50376
1990
259

LIFL U.A 369 du C.N.R.S

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

N° d'ordre: 647

THESE de DOCTORAT
en
INFORMATIQUE

présentée par Rémi Gilleron
à l'Université de Lille Flandres Artois

RECONNAISSABILITE et FRAGMENTS DECIDABLES EN REECRITURE

AUTOMATES A PILES et CALCULS DE FORMES NORMALES

Soutenue le 12 Décembre 1990 devant la commission d'examen

Président:	M.Nivat
Rapporteurs:	H.Kirchner
	J.Berstel
Examinateurs:	H.Comon
	M.Dauchet
	M.Latteux
	S.Tison

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M.H. LEFEBVRE, M. PARREAU.

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

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

PROFESSEUR EMERITE

M. A. LEBRUN

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

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

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

M. A. DUBRULLE.

PROFESSEURS - CLASSE EXCEPTIONNELLE

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

PROFESSEURS - 1ère CLASSE

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

M. BRIDOUX Michel	Chimie-Physique
M. CELET Paul	Géologie Générale
M. CHAMLEY Hervé	Géotechnique
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. DAUCHET Max	Informatique
M. DEBOURSE Jean-Pierre	Gestion des Entreprises
M. DHAINAUT André	Biologie Animale
M. DOUKHAN Jean-Claude	Physique du Solide
M. DYMENT Arthur	Mécanique
M. ESCAIG Bertrand	Physique du Solide
M. FAURE Robert	Mécanique
M. FOCT Jacques	Métallurgie
M. FRONTIER Serge	Ecologie Numérique
M. GRANELLE Jean-Jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. LABLACHE-COMBIER Alain	Chimie Organique
M. LACOSTE Louis	Biologie Végétale
M. LAVEINE Jean-Pierre	Paléontologie
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique Atomique et Moléculaire
M. LEROY Jean-Marie	Spectrochimie
M. LHOMME Jean	Chimie Organique Biologique
M. LOMBARD Jacques	Sociologie
M. LOUCHEUX Claude	Chimie Physique
M. LUCQUIN Michel	Chimie Physique
M. MACKE Bruno	Physique Moléculaire et Rayonnements Atmosph.
M. MIGEON Michel	E.U.D.I.L.
M. PAQUET Jacques	Géologie Générale
M. PETIT Francis	Chimie Organique
M. POUZET Pierre	Modélisation - calcul Scientifique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. SALMER Georges	Electronique
M. SCHAMPS Joel	Spectroscopie Moléculaire
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
Melle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. TILLIEU Jacques	Physique Théorique
M. TOULOTTE Jean-Marc	Automatique
M. VIDAL Pierre	Automatique
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ALLAMANDO Etienne	Composants Electroniques
M. ANDRIES Jean-Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BART André	Biologie animale
M. BASSERY Louis	Génie des Procédés et Réactions Chimiques

Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique Atomique et Moléculaire
M. BERTRAND Hugues	Sciences Economiques et Sociales
M. BERZIN Robert	Analyse
M.BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie Végétale
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie Civil
M. BOIVIN Jean-Claude	Spectroscopie
M. BOUQUELET Stéphane	Biologie Appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BRASSELET Jean-Paul	Géométrie et Topologie
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie Animale
M. CATTEAU Jean-Pierre	Chimie Organique
M. CAYATTE Jean-Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie Structurale
M. CHIVE Maurice	Composants Electroniques Optiques
M. COMYN Gérard	Informatique Théorique
M. COQUERY Jean-Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques et Sociales
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique Nucléaire et Corpusculaire
M. COUTURIER Daniel	Chimie Organique
M. CRAMPON Norbert	Tectolique Géodynamique
M. CROSNIER Yves	Electronique
M. CURGY Jean-Jacques	Biologie
Melle DACHARRY Monique	Géographie
M. DEBRABANT Pierre	Géologie Appliquée
M. DEGAUQUE Pierre	Electronique
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELAHAYE Jean-Paul	Informatique
M. DELORME Pierre	Physioliogie Animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
M. DENEL Jacques	Informatique
M. DE PARIS Jean Claude	Analyse
M. DEPREZ Gilbert	Physique du Solide - Cristallographie
M. DERIEUX Jean-Claude	Microbiologie
Melle DESSAUX Odile	Spectroscopie de la réactivité Chimique
M. DEVRAINNE Pierre	Chimie Minérale
Mme DHAINAUT Nicole	Biologie Animale
M. DHAMELINCOURT Paul	Chimie Physique
M. DORMARD Serge	Sciences Economiques
M. DUBOIS Henri	Spectroscopie Hertzienne
M. DUBRULLE Alain	Spectroscopie Hertzienne
M. DUBUS Jean-Paul	Spectrométrie des Solides
M. DUPONT Christophe	Vie de la firme (I.A.E.)
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FAUQUAMBERGUE Renaud	Composants électroniques

M. FONTAINE Hubert	Dynamique des cristaux
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie Sturcturale
M. GAMBLIN André	Géographie urbaine, industrielle et démog.
M. GLORIEUX Pierre	Physique moléculaire et rayonnements Atmos.
M. GOBLOT Rémi	Algèbre
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie Physique
M. GOURIEROUX Christian	Probabilités et Statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean-Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie Organique
M. GUILBAULT Pierre	Physiologie animale
M. HENRY Jean-Pierre	Génie Mécanique
M. HERMAN Maurice	Physique spatiale
M. HOUDART René	Physique atomique
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et Statistiques
M. Jean Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme (I.A.E.)
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KREMBEL Jean	Biochimie
M. LANGRAND Claude	Probabilités et statistiques
M. LATTEUX Michel	Informatique
Mme LECLERCQ Ginette	Catalyse
M. LEFEBVRE Jacques	Physique
M. LEFEBVRE Christian	Pétrologie
Melle LEGRAND Denise	Algèbre
Melle LEGRAND Solange	Algèbre
M. LEGRAND Pierre	Chimie
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne
M. LE MAROIS Henri	Vie de la firme (I.A.E.)
M. LEROY Yves	Composants électroniques
M. LESENNE Jacques	Systèmes électroniques
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique théorique
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU Jean-Marie	Optique-Physique atomique
M. MAIZIERES Christian	Automatique
M. MAURISSON Patrick	Sciences Economiques et Sociales
M. MESMACQUE Gérard	Génie Mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie Organique
M. MORTREUX André	Chimie Organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
Mme MOUYART-TASSIN Annie Françoise	Informatique
M. NICOLE Jacques	Spectrochimie
M. NOTELET Françis	Systèmes électroniques
M. PARSY Fernand	Mécanique

M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée
M. STEEN Jean-Pierre	Informatique

Je remercie Maurice Nivat de l'honneur qu'il me fait en présidant ce jury.

Je suis reconnaissant à Hélène Kirchner et Jean Berstel d'avoir accepté d'être rapporteurs de cette thèse. Je les remercie pour les remarques, suggestions et critiques qu'ils ont pu faire.

Je remercie Hubert Comon d'avoir accepté de faire partie de ce jury.

Je remercie Michel Latteux et Max Dauchet pour m'avoir accordé leur confiance en DEA puis en thèse.

Mes remerciements vont tout particulièrement à Max Dauchet pour sa passion communicative de la recherche, sa compétence et son imagination et à Sophie Tison pour l'intérêt constant qu'elle a porté à mes travaux.

J'exprime, bien sûr, ma reconnaissance à Jean-Luc Coquidé pour son étroite collaboration.

Je tiens, enfin, à exprimer ma gratitude à tous ceux qui m'ont apporté aide et soutien au cours de l'élaboration de cette thèse et, en particulier, aux membres du 332 et à tous mes collègues du département informatique de l'IUT.

Ce qui, apparemment, ne sert jamais à rien pour les uns peut,
effectivement, servir toujours à tout pour les autres.

Pierre Dac

A Catherine,
A Martin, Justine et Clément.

Avant Propos

* * * * * * * *

Une partie de ce travail a été réalisée en commun avec J.L.Coquidé et l'approche globale est la même dans les deux thèses.

La structure de l'introduction de cette thèse reflète cette collaboration. En effet, la partie générale de l'introduction et la partie relative au travail commun sur les automates à piles sont les mêmes dans les deux thèses. Par contre, les parties relatives à l'étude de fragments décidables en réécriture et à l'étude de certaines classes de systèmes de réécriture sont propres à cette thèse.

1 INTRODUCTION

La réécriture est, à plusieurs titres, un paradigme bien connu de la programmation. En programmation fonctionnelle (comme LISP), un interprète peut être vu naturellement comme un système de réécriture; en programmation équationnelle (comme OBJ), la réécriture permet de calculer des représentants canoniques de types spécifiés par des équations; en programmation logico-fonctionnelle, elle intervient dans l'unification modulo par surréduction. Notre approche se situe en amont de la programmation. Notre but est de contribuer à mieux comprendre les structures algébriques complexes manipulées. Nous participons, en celà, à un courant récent de recherche (citons, parmi d'autres, R.V.Book et J.H.Gallier aux USA, M.Oyamaguchi au Japon, F.Otto et R.Treinen en Allemagne, K.Salomaa en Finlande, H.Comon en France,...).

Nous commencerons, néanmoins, par introduire la notion de réécriture à travers la problématique classique des théories équationnelles avant de passer à l'aspect algébrique et de présenter nos propres résultats.

****

L'idée de simplifier des axiomes équationnels est partout présente en algèbre. La réécriture formalise cette idée en remplaçant des égaux par des égaux plus simples.

Dans le cadre des *théories équationnelles*, le résultat le plus célèbre, qui est une des bases du développement actuel de la réécriture, est le *théorème de Knuth et Bendix* ([KNBE70]). Il s'agit de vérifier si une égalité u=v donnée est conséquence d'un ensemble E d'équations données (axiomes). Pour celà, on oriente, selon un certain critère, les équations de E en règles de réécriture. On calcule alors les paires critiques, obtenues en réécrivant de deux façons différentes, qui se chevauchent, un membre gauche de règle. Si une paire critique n'est pas convergente, c'est-à-dire si les deux termes se réduisent en deux termes irréductibles distincts, une nouvelle règle est ajoutée, en orientant si possible la paire critique selon le critère préalablement choisi. Cette construction est récursive. Le succès de cet algorithme nécessite la propriété de terminaison du système de réécriture.

On obtient par cette procédure, lorsqu'elle converge, un système de réécriture convergent S (tout terme possède une forme normale unique pour S) qui *fournit un algorithme de décision pour la théorie équationnelle d'ensemble d'axiomes E.*

Il suffit, en effet, pour prouver qu'une égalité u=v est conséquence des équations de E, de vérifier que les formes normales de u et v pour S sont égales.

Un exemple algébrique classique est:
à partir du système d'équations (E) suivant,

$0 + x = x$;

$(-x) + x = 0$;

$(x + y) + z = x + (y + z)$;

on oriente les règles de gauche à droite (en utilisant une notion de poids d'un arbre que nous ne détaillons pas ici) et on obtient le système de réécriture convergent constitué des trois règles précédentes et des règles,

$(-x) + (x + y) \rightarrow y$;

$x + 0 \rightarrow x$;

$-0 \rightarrow 0$;

$-(-x) \rightarrow x$;

$x + (-x) \rightarrow 0$;

$x + ((-x) + y) \rightarrow y$;

$-(x + y) \rightarrow (-y) + (-x)$.

Cet algorithme fait émerger l'importance des propriétés de terminaison et de confluence en réécriture.

Un système de réécriture est *noethérien* (termine) s'il n'existe pas de calcul infini. La terminaison est indécidable même dans le cas où les termes apparaissant dans les règles sont des arbres filiformes ([HULA78]) ou dans le cas où le système est composé d'une seule règle linéaire à gauche ([DAUC89]). Le problème de l'arrêt d'un système de réécriture de mots composé d'une seule règle reste ouvert. Une étude de synthèse recouvrant les problèmes de terminaison se trouve dans [DERS87].

Un système de réécriture est *confluent* si, pour toute réécriture d'un terme de deux façons différentes, on peut réécrire ces deux termes en un même terme. La confluence est en général indécidable, même dans le cas où les termes apparaissant dans les règles sont des arbres filiformes ([BJW81]); elle est décidable pour les systèmes de réécriture noethériens ([KNBE70], grâce au théorème de Newman [NEWM42])), et aussi pour les systèmes de réécriture clos (sans variables) même sans l'hypothèse de terminaison ([DATI85]).

Les algorithmes de complétion ont été largement étudiés (L.Bachmair et N.Dershowitz [BADE86], N.Dershowitz [DERS89] pour des articles de synthèse) et implantés (REVE [LESC83], FORMEL [FAGE84], KADS [STIC86], RRL [KAZH88]). Il existe cependant des systèmes de réécriture noethériens et non confluents qui ne peuvent être complétés en un nombre fini d'étapes ([KANA85]).

Les notions de confluence et de terminaison ont été étendues à des sytèmes dont certaines égalités ne peuvent être orientées, comme la commutativité, des algorithmes de complétion, essentiellement dans le cas Associatif et Commutatif, ont été étudiés et implantés. Citons, entre autres, D.S.Lankford et A.M.Ballantyne [LABA77], G.Huet [HUET80], G.E.Peterson et M.E.Stickel [PEST81], J.P.Jouannaud et H.Kirchner [JOKI86], L.Bachmair et N.Dershowitz [BADE87], H.Kirchner [KIRC85], C.Kirchner [KIRC85].

L'exemple précédent était tiré de la tradition algébrique. En programmation, on retrouve souvent des spécifications du type suivant:

Soit Σ={0, succ, +, *, exp}, C={0, succ} et F={+, *, exp} et soit (E') le système d'équations défini par:

+(0, x) = x;
+(succ(x), y) = succ(+(x, y));
*(0, x) = 0
*(succ(x), y) = +(*(x, y), y);
exp(0, y) = 1;
exp (succ(x), y) = *(exp(x, y), y).

Soit le système de réécriture S' obtenu en orientant les équations de la gauche vers la droite. Ce système de réécriture est noethérien et *régulier* (il est linéaire à gauche et sans paires critiques). En terme de spécification, on est dans le cas type entier avec les constructeurs 0 et succ (ensemble C) et la *spécification E' est suffisamment complète par rapport à C*, en ce sens que, tout terme clos peut être prouvé égal à un terme clos s'écrivant uniquement à l'aide des éléments de C. Ce caractère régulier est assez naturel en programmation, un cas particulier évident de cette situation étant la programmation fonctionnelle.

