(X,X,f(X)) \rightarrow s(f(X),f(X),X)$

   $g_R : s(x,x,y) \rightarrow s(f(x),y,x)$

2. on calcule σ'_n et σ_n par unification entre $s(f(x),y,x)$ et $s(X,X,f(X))$.

   Mais elles n'existent pas, donc $g_{R^{3.1}} = \Omega$ et par conséquent

   $\forall \quad n>2 \quad . \quad g_{R^{n.1}} = \Omega$

▲

La proposition 3.8. peut être étendue au calcul de

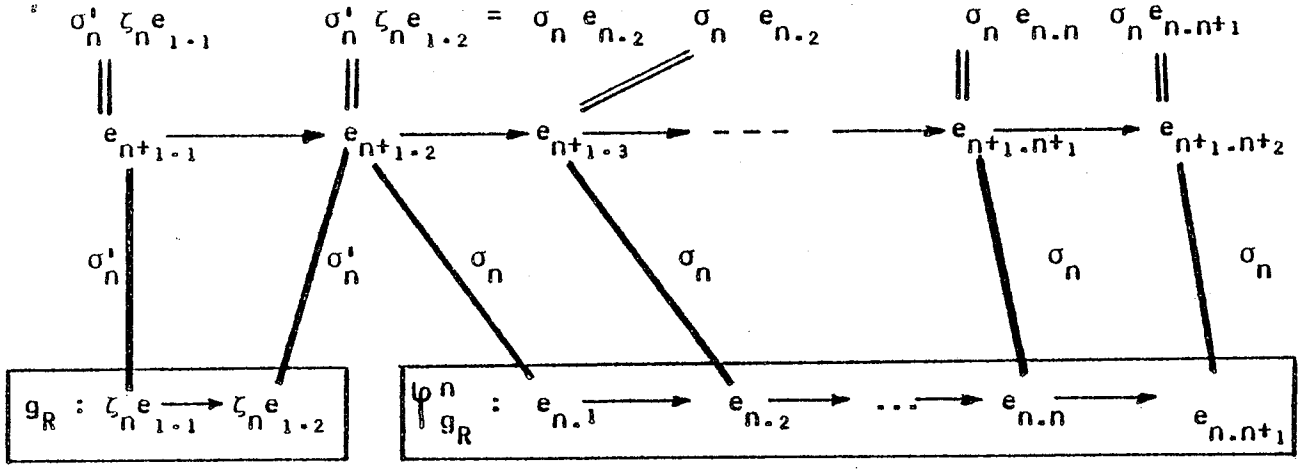
$$\varphi^{n+1}_{g_R} = g_{R^{n+1.n+1}} \circ \ldots \circ g_{R^{n+1.1}} \quad \text{à partir de} \quad \varphi^{n}_{g_R} \quad \text{et} \quad \varphi^{1}_{g_R}$$

appelé dorénavant "<u>calcul en hauteur</u>". Il peut être résumé comme le calcul de la fonction $\varphi^{n+1}_{g_R} = \varphi^{n}_{g_R} \circ g_R$. Ainsi, dans ce calcul on a

$$\varphi^{n}_{g_R} : e_{n.1} \xrightarrow{Rn.1} e_{n.2} \xrightarrow{Rn.2} \ldots\ldots \xrightarrow{Rn.n-1} e_{n.n} \xrightarrow{Rn.n} e_{n.n+1}$$

et nous calculons les termes possibles associés aux variables dans $e_{n.1}$ qui peuvent être réécrits n+1 fois. Le schéma suivant illustre le calcul de $\varphi^{n+1}_{g_R}$ dans le cas où var $e_{1.2} \subseteq$ var $e_{1.1}$:

(avec $\bigcup_{j=1}^{n+1}$ var $e_{n.j}$) $\cap$ (var(ζ_n $e_{1.1}$) $\cup$ var(ζ_n $e_{1.2}$)) = $\emptyset$)



On peut aussi dire que le calcul de $\varphi^{n+1}_{g_R}$ consiste à calculer $e_{n+1.1} = \sigma'_n \zeta_n e_{1.1}$ et à le réécrire n+1 fois dans le cas où var($e_{1.2}$) $\subseteq$ var($e_{1.1}$) (voir section 3.2).

Exemple 3.9. (suite) :

▼

Calcul de $\varphi^1_{g_R}$ trivial $\varphi^1_{g_R} = g_{R^{1.1}} = g_{R_{1.1}}$

Calcul de $\varphi^2_{g_R}$. Puisque $\sigma_1 = x\,|^{f(X)}\,y|^{X}$, alors

$\varphi^2_{g_R}$: $e_{2.1}$: s(X,X,f(X)) → $e_{2.2}$ = s(f(X),f(X),X)

→ $e_{3.3}$ = s(f^2(X),X,f(X))

$\sigma_1 e_{3.2}$

Calcul de $\varphi^3_{g_R}$. Puisque σ_2 n'existe pas $\varphi^n_{g_R} = \Omega$ pour tout $n > 2$

▲

3.2.2.2. Les suites $g_R{}^{1.1}$, $g_R{}^{2.1}$, ..., $g_R{}^{n.n}$, ... et $\varphi^1_{g_R}$, $\varphi^2_{g_R}$, ..., $\varphi^n_{g_R}$, ...

et la Propriété Non-Noethérienne

La preuve constructive de la proposition 3.8. permet de calculer itérativement la suite

$$g_R^{1.1} \quad , g_R^{2.1} \quad , \ldots, \quad g_R^{n.n} \quad , \ldots$$

Par ailleurs, d'après les propositions 3.6. et 2.14 nous savons que

$$g_R \text{ non-noethérienne} \Leftrightarrow \exists\, n < \omega \,.\; g_R{}^{n.1} = g_R{}^{n+1.1} \neq \Omega$$

Donc, nous pouvons construire une procédure qui nous permet de "semi-décider" la propriété non-noethérienne. Cette procédure peut être exprimée par :

Procédure 1

Entrée : $g_R : e_{1.1} \longrightarrow e_{1.2}$ ou $g_R : \Omega$

Sortie : g_R est noethérienne ou g_R est non-noethérienne

Méthode

Soit $n := 2$, $g_R{}^{1.1} := g_R$

Répéter

Calculer $g_R{}^{n.1}$ au moyen de l'algorithme 1

Si $g_R{}^{n.1} = \Omega$ alors g_R est noethérienne

Sinon soit ζ'_n un renommage de var $e_{n.1}$ t.q.

var $(e_{n.1})$ ∩ var $(e_{n-1.1}) = \emptyset$

si $\zeta'_n\, e_{n.1} \equiv e_{n-1.1}$ alors g_R est non-noethérienne

sinon $n := n+1$

fsi

fsi

jusqu'à : g_R noethérienne ou non-noethérienne

finrépéter

Les exemples suivants montrent cette semi-décidabilité

Exemple 3.9. (suite) : ($\exists n < \omega \;.\; g_R{}^{n.1} = \Omega$ avec $n=3$)

▼

$g_R{}^{1.1}$: $s(x,x,y) \rightarrow s(f(x),y,x)$

$g_R{}^{2.1}$: $s(X,X,f(X) \rightarrow s(f(X),f(X),X)$

$g_R{}^{3.1}$: Ω

donc g_R est noethérienne

▲

Dorénavant , nous montrerons seulement les résultats de l'application successive de la preuve de la proposition 3.8.

Exemple 3.10. : ($\forall\; n < \omega \;.\; g_R{}^{n.1} \neq g_R{}^{n+i.1} \neq \Omega$)

▼

$g_R{}^{1.1}$: $s(f(x),y) \rightarrow s(x,f(y))$

$g_R{}^{2.1}$: $s(f^2(x),y) \rightarrow s(f(x),f(y))$

$g_R{}^{3.1}$: $s(f^3(x),y) \rightarrow s(f^2(x),f(y))$

$g_R{}^{4.1}$: $s(f^4(x),y) \rightarrow s(f^3(x),f(y))$

.

.

.

donc nous ne pouvons pas décider de la propriété non-noethérienne.

▲

Exemple 3.11. $(\exists n < \omega \ . \ g_{R^{n.i}} = g_{R^{n.1.1}} \neq \Omega)$

▼

$g_{R^{1.1}}$: $s(x,y,z,f(w)) \rightarrow s(f(w),f(x),y,z)$

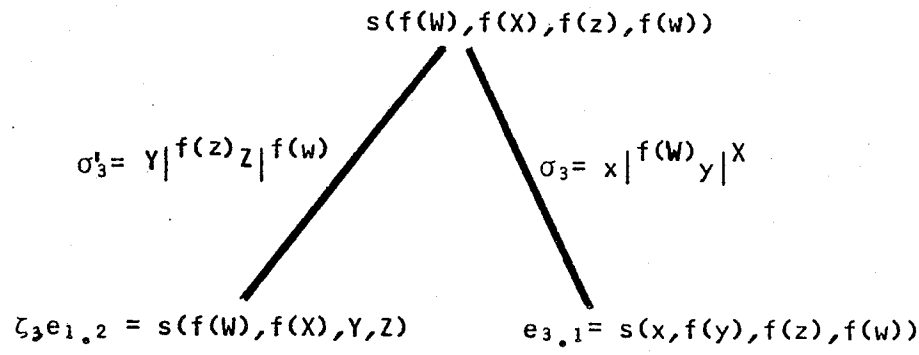
$g_{R^{2.1}}$: $s(x,y,f(z),f(w)) \rightarrow s(f(w),f(x),y,f(z))$

$g_{R^{3.1}}$: $s(x,f(y),f(z),f(w)) \rightarrow s(f(w),f(x),f(y),f(z))$

$g_{R^{3.1}} = g_{R^{4.1}}$ puisque

l'algorithme 1 procède pour $g_{R^{4.1}}$ de la façon suivante

$\zeta_3\ x = X$
$\zeta_3\ y = Y$
$\zeta_3\ z = Z$
$\zeta_3\ w = W$



donc

$g_{R^{4.1}} : s(X,f(z),f(w),f(W)) \rightarrow s(f(W),f(X),f(x),f(w))$

$\sigma'_3\ \zeta e_{i.1}$

mais $\zeta'_3\ X = x$, $\zeta'_3\ W = w$

donc $\zeta'_3\ e_{4.1} \equiv e_{3.1}$

alors $g_{R^{3.1}} = g_{R^{4.1}}$ et g_R est <u>non-noethérienne</u>

▲

Exemple 3.12. $(\exists n < \omega \,.\, g_R{}^{n}{}_{\cdot 1} = g_R{}^{n+1}{}_{\cdot 1} \neq \Omega)$

▼

$g_R{}^{1}{}_{\cdot 1}$: $s(x,y,z,f(w)) \rightarrow s(f(w),x,y,z)$

$g_R{}^{2}{}_{\cdot 1}$: $s(x,y,f(z),f(w) \rightarrow s(f(w),x,y,f(z))$

$g_R{}^{3}{}_{\cdot 1}$: $s(x,f(y),f(z),f(w)) \rightarrow s(f(w),x,f(y),f(z))$

$g_R{}^{4}{}_{\cdot 1}$: $s(f(x),f(y),f(z),f(w) \rightarrow s(f(w),f(x),f(y),f(z))$

$g_R{}^{5}{}_{\cdot 1} = g_R{}^{4}{}_{\cdot 1}$

donc g_R est <u>non-noethérienne</u>

▲

Exemple 3.13. $(\forall n < \omega \,.\, g_R{}^{n}{}_{\cdot 1} \neq g_R{}^{n+1}{}_{\cdot 1} \neq \Omega)$

▼

$g_R{}^{1}{}_{\cdot 1}$: $s(x,f(h(z),y),w) \longrightarrow s(x,f(h(z),y),w)$

$g_R{}^{2}{}_{\cdot 1}$: $s(w,f(h(x),p),f(h(y),z)) \rightarrow s(x,f(h(y),z),f(w,x))$

$g_R{}^{3}{}_{\cdot 1}$: $s(h(y),f(h(w),p),f(h(x),z)) \rightarrow s(w,f(h(x),z),f(h(y),w))$

$g_R{}^{4}{}_{\cdot 1}$: $s(h(y),f(h^2(x),p),f(h(w),z)) \longrightarrow s(h(x),f(h(w),z),f(h(y),h(x)))$

$g_R{}^{5}{}_{\cdot 1}$: $s(h(w),f(h^2(y),p),f(h^2(x),z)) \longrightarrow s(h(y),f(h^2(x),z),f(h(w),h(y)))$

$G_R{}^{6}{}_{\cdot 1}$: $s(h^2(w),f(h^2(y),p),f(h^2(x),z)) \longrightarrow s(h(y),f(h^2(x),z),f(h^2(w),h(y)))$

$G_R{}^{7}{}_{\cdot 1}$: $s(h^2(w),f(h^3(y),p),f(h^2(x),z)) \longrightarrow s(h^2(y),f(h^2(x),z),f(h^2(w),h^2(y)))$

▲

Nous <u>ne pouvons pas décider</u> de la propriété non-noethérienne .

Il est également intéressant de considérer ces cinq exemples comme une illustration de la proposition 2.1.3. et du calcul en hauteur. Nous pouvons remarquer que chaque exemple appartient à une des familles suivantes de fonctions :

famille 1 : $\exists\, n < \omega \,.\, \mathrm{dom}\, f^n \cap \mathrm{codom}\, f^n = \emptyset$

famille 2 : $\forall\, n < \omega \,.\, \mathrm{dom}\, f^n \cap \mathrm{codom}\, f^n \neq \emptyset \wedge \mathrm{codom}\, f^n \not\subseteq \mathrm{dom}\, f^n$

famille 3 : $\exists\, n < \omega \,.\, \mathrm{codom}\, f^n \subset \mathrm{dom}\, f^n$

famille 4 : $\exists\, n < \omega \,.\, \mathrm{codom}\, f^n = \mathrm{dom}\, f^n$

Exemple 3.9. (suite) (famille 1)

▼

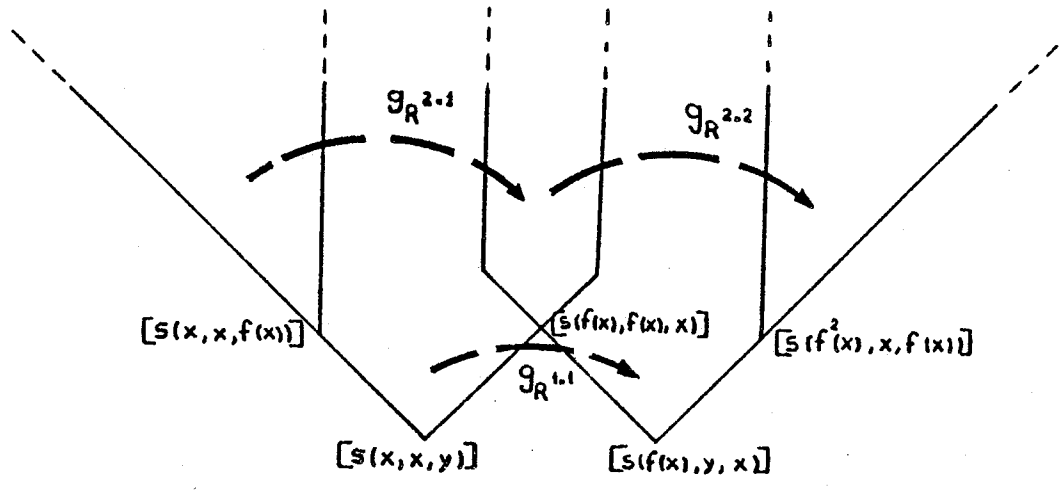
Calcul de $\varphi^n_{g_R}$

$\varphi^1_{g_R}$: $s(x,x,y) \xrightarrow{R_{1.1}} s(f(x),y,x)$

$\varphi^2_{g_R}$: $s(x,x,f(x) \xrightarrow{R_{2.1}} s(f(x),f(x),x \xrightarrow{R_{2.2}} s(f^2(x),x,f(x))$

$\varphi^3_{g_R} = \Omega$

Illustration de la proposition 2.13.