Le système S' étant régulier et noethérien, on sait que dans ce cas, il est aussi confluent et, par conséquent, on dispose d'un calcul de formes normales permettant de vérifier la validité de toute égalité. Mais, on peut plus largement chercher à vérifier des formules dites du *premier ordre* comme:

$\forall n>2, \nexists (x,y,z)\neq(0,0,0)\ x^n + y^n = z^n$.(théorème de Fermat)

Cette formule amène à résoudre un problème ouvert, célèbre s'il en est, et illustre toute la différence de problématique entre la décidabilité d'une théorie équationnelle et la décidabilité d'une formule du premier ordre.

Une formule du premier ordre est une formule qui s'écrit à l'aide des quantificateurs, des connecteurs logiques, d'opérateurs et de prédicats définis. *Une théorie du premier ordre est dite décidable* si on peut décider de la validité de toute formule du premier ordre pour cette théorie.

Par exemple, l'arithmétique de Presburger (opérateurs 0, succ et + avec leurs axiomes de définition et pour prédicat l'égalité) est décidable. Lorsqu'une théorie est décidable, le point de vue pessimiste est de dire "la théorie est pauvre"; le point de vue optimiste est de dire "on sait tout faire", mais en général, même si on sait tout faire, on a souvent des hiérarchies de complexité irréalistes avec des tours d'exponentielles.

Mais, on sait également qu'une théorie est vite indécidable. Il suffit par exemple d'ajouter l'opérateur * et ses axiomes de définition à l'arithmétique de Presburger pour obtenir une théorie indécidable.

Lorsqu'une théorie est indécidable, on peut se demander ce qu'il en est pour un problème donné ou pour des fragments de la théorie (expressions d'une certaine forme).

Nous avons, jusqu'à présent, toujours considéré des termes avec variables, nous allons maintenant montrer l'intérêt de considérer des *termes clos*, des *substitutions closes*.

D'un point de vue sémantique, l'avantage est que l'on dispose alors d'une algèbre initiale dans le cas des théories équationnelles. On parle alors de la *théorie inductive* d'ensemble d'axiomes E.

On définit la notion de *confluence close* à partir de la notion de confluence en se limitant aux termes clos. Il est évident qu'un système de réécriture confluent est confluent sur les termes clos. La confluence des systèmes de réécriture noethériens est décidable alors que la confluence sur les termes clos des systèmes de réécriture noethériens est indécidable.

Précisons sur l'exemple suivant ce qu'est un théorème inductif:
$\Sigma=\{0, succ\}$ et $E''=\{succ(succ(0)) = 0\}$.
Le théorème $succ(succ(x)) = x$ n'est pas un théorème équationnel mais est un théorème inductif (on peut le prouver par une induction structurelle sur la structure des termes clos).

Pour décider de la validité d'un théorème dans l'algèbre initiale, on utilise le principe de la preuve par induction sans induction (ou preuve par consistance). Il consiste à ajouter le théorème u=v à démontrer à l'ensemble E et à vérifier que les relations de congruence sur les termes clos définies par E, et, E auquel on a ajouté le théorème sont les mêmes. On peut prouver ce résultat en prouvant que E est confluent sur les termes clos et que les termes u et v peuvent se réécrire en un même terme pour toute substitution close ([MUSS80], [HUHU82], [GOGU80], [BACH88], [JOKO86]).

Les substitutions closes ont donc des applications importantes. Dans ce domaine, on voit encore la nécessité d'apporter un éclairage algébrique à ces notions: par exemple, l'étude du point de vue algébrique de l'algorithme de décision de l'inductive réductibilité ([PLAI85]).

Un autre domaine de recherche important de la théorie de la réécriture concerne les *problèmes d'unification*. Une application en est la réécriture équationnelle et la programmation dans des langages de type clausal auxquels on ajoute des équations. L'unification modulo AC est décidable, modulo AD, elle est indécidable et modulo D, le problème est ouvert.

****

Nous venons d'esquisser quelques champs d'étude de la réécriture du point de vue de la programmation. Nous avons rencontré au passage quelques problèmes ponctuels non résolus (terminaison d'une seule règle linéaire, unification modulo D) ou mal élucidés (inductive réductibilité, égalité d'un ensemble de formes normales à une sorte). Notre but est, rappelons le, de contribuer à une meilleure connaissance structurelle des arbres et de la réécriture dans les arbres, et de contribuer ainsi à élaborer des outils pour la réécriture en programmation.

Pour conforter notre démarche, rappelons deux outils structurels célèbres sous-jacents à la structure d'arbre: le théorème de Kruskal ([KRUS60]) et les automates de Rabin ([RABI69]) et citons quelques résultats récents, autres que les nôtres, obtenus à partir de cette approche "automates et langages".

Le théorème de Kruskal, qui dit que de toute suite d'arbres on peut extraire une sous-suite croissante pour une certaine notion d'ordre sur les arbres, est à la base de nombreux résultats et concepts sur la terminaison des systèmes de réécriture ([DERS87],[DEJO90]).

Un deuxième résultat fondamental est le théorème de Rabin [RABI69], [RABI77])qui est une mine de résultats de décisions. On trouvera dans [THOM90] une présentation de ce théorème ainsi que de nombreuses applications.

On comprend donc pourquoi l'approche structurelle des problèmes liés à la théorie de la réécriture s'est récemment développée et a amené de nombreux résultats. Parmi ces travaux citons les études de:

A.I.Mal'cev ([MALC71]), M.J.Maher ([MAHE88]), H.Comon ([COMO88, 89,90]), sur une axiomatisation des algèbres d'arbres et les résultats de décidabilité associés;

R.V.Book & all, D.Kapur, P.Narendran, F.Otto sur les systèmes de Thue avec de nombreux résultats de décidabilité induits sur les systèmes de réécriture ([KNO90], [NAOT90]);

R.V.Book et J.H.Gallier ([BOGA85]), K.Salomaa ([SALO88]) sur les systèmes de réécriture et les automates à piles d'arbres;

M.Oyamaguchi [OYAM87, 90] sur différentes classes de systèmes de réécriture (en particulier clos et clos à droite).

Pour les résultats de l'équipe de Lille, citons, en ce qui concerne la réécriture, la décidabilité de la confluence close ([DATI85]), la décidabilité de la réécriture close (résultat généralisant le précédent [DATI90]), la décidabilité de la terminaison de la surréduction itérée en tête ([LEBE88], [DEVI90]), la décidabilité de la terminaison équitable des systèmes de réécriture clos [TISO89], qui font le pendant à l'indécidabilité de savoir si un système d'équations fini préserve la reconnaissabilité ([CDT89]) et la simulation d'une machine de Turing par une seule règle linéaire à gauche ([DAUC89]).

La notion d'*automate fini d'arbres et de forêt reconnaissable* est centrale dans nos travaux.

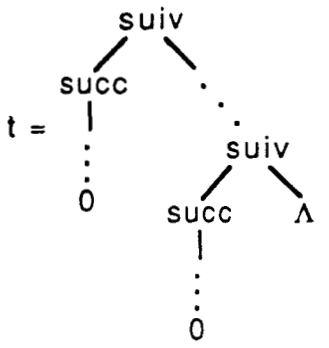
Nous rappelons que les automates finis d'arbres (J.W.Thatcher [THAT67], W.S.Brainerd [BRAI69]) sont, exprimés en un autre vocabulaire, des définitions de sortes.

Nous illustrons cette remarque sur l'exemple suivant:

Soit la sorte ListeEntier définie par:

$O : \rightarrow Nat.$
$succ : Nat \rightarrow Nat.$
$\Lambda : \rightarrow ListeEntier$
$suiv : Nat \times ListeEntier \rightarrow ListeEntier$

Un terme t de la sorte ListeEntier a pour configuration :



A la sorte ListeEntier est associée la forêt reconnaissable F reconnue par l'automate $A=(\Sigma,Q,Q_f,R)$ avec $Q = \{q_{Nat}, q_{ListE}\}$; $Q_f = \{q_{ListE}\}$ et

$R = \{ 0 \rightarrow q_{Nat}$; succ $\rightarrow q_{Nat}$; $\Lambda \rightarrow q_{ListE}$; suiv $\rightarrow q_{ListE}$ $\}$.

Cette remarque, trop souvent méconnue, a des applications importantes. Par exemple, H.Comon dans [COMO90] remarque qu'une signature avec sortes ordonnées peut être considérée comme un automate ascendant d'arbres. L'auteur propose un ensemble de règles de transformation correct, complet, qui termine pour les formules du premier ordre dont les atomes sont, soit des équations entre termes, soit de la forme $t \in s$ ou s est une sorte (c'est-à-dire une forêt reconnaissable). Dans la preuve, sont utilisées les propriétés de clôture booléenne de la classe des forêts reconnaissables et la décidabilité du vide.

Les automates d'arbres ont donc un intérêt, tant par des applications éventuelles d'un point de vue programmation, que par les algorithmes connus permettant de résoudre des problèmes de décision en réécriture.

****

Nous présentons maintenant le travail propre à cette thèse. Nous rappelons que la seconde partie de la thèse est commune avec la thèse de J.L.Coquidé.

Nos principaux résultats sont:

1) Nous étudions les problèmes de décidabilité sur les systèmes de réécriture et les forêts reconnaissables. En particulier, nous étudions la propriété (P): Pour toute forêt reconnaissable F, l'ensemble des formes normales des termes de F pour S est reconnaissable. Nous prouvons l'indécidabilité de cette propriété.

2) Soit un ensemble E d'équations pour lequel il existe un système de réécriture S convergent, satisfaisant la propriété (P) (i.e. l'ensemble des formes normales pour S des termes de toute forêt reconnaissable est reconnaissable), et vérifiant $\overset{*}{\leftrightarrow}_E = \overset{*}{\leftrightarrow}_S$. Nous prouvons que le fragment existentiel de théorie des algèbres de termes clos modulo la congruence générée par un ensemble E d'équations vérifiant ces propriétés est indécidable. Nous présentons cependant un algorithme de décision pour une classe de formules linéaires closes.

3) Nous montrons que le problème d'accessibilité est décidable pour les systèmes de réécriture clos modulo la commutativité (complexité polynomiale), est indécidable modulo l'associativité et décidable dans le cas associatif et commutatif (avec toute la complexité du problème d'accessibilité pour les réseaux de Pétri). Nous n'avons pu établir ce dernier résultat que moyennant certaines restrictions sur la configuration des termes.

4) Nous définissons la propriété suivante: un système de réécriture S possède *la propriété trf* si et seulement si, pour tout membre droit $r=l(t_1,...,t_n)$ de règle de S avec l lettre d'arité n, pour toute substitution close et irréductible σ (pour S), alors, pour tout i, $\sigma(t_i)$ est irréductible. Nous démontrons que, à tout système de réécriture linéaire et convergent S possédant la propriété trf, on peut associer un automate à piles déterministe qui calcule les formes normales pour S. Ce résultat permet d'éclairer et de généraliser les études antérieures sur les liens entre systèmes de réécriture et automates à piles (R.V.Book et J.H. Gallier dans [BOGA85] et K. Salomaa dans [SALO88]).

Reprenons, par le détail, ces différents résultats.

Le point de départ de la première partie de ce travail était d'étudier des *extensions de systèmes de réécriture clos*. Ces extensions consistent à considérer ces systèmes en présence d'opérateurs commutatifs, associatifs ou associatifs et commutatifs.

Le problème étudié était le *problème d'accessibilité* pour ce type de systèmes. Etant donné un système de réécriture S, le problème d'accessibilité du premier ordre est: étant donnés deux termes clos t et t', t peut-il se réécrire en t' par S?

Ce problème est indécidable pour les systèmes de réécriture (généraux) mais décidable pour les systèmes de réécriture clos ([DATI85], [DEGI89]).

Dans cette étude une propriété importante se met en évidence: la préservation de la reconnaissabilité. Un système de réécriture *préserve la reconnaissabilité* si, pour toute forêt reconnaissable F, l'ensemble S(F) des réductions de termes de F par S est reconnaissable. En effet, la décidabilité du problème d'accessibilité se déduit de façon immédiate de cette propriété. De plus, si on dispose d'un algorithme effectif de construction d'un automate associé à S(F), on en déduit un algorithme pour la résolution du problème d'accessibilité.

Nous avons également rappelé dans la première partie de l'introduction qu'une sorte est un automate d'arbres. Nous nous sommes donc également intéressés aux résultats généraux de décidabilité liant les systèmes de réécriture et les forêts reconnaissables.

Le plan et les résultats obtenus dans la première partie sont les suivants:

Dans la section 3.2, nous étudions les liens entre réécriture et reconnaissabilité. Nous étudions les différents *problèmes de décision* qui peuvent être étudiés quant à la *préservation de la reconnaissabilité* par un système de réécriture. Il n'est pas surprenant que la plupart des résultats soient des résultats d'indécidabilité.

Nous prouvons, en particulier, que le problème suivant est indécidable:
Donnée: système de réécriture S
Question: Pour toute forêt reconnaissable F, l'ensemble S!(F) des formes normales des termes de F pour S est reconnaissable.

Notons également que certains des problèmes posés restent ouverts. En particulier: décider si l'ensemble IRR(S) des termes clos irréductibles pour un système de réécriture S est reconnaissable.

Dans la section 3.3, nous étudions (des fragments) de théorie des algèbres de termes clos modulo certaines congruences $\overset{*}{\leftrightarrow}_E$ engendrées par un ensemble d'équations. Des travaux connexes sont les résultats de décidabilité obtenus par H.Comon dans [COMO88] et d'indécidabilité de R.Treinen dans [TREI90].

R.Treinen propose un méthode générale de preuve pour établir l'indécidabilité de théories du premier ordre (l'idée est de coder le problème de Post dans une formule du premier ordre d'une théorie pour un modèle donné). Par cette méthode, l'auteur prouve l'indécidabilité du Σ_3-fragment de théorie des algèbres de termes clos dans le cas AC et du Σ_2-fragment dans le cas A.

H.Comon montre la décidabilité du Σ_1-fragment dans le cas AC. A partir des résultats de [COMO88], on peut déduire la décidabilité des théorie des algèbres de termes clos modulo des congruences $\overset{*}{\leftrightarrow}_E$ engendrées par un ensemble d'équations closes.

Pour notre part, nous considérons des ensembles d'équations E pour lesquels il existe un système de réécriture S convergent, vérifiant $\overset{*}{\leftrightarrow}_E = \overset{*}{\leftrightarrow}_S$ et satisfaisant la propriété (P) suivante: l'ensemble des formes normales pour S des termes de toute forêt reconnaissable est reconnaissable.

Nous prouvons que le fragment existentiel de théorie des algèbres de termes clos modulo la congruence générée par un ensemble E d'équations vérifiant ces propriétés est indécidable.

Nous présentons cependant un algorithme de décision pour une classe de formules linéaires closes, étendant, en celà, un résultat de R.V.Book [BOOK83], du cas des mots au cas des arbres. Nous montrons comment ce résultat peut être étendu aux algèbres avec sortes ordonnées. Nous généralisons ce résultat à des systèmes de réécriture équationnels.

Dans la section 4, nous étudions différents problèmes de décidabilité pour certaines classes de systèmes de réécriture.

Dans la section 4.1, nous présentons et prouvons l'*algorithme de compilation d'un système de réécriture clos* implanté dans le logiciel VALERIAN. C'est-à-dire que nous présentons l'algorithme de construction d'un transducteur d'arbres V associé à un système de réécriture clos R tel que la transformation associée à V soit égale à la relation de réécriture engendrée par R.

Différents problèmes d'accessibilité du second ordre sont, par exemple: étant donné un système de réécriture clos S, étant données deux forêts reconnaissables F et F', existe-t-il un terme de F qui se réécrit en un terme de F' par S? L'ensemble des transformés des termes de F par S est-il inclus dans F'?...

Nous présentons également l'algorithme associant à un automate reconnaissant une forêt reconnaissable F et au système de réécriture clos compilé, l'automate reconnaissant l'ensemble S(F) des réductions des termes de F par S. C'est cet algorithme qui est implanté dans VALERIAN et qui permet de résoudre les problèmes d'accessibilité du premier ordre et du second ordre associé à un système de réécriture clos.

La section 4.2 a pour but de s'intéresser à des *systèmes de réécriture clos modulo la commutativité*. Nous prouvons que la reconnaissabilité est préservée et que les problèmes d'accessibilité, la terminaison modulo sont décidables. De plus, la confluence est décidable, sous l'hypothèse de terminaison modulo, ce résultat étant une conséquence immédiate des résultats généraux sur les systèmes de réécriture équationnels ([JOKI86]).

Par contre, restent posées la question de la décidabilité de la confluence sans cette hypothèse de terminaison modulo, et, de façon plus générale, la décidabilité de la réécriture close commutative au sens de M.Dauchet et S.Tison pour la réécriture close ([DATI90]).

C'est-à-dire la décidabilité de la théorie du premier ordre où les prédicats sont $\rightarrow_S$, $\overset{*}{\rightarrow}_S$, pour tout S système de réécriture clos modulo la commutativité, les variables étant des termes. Ce résultat aurait, en effet, pour conséquence, par exemple: la décidabilité de l'équivalence de tels systèmes, la décidabilité de la confluence... Il semble, cependant, que de nouveaux outils soient à mettre en oeuvre pour essayer d'obtenir un tel résultat, les techniques de codage canonique par automates finis ne s'appliquant plus dans le cas modulo la commutativité.

Dans la section 4.3, nous montrons que, *pour les systèmes de réécriture clos modulo l'asssociativité*, les résultats deviennent des résultats d'indécidabilité en prouvant les problèmes équivalents aux mêmes problèmes sur les mots.

Dans la section 4.4 (respectivement 4.5), nous obtenons un résultat partiel de décidabilité du problème d'accessibilité pour les *systèmes de réécriture clos modulo la commutativité et l'associativité* (respectivement l'associativité, la commutativité et l'idempotence).

Dans la section 4.6, nous étudions *les systèmes de réécriture monadiques*. Un système de réécriture est dit monadique si les membres gauches de règle sont de hauteur supérieure ou égale à 1 et les membres droits sont une constante, une variable, ou de la forme $b(y_1,...,y_n)$ où, pour tout i, y_i est une variable. La reconnaissabilité est préservée dans le cas linéaire à droite ([SALO88]) et nous prouvons que la terminaison et la confluence sont décidables dans le cas linéaire à droite. Ces résultats ont été prouvés indépendamment par K.Salomaa qui a de plus prouvé que la terminaison est indécidable dans le cas non linéaire à droite.

****

La seconde partie de la thèse est la présentation d'un travail qui a été réalisé à l'occasion de la venue de S.Vàgvölgyi à Lille. Cette partie est commune à cette thèse et à la thèse de J.L.Coquidé.

Nous étudions dans cette partie *les liens entre les automates à piles d'arbres et les systèmes de réécriture.*

Nous poursuivons, en cela, une étude commencée par R.V.Book et J.H. Gallier dans [BOGA85] et K. Salomaa dans [SALO88]. Les automates à piles d'arbres peuvent être considérés comme le point de vue algorithmique de problèmes spécifiés à l'aide de certains systèmes de réécriture. On peut alors parler de compilation d'un système de réécriture à l'aide d'automates à piles d'arbres.

Plus précisément, R.V. Book, J.H. Gallier et K. Salomaa ont montré que l'on pouvait associer, à tout système de réécriture monadique linéaire et convergent S, un automate à piles déterministe d'arbres qui calcule les formes normales pour S.

L'originalité du travail présenté ici est que nous mettons en évidence l'intérêt d'une nouvelle propriété des systèmes de réécriture: la propriété trf (pour "tail reduction free").

Un système de réécriture S possède *la propriété trf* si et seulement si, pour tout membre droit $r=l(t_1,...,t_n)$ de règle de S avec l lettre d'arité n, pour toute substitution close et irréductible σ (pour S), alors, pour tout i, $\sigma(t_i)$ est irréductible. C'est-à-dire, pour tout membre droit r de règle de S, pour toute substitution close irréductible (pour S) σ, le terme $\sigma(r)$ n'est réductible par S qu'en tête.

Nous démontrons que, à tout système de réécriture linéaire et convergent S possédant la propriété trf, on peut associer un automate à piles déterministe qui calcule les formes normales pour S. Les systèmes de réécriture monadiques possèdent, de façon évidente, la propriété trf; les résultats précédemment démontrés sont alors des cas particuliers des résultats que nous obtenons.

Dans la section 5.1, nous comparons les différentes définitions possibles d'automates à piles ascendants d'arbres en fonction de l'arité des états, de la profondeur des termes dépilés et de la forme des règles. Nous montrons l'équivalence des différentes définitions.

Nous définissons également la notion de déterminisme associé de la façon suivante: un automate à piles est déterministe si le système de réécriture (définissant les règles) est linéaire à gauche et sans paire critique. Nous montrons, dans le cas déterministe, l'équivalence des différentes définitions.

Dans la section 5.2, nous introduisons la propriété trf. Nous montrons que la classe des systèmes de réécriture possédant cette propriété est large (contient, par exemple, la classe des systèmes de réécriture monadiques). Nous prouvons que l'on peut simuler une machine de Türing à l'aide d'un système de réécriture ayant cette propriété.

Nous démontrons la décidabilité de la propriété trf en utilisant la décidabilité de l'inductive réductibilité ([PLAI85]).

Nous associons à tout système de réécriture S possédant la propriété trf un automate à piles d'arbres qui calcule les formes normales des termes clos pour S. Nous précisons les relations entre la linéarité du système de réécriture et la nécessité d'un contrôle pour l'automate à piles, en effet, dans le cas linéaire à gauche, l'automate à piles associé est déterministe, et, dans le cas non linéaire à gauche, il est nécessaire d'introduire un contrôle ("priorité à la réduction") dans l'automate à piles.

Dans la section 5.3, nous étudions les systèmes de réécriture possédant la propriété trf et les automates à piles d'arbres du point de vue théorie des langages (d'arbres). Un système de réécriture est *semi-monadique* si tout membre droit de règle a pour configuration $r=l(y_1,...,y_n)$ avec l lettre d'arité n et, pour tout i, y_i est une variable ou un terme clos. Nous étendons un résultat de K. Salomaa ([SALO88]) sur la préservation de la reconnaissabilité des systèmes de réécriture monadiques aux systèmes de réécriture semi-monadiques.

****

Citons quelques prolongements et perspectives à ce travail.

Ponctuellement, des problèmes de décidabilité sont laissés ouverts.

Par exemple, dans la première partie, le problème (difficile) de décider si un système de réécriture S est tel que IRR(S), ensemble des termes clos irréductibles pour S, est reconnaissable.

De même, dans l'étude des classes de systèmes de réécriture, certains problèmes ont été laissés non résolus tels, par exemple, la décidabilité de la confluence des systèmes de réécriture monadiques, la décidabilité de la confluence close des systèmes de réécriture monadiques linéaires.

D'une part, H.Comon montre que la décidabité de la "théorie des arbres" (au sens de H.Comon, A.I.Mal'cev, M.J.Maher) demeure quand on la fait modulo un système d'équations closes ([COMO88]) ou sur des algèbres sortées ([COMO90]).

D'autre part, nous montrons, d'une certaine façon, que ces résultats ne s'étendent pas, même si la reconnaissabilité est préservée. Cependant, nous obtenons un résultat de décision pour une sous-classe de formules linéaires. La recherche de nouveaux fragments décidables pour les théories des algèbres de termes clos modulo des congruences engendrées par des ensembles d'équations est à poursuivre.

****

D'une façon générale, nous comptons développer nos efforts dans une double perspective: mieux connaître la structure de classes de systèmes de réécriture et de classes de problèmes; trouver des algorithmes de transformation (de "compilation") de ces systèmes.

Nous allons développer, avec nos méthodes, l'étude de fragments de théories décidables (éventuellement sous des hypothèses indécidables), la combinaison modulaire de celles-ci, une généralisation et un éclaircissement des problèmes liés à l'inductive réductibilité.

2 PRELIMINAIRES

2.1 Termes et Substitutions

2.1.1 Termes

Σ est un alphabet fini gradué si et seulement si chaque élément b de Σ a une arité unique dans l'ensemble des entiers naturels notée r(b). Les éléments d'arité 0 sont appelés les constantes. Nous supposons que Σ contient toujours une constante. Pour tout entier positif n, Σ_n désigne le sous-ensemble de Σ qui contient tous les éléments d'arité n.

Soit X un ensemble dénombrable de variables. *L'ensemble $T_\Sigma(X)$ des arbres (ou termes) sur $\Sigma \cup X$* est défini comme étant le plus petit ensemble pour lequel:

i) $X \subseteq T_\Sigma(X)$

ii) $b(t_1,\ldots,t_n) \in T_\Sigma(X)$, pour tout $b \in \Sigma_n$ avec $n \geq 0$ et $t_1,\ldots, t_n \in T_\Sigma(X)$.

Posons $X_n=\{x_1,\ldots,x_n\}$ pour tout entier positif n. $T_\Sigma(X_n)$ est l'ensemble des arbres sur $\Sigma \cup X_n$. L'ensemble $T_\Sigma(\varnothing)$, noté T_Σ, est *l'ensemble des arbres clos* sur Σ.

L'ensemble des variables apparaissant dans t est noté V(t).

Un arbre t est linéaire si et seulement si aucune variable n'apparait deux fois dans t.

La racine d'un arbre $t=c(t_1,\ldots,t_n)$ est le symbole c situé en son sommet, *une feuille* est un symbole d'arité 0 apparaissant dans t (c'est une constante ou une variable).

Une occurrence dans un arbre est une position dans l'arbre. Elle peut être déterminée par une suite d'entiers positifs (notation de Dewey) donnant le chemin permettant d'aller de la racine au symbole situé à la position considérée. *L'ensemble des occurrences d'un arbre est notée O(t). Nous notons $t_{/p}$ le terme situé à l'occurrence p.*

La hauteur h(t) (ou profondeur) d'un arbre t correspond à la longueur de sa plus grande branche dans t. La *taille ||t||* d'un arbre est le nombre de noeuds que contient l'arbre. La *frontière fr(t)* d'un arbre est le mot constitué de ses feuilles. *La largeur d'un arbre* est le nombre de feuilles de l'arbre. Elle est notée largeur(t). Nous définissons *l'ensemble Sub(t) comme l'ensemble des sous-arbres d'un arbre t* et *Chemin(t)* désigne l'ensemble de toutes les chaînes de symboles de Σ qui apparaissent le long d'un chemin allant de la racine de l'arbre t jusqu'à une feuille.

Elles sont définies de la façon suivante:

i) si $t \in \Sigma_0 \cup X$ alors $h(t) = 0$, $fr(t) = t$, $||t|| = 1$, $Sub(t) = \{t\}$, $Chemin(t) = \{t\}$

ii) si $t = b(t_1,\ldots,t_n)$ avec $n \geq 1$, $b \in \Sigma_n$ et $t_1,\ldots t_n \in T_\Sigma(X)$ alors

$$h(t) = 1 + \max(\{h(t_i) \;/\; 1 \leq i \leq n\}),$$

$$||t|| = 1 + \sum_{i=1}^{n} ||t_i||,$$

$$fr(t) = \prod_{i=1}^{n} fr(t_i).$$

$$Sub(t) = \{t\} \cup \left(\bigcup_{i=1}^{n} Sub(t_i) \right)$$

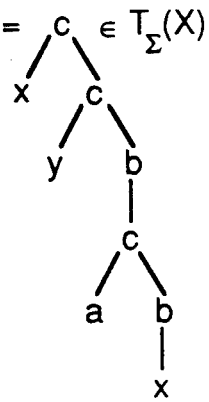
$$Chemin(t) = \{b.ch \;/\; ch \in \left(\bigcup_{i=1}^{n} Chemin(t_i) \right)\}.$$

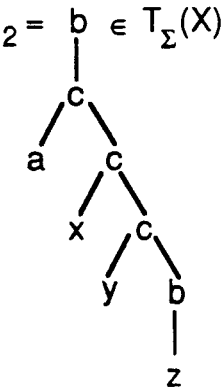
et largeur(t)= |fr(t)|.

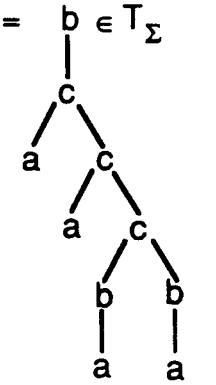
Exemple: Voir figure 2.1.1.

Un sous-terme propre de t est un élément de Sub(t) distinct de t.

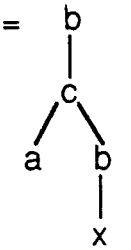
Figure 2.1.1.