Remarquer que dom $\varphi^{2}_{g_R}$ ∩ codom $\varphi^{2}_{g_R}$ = ∅ . En particulier

dom $g_{R^{2\cdot 1}}$ ∩ codom $g_{R^{2\cdot 2}}$ = ∅ donc d'après la proposition 2.13 la fonction g_R est noethérienne.

▲

Exemple 3.10. (suite) (famille 2)

▼

Calcul de $\varphi^{n}_{g_R}$

$\varphi^{1}_{g_R}$: $s(f(x),y) \to s(x,f(y))$

$\varphi^{2}_{g_R}$: $s(f^2(x),y) \to s'f(x),f(y)) \to s(x,f^2(y))$

$\varphi^{3}_{g_R}$: $s(f^3(x),y) \to s(f^2(x),f(y)) \to s(f(x),f^2(y)) \to s(x,f^3(y))$

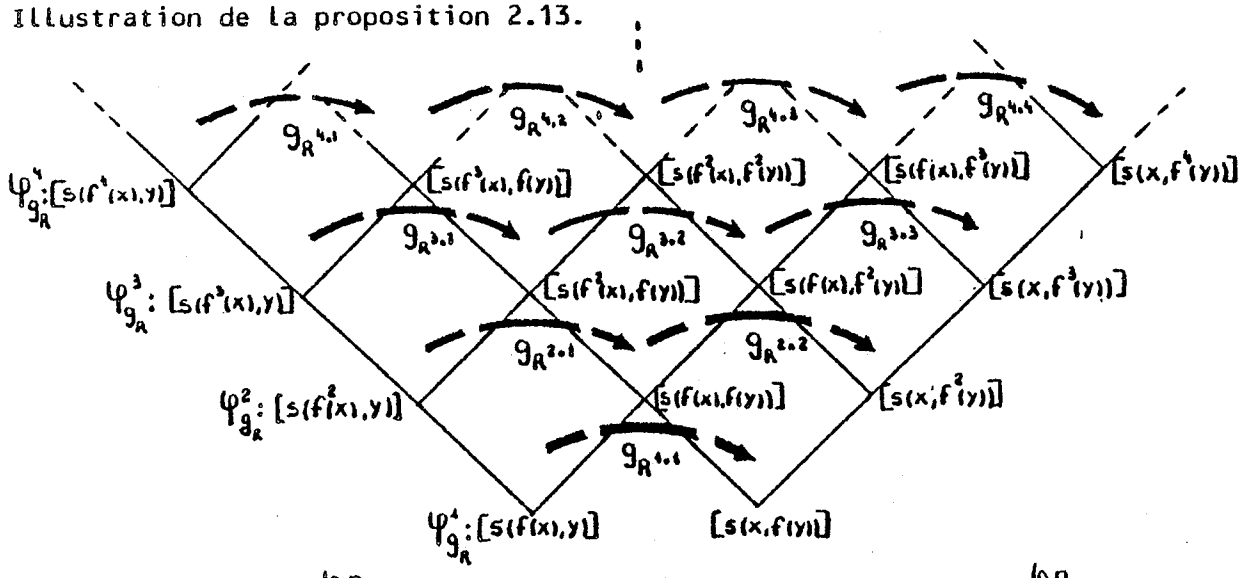
$\varphi^{4}_{g_R}$: $s(f^4(x),y) \to s(f^3(x),f(y) \to s(f^2(x),f^2(y)) \to s(f(x),f^3(y)) \to s(x,f^4(y))$

.

.

.

Illustration de la proposition 2.13.



Remarquer que codom $\varphi^{n}_{g_R}$ ne sera jamais un sous-ensemble de dom $\varphi^{n}_{g_R}$ pour $n < \omega$ donc d'après la proposition 2.13 la fonction g_R est noethérienne.

▲

Exemple 3.11. (*suite*) (famille 3)

▼

Calcul de $\varphi^{n}_{g_R}$

$\varphi^{1}_{g_R}$: $s(x,y,z,f(w)) \to s(f(w),f(x),y,z)$

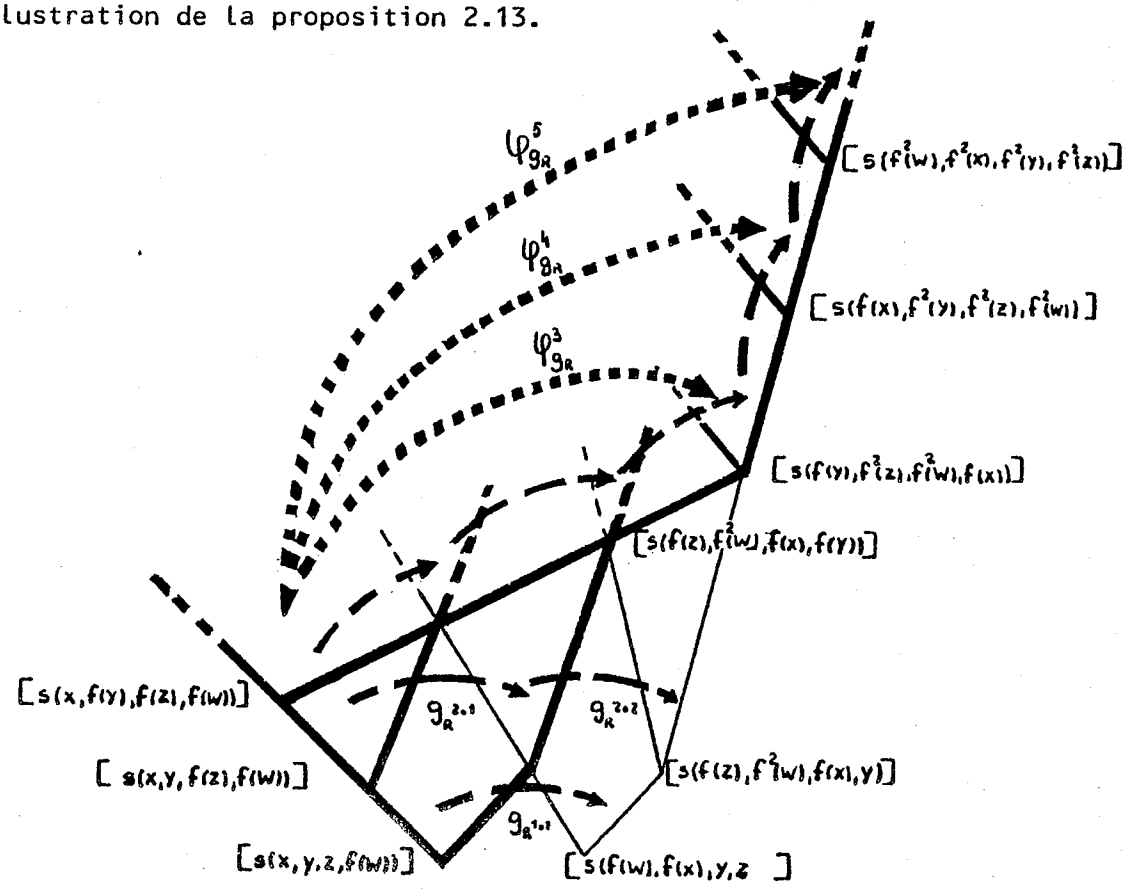
$\varphi^{2}_{g_R}$: $s(x,y,f(z),f(w)) \to s(f(w),f(x),y,f(z)) \to s(f(z),f^2(w),f(x),y)$

$\varphi^{3}_{g_R}$: $s(x,f(y),f(z),f(w)) \to s(f(w),f(x),f(y),f(z)) \to$
$s(f(z),f^2(w),f(x),f(y)) \to s(f(y),f^2(z),f^2(w),f(x))$

$\varphi^{4}_{g_R}$: $s(x,f(y),f(z),f(w)) \to s(f(w),f(x),f(y),f(z)) \to$
$s(f(z),f^2(w),f(x),f(y)) \to s(f(y),f^2(z),f^2(w),f(x) \to$
$s(f(x),f^2(y),f^2(z),f^2(w))$

.
.
.

Illustration de la proposition 2.13.



Remarquer que codom $\varphi^3_{g_R} \subset$ dom $\varphi^3_{g_R}$ donc d'après la proposition 2.13., g_R est non-noethérienne.

▲

Exemple 3.12. (*suite*) (famille 4)

▼

Calcul de $\varphi^n_{g_R}$

$\varphi^1_{g_R}$: $s(x,y,z,f(w)) \to s(f(w),x,y,z)$

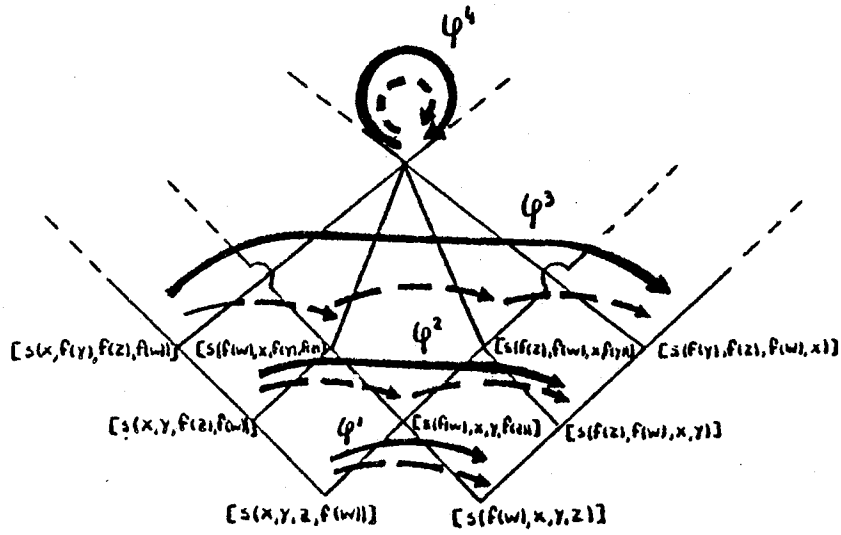
$\varphi^2_{g_R}$: $s(x,y,f(z),f(w)) \to s(f(w),x,y,f(z) \to s(f(z),f(w),x,y)$

$\varphi^3_{g_R}$: $s(x,f(y),f(z),f(w)) \to s(f(w),x,f(y),f(z)) \to s(f(z),f(w),x,f(y)) \to$
$s(f(y),f(z),f(w),x)$

$\varphi^4_{g_R}$: $s(fx),f(y),f(z),f(w)) \to s(f(w),f(x),f(y),f(z)) \to$
$s(f(z),f(w),f(x),f(y) \to s(f(y),f(z),f(w),f(x) \to$
$s(f(x),f(y),f(z),f(w))$

$\varphi^n_{g_R}$ pour $n \geq 4$

Illustration de la proposition 2.13.



Remarquer que codom $\varphi^4_{g_R}$ = dom $\varphi^4_{g_R}$ donc d'après la proposition 2.13 g_R est non-noethérienne

▲

Exemple 3.13. *(suite)* (famille 2)

▼

Calcul de $\varphi^n_{g_R}$

$\varphi^1_{g_R}$: $s(x,f(h(z),p),w) \to s(z,w,f(x,z))$

$\varphi^2_{g_R}$: $s(w,f(h(x),p),f(h(y),z)) \to s(x,f(h(y),z),f(w,x)) \to s(y,f(w,x),f(x,y))$

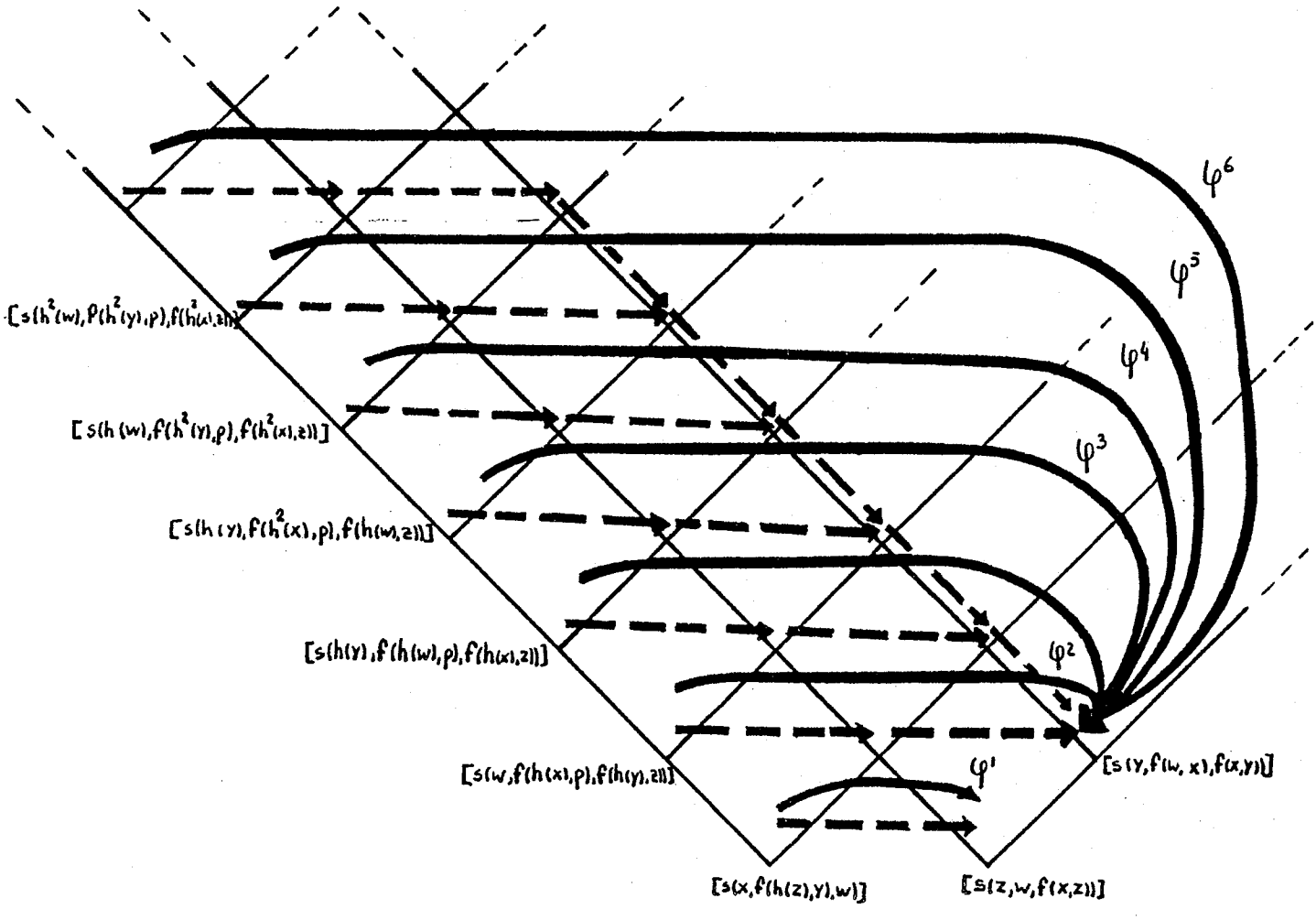
$\varphi^3_{g_R}$: $s(h(y),f(h(w),p),f(h(x),z)) \to s(w,f(h(x),z),f(h(y),w)) \to s(x,f(h(y),w),f(w,x)) \to s(y,f(w,x),f(x,y))$

$\varphi^4_{g_R}$: $s(h(y),f(h^2(x),p),f(h(w),z)) \to s(h(x),f(h(w),z),f(h(y),h(x))) \to s(w,f(h(y),h(x)),f(h(x),w)) \to s(y,f(h(x),w),f(w,y)) \to s(x,f(w,y),f(y,x))$

.

.

.



Remarquons qu'à première vue il n'est pas évident de dire quel est le résultat de la proposition 2.13. Intuitivement nous pouvons croire que le codom φ^n ne sera jamais un sous-ensemble du dom φ^n.
Dans le prochain paragraphe nous verrons comment justifier cette intuition.

▲

En outre, la méthode proposée pour "semi-décider" de la propriété non-noethérienne peut être utilisée pour connaître l'ensemble de termes fermés $\complement f_R$ qui font que la règle R ne termine pas. Ainsi, dans le cas où

- g_R est noethérienne, évidemment $\complement f_R = \emptyset$ (prop. 3.1., 3.7.)