$t_1 =$  $\in T_\Sigma(X)$

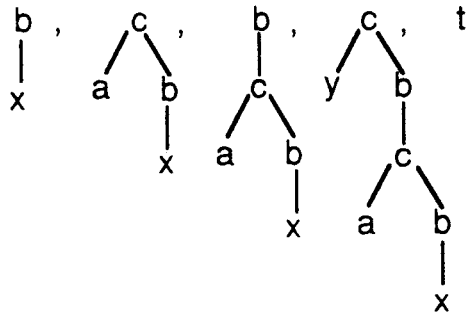
$t_2 =$  $\in T_\Sigma(X)$

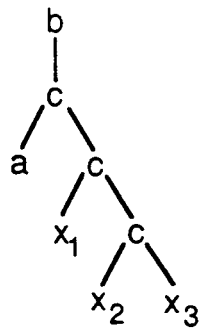
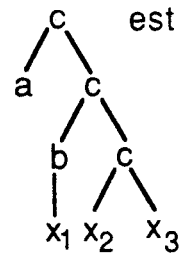
$t_3 =$  $\in T_\Sigma$

t_1 n'est pas linéaire, t_2 et t_3 sont linéaires; $V(t_1)=\{x,y\}$; $V(t_2)=\{x,y,z\}$;

$t_{1/2.2} =$  ; $h(t_1)=5$; $||t_1|| = 9$; $fr(t_1) = xyax$; $largeur(t_1) = 4$;

; $chemin(t_1) = \{ cx, ccy, ccbca, ccbcbx\}$

$Sub(t_1) = \{ x, y,$  $t_1 \}$

$tronc(t_2, 4) =$  ;  est un arbre 3-normalisé

Un contexte c est un terme de $T_\Sigma(X_n)$ tel que chaque variable apparaît exactement une fois dans c et nous notons $c(t_1,...,t_n)$ le résultat de la substitution de chaque x_i par un terme t_i.

Soit t un terme et p une position de t, *le contexte c associé à p dans t* est le terme c défini par $t=c(t_{/p})$. c(u) désigne alors le terme obtenu en substituant u à $t_{/p}$ à l'occurrence p dans t.

La troncature d'un arbre à la profondeur k est notée tronc(t,k). Elle est définie de la façon suivante: Pour toute occurrence p de O(tronc(t,k)), soit |p| = k et $tronc(t,k)_{/p}$ est élément de X, soit |p| < k et la racine de $tronc(t,k)_{/p}$ est égale à la racine de $t_{/p}$ et de plus si tronc(t,k) est un élément de $T_\Sigma(X_n)$ alors $fr(tronc(t,k)) \in \Sigma^* x_1 \Sigma^* ... x_n \Sigma^*$.

Exemple: Voir figure 2.1.1.

Un arbre k-normalisé est un arbre linéaire de hauteur k tel que les variables soient ordonnées et apparaissent toutes à la profondeur k et tel que les opérateurs apparaissent à un niveau inférieur à k. Formellement, un arbre k-normalisé (k entier positif) est un arbre linéaire de $T_\Sigma(X_n)$ tel que pour toute occurrence p de O(t) soit |p| = k et $t_{/p}$ est un élément de X soit |p| < k et la racine de $t_{/p}$ est un élément de Σ et de plus $fr(t) \in \Sigma^* x_1 \Sigma^* ... x_n \Sigma^*$.

Exemple: Voir figure 2.1.1.

2.1.2 Substitutions

Une *substitution* σ est une application de X dans $T_\Sigma(X)$ distincte de l'identité pour un nombre fini de variables, on note $\sigma=\{x_1 \rightarrow s_1,\ldots,x_n \rightarrow s_n\}$. Le *domaine* $D(\sigma)$ d'une substitution σ est l'ensemble des variables qui n'ont pas pour image elle-même.

Une substitution σ s'étend en une application de $T_\Sigma(X)$ dans lui-même par: pour tout f de Σ_n, pour tous termes $t_1,\ldots,t_n$ de $T_\Sigma(X)$, $\sigma(f(t_1,\ldots,t_n)) = f(\sigma(t_1),\ldots,\sigma(t_n))$. Nous confondons toujours, dans les notations, une substitution et son extension.

Une substitution close σ est telle que toute variable de $D(\sigma)$ a son image qui est un terme clos de T_Σ.

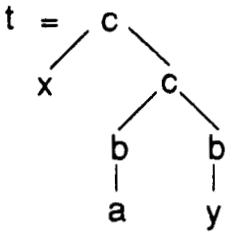
Un terme t filtre un terme t' si il existe une substitution σ telle que $\sigma(t)=t'$. La relation ainsi définie est une relation de préordre, et la relation d'équivalence associée correspond à l'égalité des termes modulo un renommage des variables. Nous utiliserons toujours, sans le repréciser, le fait que nous travaillons modulo un renommage des variables.

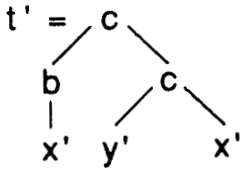
Deux termes t et t' sont unifiables si il existe une substitution σ telle que $\sigma(t)=\sigma(t')$.

La *composition des substitutions* est la composition des applications.

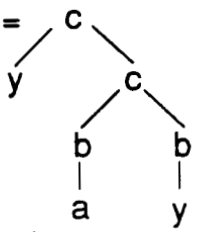
σ est plus générale que τ si et seulement si il existe une substitution ρ telle que $\rho \circ \sigma = \tau$. Cette relation est une relation de préordre. Quels que soient les termes t et t', il existe un unificateur le plus général de t et t' ([PLOT72]).

Figure 2.1.2.

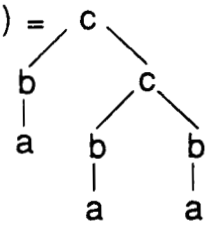




$\sigma_1 = \{ x \longrightarrow y \}$; $\sigma_1(t) =$

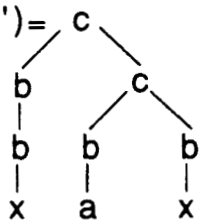


$\sigma_2 = \{ x \longrightarrow b(a),\ y \longrightarrow a \}$ est close et $\sigma_2(t) =$



$\sigma_3 = \{ x \longrightarrow b(b(x)),\ y \longrightarrow x,\ x' \longrightarrow b(x),\ y' \longrightarrow b(a) \}$

σ_3 est l'unificateur le plus général de t et t'; $\sigma_3(t) = \sigma_3(t') =$



2.2 Langages d'arbres

2.2.1 Grammaires

Une grammaire algébrique d'arbres G est un quadruplet $G=(\Sigma,V,A_0,R)$ où:

Σ est un alphabet gradué fini (de symboles terminaux);

V est un alphabet gradué fini (de variables);

A_0 est l'axiome et appartient à V_0 (non terminal d'arité 0) et

R un ensemble fini de règles de production $l \rightarrow r$ avec $l=B(x_1,...,x_n)$ où B appartenant à V est d'arité n, et, r est un terme de $T_{\Sigma \cup V}(X_n)$.

On définit, de façon usuelle, la relation $\rightarrow_G$ comme la relation de réécriture $\rightarrow_R$ sur $T_{\Sigma \cup V}(X)$ et sa clôture réflexive et transitive est $\overset{*}{\rightarrow}_G$ Le *langage L(G) généré par G* est $L(G)=\{ t \in T_\Sigma / A_0 \overset{*}{\rightarrow}_G t\}$. Un *langage est algébrique (ou à contexte libre)* si et seulement si il existe une grammaire algébrique qui l'engendre.

Une grammaire régulière d'arbres G est un quadruplet $G=(\Sigma,W,A_0,R)$ où:

Σ est un alphabet gradué fini (de symboles terminaux);

W est un alphabet fini de symboles d'arité 0 (variables);

A_0 est l'axiome et appartient à W et

R un ensemble fini de règles de production $l \rightarrow r$,avec $l=B \in W$ et r est un terme de $T_{\Sigma \cup W}$.

Un langage est régulier si et seulement si il existe une grammaire régulière qui l'engendre.

Exemple: Voir figure 2.2.2.a.

Pour de plus amples développements sur les grammaires régulières, se reporter à [BRAI69], [GEST84], sur les grammaires algébriques, voir [ARDA76], [ARDA78], [LEGU80].

2.2.2 Automates et forêts reconnaissables

Un *automate ascendant d'arbres* A est un quadruplet $A=(\Sigma,Q,Q_f,R)$ où: Σ est un alphabet fini gradué;

Q est un ensemble fini d'états d'arité 0;

Q_f est un sous ensemble de Q (ensemble des états finaux) et

R est un ensemble fini de règles de l'une des deux configurations suivantes:

(i) $f(q_1,\ldots,q_n)\rightarrow q$ avec $f\in\Sigma$, f d'arité n, $q_1,\ldots,q_n$ dans Q;

(ii) $q\rightarrow q'$ avec q,q' dans Q (ε-règles).

R est un système de réécriture sur $T_{\Sigma\cup Q}$, $\rightarrow_A$ est la relation de réécriture $\rightarrow_R$.

Un automate est *déterministe* s'il n'existe pas deux règles de type (i) ayant même partie gauche et s'il n'existe pas de règles de type (ii) (ε-règles).

Le *langage d'arbres reconnu par A* est $L(A)=\{t\in T_\Sigma / t \stackrel{*}{\rightarrow}_A q, q\in Q_f\}$. Un *langage d'arbres (une forêt) F est reconnaissable* s'il existe un automate ascendant A tel que L(A)=F. On appelle Rec la classe des langages d'arbres reconnaissables.

Exemples: Voir figure 2.2.2.a. Dans l'introduction, nous présentons un exemple de forêt reconnaissable pour rappeler le lien existant entre la notion de "sorte" et de forêt reconnaissable, et, dans la figure 2.2.2.b un automate associé à la notion de signature avec sortes ordonnées.

Soit A un automate ascendant d'arbres, il existe un automate ascendant sans ε-règles tel que L(B)=L(A) et il existe un automate ascendant deterministe (i.e sans ε-règles et n'ayant pas deux règles avec le même membre gauche) C tel que L(C)=L(A).

La classe Rec des langages d'arbres reconnaissables est close par union, intersection et complémentation. On peut décider si un langage d'arbres reconnaissable est vide, l'inclusion, l'égalité de langages d'arbres reconnaissables.

Figure 2.2.2.a

$\Sigma = \{ a, f, g \}$; $F = \{ f(g^n, g^{2p}) \;/\; n \geq 0 , p \geq 0 \}$

F est reconnu par l'automate ascendant déterministe $A=\{\Sigma,Q,Q_f,R\}$ avec

$Q = \{q, q_{pair}, q_{imp}\}$; $Q_f = \{q\}$ et

$R = \{ a \to q_{pair} ; g(q_{pair}) \to q_{imp} ; g(q_{imp}) \to q_{pair} ; f(q_{pair}, q_{pair}) \to q ; f(q_{imp}, q_{pair}) \to q \}$

F est reconnu par l'automate descendant déterministe $B=\{\Sigma,Q,Q_i,R\}$ avec

$Q = \{q, q_{ind}, q_{pair}, q_{imp}\}$; $Q_i = \{q\}$ et

$R = \{ q \to f(q_{ind}, q_{pair}) ; q_{ind} \to a ; q_{ind} \to g(q_{ind}) ; q_{pair} \to a ; q_{pair} \to g(q_{imp}) ; q_{imp} \to g(q_{pair}) \}$

F est engendrée par la grammaire régulière $G=(\Sigma,W,A_0,R)$ avec

$W = \{A_0, B, C\}$ et $R = \{ A \to f(B, C) ; B \to a \mid g(B) ; C \to a \mid g(g(C)) \}$

La classe Rec des langages reconnaissables (reconnus par automates ascendants d'arbres) est égale à la classe des langages réguliers (engendrés par grammaire régulière).

Un *automate descendant d'arbres* est un quadruplet $A=(\Sigma,Q,Q_i,R)$ où:

Σ est un alphabet fini gradué,

Q un ensemble fini d'états d'arité 1,

Q_i un sous ensemble de Q (ensemble d'états initiaux) et

R un ensemble fini de règles de l'une des deux configurations suivantes:

(i) $q[f(x_1,\ldots,x_n)] \to f(q_{i_1}[x_1],\ldots,q_{i_n}[x_n])$ avec $n \geq 0$, $q_{i_1},\ldots,q_{i_n}$ dans Q;

(ii) $q(x) \to q'(x)$ avec q,q' dans Q (ε-règles).

Le *langage d'arbres reconnu par A* est $L(A)=\{t\in T_\Sigma / q[t] \xrightarrow{*}_A t, q\in Q_i\}$.

Un automate descendant d'arbres est *déterministe* s'il n'existe pas deux règles de type (i) de même partie gauche et s'il n'existe pas de règles de type (ii).

La classe des langages reconnus par automates descendants d'arbres est égale à Rec mais, il n'existe pas d'algorithme de réduction du non-déterminisme pour les automates descendants d'arbres, en effet, si on considère la forêt F={ f(a,b), f(b,a)}, qui est finie et donc évidemment reconnaissable, elle ne peut être reconnue par automate déterministe descendant d'arbres.

Figure 2.2.2.b

Soit la signature avec sortes ordonnées définie par:

les constantes 0 et Λ, les opérateurs succ d'arité 1 et suiv d'arité 2, les trois sortes Nat, ListeEntier et ListeEntierNonVide,

$0 : \rightarrow$ Nat.
succ : Nat $\rightarrow$ Nat.
$\Lambda : \rightarrow$ ListeEntier.
suiv : Nat x ListeEntier $\rightarrow$ ListeEntierNonVide. et

ListeEntierNonVide < ListeEntier.

A cette signature est associée l'automate $A=(\Sigma,Q,Q_f,R)$ avec

$$Q = \{q_{Nat}, q_{ListE}, q_{LENV}\} = Q_f \text{ et}$$

$$R = \{ 0 \rightarrow q_{Nat}; \underset{\mid}{succ} \rightarrow q_{Nat} ; \Lambda \rightarrow q_{ListE} ; \underset{q_{Nat}\ q_{ListE}}{suiv} \rightarrow q_{LENV} ; q_{LENV} \rightarrow q_{ListE} \}.$$

Pour de plus amples développements, voir [GEST84].

2.2.3 Transducteurs d'arbres

Les transducteurs d'arbres sont une généralisation des transducteurs de mots. Un livre de synthèse sur les transductions est [BERS79].

Un *transducteur descendant d'arbres* (tda) est un quintuplet $T=(\Sigma,\Delta,Q,I,R)$ avec:

Σ et Δ alphabets gradués finis;

Q ensemble fini d'états (alphabet distingué, lettres d'arité 1);

I ensemble des états initiaux (I est inclus dans Q) et

R ensemble de règles de réécriture de la forme:

$q[c(x_1,\ldots,x_n)]\rightarrow t(q_{i1}[x_{i1}],\ldots,q_{ip}[x_{ip}])$, $c\in\Sigma_n, t(x_{i1},\ldots,x_{ip})\in T_\Delta(X_n)$

Exemple: voir figure 2.2.3.a

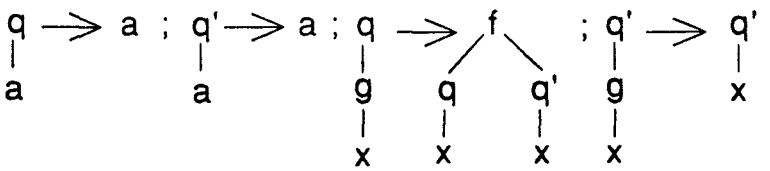
On note $\rightarrow_T$ la relation de réécriture associée à R et $\overset{*}{\rightarrow}_T$ sa cloture réflexive et transitive. *La transformation associée à T* est $r(T)=\{(t,t')/t\in T_\Sigma,\ t'\in T_\Delta,\ \exists q\in I$ tel que $q(t)\overset{*}{\rightarrow}_T t'\}$.

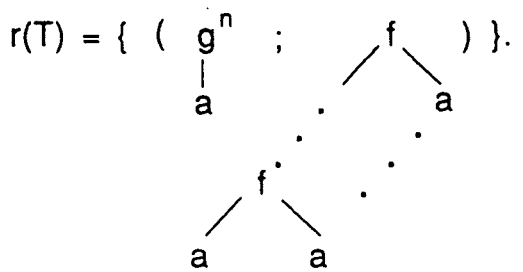
Figure 2.2.3.a

Exemple de transducteur descendant d'arbres:

Soit $\Sigma=\{a,g,f\}$ avec a, g et f d'arités respectives 0, 1 et 2.

Soit $T=(\Sigma,\Delta,Q,I,R)$ avec $\Sigma=\Delta$, $Q=\{q,q'\}$, $I=\{q\}$ et R défini par





Un *transducteur ascendant d'arbres* (taa) est un quintuplet $U=(\Sigma,\Delta,Q,J,R)$ avec

Σ et Δ alphabets gradués finis;

Q ensemble fini d'états (alphabet distingué, lettres d'arité 1);

J ensemble des états finaux (J est inclus dans Q) et

R ensemble de règles de réécriture de la forme:

$c(q_1[x_1],\ldots,q_n[x_n])\rightarrow q[t(x_{j1},\ldots,x_{jp})]$, $c\in\Sigma_n$, $t\in T_\Delta(X_n)$.

On note $\rightarrow_U$ la relation de réécriture associée à R et $\overset{*}{\rightarrow}_U$ sa cloture réflexive et transitive. *La transformation associée à U* est $r(U)=\{(t,t')/t\in T_\Sigma,\ t'\in T_\Delta,\ \exists q\in J$ tel que $t\overset{*}{\rightarrow}_U q(t')\}$.

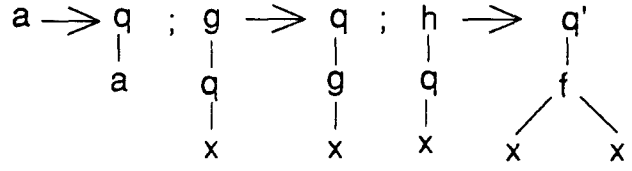
Exemple: voir figure 2.2.3.b.

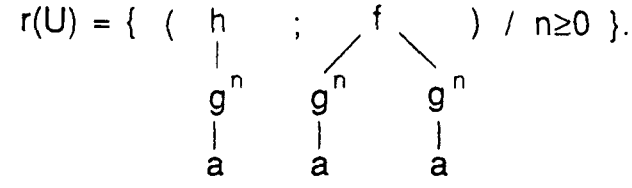
Figure 2.2.3.b.

Exemple de transducteur ascendant d'arbres:

Soit $\Sigma=\{a,g,h,f\}$ avec a, g, h et f d'arités respectives 0, 1, 1 et 2.

Soit $U=(\Sigma,\Delta,Q,J,R)$ avec $\Sigma=\Delta$, $Q=\{q,q'\}$, $J=\{q'\}$ et R défini par





U n'est pas linéaire, on peut remarquer que l'image de la forêt reconnaissable $F=\{h(g^n(a))\ /\ n\geq 0\}$ est $U(F)=\{f(g^n(a),g^n(a))\ /n\geq 0\}$ qui n'est pas une forêt reconnaissable.

Un transducteur descendant T (respectivement transducteur ascendant U) est *linéaire* si et seulement si, pour toute règle de R, $t(x_{i1},\ldots,x_{ip})$ (respectivement $t(x_{j1},\ldots,x_{jp})$) est linéaire.

La *classe des forêts reconnaissables est close par transducteurs linéaires*, c'est-à-dire, pour tout transducteur (ascendant ou descendant) linéaire T et toute forêt reconnaissable F, l'ensemble $T(F)=\{t'/t\in F$ et $(t,t')\in r(T)\}$ est une forêt reconnaissable.

La *classe des forêts reconnaissables est close par inverse de transducteurs*, c'est-à-dire, pour tout transducteur (ascendant ou descendant) T et toute forêt reconnaissable F, l'ensemble $T^{-1}(F)=\{t'/t\in F$ et $(t',t)\in r(T)\}$ est une forêt reconnaissable.

Les classes de transducteurs ascendants linéaires et de transducteurs descendants linéaires complets sont closes par composition.

Pour de plus amples développements, se reporter à [ENGE75], [GEST84].

2.3 Systèmes de réécriture

Les notations et définition utilisées dans cette partie sont, pour la plupart, celles de [DEJO90].

2.3.1 Résultats généraux

Un *système de réécriture* S est un ensemble de couples de termes. On note, en général, $S=\{ l_i \rightarrow r_i / l_i, r_i \in T_\Sigma(X), V(r_i) \subset V(l_i), i \in I\}$. Dans toute notre étude, nous ne considérerons que le cas I fini.

$\rightarrow_S$ *est la relation de réécriture induite par S*, c'est-à-dire, la clôture de la relation S par application au contexte et substitution. Soit encore, $t \rightarrow_S t'$ si et seulement si il existe une occurence p de t, une règle $l \rightarrow r$ de S et une substitution σ tels que $t/_p=\sigma(l)$ et $t'=c(\sigma(r))$, où c désigne le contexte associé à p dans t. $\stackrel{*}{\rightarrow}_S$ est la clôture réflexive et transitive de S.

Etant donnés un système de réécriture S et une relation d'ordre sur S, nous définissons *la relation de réécriture IO induite par S respectant* $<$ par $t \rightarrow_{i,S,<} t'$ si et seulement si t se réécrit en t' par S à une occurence p de t telle que aucune règle de S ne soit applicable en dessous de p et qu'il n'existe aucune règle supérieure pour $<$ applicable à cette occurence p. Si la relation d'ordre est vide, cette relation est la relation de réécriture IO (par valeur) induite par S. Dans le cas d'un ordre total, on retrouve la définition de R.V.Book et J.H.Gallier dans [BOGA85]. Cette définition nous sera utile dans la partie 5, lors de la construction d'un automate à piles associé à un système de réécriture.

Un terme t est *irréductible* pour S si il n'existe pas de t' tel que $t \rightarrow_S t'$. Nous notons *IRR(S)* l'ensemble des termes irréductibles pour S.

Un terme t' est *une forme normale de t pour S* si $t \stackrel{*}{\rightarrow}_S t'$ et $t' \in IRR(S)$, dans ce cas, nous noterons $t \stackrel{!}{\rightarrow} S\ t'$.

Nous notons *S(t)* l'ensemble des réductions d'un terme t par S et *S!(t)* l'ensemble des formes normales d'un terme t pour S. Formellement, $S(t)=\{t' \in T_\Sigma(X) / t \xrightarrow{*}_S t'\}$ et $S!(t)=S(t)\cap IRR(S)$.

Une substitution σ est irréductible pour S si, pour tout x de $D(\sigma)$, le terme $\sigma(x)$ est irréductible pour S.

Un système de réécriture *clos* est tel que tous les membres gauches et droits de règles sont des termes clos ([ROSE73]).

Un système de réécriture est *linéaire (respectivement linéaire à gauche, linéaire à droite)* si tous les termes apparaissant dans les membres gauches et droits (respectivement les membres gauches, les membres droits) de règles sont linéaires.

Un système de réécriture est *noethérien* s'il n'existe pas de séquence infinie de réduction.

Un système de réécriture S est *Church-Rosser* si $\xleftrightarrow{*}_S \subset \xrightarrow{*}_S \circ \xleftarrow{*}_S$, *confluent* si $\xleftarrow{*}_S \circ \xrightarrow{*}_S \subset \xrightarrow{*}_S \circ \xleftarrow{*}_S$. Ces deux notions sont équivalentes ([NEWM42]).

Un système de réécriture est dit *convergent* s'il est confluent et noethérien.

Si $l \to r$ et $s \to t$ sont deux règles avec des ensembles de variables distincts, p une occurence de s tel que $s_{/p}$ ne soit pas réduit à une variable, et σ un unificateur le plus général de $s_{/p}$ et de l, alors le couple (u,v), avec $u=\sigma(t)$ et $v=\sigma(c)(\sigma(r))$ où c désigne le contexte associé à p dans s, est une *paire critique* obtenue à partir de ces deux règles. On note *cp(S)* l'ensemble des paires critiques des règles de S.

En utilisant le lemme du diamant ([NEWM42]) et le lemme sur les paires critiques ([KNBE70]), on prouve alors qu'un système de réécriture noethérien S est confluent si et seulement si $cp(S) \subset \xrightarrow{*}_S \circ \xleftarrow{*}_S$.

Figure 2.3.1.

Exemple de système de réécriture clos

Soit Σ={a,b,g,h,f} avec a et b d'arité 0, g et h d'arité 1 et f d'arité 2

Soit S = { g → a ; h → b } ; f(g*, h*) →* S(a, b)

Exemple de système de réécriture régulier

Soit Σ={0,inf,succ} où 0 est d'arité 0, succ d'arité 1 et inf d'arité 2

Soit S = { inf(succ(x), succ(y)) → succ(inf(x, y)) ; inf(x, 0) → 0 ; inf(0, x) → 0 }

S est linéaire et sans paires critiques.

Exemple de système sans paire critique et non confluent

Soit Σ={a,b,c,g,f} où a,b,c sont d'arité 0, g d'arité 1 et f d'arité 2

Soit S = { f(x, x) → a ; f(x, g(x)) → b ; c → g(c) }

S ne termine pas en raison de la règle c→g(c),

c n'a pas de forme normale pour S,

le terme f(c,c) en a deux, en effet, f(c, c) →! a et f(c, c) →! b.

Remarque: Sans l'hypothèse de terminaison, un système de réécriture peut être sans paires critiques et être non confluent (un exemple tiré de [HUET80] est donné dans la figure 2.3.1). Cependant, un système de réécriture *régulier* (linéaire à gauche et sans paires critiques) est confluent ([HUET80]) ,ceci, même sans l'hypothèse de terminaison. Cette remarque sera utilisée, dans la section 5, pour définir la notion de déterminisme associé à un automate à piles d'arbres.

Un système de réécriture est *confluent sur les termes clos* si la propriété de confluence restreinte aux termes clos est vérifiée. Un système de réécriture confluent est confluent sur les termes clos.

2.3.2 Systèmes de réécriture modulo

Nous utilisons, en général, les notations et définitions utilisées dans [DEJO90] et [JOKI86].

Soit R un système de réécriture sur $T_\Sigma(X)$ et E un ensemble d'équations sur $T_\Sigma(X)$.

La relation R/E.

La relation R/E simule la relation induite par R sur l'ensemble des classe d'équivalence définies par la relation de congruence $\overset{*}{\leftrightarrow}_E$.

Définition 1: $\rightarrow_{R/E} = \overset{*}{\leftrightarrow}_E \circ \rightarrow_R \circ \overset{*}{\leftrightarrow}_E$. C'est-à-dire que $t \rightarrow_{R/E} t'$ si et seulement si il existe un terme u, un terme v, une occurence p de u, un contexte c, une règle $l \rightarrow r$ de R, une substitution σ tels que: $t \overset{*}{\leftrightarrow}_E u$, $u=c(\sigma(l))$, $v=c(\sigma(r))$ et $v \overset{*}{\leftrightarrow}_E t'$.

Les notions usuelles de terminaison, de propriété Church-Rosser, de confluence, de forme normale s'étendent de façon naturelle à la relation R/E.

Définition 2: *R termine modulo E* si et seulement si R/E termine, c'est à-dire R termine modulo E s'il n'existe pas de séquence infinie de la forme: $t_0 \overset{*}{\leftrightarrow}_E t'_0 \rightarrow_R t_1 \overset{*}{\leftrightarrow}_E t'_1 \ldots t_n \overset{*}{\leftrightarrow}_E t'_n \rightarrow_R t_{n+1} \ldots$

On peut remarquer que pour que R termine modulo E, les équations de E doivent être complètes (toute variable qui occure dans un membre doit occurer dans l'autre) et il ne doit pas y avoir d'équations dont un membre soit réduit à une variable et l'autre soit non linéaire en cette variable (par exemple $x=f(x,x)$).

Définition 3:
R est Church-Rosser modulo E si et seulement si R/E est Church-Rosser, c'est-à-dire si et seulement si $\overset{*}{\leftrightarrow}_{R\cup E} \subset \overset{*}{\rightarrow}_{R/E} \circ \overset{*}{\leftrightarrow}_{E} \circ {}_{R/E}\overset{*}{\leftarrow}$,
c'est à dire encore si et seulement si, pour tout t,t' de $T_\Sigma(X)$, on a: $(t \overset{*}{\leftrightarrow}_{R\cup E} t') \Rightarrow (\exists t_1,\ t'_1 \in T_\Sigma(X),\ t \overset{*}{\rightarrow}_{R/E} t_1,\ t' \overset{*}{\rightarrow}_{R/E} t'_1$ et $t_1 \overset{*}{\leftrightarrow}_{E} t'_1)$.

Définition 4:
R est confluent modulo E si et seulement si R/E est confluent,
c'est à dire si et seulement si ${}_{R/E}\overset{*}{\leftarrow} \circ \overset{*}{\rightarrow}_{R/E} \subset \overset{*}{\rightarrow}_{R/E} \circ \overset{*}{\leftrightarrow}_{E} \circ {}_{R/E}\overset{*}{\leftarrow}$,
c'est à dire encore si et seulement si, pour tout t, t_1,t_2 de $T_\Sigma(X)$, on a: $(t \overset{*}{\rightarrow}_{R/E} t_1$ et $t \overset{*}{\rightarrow}_{R/E} t_2) \Rightarrow (\exists t',t'' \in T_\Sigma(X),\ t_1 \overset{*}{\rightarrow}_{R/E} t',\ t_2 \overset{*}{\rightarrow}_{R/E} t''$ et $t' \overset{*}{\leftrightarrow}_{E} t'')$.