- g_R est non-noethérienne alors grâce au corollaire 2.12 et à la proposition 2.11. nous pouvons dire que

$$\exists n < \omega .\ g_R n.1 = g_R n+1.1 \wedge g_R n.1 \ : e_{n.1} \rightarrow e_{n.2} \complement g_R = \mathrm{dom}\ g_R n.1 = \nabla_{[e_{n.1}]}$$

En particulier pour l'ensemble

$$C = \{[e] \,/ [e] \in \nabla_{[e_{n.1}]} \wedge \mathrm{var}(e) = \emptyset \} \subseteq \complement g_R$$

La fonction g_R ne termine pas. Donc la fonction f_R ne termine pas avec les termes dans l'ensemble

$$\complement f_R = \{ e \ / \ [e] \in C\}$$

Par conséquent (prop 3.1, 3.7) la règle R ne termine pas pour les termes (fermés) dans $\complement f_R$.
Donc l'ensemble des termes fermés tels que la règle R ne termine pas est caractérisé par les termes fermés unifiables avec $e_{n.1}$

$$(\complement f_R = \{ e \ / \ [e] \in \nabla_{[e_{n-1}]} \wedge \mathrm{var}\ (e) = \emptyset \}).$$

La procédure 1 peut être modifiée pour donner aussi cette information.

Exemple 3.11. (suite)

▼

$$g_R : s(x,y,z,f(w)) \rightarrow s(f(w),f(x),y,z)$$

▲ $$\circlearrowleft f_R : \{ e \;/\; s(x,f(z),f(w),f(w)) \sqsubset e \wedge \text{var } e = \emptyset \}$$

Exemple 3.12. (suite)

▼

$$g_R : s(x,y,z,f(w)) \rightarrow s(f(w),x,y,z)$$

▲ $$\circlearrowleft f_R : \{ e \;/\; s(f(x),f(y),f(z),f(w)) \sqsubset e \wedge \text{var } e = \emptyset \}$$

Finalement, et par rapport à l'ensemble $\circlearrowleft f_R$, il est intéressant de signaler que dans le cas où les termes $e_{1.1}$ et $e_{1.2}$ (où $R = e_{1.1} \rightarrow e_{1.2}$) sont unifiables dans les termes l'unification $\Theta e_{1.1} = \Theta e_{1.2}$ ne caractérise pas l'ensemble $\circlearrowleft f_R$.

Exemple 3.12. (suite)

▼

$$g_R : e_{1.1} = s(x,y,z,f(w)) \rightarrow e_{1.2} = s(f(w),x,y,z)$$

$e_{1.1}$ et $e_{1.2}$ sont unifiables avec l'unificateur

$$\Theta = x|^{f(w)} \quad y|^{f(w)} \quad z|^{f(w)}$$

donc $\Theta e_{1.1} = \Theta e_{1.2} = s(f(w),f(w),f(w),f(w))$

et par contre

▲ $$\circlearrowleft f_R = \{ e \;/\; s(f(x),f(y),f(z),f(w)) \sqsubset e \wedge \text{var } e = \emptyset \}$$

3.2.3. Sur le calcul borné pour la suite $g_R^{1\cdot 1}$, $g_R^{2\cdot 1}$, ..., $g_R^{n\cdot 1}$, ...

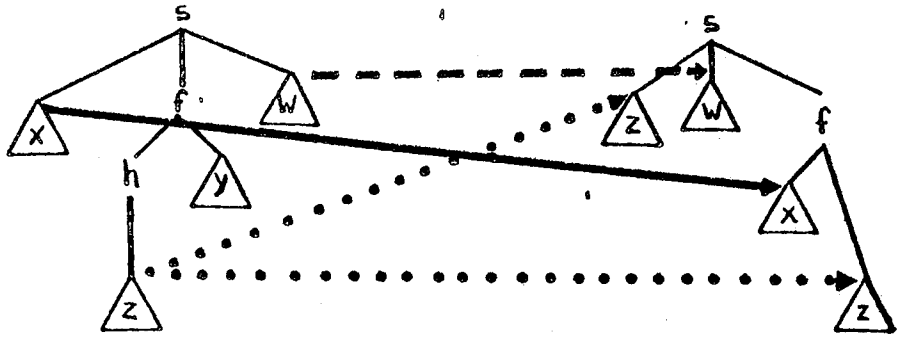
Nous montrons ici certaines idées, illustrées par des exemples, qui nous font penser que le calcul de la suite $g_R^{1\cdot 1}$, $g_R^{2\cdot 1}$, ..., $g_R^{n\cdot 1}$, ... peut être borné pour décider la propriété non-noethérienne. Nous avons déjà commencé les recherches pour formaliser ces idées et vérifier notre intuition. Nous espérons présenter dans l'avenir les résultats de telles recherches.

Nous pouvons considérer la réécriture définie par la règle $R = e_{1.1} \longrightarrow e_{1.2}$ comme un "déplacement" ou "disparition" des termes associés à chaque variable dans $e_{1.1}$ dans le terme $e_{1.2}$.

Exemple 3.14.

▼

$$R_1 = s(x,f(h(z),y),w) \longrightarrow s(z,w,f(x,z))$$

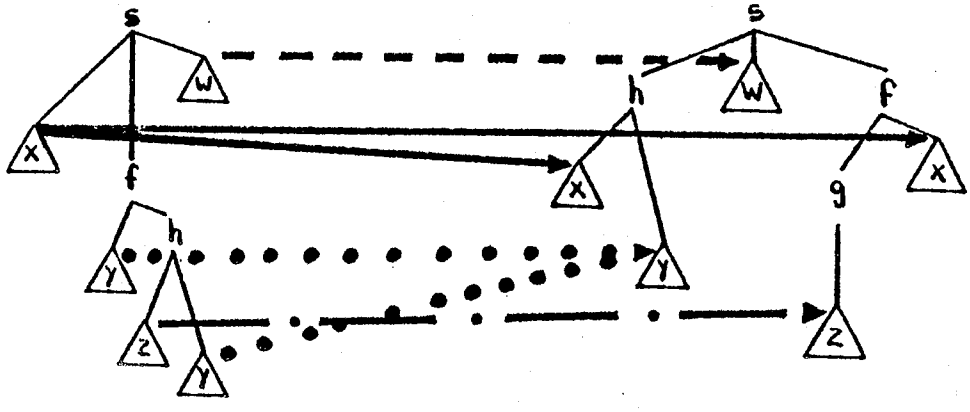


▲

Exemple 3.15.

▼

$$R = s(x,f(y,h(z,y)),w) \longrightarrow s(h(x,y),w,f(g(z),x))$$

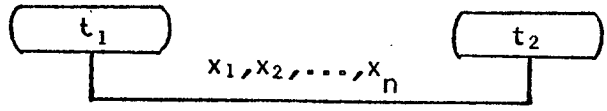


▲

Nous avons remarqué que le calcul de $\varphi_{g_R}^{n+1}$ peut être défini comme le calcul du terme $e_{n+1.1} = \sigma_n^1 \xi_n e_{1.1}$ et sa réécriture n+1 fois. Le but du calcul de $e_{n+1.1}$ est de caractériser la plus petite classe d'équivalence de termes qui peuvent être réécrits au moins n+1 fois. Alors, le déplacement a été réalisé dans chacune de ces n+1 réécritures.

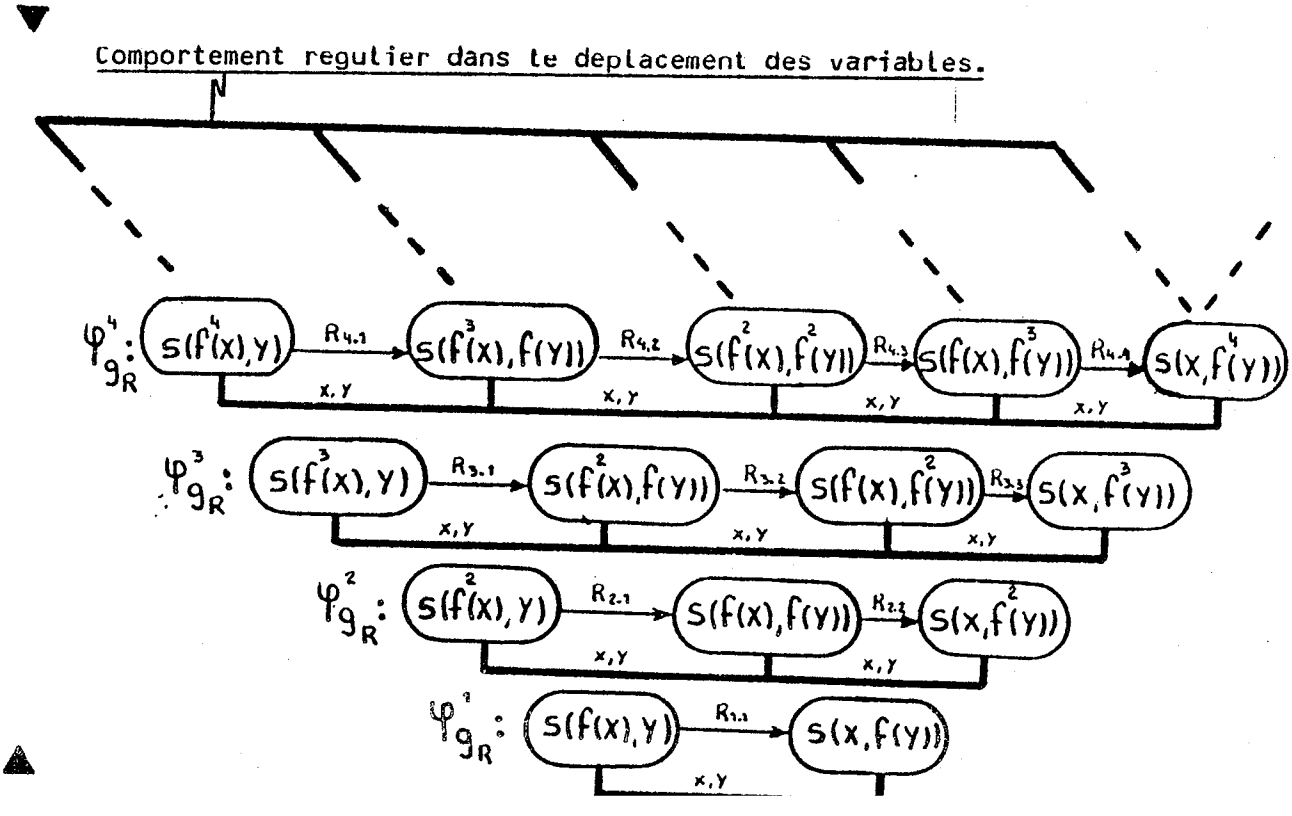
Donc, nous pouvons constater que les termes associés aux variables dans $e_{n.1}$ par σ_n^1 vont suivre le même déplacement (dans les n premières réécritures de $e_{n+1.1}$) qu'ont suivi les variables dans dom σ_n^1 dans les n réécritures de $e_{n.1}$.

Par ailleurs, et puisque le déplacement est toujours le même, nous pensons que pour un n suffisamment grand, on doit avoir une régularité dans le déplacement des variables dans $e_{n.1}$ au cours de ses n+1 réécritures. Dans les exemples qui suivent, cette régularité est limitée a l'apparition des variables dans les mêmes sous-termes d'une façon périodique. Pour montrer cela d'une façon plus visuelle, nous utilisons le schéma

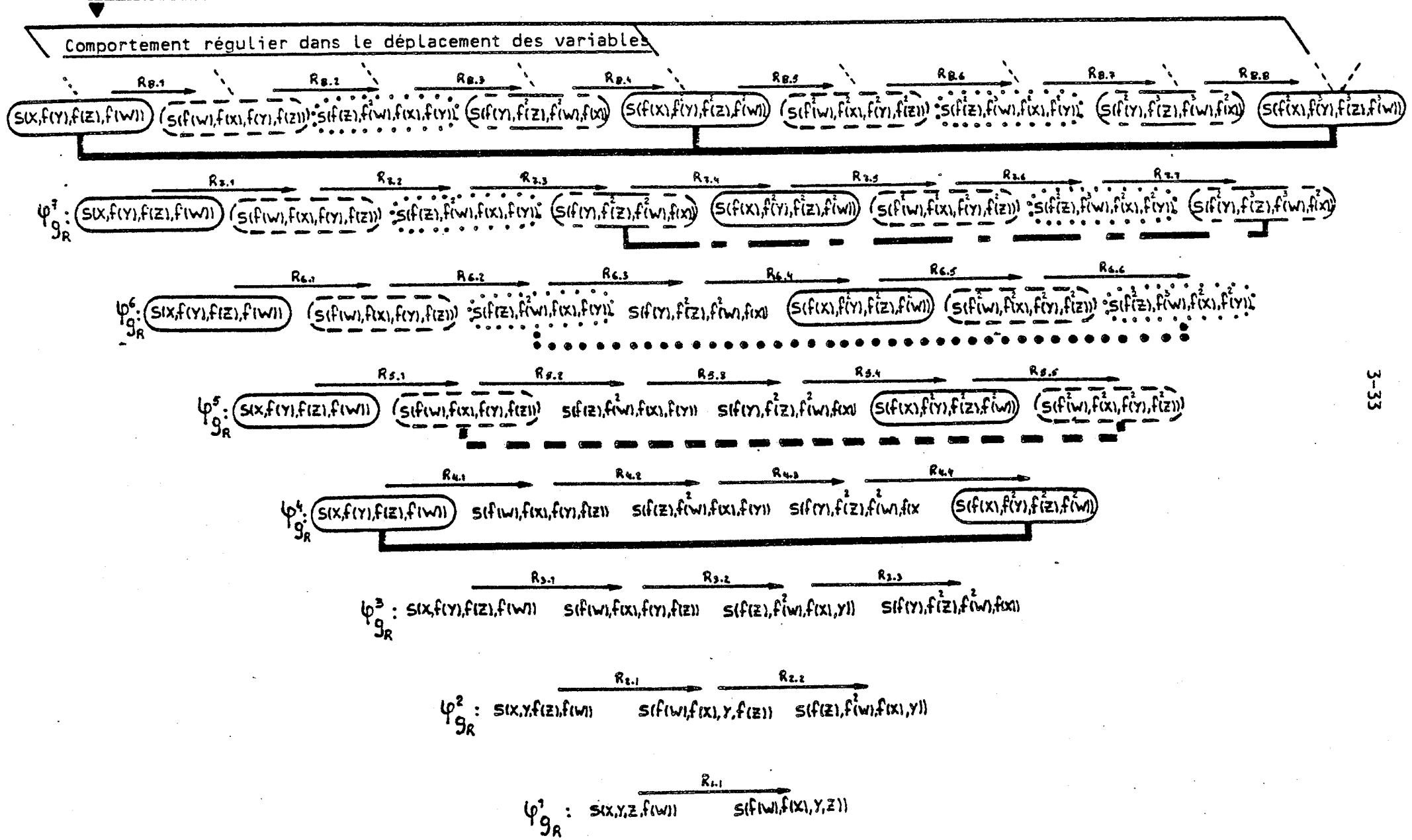


pour indiquer que les variables $x_1, x_2, \ldots, x_n$ dans t_1 apparaissent dans les mêmes sous-termes que dans t_2

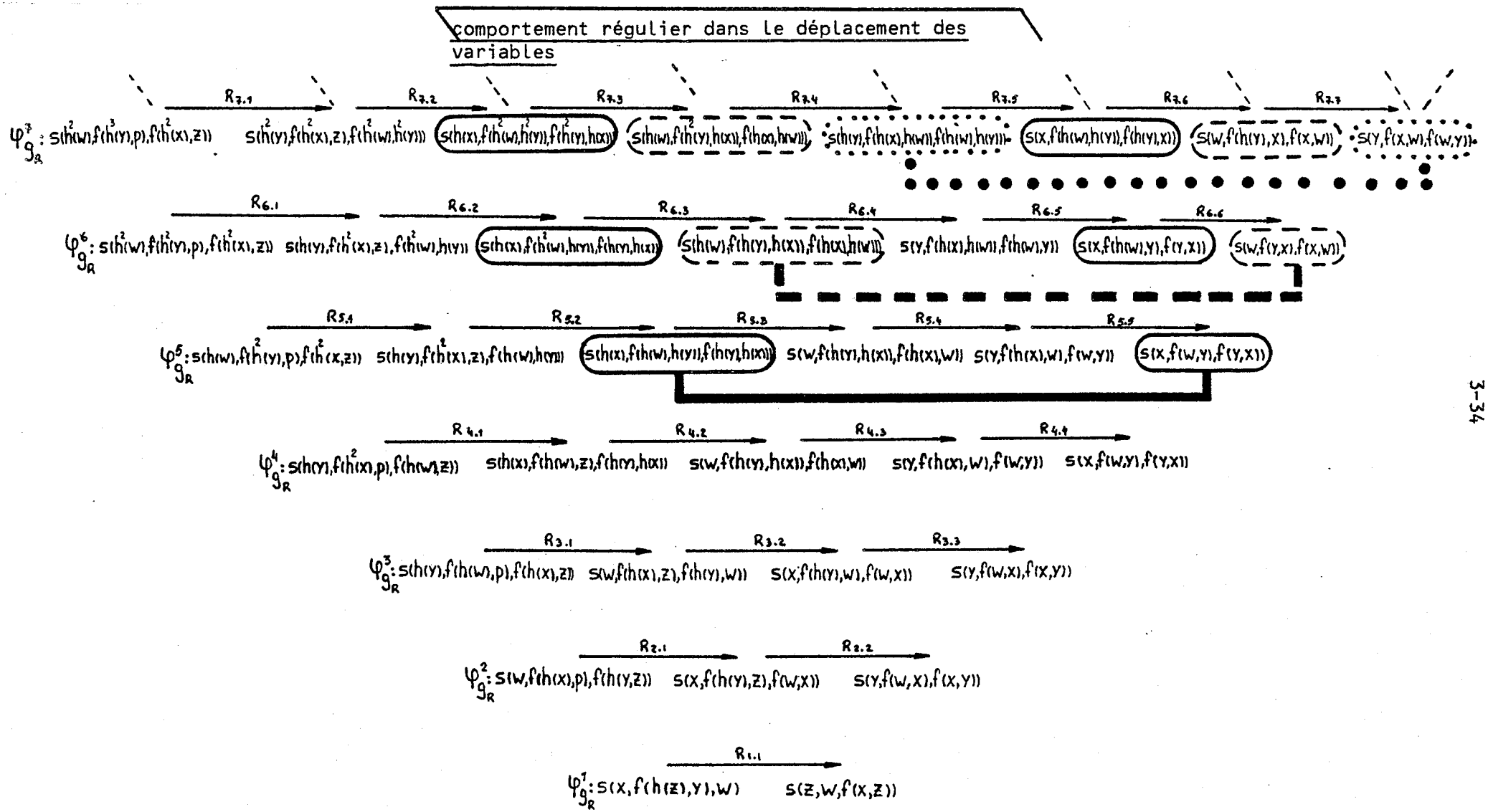
Exemple 3.10 (suite)



Exemple 3.11. (suite)



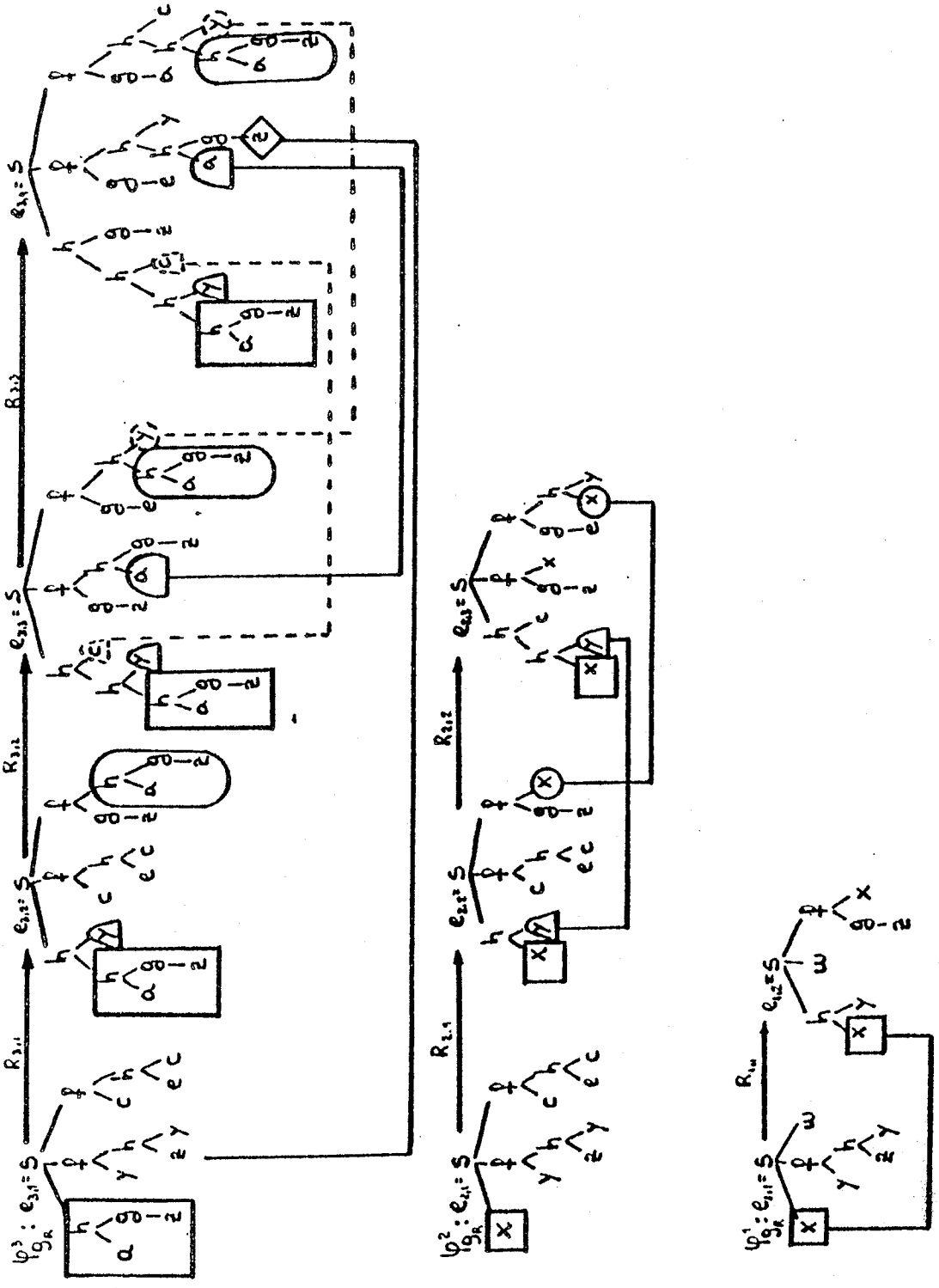
Exemple 3.13. (suite)



Les trois exemples précédents ont en commun le fait que toutes les variables ont la même périodicité d'apparition dans les sous-termes. Ils ont été choisis pour une meilleure compréhension de notre intuition. Néanmoins, dans le cas général, chaque variable a sa propre période. L'exemple suivant illustre cela

Exemple 3.15. (suite)

▼



▲

Cette "périodicité" de chaque variable d'apparition dans le même sous-terme est une propriété "en largeur", c-à-d le calcul

$$\varphi^{n}_{g_R} = g_R{}^{n.n} \circ \ldots \circ g_R{}^{n.1}$$

Par contre, les caractérisations de la propriété non-noethérienne démontrées dans le chapitre 2 sont en "hauteur", c-à-d le calcul

$$\varphi^{1}_{g_R}, \varphi^{2}_{g_R}, \ldots, \varphi^{n}_{g_R}, \ldots$$

ou

$$g_R{}^{1.1}, g_R{}^{2.1}, \ldots, g_R{}^{n.1}, \ldots$$

Néanmoins, nous pensons que cette "périodicité" induit une "régularité" dans le calcul en hauteur. Cette régularité porte sur les filtrages qui sont réalisés entre $e_{n.m}$ et $e_{n+1.m}$. Ces filtrages, pour la suite

$$e_{1.1}, \quad e_{2.1}, \ldots, e_{n_1.1} \ldots$$

sont finis dans le cas : g_R est non-noethérienne (voir prop. 2.14) ou g_R noethérienne ssi $\exists n < \omega \cdot \varphi^{n}_{g_R} = \Omega$. Dans les autres cas, les filtrages sont réguliers non-finis.

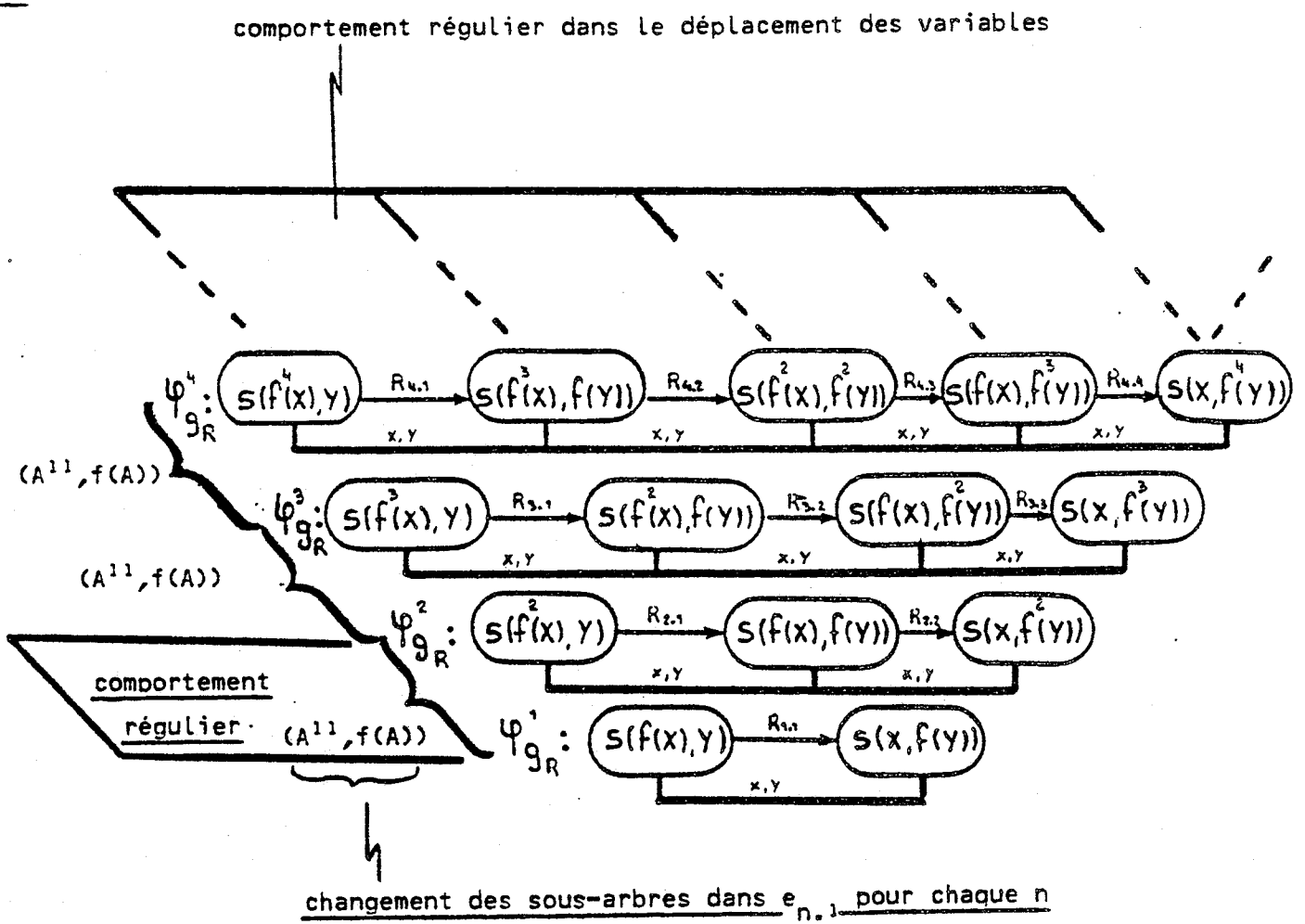
Une des grandes difficultés pour formaliser ces idées est le renommage de variables nécessaire dans le calcul en hauteur.

Nous montrons sur les exemples précédents le sens de cette régularité et, pour éviter le renommage des variables, nous considérons l'arbre correspondant associé au terme $e_{n.1}$ (voir annexe 1), et nous utilisons la notation :

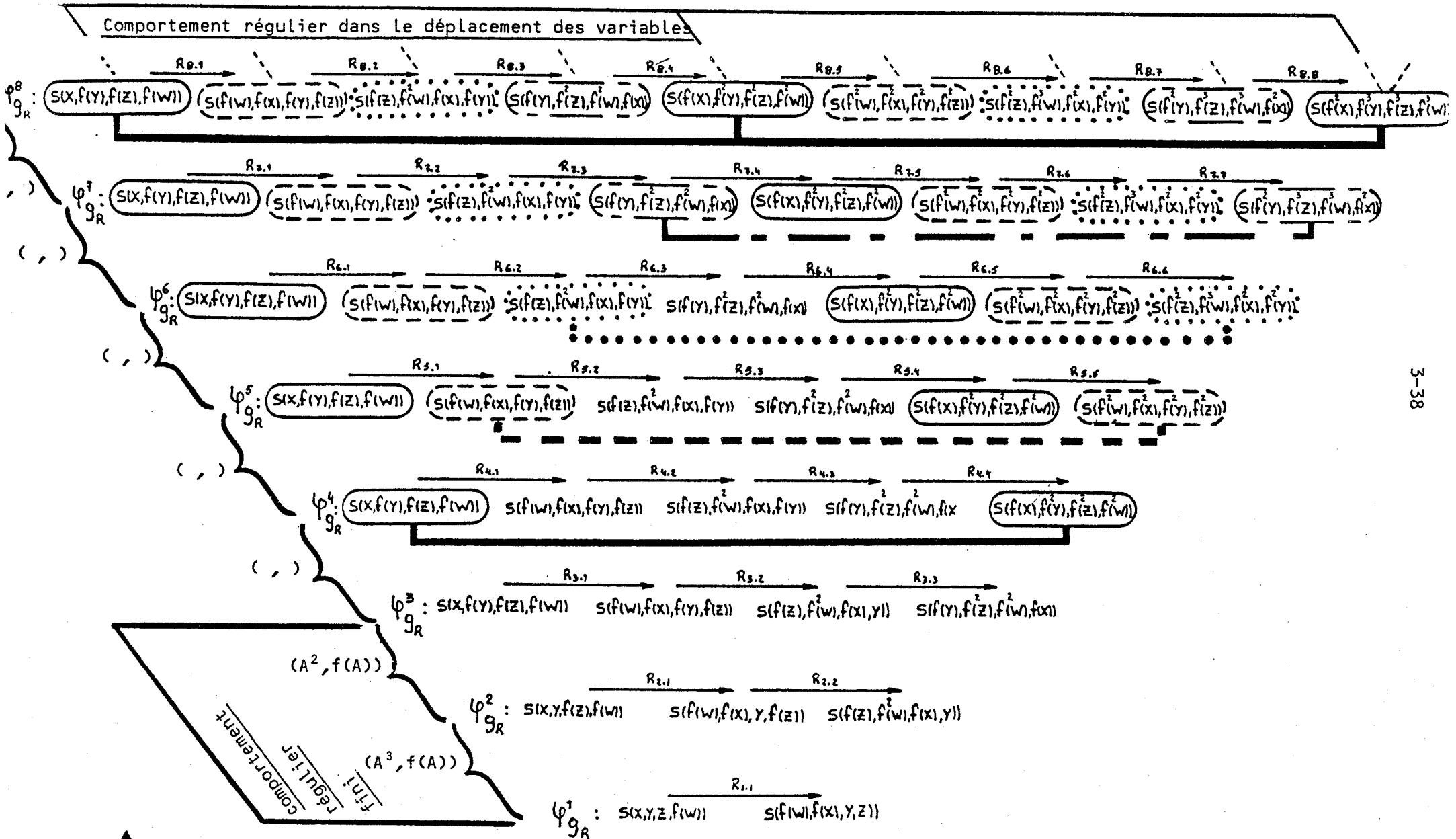
$$(A^{\alpha}, t) \quad \text{où} \quad t \in H_{F.V}$$

pour indiquer que dans l'arbre A accroché au chemin α, il y a une substitution d'un de ses sous-arbres par l'arbre t.