On prouve sans difficultés, comme dans le cas d'un système de réécriture, que ces deux propriétés sont équivalentes.

Soit A un ensemble d'équations. On partitionne A en R et E en plaçant dans R les règles (obtenues par orientation des équations) qui assurent la terminaison.

Si R termine modulo E et si R est confluent modulo E alors $s \overset{*}{\leftrightarrow}_{A} t$ si et seulement si s et t ont leurs formes normales pour la relation R/E qui sont équivalentes pour la congruence $\overset{*}{\leftrightarrow}_{E}$.

On obtient donc un algorithme de décision pour la théorie équationnelle ayant pour ensemble d'axiomes A si les deux conditions suivantes sont vérifiées: Il faut que l'équivalence modulo E soit décidable et que la R/E réductibilité soit décidable.

Mais, même lorsque ces deux conditions sont vérifiées, l'utilisation de la relation R/E se révèle inefficace dès que les classes d'équivalence modulo E sont grandes (même finies), en effet, il faut à chaque pas de réécriture, parcourir la classe d'équivalence jusqu'à trouver un terme qui soit réductible par R, s'il existe.

L'idée est donc d'introduire une relation R^E plus faible vérifiant $R \subset R^E \subset R/E$. Je ne m'intéresse ici qu'à la relation R^E=R,E introduite dans [PEST81].

La relation R,E.

Définition 5: $t \rightarrow_{R,E} t'$ si et seulement si il existe une occurence p de t, une règle $l \rightarrow r$ de R, une substitution σ tels que $\sigma(l) \overset{*}{\leftrightarrow}_E t/_p$ et $t'=c(\sigma(r))$, où c désigne le contexte associé à p dans t.

La R,E réductibilité est décidable si on dispose d'un algorithme de filtrage modulo E. La relation R,E vérifie $R \subset R,E \subset R/E$.

Figure 2.3.2.

Exemple 1: R={ f(a,b) → c} ;E=C={ f(x,y) = f(y,x) }

f(b,a) est irréductible pour R.; f(b,a) $\rightarrow_{R,C}$ c et f(b,a) $\rightarrow_{R/C}$ c .

Exemple 2: R={ f(f(b,b),a) → c} ;E=AC={ f(x,y) = f(y,x) ; f(f(x,y),z) = f(x,f(y,z)) }

f(b,f(a,b)) est irréductible pour R.; f(b,f(a,b)) $\rightarrow_{R,AC}$ c .

f(f(a,b),f(c,b)) est irréductible pour R. et R,AC; f(f(a,b),f(c,b)) $\rightarrow_{R/AC}$ f(c,c) .

Le but est toujours d'obtenir un algorithme de décision pour une théorie équationnelle ayant pour ensemble d'axiomes A partitionné en R et E. La relation de réécriture $\rightarrow_{R,E}$ est plus faible (en général) que la relation $\rightarrow_{R/E}$. Il faut donc définir une propriété Church-Rosser associée à cette relation qui soit plus forte de façon à faire coïncider les formes normales calculées par ces deux relations.

Définition 6:
R,E est Church-Rosser modulo E si $\overset{*}{\leftrightarrow}_{R\cup E} \subset \overset{*}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E \circ {}_{R,E}\overset{*}{\leftarrow}$,
c'est à dire encore si et seulement si, pour tout t,t' de $T_\Sigma(X)$, on a: $(t \overset{*}{\leftrightarrow}_{R\cup E} t') \Rightarrow (\exists t_1, t'_1 \in T_\Sigma(X),\ t \overset{*}{\rightarrow}_{R,E} t_1,\ t' \overset{*}{\rightarrow}_{R,E} t'_1$ et $t_1 \overset{*}{\leftrightarrow}_E t'_1)$.

Lorsque la relation R termine modulo E (c'est-à-dire R/E termine), cette propriété est démontrée équivalente à deux propriétés locales de la relation R,E.

Définition 7: *R,E est localement confluent avec E modulo R* si et seulement si ${}_{R,E}\leftarrow \circ \rightarrow_R \subset \overset{*}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E \circ {}_{R,E}\overset{*}{\leftarrow}$.

R,E est localement cohérent modulo E si et seulement si ${}_{R,E}\leftarrow \circ \leftrightarrow_E \subset \overset{*}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E \circ {}_{R,E}\overset{*}{\leftarrow}$.

Théorème 1: [JOKI86] *Si R termine modulo E, R,E est Church-Rosser modulo E si et seulement si R,E est localement confluent modulo E avec R et R,E est localement cohérent modulo E.*

Nous supposons maintenant qu'il existe un algorithme fini et complet d'unification pour E.

Définition 8: Soit E un ensemble d'équations et R un système de réécriture sur $T_\Sigma(X)$. L'ensemble $cp_E(R)$ des paires critiques de R modulo E est l'ensemble des couples (u,v) défini par:

*/ Soit $g \rightarrow d$ et $l \rightarrow r$ deux règles de R (avec des ensembles de variables disjoints), soit p une occurence de g ($g_{/p}$ n'est pas réduit à une variable), soit σ un unificateur le plus général modulo E tel que $\sigma(g)_{/p} =_E \sigma(l)$, soit $g=c(g_{/p})$, $u=\sigma(d)$ et $v=\sigma(c)(\sigma(r))$.

*/ Soit s=t une équation de E et $l \rightarrow r$ une règle de R, soit p une occurence de s ($s_{/p}$ sous-terme propre de s non réduit à une variable), soit σ un unificateur le plus général modulo E vérifiant $\sigma(s)_{/p} =_E \sigma(l)$, soit $s=c(s_{/p})$, $u=\sigma(t)$ et $v=\sigma(c)(\sigma(r))$.

Théorème 2: [JOKI86] *Si R termine modulo E et s'il existe un algorithme fini et complet d'unification pour $=_E$ alors R,E est Church-Rosser modulo E si et seulement si les ensembles $cp_E(R)$ et $ex_E(R)$ est contenu dans* $\overset{!}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E \circ {}_{R,E}\overset{!}{\leftarrow}$.

Les preuves des théorèmes 1 et 2 peuvent être trouvées dans [JOKI86]. Ces résultats sont des généralisations des résultats obtenus par G.Huet ([HUET80], cas des systèmes de réécriture linéaires à gauche) et G.Peterson et M.Stickel ([PEST81], équations linéaires). Ces résultats sont à la base des algorithmes de complétion en réécriture équationnelle (en particulier pour le cas AC).

3 FORETS RECONNAISSABLES ET SYSTEMES DE REECRITURE

3.1 Motivations de cette étude

Dans toute cette section, étant donnés une forêt reconnaissable F et un système de réécriture S, on note IRR(S) l'ensemble des termes clos irréductibles pour S, S(F) l'ensemble des transformés des termes de F par S et S!(F)=S(F)∩IRR(S) l'ensemble des formes normales des termes de F pour S.

Citons tout d'abord le résultat suivant.

Proposition: *Pour tout système de réécriture S linéaire à gauche, l'ensemble IRR(S) des termes irréductibles pour S est une forêt reconnaissable.*

Une preuve de ce résultat peut être trouvée dans [BOGA85]. L'idée de la preuve est que l'on peut, dans le cas d'un système de réécriture linéaire à gauche, caractériser par un ensemble fini d'arbres le fait d'être unifiable ou pas avec un membre gauche de règle.

Signalons également la preuve de Z.Fülop and S.Vàgvölgyi [FUVA89] dans laquelle les auteurs donnent une caractérisation des ensembles de termes qui sont formes irréductibles d'un système de réécriture linéaire à gauche comme langages reconnus par une certaine classe d'automates.

Ce résultat s'étend, sans difficultés, au cas d'algèbres sortées en utilisant la clôture de la classe des forêts reconnaissables par les opérations booléennes et la définition de sortes en termes d'automates d'arbres.

Pour les applications éventuelles de ce travail, nous rappelons les liens entre sortes et forêts reconnaissables et montrons comment cette remarque s'applique pour résoudre le problème de l'inductive réductibilité dans le cas linéaire.

Fait: *Il y a identité entre la notion d'algèbre avec sortes ordonnées et celle d'automate fini d'arbres.*

Différentes illustrations de ce fait ont été faites dans l'introduction et les préliminaires. Une justification du fait peut être trouvée dans [COMO90].

Fait: *Le "point de vue" théorie des automates nous donne, de façon immédiate, un algorithme pour tester l'inductive réductibilité d'un terme linéaire dans le cas des systèmes de réécriture linéaires à gauche.*

- Cet algorithme s'adapte au cas sorté

- Par précompilation du système de réécriture, on obtient un algorithme linéaire en temps par rapport au terme linéaire t.

Preuve: Soit S un système de réécriture linéaire à gauche, soit t un terme, t est *inductivement réductible pour S* si et seulement si, pour toute substitution close σ, $\sigma(t)$ est réductible pour S.

Phase 1: Précompilation du problème.

donnée: Système de réécriture linéaire à gauche S.

étape 1: Pour toute règle $l_i \rightarrow r_i$ de S, on construit l'automate reconnaissant la forêt reconnaissable F_i des termes contenant au moins une occurrence de l_i (c'est-à-dire des termes réductibles par la règle considérée.

étape 2: On construit l'automate reconnaissant l'union des forêts F_i précédemment définies (clôture par union de REC). On obtient ainsi un automate reconnaissant la forêt RED(S) des termes réductibles par S.

étape 3: Dans le cas d'algèbres sortées, on considère alors l' intersection avec les automates de vérification de sortes.

étape 4: On rend alors déterministe l'automate obtenu (compilation du problème). Nous appelons A l'automate déterministe obtenu reconnaissant RED(S).
Finphase1.

Phase 2: Inductive réductibilité d'un terme linéaire t pour S.

donnée: Soit $t=t(x_1,...,x_n)$ un terme *linéaire,*

étape 1: Soit l'ensemble F(t) des termes $t(t_1,...,t_n)$, pour tous $t_1,...,t_n$ de T_Σ. Le terme t étant linéaire, F(t) est une forêt reconnaissable en utilisant la propriété de clôture de la classe des forêts reconnaissables par substitution. On construit donc un automate A(t) associé à t qui, de plus, est de taille linéaire par rapport à t (il suffit de reconnaître t, la construction est immédiate par automate descendant).

Rappelons également que ce résultat est faux pour t non linéaire, il suffit, en effet, de considérer l'exemple suivant:

Exemple: $\Sigma=\{f,g,a\}$ avec f,g et a d'arités respectives 2, 1 et 0. Soit le terme t=f(x,x), l'ensemble F(t) des termes de la forme f(t',t') n'est pas reconnaissable.

étape 2: t est alors inductivement réductible par rapport à S si et seulement si F(t) est inclus dans RED(S), soit encore, si et seulement si, $F(t)\cap \overline{RED(S)}=\varnothing$, ce qui peut être décidé en temps linéaire par rapport à t.
❑ *Finpreuvefait.*

Nous rappelons que le cas général a été prouvé décidable par D.A.Plaisted ([PLAI85]) et que différents algorithmes ont été donnés par D.Kapur, P.Narendran & H.Zhang (avec une étude de complexité, [KNZ87]), par H.Comon ([COMO88]), par E.Kounalis ([KOUN90]).

Nous nous intéresserons également à l'étude de la conservation de la reconnaissabilité par un système de réécriture.

Définition: Un système de réécriture *préserve la reconnaissabilité* si et seulement si pour toute forêt reconnaissable F, l'ensemble S(F) des transformés des termes de F par S est une forêt reconnaissable.

B.Courcelle [COUR89] a mis en évidence l'importance de cette propriété, pour les congruences engendrées par un ensemble d'équations E, dans l'étude des parties reconnaissables des algèbres libres $T_\Sigma(X)_{/=E}$.

Cette propriété est également importante pour la résolution des différents problèmes d'accessibilité.

Définition: Soit S un système de réécriture, le *problème d'accessibilité du premier ordre* est: étant donné deux termes clos t et t', peut-on réécrire t en t' par S.

Définition: Différents *problèmes d'accessibilité du second ordre* sont: étant donné deux forêts reconnaissables F et F', l'ensemble S(F) des transformés des termes de F par S est-il inclus dans F'?, existe-t-il un terme de F qui se transforme en un terme de F' par S?...

En effet, si S préserve la reconnaissabilité, on en déduit des algorithmes pour la résolution des différents problèmes d'accessibilité en utilisant les algorithmes classiques sur les forêts reconnaissables (clôture de REC par les opérations booléennes et décision du vide).

Une dernière application importante de ce type d'étude est le résultat obtenu par M.Dauchet et S.Tison. Ils ont en effet prouvé que la théorie de la réécriture close est décidable. L'idée de la preuve de ce résultat est de coder de façon canonique la transformation associée à un système de réécriture clos par une relation définissable par automates finis (voir [DATI90]).

Dans la section 3.2, nous étudions la décidabilité des différentes questions que l'on peut se poser sur le lien entre systèmes de réécriture et forêts reconnaissables (préservation de la reconnaissabilité par un ensemble d'équations, par un système de réécriture convergent,...). Les problèmes traités peuvent être considérés d'un point de vue programmation en remplaçant forêt reconnaissable par sorte.

Dans la section 3.3, nous étudions (des fragments) de théorie des algèbres de termes clos modulo des congruences engendrées par des ensembles d'équations. Nous considérons des ensembles d'équations E pour lesquels il existe un système de réécriture convergent S, vérifiant $\overset{*}{\leftrightarrow}_S = \overset{*}{\leftrightarrow}_E$, et, tels que les ensembles de formes normales des termes de toute forêt reconnaissable soient encore reconnaissables.

3.2 Résultats généraux de décidabilité

Q1: *Donnée: système de réécriture S.*
Question: pour toute forêt reconnaissable F, S(F) est reconnaissable.
Résultat: Ce problème est indécidable.

Preuve: Ce résultat a été prouvé par S.Tison dans [CDT89]. On peut remarquer que cette preuve est faite dans le cas d'une relation de congruence et donc pour un système de réécriture non noethérien. Le même problème reste posé en prenant pour donnée un système de réécriture convergent.

Q2: *Donnée: système de réécriture S et une forêt reconnaissable F.*
Question: S(F) est reconnaissable.
Résultat: Ce problème est indécidable.