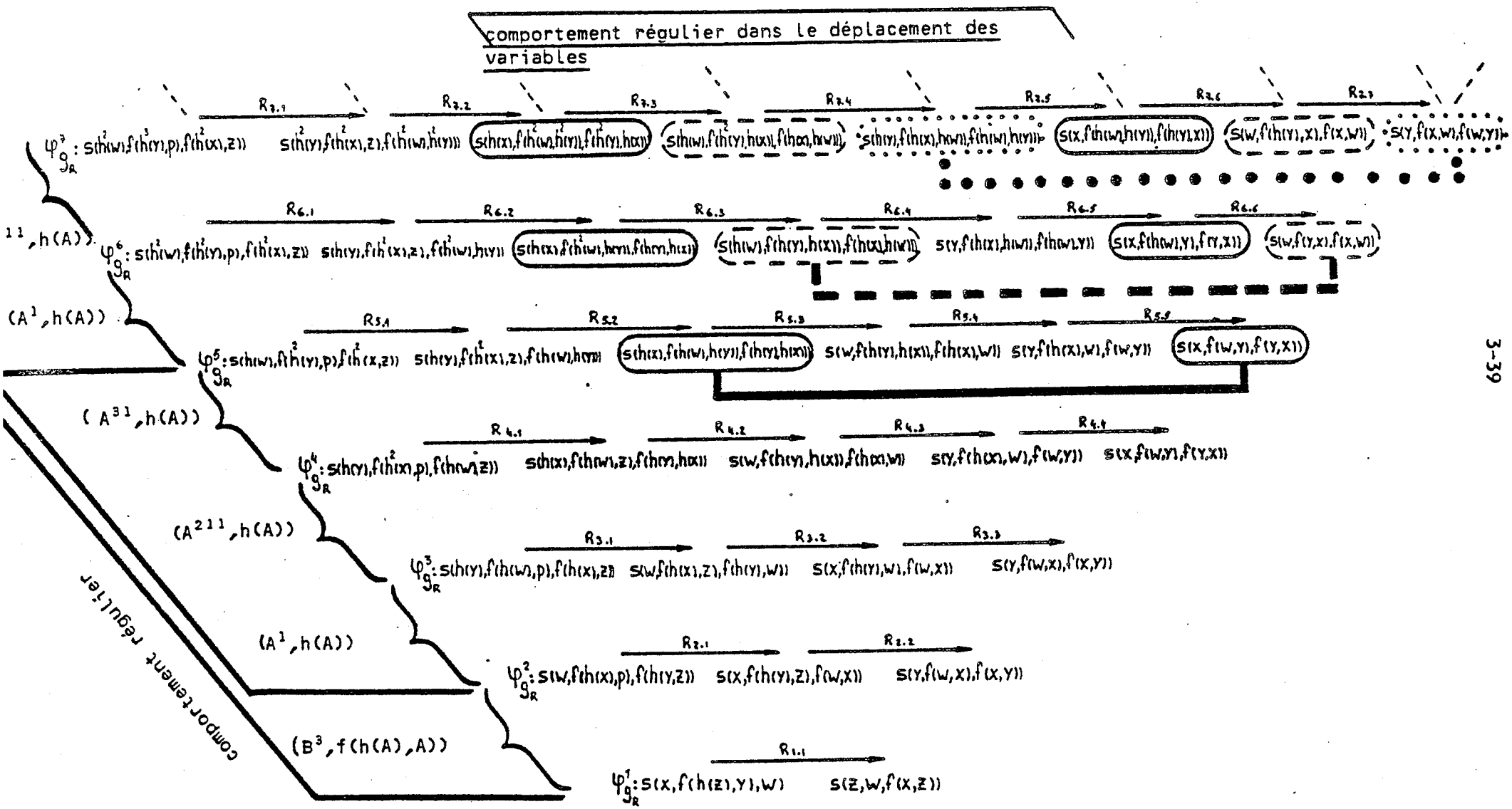
Exemple 3.10. (suite)



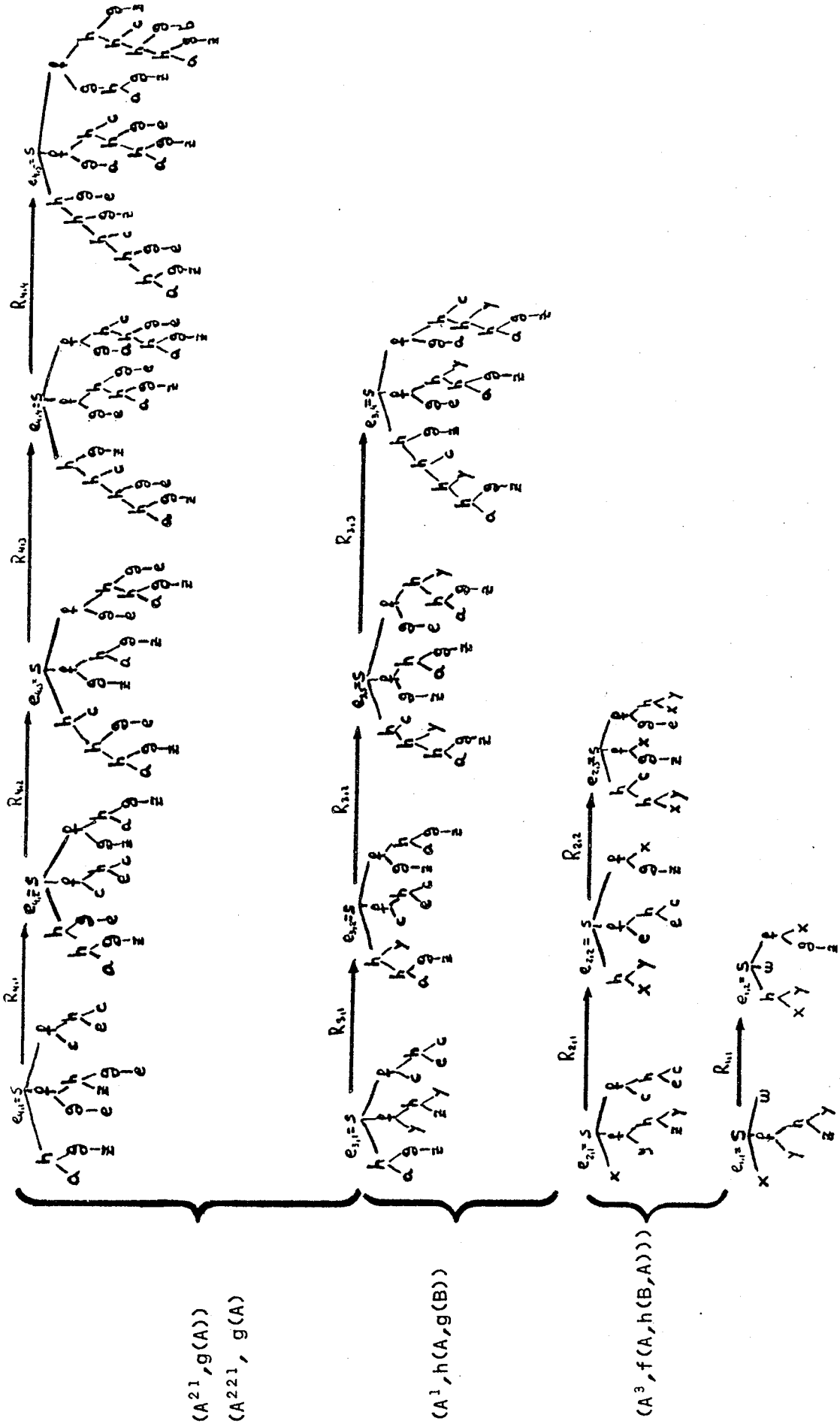
Exemple 3.11. (suite)



Exemple 3.13. (suite)



Exemple 3.15. (*suite*)



Nous avons déjà commencé à formaliser ces idées. Pour éviter des variables dans le calcul en hauteur et pour cerner la régularité du déplacement des variables, nous utilisons une extension de la réécriture aux arbres rationnels. Ainsi, nous travaillons sur la solution dans les arbres rationnels à l'équation $e_{1\cdot 1} = e_{1\cdot 2}$ où $R = e_{1\cdot 1} \to e_{1\cdot 2}$.

L'étude du déplacement sur cet arbre rationnel nous permettra de généraliser le calcul en "largeur". Nous donnons dans l'annexe , certains faits dans cette voie.

Par ailleurs, du point de vue strictement théorique, nous pensons que si cette régularité est confirmée, elle rejoindra les travaux de Koening [Koe 1884] et Julia [1918] sur les groupes circulaires limités et les travaux de Cosnard [Cos 83] sur l'ensemble des points d'accumulation dans la méthode des approximations successives. Mais, dans notre cas, nous aurons une généralisation de la circularité à la régularité. Autrement dit, les arbres supremums des suites

$\langle e_{n\cdot 1}, \trianglelefteq \rangle$ pour tout $n \geq 1$

$\langle e_{n\cdot 2}, \trianglelefteq \rangle$ pour tout $n \geq 1$

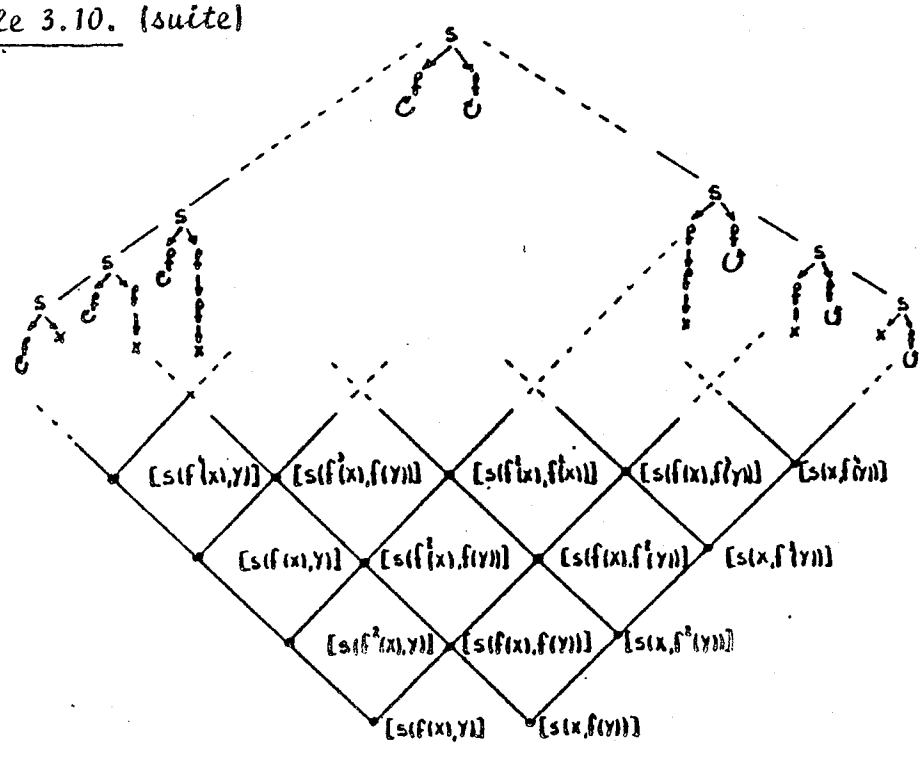
$\vdots$

$\langle en.m, \trianglelefteq \rangle$ pour tout $n \geq 1$

sont approximés d'une façon régulière.

Exemple 3.10. (suite)

▼



▲

3.3. LES REGLES DE TRANSITIONS DE TERMES (g → d) SANS RESTRICTION

Dans la section 3.2., nous avons défini la relation de réécriture engendrée par $R = e_{1.1} \rightarrow e_{1.2}$ comme

$\vdash_R \subseteq H_{\Sigma,F} \times H_{\Sigma,F}$ t.q $c \vdash_R c' \iff \exists \sigma \in$ Subs-fermés t.q. $c = \sigma e_{1.1} \wedge c' = e_{1.2}$

Or, dans le cas où $var(e_{1.2}) \not\subseteq var(e_{1.1})$, alors la réécriture engendrée par R ne peut pas être définie comme $\vdash_R$. Car la réécriture à travers R d'un terme ferme n'est pas en général un terme ferme.

Néanmoins, cette réécriture de règles sans restriction peut être définie sur des termes non-fermés. Autrement dit, nous pouvons définir la réécriture d'une règle $e \rightarrow e'$ sans restriction, par :

$$\Vdash_R \subseteq H_{\Sigma,F,\mathcal{V}} \times H_{\Sigma,F,\mathcal{V}}$$

$$e_1 \Vdash_R e_2 \iff \exists\ \sigma \in \text{Subs t.q.}\quad e_1 = \sigma e_{1.1} \wedge e_2 = \sigma e_{1.2}$$

Notre intérêt réside toujours en la connaissance de l'ensemble de termes fermés qui peuvent faire que l'application de la règle ne termine pas, c-à-d, même dans le cas où la règle est de la forme $var(e_{1.2}) \not\subseteq var(e_{1.1})$, nous voulons considérer le cas où les variables qui apparaissent dans la partie droite sans apparaître dans la partie gauche reçoivent des termes fermés avant d'être réécrits de nouveau. Nous pouvons illustrer ce fait de la façon suivante :

Soit $e_1 \in H_{\Sigma,F}$,

$e_1 \Vdash_R e_2$

$\downarrow \xi_1 : var(e_{1.2}) - var(e_{1.1}) \rightarrow H_F$

$\xi_1 e_2 \Vdash_R e_3$

$\downarrow \xi_2 : var(e_{1.2}) - var(e_{1.1}) \rightarrow H_F$

$\xi_2 e_4 \Vdash_R e_4$

$\vdots$

$e_n \in H_{\Sigma,F,\mathcal{V}}$ pour $n>1$

ξ_n sont fonctions totales et $\downarrow\xi_n$ indique l'affectation de termes fermés aux variables dans e' au pas n de la réécriture, donc

$\xi_n e_n \in H_{\Sigma,F}$, $n>1$.

Cette affectation de termes fermés, au fur et à mesure que l'on réalise la réécriture, peut être étudiée avec la méthode proposée dans le paragraphe précédent.

Ainsi, la proposition 3.8. caractérise d'abord le

codom $g_R{}^{n+1.1} = \nabla_{[\sigma_n e_{n.1} \leftarrow \sigma^1{}_n \xi_n e_{1.2}]}$

Ensuite, elle définit : le dom $g_R{}^{n+1.1}$ comme le filtre principal $\nabla_{[\sigma^1_n \xi_n e_{1.1}]}$ et la règle $R_{n+1.1} = \sigma^1_n \xi_n e_{1.1} \rightarrow \sigma^1_n \xi_n e_{1.2}$

comme une façon de représenter la fonction $g_R{}^{n.1}$.

La caractérisation du codom $g_R{}^{n.1}$ nous permet de connaître l'ensemble de valeurs possibles associées aux variables pour pouvoir appliquer de nouveau la fonction g_R sur l'ensemble de valeurs dans $\nabla_{[\sigma^1_n \xi_n e_{1.2}]}$

Or, le calcul de codom $g_R{}^{n+1} = \nabla_{[e_{1.2}]} \cap \nabla_{[e_{n.1}]}$ ne prend pas compte du fait que var $e_{1.2} \not\subseteq$ var $e_{1.1}$.
Donc, la caractérisation des variables nouvelles dans $e_{1.2}$ pour le calcul du codom $g_R{}^{n+1}$ indique l'ensemble de termes fermes qui peuvent être associés à ces variables pour pouvoir appliquer de nouveau la règle R.

Autrement dit, la composition itérée de g_R

$$\varphi^n_{g_R} : g_R \circ g_R \circ \ldots \circ g_R$$

reçoit dans chaque composition une information supplémentaire sur le résultat de la composition précédente. Plus exactement,

si $g^m_R(x) = y$ alors $g^{m+1}_R(x) = g(y')$ où y' est une restriction sur y.

Nous montrons, dans les exemples suivants, ces idées. Ces exemples sont très simples pour faire comprendre les idées.

Une formalisation plus rigoureuse s'impose pour définir le rapport entre g_R et la réécriture. Nous pensons que cette formalisation doit être abordée en même temps que l'étude de la régularité dans le calcul en hauteur.

Exemple 3.16. (si la variable nouvelle reçoit dans chaque itération un élément dans $\nabla_{[f(z)]}$ alors la fonction g_R est non-noethérienne).

▼

$$R_{1-1} : s(x,f(y)) \rightarrow s(f(x),z)$$

donc

$$\varphi^1_{R_{1-1}} = g_{R_{1-1}}$$

$$\varphi^2_{R_{1-1}} : s(X,f(Y)) \longrightarrow s(f(X),f(y)) \longrightarrow s(f^2(X),z)$$

$$\sigma_1 = x\,|^{f(X)} \qquad \sigma_1$$

$$\sigma_1' = z\,|^{f(y)} \qquad \varphi^1_{R_{1-1}} : s(x,f(y)) \longrightarrow s(f(x),z)$$

$$g_R : s(X,f(Y)) \rightarrow s(f(X),Z)$$

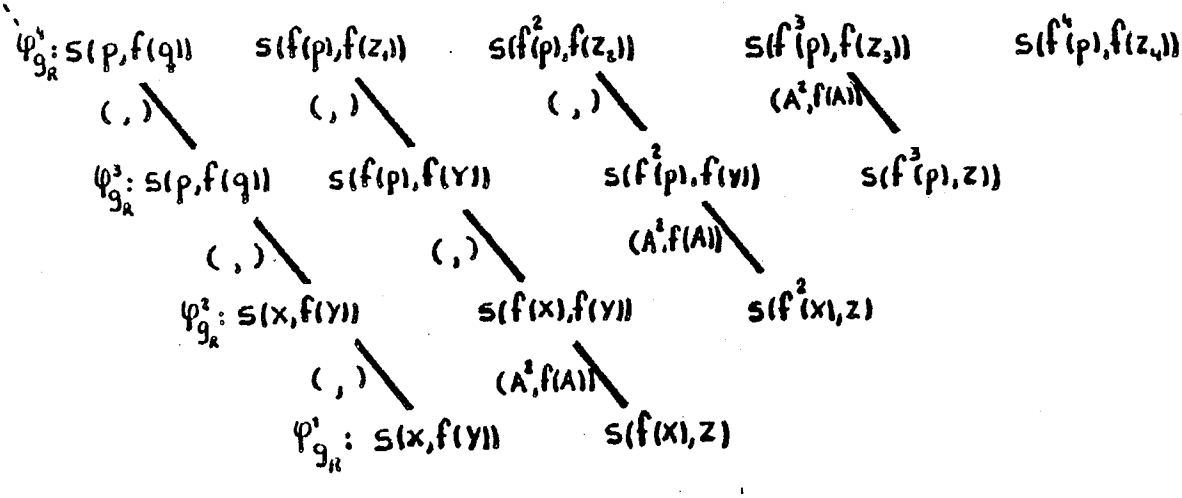
$$\varphi^3_{R_{1-1}} : s(p,f(q)) \longrightarrow s(f(p),f(Y)) \longrightarrow s(f^2(p),f(y) \longrightarrow s(f^3(p),z)$$

$$\sigma_2 = X\,|^{f(p)} \qquad \sigma_2 \qquad \sigma_2$$

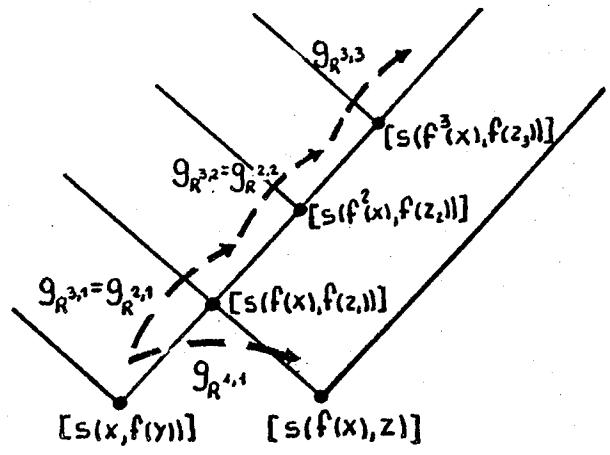
$$\sigma_2' = r\,|^{f(Y)} \qquad \varphi^2_R : s(X,f(Y)) \longrightarrow s(f(X),f(y) \longrightarrow s(f^2(X),z)$$

$$g_R : s(p,f(q)) \rightarrow s(f(p),r)$$

Noter que $g_R{}^{2-1} = g_R{}^{3-1}$



Par conséquent, on peut observer que, si dans chaque application itérée de la fonction g_R, la fonction g_R reçoit pour la nouvelle variable z une valeur dans $\nabla_{[f(z_1)]}$, alors la fonction g_R est non-noethérienne pour l'ensemble de valeurs $\nabla_{[s(x,f(y))]}$.



▲

Exemple 3.17. (Quelle que soit la valeur reçue par la nouvelle variable, la fonction g_R est non-noethérienne).

▼

$$R_{1\cdot1} = s(x,f(y)) \rightarrow s(f(z),f^2(y))$$

donc

$$\varphi^1_{R_{1\cdot1}} = g_{R_{1\cdot1}}$$

$$\varphi^2_{R_{1\cdot1}} : s(X,f(Y)) \rightarrow s(f(Z),f^2(Y)) \longrightarrow s(f(Z),f^3(Y))$$

$\sigma_1 = x \mid f(z)_y \mid f(Y)$; σ_1

$$\varphi^1_{R_{1\cdot1}} : s(x,f(y)) \rightarrow s(f(z),f^2(y))$$

$$g_R : s(X,f(Y)) \rightarrow s(f(Z),f^2(Y))$$

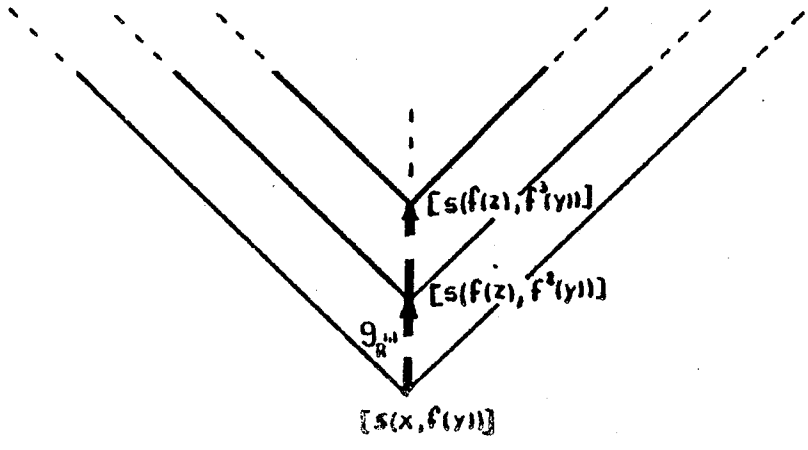
Noter que $g_R{}^{2\cdot1} = g_R{}^{1\cdot1}$

$$\varphi^3_R : s(x,f(y)) \rightarrow s(f(z_1),f^2(y)) \rightarrow s(f(z_2),f^3(y)) \rightarrow s(f(z_3),f^4(y))$$

$$\varphi^2_R : s(X_1f(Y)) \rightarrow s(f(z),f^2(Y)) \rightarrow s(f(z),f^3(Y))$$

$$\varphi^1_R : s(x_1f(y)) \rightarrow s(f(z),f^2(y))$$

Par conséquent, on peut observer que la valeur reçue par la nouvelle variable z dans chaque itération n'intervient pas dans la non-noethérienneté de g_R.



▲

Exemple 3.18. (Quelle que soit la valeur reçue pour la nouvelle variable, la fonction g_R est noethérienne). ▼

$$R_{1\cdot 1} = s(x,f(y)) \to s(f(z),y)$$

donc

$$\varphi^1 g_R = g_{R_{1\cdot 1}}$$

$$\varphi^2 g_R : s(X,f^2(y)) \to s(f(Z),f(y)) \to s(f(z),y)$$

$\sigma_1 = x \mid f(Z)$; σ_1 ; σ'_1 ; $\sigma'_1 = Y \mid f(y)$; $\varphi^1 g_R : s(x,f(y)) \to s(f(z),y)$

$$g_R : s(X,f(Y)) \to s(f(Z),Y)$$

$$\varphi^3 g_R : s(p,f^3(y)) \to s(f(r),f^2(y)) \to s(f(Z),f(y)) \to s(f(z),y)$$

$\sigma_2 = X \mid f(r)$; σ_2 ; σ_2 ; σ'_2 ; $\sigma'_2 = q \mid f^2(y)$; $\varphi^2 g_R : s(X,f^2(y)) \to s(f(Z),f(y)) \to s(f(z),y)$

$$g_R : s(p,f(q)) \to s(f(r),q)$$

$$\varphi^4 g_R : s(t,f^4(y)) \to s(f(v),f^3(y)) \to s(f(r),f^2(y)) \to s(f(Z),f(y)) \to s(f(z),y)$$

$\sigma_3 = p \mid f(v)$; σ_3 ; σ_3 ; σ_3 ; σ'_3 ; $\sigma'_1 = u \mid f^3(y)$; $\varphi^3 g_R : s(p,f^3(y)) \to s(f(r),f^2(y)) \to s(f(Z),f(y)) \to s(f(z),y)$

$$g_R : s(t,f(u)) \to s(f(v),u)$$

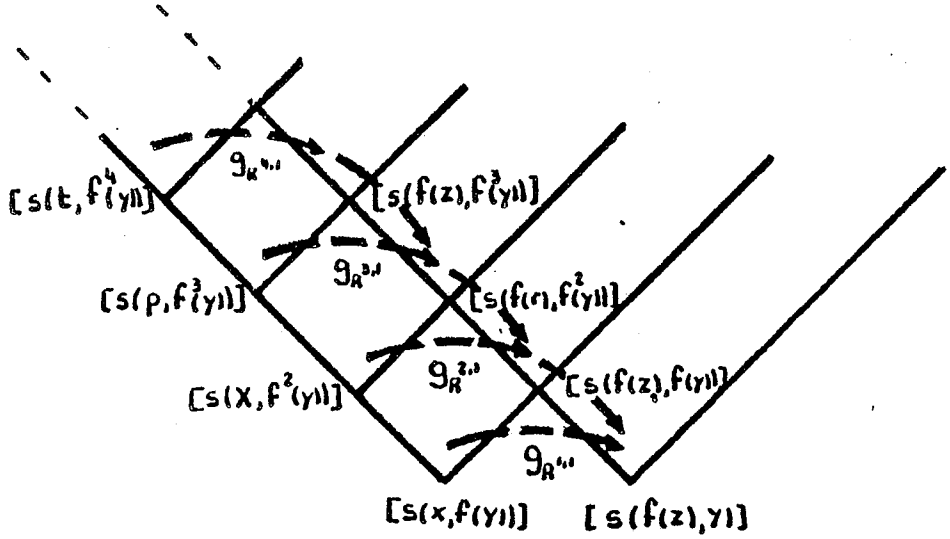
$$\varphi^{4}_{g_R} : s(t,f^{4}(y)) \to s(f(v),f^{3}(y) \to s(f(r),f^{2}(y) \to s(f(Z),f(y)) \to s(f(z),y)$$

$$\varphi^{3}_{g_R} : s(p,f^{3}(y)) \to s(f(r),f^{2}(y) \to s(f(Z),f(y)) \to s(f(z),y)$$

$$\varphi^{2}_{g_R} : s(X,f^{2}(y)) \to s(f(Z),f(y)) \to s(f(z),y)$$

$$\varphi^{1}_{g_R} : s(x,f(y)) \to s(f(z),y)$$

Remarquer que jusqu'à $\varphi^{4}_{g_R}$, nous n'avons pas trouvé que $g_R n.1 = g_R n+1.1$. Par contre, nous avons la régularité de $(A^2_{;}f(A))$. Donc, (puisque pendant cette régularité $\mathfrak{N}$ dom $g_R n.1 \not\subseteq \mathfrak{N}$ codom $g_R n.1$ (propo 2.17)), la fonction g_R doit être noethérienne pour n'importe quelle valeur de la variable nouvelle.



▲

Exemple 3.19.

▼

Ce dernier exemple montre un aspect très intéressant de l'étude de la réécriture. Ainsi, le terme fermé associé aux variables peut être une fonction des termes fermés réécrit auparavant.

$$R_{1.1} : s(x,f(y),x) \rightarrow s(f(z),f(x),y)$$

donc

$$\varphi^1_{g_R} = g_{R^{1.1}}$$

$$\varphi^2_{g_R} : s(y,f^2(Z),y) \rightarrow s(f(Z),f(y),f(Z)) \rightarrow s(f(z),f^2(Z),y)$$

$\sigma_1 = x|f(Z)$; σ_1

σ'_1 : $\sigma'_1 = y|f(Z)_X|y$; $\varphi^1_{g_R} : s(x,f(y),x) \rightarrow s(f(z),f(x),y)$

$g_R : s(X,f(Y),X) \rightarrow s(f(Z),f(X),Y)$

$$\varphi^3_{g_R} : s(f(Z),f^2(r),f(Z)) \rightarrow s(f(r),f^2(Z),f(r)) \rightarrow s(f(Z),f^2(r),f(Z) \rightarrow (s(f(z),f^2(Z),f(r))$$

$\sigma_2 = y|f(r)$; σ_2 ; σ_2

σ'_2 : $\sigma'_2 = p|f(Z)_q|f(r)$; $\varphi^2_{g_R} : s(y,f^2(Z),y) \rightarrow s(f(Z),f(y),f(Z)) \rightarrow s(f(z),f^2(Z),y)$

$g_R : s(p,f(q),p) \rightarrow s(f(r),f(p),q)$

$$\varphi^4_{g_R} : s(f(r),f^2(Z),f(r)) \rightarrow s(f(Z),f^2(r),f(Z)) \rightarrow s(f(r),f^2(Z),f(r)) \rightarrow s(f(Z),f^2(r),f(Z)) \rightarrow s(f(z),f^2(Z),f(r))$$

σ'_3 : $\sigma'_3 = t|f(r)_u|f(v)$; $\varphi^3_{g_R} : s(f(Z),f^2(r),f(Z)) \rightarrow s(f(r),f^2(Z),f(r)) \rightarrow s(f(Z),f^2(r),f(Z)) \rightarrow s(f(z),f^2(Z),f(r))$

$g_R : s(t,f(u),t) \rightarrow s(f(v),f(t),u)$

alors $g_R^{4\cdot 1} = g_R^{3\cdot 1}$

$$\varphi^4_{g_R} : s(f(r),f^2(Z),f(r)) \to s(f(Z),f^2(r),f(Z)) \to s(f(r),f^2(Z),f(r)) \to s(f(Z),f^2(r),f(Z)) \to s(f(z),f^2(Z),f(r))$$

$$(\ ,\ ) \qquad (\ ,\ ) \qquad (\ ,\ ) \qquad (\ ,\ )$$

$$\varphi^3_{g_R} : s(f(Z),f^2(r),f(Z)) \to s(f(r),f^2(Z),f(r)) \to s(f(Z),f^2(r),f(Z)) \to s(f(z),f^2(Z),f(r))$$

$$(A^1;f(A)) \quad (A^3;f(A)) \qquad (A^2;f(A)) \qquad (A^3;f(A^1))$$

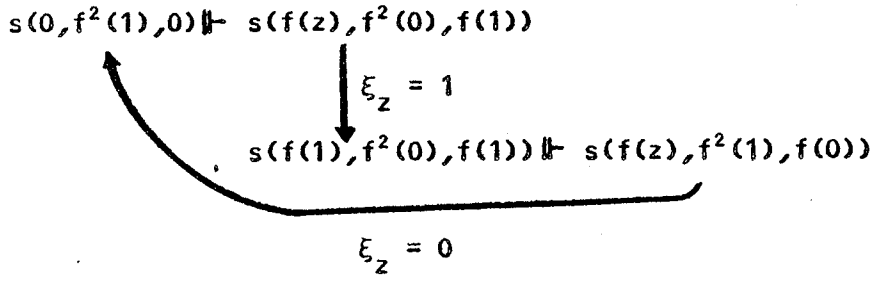
$$\varphi^2_{g_R} : s(y,f^2(Z),y) \to s(f(Z),f(y),f(Z)) \to s(f(z),f^2(Z),y)$$