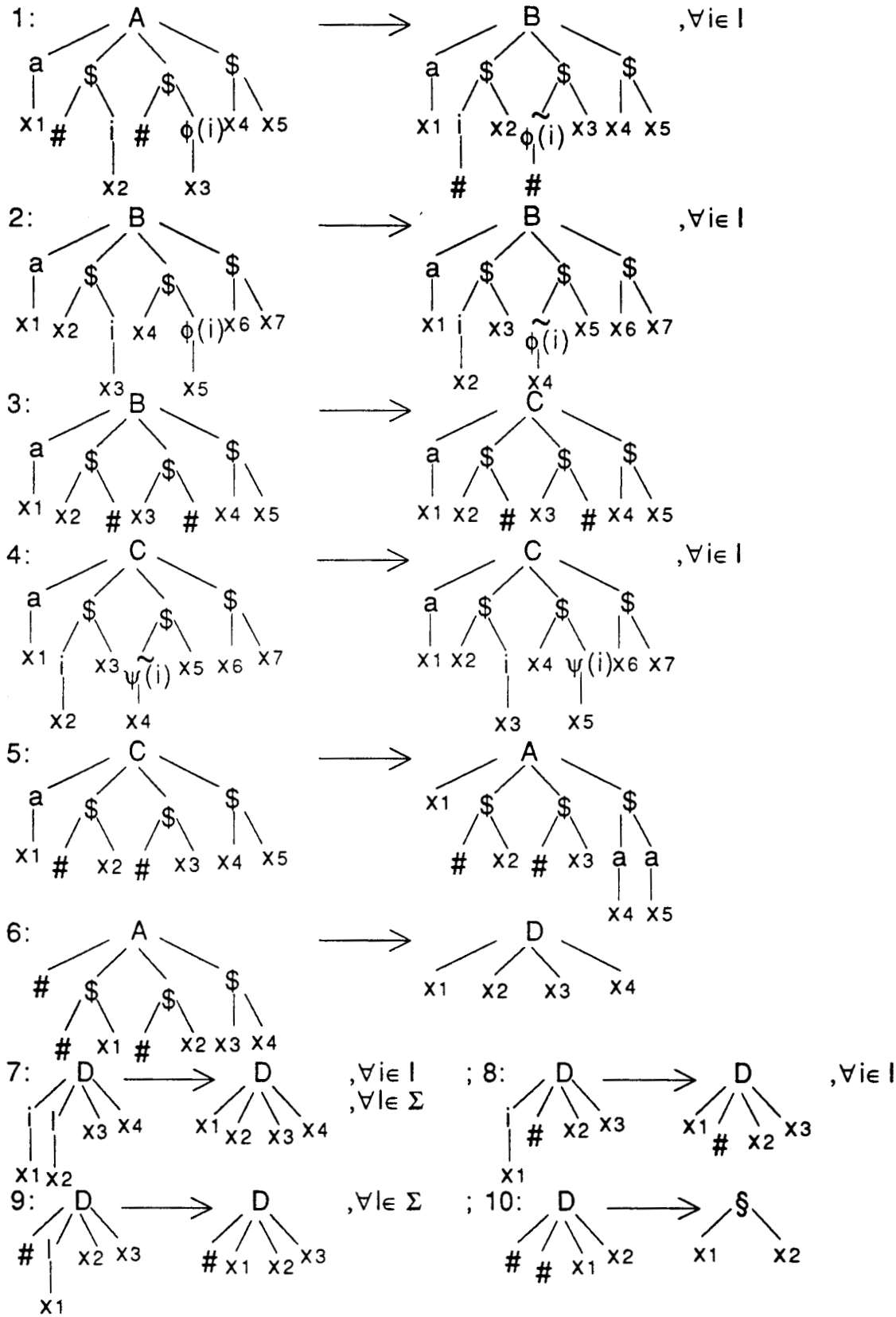
Preuve: La construction utilisée par S.Tison convient également dans ce cas. (voir [CDT89]).

Q3: *Donnée: système de réécriture S confluent noethérien.*
Question: pour toute forêt reconnaissable F, S!(F) est reconnaissable.
Résultat: Ce problème est indécidable.

Preuve: Soit $I=\{1,...,n\}$ et Σ un alphabet. Soit ϕ et ψ deux morphismes de I^* dans Σ^*. Le problème de Post $PCP=(\Sigma,n,\phi,\psi)$ correspondant est l'existence d'un mot m dans I^+ tel que $\phi(m)=\psi(m)$. De façon classique I et Σ peuvent être mis en correspondance avec des alphabets unaires que nous noterons également I et Σ.

Soit $\Delta=I\cup\Sigma\cup\{\#,0,a,\$,\S,A,B,C,D\}$ où # et 0 sont des nouvelles lettres d'arité 0, \$ et § des nouvelles lettres d'arité 2, A, B, C et D de nouvelles lettres d'arité 4. Pour tout mot m, nous notons $\tilde{m}$ l'image miroir de m. Soit R le système de réécriture linéaire défini par les méta-règles de la figure 3.2.a.

Figure 3.2.a



Nous construisons, à partir de R, le système de réécriture S. Le but de cette construction est que tout terme de sommet A, B, C ou D qui ne filtre pas un membre gauche de règle de R se réécrive en 0 par S. Cette construction est possible car R est linéaire à gauche. Nous ne donnons ici que les règles de S de racine A, en effet, on définit facilement, par la même méthode, toutes les autres règles de S.

$A(l(x_1,\ldots,x_n),x,y,z)\to 0$, $\forall l\in\Delta$, $l\neq\#$ et $l\neq a$.
$A(a(x),l(x_1,\ldots,x_n),y,z)\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(a(x),y,l(x_1,\ldots,x_n),z)\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(a(x),y,z,l(x_1,\ldots,x_n))\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(a(x),\$(l(x_1,\ldots,x_n),y),\$(z,t),\$(x',y'))\to 0$, $\forall l\in\Delta$, $l\neq\#$.
$A(a(x),\$(y,z),\$(l(x_1,\ldots,x_n),t),\$(x',y'))\to 0$, $\forall l\in\Delta$, $l\neq\#$.
$A(a(x),\$(\#,l(x_1,\ldots,x_n)),\$(\#,y),\$(x',y'))\to 0$, $\forall l\in\Delta$, $l\notin I$.
$A(a(x),\$(\#,i(y)),\$(\#,t(x_1,\ldots,x_n)),\$(x',y'))\to 0$, $\forall i\in I$, $h(t)\leq h(\Phi(i))$ et t ne filtre pas $\Phi(i)(x)$.
$A(\#,l(x_1,\ldots,x_n),x,y)\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(\#,x,l(x_1,\ldots,x_n),y)\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(\#,x,y,l(x_1,\ldots,x_n))\to 0$, $\forall l\in\Delta$, $l\neq\$$.
$A(\#,\$(l(x_1,\ldots,x_n),x),\$(y,z),\$(x',y'))\to 0$, $\forall l\in\Delta$, $l\neq\#$.
$A(\#,\$(x,y),\$(l(x_1,\ldots,x_n),z),\$(x',y'))\to 0$, $\forall l\in\Delta$, $l\neq\#$.

S est donc constitué des règles de R, des règles précédemment définies et on définit, de plus, dans S les règles suivantes: $i(0)\to 0$, pour tout i de I; $l(0)\to 0$, pour tout l de Σ; $a(0)\to 0$; $\$(0,x)\to 0$; $\$(x,0)\to 0$; $\S(0,x)\to 0$; $\S(x,0)\to 0$. Par cette construction, nous obtenons un système de réécriture S qui est linéaire à gauche.

Idée intuitive des définitions de R et S: L'idée de la construction de R est la suivante: Partant d'un arbre de racine A, les règles 1 et 2 permettent de vérifier l'égalité d'un mot u et de $\Phi(m)$, en recopiant les images miroir des mots u et m. Les règles 3 et 4 vérifient l'égalité de u et de $\psi(m)$ en restaurant la configuration initiale avec un arbre de racine C. On peut alors revenir sur un arbre de racine A par la règle 5 en dupliquant un a. Dans ces conditions, on pourra générer à partir de $a^n(\#)$, un arbre de la forme $\$(a^n(\#),a^n(\#))$ pour sortir des reconnaissables.

Mais, il nous faut obtenir le résultat pour toute forêt reconnaissable F. Le but de la construction de S est d'envoyer tout terme ne satisfaisant pas les conditions précédentes sur 0. Les règles 6 à 10 de R ont été ajoutées pour supprimer d'éventuels phénomènes de non reconnaissabilité parasites (la boucle constituée par l'application des règles 1 à 5 n'étant pas réalisée en totalité).

Fait 1: *S est noethérien.*

Preuve: Toutes les règles sont strictement décroissantes en taille (nombre de noeuds) à l'exception des règles 1 à 5 de R. Pour ces règles, on ne peut avoir un cycle que lorsqu'on passe par deux termes t et t' de même sommet A en appliquant, dans l'ordre, les règles 1, 2, 3, 4 et 5. Un tel cycle fait décroître strictement le nombre de a sur la branche la plus à gauche du terme t, ce qui assure la terminaison dans ce cas.❑

Fait 2: *S est confluent.*

Preuve: S est noethérien donc il suffit de prouver que pour toute paire critique (u,v), on a $u \xrightarrow{*}_S \circ \xleftarrow{*}_S v$. Soit $l \to r$ et $g \to d$ les deux règles de réécriture (avec des ensembles de variables distincts), soit o l'occurence de g et σ l'unificateur le plus général de $g_{/o}$ et de l, on pose $u=\sigma(d)$ et $v=\sigma(c).\sigma(r)$ avec $g=c(g_{/p})$.

Cas d'une paire critique où g a pour racine A, B, C ou D et $l \to r$ l'une des règles $i(0) \to 0, \ldots, \S(0,x) \to 0$. En examinant cas par cas, on peut prouver que, pour toute paire critique (u,v) associée, on a $u \xrightarrow{*}_S 0 \xleftarrow{*}_S v$.

Exemple: Considérons la règle 1 de R et la règle $i(0) \to 0$. On obtient $u=B(a(x),\$(i(\#),0),\$(\Phi(i)(\#),z),\$(t,u))$ et $v=A(a(x),\$(\#,0),\$(\#,\Phi(i)(z)),\$(t,u))$. On a $u \to_S 0$ par la règle $B(a(x),\$(y,0),\$(z,t),\$(u,v)) \to 0$ et $v \to_S 0$ par la règle $A(a(x),\$(\#,0),\$(\#,y),\$(u,v)) \to 0$.

Cas d'une paire critique où les deux règles ont pour membre gauche un terme de racine A, B, C ou D.

Le système de réécriture R est sans paire critique donc il suffit d'étudier le cas de deux règles de S qui ne sont pas dans R et le cas d'une règle de R avec une règle de S qui n'est pas dans R.

Ici encore, une étude de tous les cas permet de prouver que pour toute paire critique (u,v), on a $u \xrightarrow{*}_S 0 \xleftarrow{*}_S v$.

Exemple: Règle 2 de R et la règle $A(a(x_1),B(x_2,x_3,x_4,x_5),x_6,x_7)\rightarrow 0$ de S. On obtient u=0 et $v=A(a(x_1),B(a(x),\$(i(y),z),\$(\Phi(i)(t),u),\$(v,w)),x_6,x_7)$ et $v\rightarrow_S 0$ par la même règle de S.

Fait3: *Soit (P): Pour toute forêt reconnaissable F, l'ensemble S!(F) des formes normales des termes de F pour S est reconnaissable.*
S satisfait (P) si et seulement si le problème de Post n'a pas de solution.

Preuve:
Preuve de $\Rightarrow$*:* Supposons que PCP possède une solution et considérons la forêt reconnaissable F définie comme suit:
$F=\{ A(a^n(\#), \$(\#,u(\#)), \$(\#,v(\#)), \$(\#,\#)) / u\in I^+, v\in \Sigma^*, n>0 \}$.
Soit $t=A(a^n(\#), \$(\#,u(\#)), \$(\#,v(\#)), \$(\#,\#))$ un terme de F, déterminons la forme normale de t pour S.

premier cas: $v=\Phi(u)$ et $v=\Psi(u)$. Nous avons alors:
$t \xrightarrow{*}_S t_1=B(a^n(\#), \$(\tilde{u}(\#),\#), \$(\tilde{v}(\#),\#), \$(\#,\#))$ en utilisant les règles 1 et 2 de R car $v=\Phi(u)$.
$t_1 \xrightarrow{*}_S t_2=C(a^n(\#), \$(\#,u(\#)), \$(\#,v(\#)), \$(\#,\#))$ en utilisant les règles 3 et 4 de R car $v=\Psi(u)$.
$t_2 \rightarrow_S t_3=A(a^{n-1}(\#), \$(\#,u(\#)), \$(\#,v(\#)), \$(a(\#),a(\#)))$ en utilisant la règle 5 de R.
Soit n-1=0, on peut alors appliquer la règle 6 de R, soit n-1>0, on peut alors réitérer cette réécriture. On obtient donc que, dans ce cas, $t \xrightarrow{*}_S D(u(\#),v(\#),a^n(\#),a^n(\#))$. Comme, de plus, u appartient à I^+ et v à Σ^*, en utilisant les règles 7, 8, 9 et 10 de R, nous obtenons $t\xrightarrow{*}_S \S(a^n(\#),a^n(\#))$. Le terme $\S(a^n(\#),a^n(\#))$ est donc la forme normale de t pour le système de réécriture S.

deuxième cas: $v \neq \Phi(u)$ ou $v \neq \Psi(u)$. Par construction du système de réécriture S, on ne pourra effacer un a sur la branche la plus à gauche de t que si $v=\Phi(u)$ et $v=\Psi(u)$. t se réécrit donc en un terme de racine A, B ou C qui n'est filtré par aucune des règles 1, 2, 3, 4 ou 5 de R, par construction de S, ce terme se réécrira donc en 0, et donc, $t \xrightarrow{*}_S 0$.

PCP ayant une solution, nous avons $S!(F)=\{0\} \cup \{\S(a^n(\#),a^n(\#)) / n>0\}$ qui n'est pas une forêt reconnaissable.❑ *finpreuve de* $\Rightarrow$.

Preuve de $\Leftarrow$*:* Nous supposons donc que PCP n'a pas de solution.

Soit F une forêt reconnaissable, on peut décomposer F en une union disjointe de forêts reconnaissables par $F=F_0 \cup F_{ABCD} \cup F'$ où:
F_0 désigne l'ensemble des termes de F contenant au moins une occurrence de 0,
F_{ABCD} l'ensemble des termes de F ne contenant aucune occurrence de 0 mais contenant au moins une occurence de A, B, C ou D et
F' l'ensemble des termes de F sans occurence de 0, A, B, C et D.