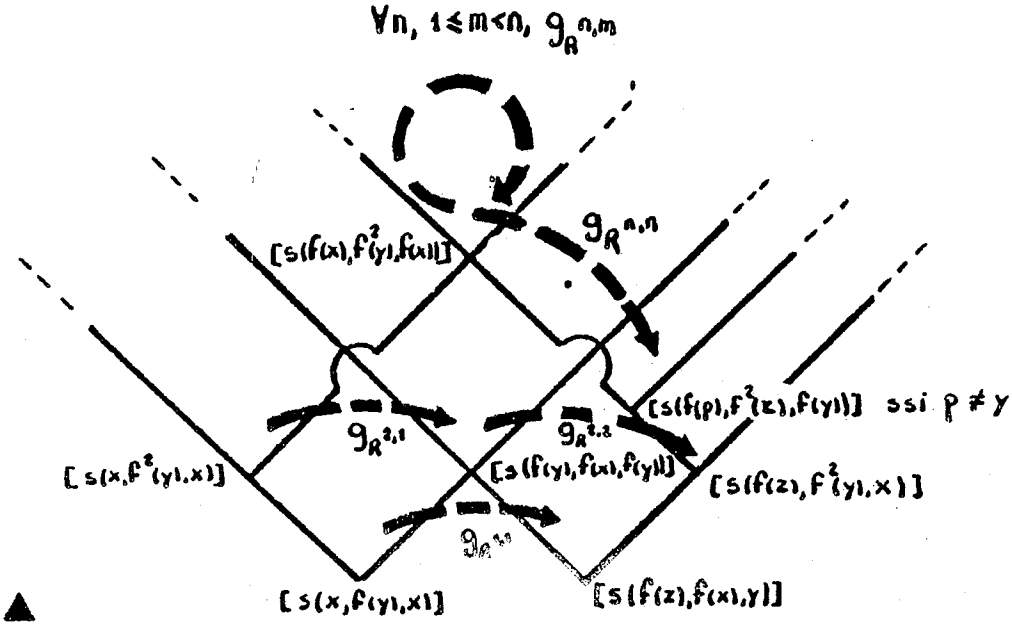
$$(A^2;f(A)) \qquad (A^3;f(A^1))$$

$$\varphi^1_{g_R} : s(x,f(y),x) \to s(f(z),f(x),y)$$

Par conséquent, nous pouvons observer que la fonction g_R est non-noethérienne pour l'ensemble de valeurs $\nabla_{[s(f(z),f^2(r),f(z))]}$.
Or, pour que la fonction g_R ne termine pas, elle doit recevoir dans chaque composition itérée pour la nouvelle variable z une valeur telle que $f(z)$ soit égale au troisième argument du calcul précédent.

Ainsi, par exemple :





▲

CONCLUSION

CONCLUSION

Nous avons donné dans le premier chapitre trois modèles pour la spécification de processus communicants. Ces modèles sont introduits de la même façon syntaxique et nous lui avons associé trois sens sémantiques différents :

- algébrique
- opérationnel
- arborescent (fini ou infini)

Ces trois conceptions différentes d'un même objet syntaxique permettra dans l'avenir :

- de faire une extension sur LPG (Langage de Programmation Générique) [Ber 79, 82a, 82b] pour spécifier un SPC, les preuves de correctitude dans le style de types abstraits de données [Lis-Zil 77]
- l'étude d'invariants, trajectoires, "deadlocks" et "livelocks", dans la même voie que les études réalisées sur les systèmes de transitions [Sif 80] .
- l'étude de la relation de l'observationalité [Mil 80] au niveau des comportements infinis dans les systèmes de processus.

Certains travaux sont déjà en cours dans ces voies.

Le langage formel proposé sert à exprimer toutes les fonctions calculables et il peut être utilisé comme un langage de spécification. Nous avons montré trois opérateurs au niveau de la construction de réseaux de processus qui communiquent. Ces trois opérateurs : union, connexion et restriction , permettent un degré d'abstraction de haut niveau dans le dessin du systèmes de processus.

La différence fondamentale avec les autres modèles de processus communicants [Mil 80] [Hoa 81-82] [Aus-Bou] (dans lesquels les systèmes de processus sont éléments d'une algèbre de processus) est que nos opérateurs sont simplement une manipulation syntaxique de description. Cela permet de ramener l'étude des propriétés d'un système de processus à l'étude d'un processus simple.

Nous avons aussi prouvé que la divergence est indécidable dans notre modèle. Néanmoins, et puisque la sémantique opérationnelle peut être interprétée au moyen d'un système de transitions de termes avec évènement (une sorte de système de réécriture), nous avons commencé une étude sur la décidabilité d'arrêt d'une règle de réécriture qui permettra dans l'avenir de décider de la divergence sur certaines classes de processus.

Cette même étude sur l'arrêt d'une règle de réécriture permettra de connaître la suite des valeurs qu'un processus peut recevoir dans l'évènement d'une règle et qui sont susceptibles de faire boucler le processus indéfiniment sur cette règle.

Dans l'étude de décidabilité d'arrêt d'une règle de transitions de termes , nous pensons avoir introduit une méthode originale dans l'étude de la propriété noethérienne. D'abord nous avons étudié la propriété noethérienne sur les fonctions définies sur un cpo filtré complet. Nous avons trouvé quatre propositions de caractérisation (conditions nécessaires et suffisantes) de cette propriété qui se révèle intéressante non seulement dans le contexte de la semi-décidabilité de la propriété non-noethérienne sur une règle de transitions de termes, mais aussi dans l'optimisation du calcul de cette propriété et dans la connaissance de l'ensemble des valeurs telles que la règle ne termine pas.

Nous pensons que cette propriété non-noethérienne sur les règles de transitions de termes est décidable. Nous pensons que cette décidabilité est fondée sur une régularité du déplacement des termes dans des réécritures successives à travers la même règle. Cette régularité nous fait penser que nous pouvons peut être généraliser les travaux de Koening [Koe 1884] , Julia [Ju 18] sur les groupes circulaires limités et les travaux de Cosnard [Cos 83] sur l'ensemble des points d'accumulation dans le méthode des approximations successives. Cette généralisation consistera à passer de la circularité à la régularité. Nous pensons que dans cette direction un bon travail peut être entrepris.

Finalement, une étude approfondie doit être réalisée pour connaître l'extension de ce travail de décidabilité d'arrêt d'une règle de transitions de termes à une règle quelconque et à un système de règles de transitions de termes.

ANNEXE

L'unification de termes dans les arbres infinis

Soient $\mathbb{S}$, $\mathbb{F}$, $\mathbb{V}$ comme dans 3.1.1.1. , $\mathbb{N}_+ = \mathbb{N} - \{0\}$ et ε le mot vide.

Un <u>arbre sur</u> $(\mathbb{S}, \mathbb{F}, \mathbb{V})$ est une fonction a où

$$dom(a) \subseteq \mathbb{N}_+^* \quad \wedge \quad codom(a) \subseteq \mathbb{S} \cup \mathbb{F} \cup \mathbb{V}$$

t.q.

i. $\forall \alpha,\beta \in \mathbb{N}_+^*.\ (\alpha\beta \in dom(a) \Rightarrow \alpha \in dom(a))$

ii. $\forall \alpha \in \mathbb{N}_+^*,\ l,j \in \mathbb{N},\ 1 \leq l \leq j.\ (\alpha j \in dom(a) \Rightarrow \alpha l \in dom(a))$

iii. $a(\varepsilon) \in \mathbb{S} \wedge (\forall \alpha \in \mathbb{N}_+^* .\ (\alpha \in dom(a) = a(\alpha) \in \mathbb{F} \cup \mathbb{V}))$

iv. $\forall \alpha.\ (a(\alpha) = g \wedge \rho(g) = K \Rightarrow \forall j \in \mathbb{N}_+,\ 1 \leq j \leq k,\ \alpha j \in dom(a))$

L'arbre a est fini, si $dom(a)$ est <u>fini</u>, sinon, il est <u>infini</u>. Puisque les arbres finis correspondent bijectivement aux termes $H_{\mathbb{S},\mathbb{F},\mathbb{V}}$ de manière évidente, nous parlerons indistinctement de termes ou d'arbres finis.

<u>Le sous-arbre</u> accroché au chemin $\alpha \in dom(a)$ est défini par

$$a|_\alpha = \lambda\beta \in \mathbb{N}_+^*.\ a(\alpha\beta)$$

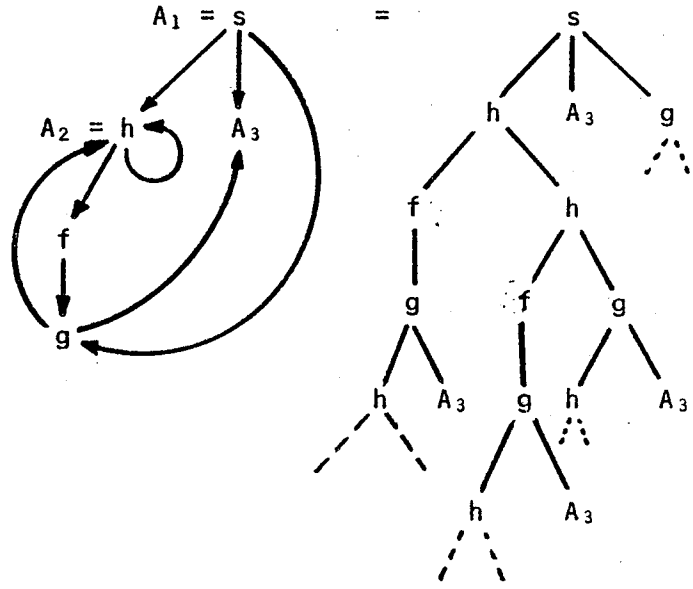
et on dénote par sous-arbres (a) l'ensemble $\{a|_\alpha \ / \ \alpha \in dom(a)\}$

Un arbre a est <u>rationnel</u> si l'ensemble sous-arbres (a) est fini. On dénote par $\mathbb{R}(\mathbb{S}, \mathbb{F}, \mathbb{V})$ l'ensemble des arbres rationnels sur $(\mathbb{S}, \mathbb{F}, \mathbb{V})$.

Les arbres rationnels sont représentés par des arbres bouclés et ils sont solution d'<u>un système d'équations simples</u> (voir [Cou 81], [Hue 76]), c-à-d, un système fini d'équations de la forme $S = \langle A_1 = u_1, \ldots, A_n = u_n \rangle$ où $A_j \in \mathbb{V}$ $1 \leq j \leq n$ et $u_j \in H_{\mathbb{S},\mathbb{F},\{A_1,\ldots,A_n\}} \cup H_{\mathbb{F},\{A_1,\ldots,A_n\}}$

Exemple 3.14.

$S = \langle A_1 = s(A_2, A_3, A_4),$
$A_2 = h(f(A_4), A_2),$
$A_4 = g(A_2, A_3) \rangle$



On appelle <u>équivalence rationnelle</u> toute équivalence $\sim$ sur $H_{\mathbb{S},\mathbb{F},\mathbb{V}} \cup H_{\mathbb{F},\mathbb{V}}$ qui vérifie les propriétés de :

1. *finitude* : $[x]_{\sim} = \{x\}$ presque partout dans $\mathbb{V}$
2. *simplification* : $g(i_1,\ldots,i_n) \sim g(i_1',\ldots,i_n') \Rightarrow i_j \sim i_j' \quad 1 \leqslant j \leqslant n$
3. *cohérence* : $g_1(i_1,\ldots,i_n) \not\sim g_2(i_1',\ldots,i_m')$ pour $g_1 \neq g_2$

Enfin, une <u>greffe</u> est une fonction γ où $\mathrm{dom}(\gamma) = \mathbb{V}$
et $\mathrm{codom}(\gamma) \subseteq \mathbb{R}(\mathbb{S}, \mathbb{F}, \mathbb{V})$

On définit <u>un préordre $\leqslant$ sur les greffes</u> par $\gamma \leqslant \gamma' \iff$
$\exists \delta.\ \gamma' = \delta\gamma$ (voir [Hue 76])

Théorème 3.15.

Soient $R = e \to e'$
$\sim_R = \{[e,e']\}$
$\sim$ la fermeture simplifiable de $\sim_R$

Si $\sim$ est cohérente, alors il existe une greffe γ t.q. l'équation $e = e'$ a une solution minimale dans $\mathbb{R}(\Sigma,F,V)$. Sinon, $e = e'$ n'a pas de solution dans les arbres infinis.

Preuve :
■

conséquence du théorème 17 de la thèse de Huet [Hue 76].
■

Nous donnons, au moyen d'exemples, la façon de trouver la solution dans les arbres de l'équation $e = e'$. L'idée est d'obtenir, à partir de la fermeture simplifiable et cohérente de $\sim_R\{[e,e']\}$, une greffe γ définissant un système d'équations simples S_1. Ce système d'équations est étendu au système d'équations simples S par une équation de la forme $T = \gamma e$ où T est une variable qui n'apparaît pas dans S_1. Le nouveau système d'équations S définit l'arbre rationnel.

Exemple 3.13. (suite)

$R = (e = s(x,f(h(z),y),w) \rightarrow e' = s(z,w,f(x,z)))$

$\sim_R = \{[e,e']\}$

la fermeture simplifiable de $\sim_R$ contient les classes suivantes :

c_1. $s(x,f(h(z),y),w) \sim s(z,w,f(x,z))$

c_2. $x \sim z \sim y \sim h(z)$

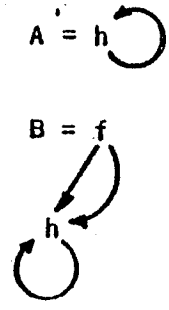
c_3. $w \sim f(h(z),y) \sim f(x,z)$

$\sim$ est cohérente, donc la greffe γ est

$\gamma_x = \gamma_z = \gamma_y = A$ où $A = h(A)$ c-à-d $A = h$

$\gamma_w = B$ où $B = h(A,A)$ $B = f$

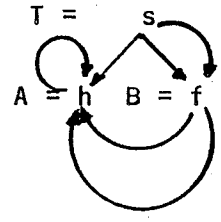
alors $S_1 = \langle A = h(A),\ B = h(A,A)$



et la solution de l'équation e = e' est :

S = <T = s(A,B,B),
A = h(A),
B = f(A,A) >

c-à-d



Exemple 3.9. (*suite*)

s(x,x,y) = s(f(x),y,x)

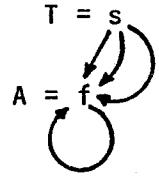
c_1. s(x,x,y) ~ s(f(x),y,x)

c_2. x ~ f(x) ~ y $\quad \gamma_x = \gamma_y = A$ où $S_1 = <A = H(A)>$

donc, la solution est l'arbre

S = <T = s(A,A,A),
A = h(A) >

c-à-d



Exemple 3.10. (*suite*)

s(f(x),y) = s(x,f(y))

c_1. s(f(x),y) ~ s(x,f(y))

c_2. x ~ f(x) $\quad \gamma_x = A$

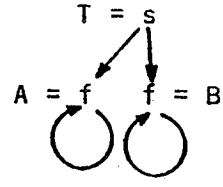
c_3. y ~ f(y) $\quad \gamma_y = B$

où $S_1 = <A = f(A),\ B = f(B)>$

donc, la solution est l'arbre

S = <T = s(A,B),
A = f(A),
B = f(B) >

c-à-d

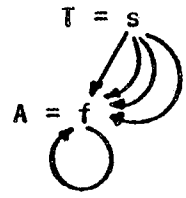


Exemple 3.11. (suite)

$s(x,y,z,f(w)) = s(f(w),f(x),y,z)$ a comme solution

$S = \langle T = s(A,A,A,A),$
$A = f(A) \rangle$

c-à-d

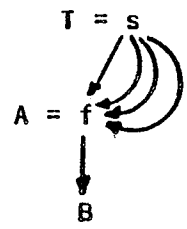


Exemple 3.12. (suite)

$s(x,y,z,f(w)) = s(f(w),x,y,z)$ a comme solution

$S = \langle T = s(A,A,A,A),$
$A = f(B) \rangle$

c-à-d



Un algorithme appelé RATIO permettant de calculer cette greffe, solution à l'équation $e = e'$, est proposé par Huet [Hue 76]

Sur l'arbre solution de l'équation $e_{1.1} = e_{1.2}$

Nous notons $\gamma e_{1.1}$ l'arbre solution, s'il existe, de l'équation $e_{1.1} = e_{1.2}$.

Nous définissons, comme dans le cas des termes, un préordre $\leqslant$ de filtrage et une relation de réécriture sur les arbres, de la façon suivante :

$a \leqslant a' \iff \exists \gamma.\ a' = \gamma a$

$a \underset{R}{\vdash} a' \iff \exists \gamma.\ a = \gamma e_{1.1} \wedge a' = \gamma e_{1.2}$

Donc, il est clair que la réécriture de l'arbre solution produit la même solution ($\gamma e_{1.1} = \gamma e_{1.2}$), c-à-d que la suite $\langle \gamma e_{1.1}, \vdash^* \rangle$ est formée par le même arbre solution. Toutefois, nous pouvons étudier sur cette suite $\langle \gamma e_{1.1}, \vdash^* \rangle$, comment les arbres composants de $\gamma e_{1.1}$ se "déplacent". Cette étude permettra d'obtenir une généralisation du calcul en largeur.

Par ailleurs, nous définissons comment associer, à la suite $g_{R^{1.1}}, \ldots, g_{R^{n.1}}, \ldots$ une suite de règles $R_{1.1}, R_{2.1}, \ldots, R_{n.1}, \ldots$ t.q. si $g_{R^{n.1}} = \Omega$ alors $g_{R^{n.1}} = g_{R_{n.1}}$. Nous cherchons à établir un rapport entre cette suite de règles et l'arbre rationnel, s'il existe, solution de l'équation $e_{1.1} = e_{1.2}$ où $R_{1.1} = e_{1.1} \rightarrow e_{1.2}$. La proposition suivante établir ce rapport et montre l'importance de travailler sur l'arbre solution comme une abstraction de règles $R_{n.m}$.

Proposition A1

Soit $R_{1.1} = e_{1.1} \rightarrow e_{1.2}$

$\forall n > 1.\ 1 \leqslant m \leqslant n \quad g_{R^{n.m}} = g_{R_{n.m}}$ où $R_{n.m} = e_{n.m} \rightarrow e_{n.m+1}$

S'il existe une solution de l'équation $e_{1.1} = e_{1.2}$

alors l'équation $e_{n.m} = e_{n.m+1}$ a la même solution

Preúve :
■

Soient γ_1 la greffe solution à $e_{1.1} = e_{1.2}$
$n.m$ t.q. $n > 1$ et $1 \leqslant m \leqslant n$

d'après le *théorème 3.13.* si l'équation $e_{1.1} = e_{1.2}$ a une solution, c'est l'arbre solution (fini ou infini). De plus, cette solution est la solution minimum.

D'autre part, la suite $\langle \gamma e_{1.1}, \vdash^* \rangle$ est infinie et constante.

Mais $\varphi_{g_R}^{n.m} = g_R n.m$ (où $R^{n.m} = e_{n.m} \rightarrow e_{n.m+1}$), ce qui signifie que $\forall a_{\bar{e}}.a_{e_{n.m}} \leqslant a_{\bar{e}}$ (où $a_{\bar{e}}$, $a_{e_{n.m}}$ sont les arbres associés aux termes $\bar{e}, e_{n.m}$).

L'arbre $a_{\bar{e}}$ peut être réécrit grâce à $\Vdash_R$ au moins n-m+1 fois, donc $a_{\bar{e}_{n.m+1}} \leqslant a_{\gamma e_{1.1}}$

Enfin, $g_R n.m$ est une restriction de $g_R^{1.1}$, donc, puisque γe est la solution minimum de $e_{1.1} = e_{1.2}$, alors γe est aussi la solution minimum $e_{n.m} = e_{n.m+1}$

■ c.q.f.d.

BIBLIOGRAPHIE

[Abr 83] J.R. Abrial
"A Practical Approach to the Analysis of Concurrents Systems"
STL/SERC Workshop on the Analysis of Concurrents Systems
August 1983.

[ADJ 78] J.A. Goguen - J.W. Thatcher - E.G. Wagner
"An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types"
"Currendt Trends in Programming Methodology" (vol IV, chap V)
Ed. : Raymond T. Yeh - Prentice-Hall Inc. 1978.

[Ark 84] E. Arkaxhiu
"Un Langage et un Environnement Graphique pour la Spécification des Processus en Parallèles Communicants"
Thèse de 3ème Cycle (à apparaître). Université de Grenoble.

[Arn-Niv 78] A. Arnold - M. Nivat
"The Metric Space of Infinite Trees - Algebraic and Topological Properties"
R.R. 323, IRIA Laboria - September 1978.

[Aus-Bou 82] D. Austry - G. Boudol
"Algèbre de Processus et Synchronisation"
A paraître dans TCS.

[Ber 79] D. Bert
"La Programmation Générique"
Thèse d'Etat (Juin 1979). Université de Grenoble.

[Ber 82a] D. Bert
"Refinements of Generic Specifications with Algebraic Tools"
R.R. 335, IMAG 1982.

[Ber 82b] D. Bert
"Generic Programming : A Tool for Designing Universal Operator Application to Program Algebra"
R.R. 336, IMAG 1982.

[Bea-Niv 81] J. Beauquier - M. Nivat
"Application of Formal Langage Theory to Problems of Security and Synchronisation"
International Summer School Theoretical Foundations of Programming Methodology
Munich 1981.

[Bur-San 81] S. Burris - H.P. Sankappanavar
"A Course in Universal Algebra"
Graduate Texts in Mathematics Springer-Verlag 78

[Bur-Gog 80] R.M. Burstall - J.A. Goguen
"An Informal Introduction to Specifications Using CLEAR"
International Summer School Theoretical Foundations of Programming Methodology
Munich 1981.

[Cha-Kei 73] C.C. Chang - H.J. Keisler
"Model Theory"
North-Holland Publishing Company 1973.

[Com 82] G. Comyn
"Objets Infinis Calculables"
Thèse d'Etat (Mars 1982). Université de Lille.

[Com-Dau 82] G. Comyn - M. Dauchet
"Approximations of Infinitary Objects"
ICALP 82, Denmark - Automates Langages and Programming
LNCS 140

[Cos 83] M. Cosnard
"Contributions à l'étude du Comportement Itératif des Transformations Unidimensionnelles"
Thèse d'Etat (Mars 1983). Université de Grenoble.

[Cou-Kah-Vui73] B. Courcelle - G. Kahn - J. Vuillemin
"Algorithmes d'Equivalence et de Réduction à des Expressions Minimales dans une Classe d'Equations Récursives Simples"
R.R. 337, IRIA Laboria - Novembre 1973.

[Cou 81] B. Courcelle
"Fundamental Properties of Infinite Trees"
International Summer School Theoretical Foundations of Programming Methodology
Munich 1981

[Cou 78] P. Cousot
"Methodes Itératives de Construction et d'Approximation de Points Fixes d'Opérateurs Monotones sur un Treillis, Analyse Sémantique des Programmes"
Thèse d'Etat (Mars 1978) - Université de Grenoble.

[Dar 82] Ph. Darondeau
"An Enlarged Definition and Complete Axiomatization of Observational Congruence of Finite Processes"
International Symposium on Programming - 5th Colloquium
Turin 1982 LNCS 137.

[Dav 58] M. Davis
"Computability & Unsolvability"
Mc-Graw-Hill Series in Information Processing and Computers, 1958.

[Der-Man 79] N. Dershowitz - Z. Manna
"Proving Termination with Multiset Orderings"
C.A.C.M. (Vol. 22, N° 8), 1979.

[Der 82] N. Dershowitz
"Orderings for Term-Rewriting Systems"
Theoretical Computer Science (vol. 17, N° 3), 1982.

[Dub-Dub 61] P. Dubreil - M.L. Dubreil-Jacotin
"Leçons d'Algèbre Moderne"
Dunod-Paris, 1961. Collection Universitaire de Mathématiques

[Dug 66] J. Dugundji
"Topology"
Allyn and Bacon, Inc 1966.

[Fer 83] J.C. Fernandez
"Structures Mathématiques pour l'Etude du Parallélisme"
R.R. 390, IMAG 1983.

[Gie 80] G. Gierz - K.H. Hofamn - K. Keimel - J.D. Lawson
M. Mislove - D.D. Scott
"A Compendium of Continuos Lattices"
Springer - Verlag 1980.

[Grä 78] G. Grätzer
"General Lattice Theory"
Academic Press 1978.

[Hen-Mil 80] M. Hennessy - R. Milner
"On Observing Non-Determinism and Concurrency"
7th ICALP Conference 1980, LNCS 74

[Hen-Plo 80] M.C.B. Hennessy - G.D. Plotkin
"A Term Model for CCS"
Mathematical Foundations of Computer Science, LNCS 88 1980.

[Hoa 78] C.A.R. Hoare
"Communicating Sequential Processes"
C.A.C.M. (vol. 21, N° 8), 1978.

[Hoa-Bro-Ros 81] C.A.R. Hoare - S.D. Brookes - A.W. Roscoe
"A Theory of Communicating Sequential Processes"
PRG Monograph N° 16, 1981 - Oxford University.

[Hoa 81a] C.A.R. Hoare
"A model for Communicating Sequential Processes"
PRG Monograph N° 22, 1981 - Oxford University.

[Hoa 81b] C.A.R. Hoare
"A Calculus of Total Correctness for Communicating Processes"
PRG Monograph N° 23, 1981 - Oxford University.

[Hue 76] G. Huet
"Résolution d'Equations dans des Langages d'ordre 1,2,...,ω"
Thèse d'Etat (Septembre 1976). Université de Paris VII.

[Hue-Lan 78] G. Huet - D.S. Lankford
"On the Uniform Halting Problem for Term-Rewriting Systems"
R.R. 359, INRIA Laboria - March 1978.

[Hue 80] G. Huet
"Confluent Reductions : Abstract Properties and Applications to Term Rewriting Systems"
JACM (vol. 27, N° 4), 1980.

[Jor 81a] Ph. Jorrand
"Bases for the Specification of Communicating Processes"
ICS 81, London - April 1981.

[Jor-Tur 81] Ph. Jorrand - F. Turini
"On the Properties of Networks of Communicating Processes"
Non publié.

[Jor 81b] Ph. Jorrand
"Specification of Communicating Processes and Process Implementation Correctness"
International Symposium on Programming.
5th Colloquium, Turin - April 1982 Proceedings LNCS 137.

[Jor 83] Ph. Jorrand
"Projet SPARC"
Non publié.

[Jor 84a] Ph. Jorrand
"Communicating Processes as Term Algèbres"
A paraître.

[Jor 84b] Ph. Jorrand
"FP2 : Functionnal Parallel Programming Based on Term Substitution"
A paraître.

[Jou-Les-Rei 81] J.P. Jouannaud - P. Lescanne - F. Reinig
"Recursive Decomposition Ordering"
A paraître.

[Jou-Kir 81] J.P. Jouannaud - H. Kirchner
"Construction d'un Plus Petit Ordre de Simplification"
Rapport Greco N° 20 - Université de Bordeaux I.

[Jul 18] G. Julia
"Mémoire sur Itération des Fonctions Rationnelles"
J. Math. Pures Appl. 4,1 (1918).

[Kel 76] R.M. Keller
"Formal Verification of Parallel Programs"
CACM (vol. 19, N° 7), 1976.

[Koe 1884] M.G. Koenigs
"Certaines Equations Fonctionnelles"
Annales de l'Ecole Normale Supérieure, 1884
Supplément N^{os} 2-24-25.

[Lis-Zil 77] B. Liskov - S. Zilles
"An Introduction to Formal Specifications of Data Abstractions"
Currends Trends in Programming Methodology
(vol. I, Software Specification and Design)
Ed. R.T. Yeh Prentice-Hall, Inc.

[Man 74] Z. Manna
"Mathematical Theory of Computation"
Mc Graw-Hill Kogakusha ltd, 1974.

[Mar 76] G. Markowsky
"Chain-Complete Posets and Directed Sets with Applications"
Algebra Universalis 6, 1976.

[Mar-Mon 82] A. Martelli - U. Montanari
"An Efficient Unification Algorithm"
ACM Transactions on Programming Languages and Systems
(vol. 4, N° 2), 1982.

[Mil 80] R. Milner
"A calculus of Communicating Systems"
LNCS 92, 1980.

[Mil 81] R. Milner
"A modal Characterisation of Observable Machine-Behaviour"
Non publié.

[Min 72] M. Minsky
"Computation : Finite and Infinite Machines"
Prentice-Hall International Inc 1972
Series in Automatic Computation.

[Moz 83] M.A. Mozota
"Certaines Méthodes de Preuve de Terminaison des Systèmes de Réécriture""
R.R. 362, IMAG 1983.

[Myc-Tay 76] J. Mycielski - W. Taylor
"A Compactification of the Algebra of Terms"
Algebra Universalis 6, 1976.

[Pre-Yeh 73] F.P. Preparata - R.T. Yeh
"Introduction to Discrete Structures for Computer Science and Engineering"
Addison - Wesley Publishing Company, 1973.

[Ras-Sik 63] H. Rasiowa - R. Sikorski
"The Mathematics of Metamathematics"
Polska Akademia Nauk - Monographie Matematyczne TOM 41, 1963

[Rob 63] A. Robinson
"Introduction to Model Theory and to the Metamathematics of Algebra"
Studies in Logic and the Foundations of Mathematics
North-Holland Publishing Company, 1963.

[Rob 65] J.A. Robinson
"A Machine-Oriented Logic Based on the Resolution Principle"
JACM 12, 1965.

[Rob 71] J.A. Robinson
"Computational Logic : The Unification Computation"
Machine Intelligence 6, 1971.

[Rob 79] J.A. Robinson
"Logic : form and function"
The mechanization of deductive reasoning
University Press, Edinburgh 1979.

[Rou-Bro 81] W.C. Rounds - S.D. Brookes
"Possibles Futures, Acceptances, Refusals and Communicating Processes"
Proceedings of 22nd Annual IEEE Symposium on Foundations of Computer Science, 1981.

[Sco 76] D. Scott
"Data Types as Lattices"
SIAM J. Comp. (vol. 5, N° 3), 1976.

[Sco 81] D. Scott
"Lectures on a Mathematical Theory of Computation"
PRG Monograph N° 19, 1981 - Oxford University.

[Sif 82] J. Sifakis
"An Unified Approach for Studying the Properties of Transition Systems"
TCS 18 (1982)

[Sol 82] R. Soler
"Une Approche de la Théorie de D. Scott et Application à la Sémantique des Types Abstraits Génériques"
Thèse de 3ème Cycle (Septembre 1982) - Université de Grenoble

[Zil 79] S. Zilles
"An Introduction to data algebras"
Abstract Software Specifications
1979, Copenhagen Winter School Proceedings - LNCS 86.

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapport de présentation de Monsieur Ph. JORRAND

Monsieur PEREIRA-FERNANDEZ Juan Manuel

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 22 mai 1984

Le Président de l'I.N.P.-G

D. BLOCH

P.O. le Vice-Président,