F_0, F_{ABCD} et F' sont des forêts reconnaissables (F est reconnaissable et clôture par intersection). $S!(F_0)=\{0\}$, en effet, pour tout terme t ayant en feuillage un 0, t se réécrit en 0 en utilisant les règles de S-R. Donc $S!(F_0)$ est reconnaissable. $S!(F')=F'$ car tout terme de F' est irréductible pour S donc S!(F') est reconnaissable. Il nous reste donc à montrer que $S!(F_{ABCD})$ est reconnaissable.

Soit F une forêt reconnaissable telle que tout terme de F ne contient aucune occurence de 0 et au moins une occurence de A, B, C ou D. Nous allons étudier la forme normale d'un terme t de F de racine A, B, C et D. Nous rappelons que si l'on réécrit un terme t de racine A, B, C ou D, si ce terme est en forme normale pour R, alors, par construction de S, ce terme se réécrit en 0 par S.

Cas 1: t est de racine A, soit $t=A(t_1,t_2,t_3,t_4)$.
Cas 1.1: $t_1 \notin a^*(\#)$, on a alors $t \xrightarrow{*}_S 0$.
Cas 1.2: $t_1 \in a^+(\#)$, par définition de S, on ne peut réécrire t en un terme de racine A par des réécritures à l'occurence ε que si PCP a une solution, donc, dans ce cas, nous avons encore $t \xrightarrow{*}_S 0$.

Cas 1.3: $t_1=\#$, si $t=A(\#,\$(\#,u),\$(\#,v),\$(p,q))$ alors $t \rightarrow_S D(u,v,p,q)$, si, de plus, $u \in I^*(\#)$ et $v \in \Sigma^*(\#)$ alors $D(u,v,p,q) \stackrel{*}{\rightarrow}_S \S(p,q)$. Dans tous les autres cas, par définition de S, on a $t \stackrel{*}{\rightarrow}_S 0$.
Conclusion du cas 1: Si t est un arbre de racine A, si $t=A(\#,\$(\#,u),\$(\#,v),\$(p,q))$ avec $u \in I^*(\#)$ et $v \in \Sigma^*(\#)$ alors $t \stackrel{*}{\rightarrow}_S \S(p,q)$ et dans tous les autres cas, on a $t \stackrel{*}{\rightarrow}_S 0$.

Cas 2: t est de racine B, en faisant une étude cas par cas, on montre que si $t=B(a(\#),\$(\tilde{m}_1(\#),m_2(\#)),\$(\tilde{u}_1(\#),u_2(\#)),\$(p,q))$ avec m_1 et m_2 dans I^*, u_1 et u_2 dans Σ^*, $u_2=\Phi(m_2)$ et $u_1u_2=\Psi(m_1m_2)$ alors $t \rightarrow_S \S(a(p),a(q))$ et dans tous les autres cas, on a $t \stackrel{*}{\rightarrow}_S 0$.

Cas 3: t est de racine C, deux cas peuvent se produire: si $t=C(a(\#),\$(m_1(\#),m_2(\#)),\$(u_1(\#),u_2(\#)),\$(p,q))$ avec m_1 et m_2 dans I^*, u_1 et u_2 dans Σ^* tels que $u_1=\Psi(m_1)$ alors $t \stackrel{*}{\rightarrow}_S \S(a(p),a(q))$ et dans tous les autres cas, on a $t \stackrel{*}{\rightarrow}_S 0$.

Cas 4: t est de racine D, si $t=D(m(\#),u(\#),p,q)$ avec m dans I^* et u dans Σ^* alors $t \stackrel{*}{\rightarrow}_S \S(p,q)$ et dans tous les autres cas, on a $t \stackrel{*}{\rightarrow}_S 0$.

Soit F une forêt reconnaissable telle que tout terme de F ne contient aucune occurence de 0 et au moins une occurence de A, B, C ou D. On peut construire un automate déterministe $M=(\Delta,Q_M,Q_{Mf},R_M)$ tel que il existe des ensembles d'états Q_I et Q_Σ tels que: $\forall t \in I^*(\#)$, $(t \stackrel{*}{\rightarrow}_M q)$ si et seulement si $(q \in Q_I)$ et $\forall t \in \Sigma^*(\#)$, $(t \stackrel{*}{\rightarrow}_M q)$ si et seulement si $(q \in Q_\Sigma)$. On construit un automate $N=(\Delta,Q_N,Q_{Nf},R_N)$ reconnaissant S!(F). L'ensemble des règles R_N de N est constitué des règles définies ci-dessous:

- La règle $0 \rightarrow q_0$, q_0 est un nouvel état.

- Toutes les règles de R_M dont le membre gauche a une racine différente de A, B, C et D.

- Pour les règles dont le membre gauche est de racine A:
Si, dans M, on a une règle $A(q_\#,q_1,q_2,q_3) \rightarrow q$ avec $\# \rightarrow_M q_\#$ telle qu'il existe des états q', q", q_4, q_5 tels que $\$(q_\#,q') \rightarrow q_1$ et $q' \in Q_I$, $\$(q_\#,q") \rightarrow q_2$ et $q" \in Q_\Sigma$, $\$(q_4,q_5) \rightarrow q_3$ alors on définit dans N la règle $\S(q_4,q_5) \rightarrow q$.

Dans tous les autres cas, on définit dans N la règle $q_0 \rightarrow q$ pour toute règle $A(q_1,q_2,q_3,q_4) \rightarrow q$ ne vérifiant pas les hypothèses précédentes.

- Pour les règles dont le membre gauche est de racine B:
Si, dans M, on a une règle $B(q_1,q_2,q_3,q_4) \rightarrow q$ avec $a(\#) \stackrel{*}{\rightarrow}_M q_1$ telle qu'il existe des états $q_I,q'_I,q_\Sigma,q'_\Sigma,q_5,q_6$ vérifiant $\$(q_I,q'_I) \rightarrow q_2$ et $q_I,q'_I \in Q_I$, $\$(q_\Sigma,q'_\Sigma) \rightarrow q_3$ et $q_\Sigma,q'_\Sigma \in Q_\Sigma$, $\$(q_5,q_6) \rightarrow q_4$ tels qu'il existe des mots $\tilde{m}_1(\#)$ et $m_2(\#)$ dans $I^*(\#)$, $\tilde{u}_1(\#)$ et $u_2(\#)$ dans $\Sigma^*(\#)$ avec $\tilde{m}_1(\#) \stackrel{*}{\rightarrow}_M q_I$, $m_2(\#) \stackrel{*}{\rightarrow}_M q'_I$, $\tilde{u}_1(\#) \stackrel{*}{\rightarrow}_M q_\Sigma$, $u_2(\#) \stackrel{*}{\rightarrow}_M q'_\Sigma$, $u_2=\Phi(m_2)$ et $u_1u_2=\Psi(m_1m_2)$ alors on définit dans N les règles $a(q_5) \rightarrow q'_5$, $a(q_6) \rightarrow q'_6$, $\S(q'_5,q'_6) \rightarrow q$ où q'_5,q'_6 sont de nouveaux états.
Dans tous les autres cas, on définit dans N la règle $q_0 \rightarrow q$ pour toute règle $B(q_1,q_2,q_3,q_4) \rightarrow q$ ne vérifiant pas les hypothèses précédentes.

La construction est identique pour les règles dont le membre gauche est de racine C et D, il suffit d'adapter cette construction aux cas 3 et 4 précédemment étudiés.

Cette construction définit Q_N et R_N. Si $F \neq \emptyset$ et si il existe une règle $I(q_1,q_2,q_3,q_4) \rightarrow q$ avec I égal à A, B ou C ou I égal à D avec q_2 n'appartenant pas à Q_I ou q_3 n'appartenant pas à Q_Σ telle qu'un état final soit accessible à partir d'une configuration contenant l'état q alors 0 appartient à S!(F) et donc on pose $Q_{Nf}=Q_{Mf} \cup \{q_0\}$, sinon on pose $Q_{Nf}=Q_{Mf}$.

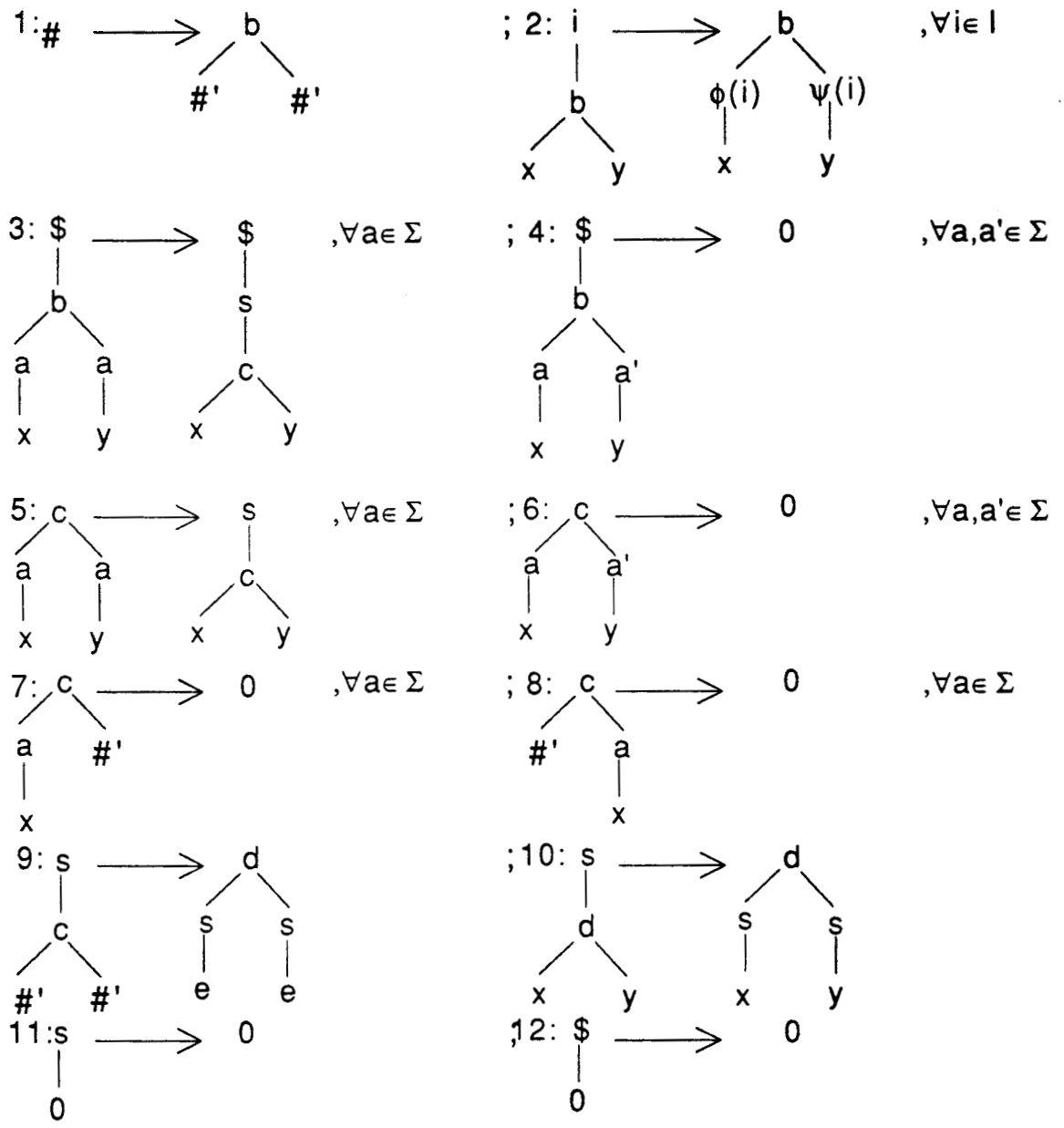
On peut alors montrer que $S!(F)=L(N)$.❑

Q4: *Donnée: système de réécriture S et une forêt reconnaissable F.*
Question: S!(F) est reconnaissable.
Résultat: Ce problème est indécidable.

Preuve: Nous utilisons les mêmes notations que dans la preuve précédente. Soit $\Delta = I \cup \Sigma \cup \{\#,\#',e,s,\$,b,c,d\}$ où $\#,\#',e$ sont des nouvelles lettres d'arité 0, $s,\$$ des nouvelles lettres d'arité 1 et b,c,d des nouvelles lettres d'arité 2.

Soit S le système de réécriture sur T_Δ défini dans la figure 3.2 et soit F la forêt reconnaissable F={ $(m(#)) / m∈ I+ }.

Figure 3.2.b



Le système de réécriture S est linéaire et sans paire critique donc S est confluent (voir [HUET80]). On peut également vérifier que S est noethérien.

Soit un terme $t=\$(m(\#))$ de F. Seule la règle 1 peut être appliquée et t se réécrit en $t_1=\$(m(b(\#',\#'))$. Pour ce terme t_1, on ne peut appliquer que la règle 2 tant que toutes les lettres de I du mot m n'ont pas été réécrites. t_1 se réécrit donc en $t_2=\$(b(\phi(m)(\#'),\psi(m)(\#')))$.

Premier cas: $\phi(m)$ et $\psi(m)$ sont égaux, dans ce cas, en utilisant les règles 3 et 5, t_2 se réécrit en $\$(s^p(c(\#',\#')))$ où p désigne la longueur de $\phi(m)$, puis par les règles 9 et 10 en $\$(d(s^p(e),s^p(e)))$ qui est irréductible pour S.

Deuxième cas: $\phi(m)$ et $\psi(m)$ sont distincts, alors par l'une des règles 4, 6, 7 ou 8, un 0 sera engendré et le terme pourra alors être réduit par les règles 11 et 12 en 0 qui est irréductible pour S.
On montre ainsi que les termes irréductibles pour S que l'on peut obtenir à partir des termes de F sont donc sous l'une de ces deux formes. Réciproquement ces termes sont irréductibles pour S et peuvent être obtenus comme réductions par S des termes de F.

Donc $S!(F) = \{\$(d(s^p(e),s^p(e))) / \phi(m)=\psi(m) \text{ et } |\phi(m)|=p\} \cup \{0\}$.

Si le problème de Post PCP n'a pas de solution, alors $S!(F)=\{0\}$ qui est reconnaissable.

Réciproquement, si le problème de Post PCP a une solution alors $S!(F) = \{\$(d(s^p(e),s^p(e))) / p \in I\} \cup \{0\}$ avec I non fini (si m est solution alors m^k est solution pour tout k entier) et donc S!(F) n'est pas reconnaissable.

On en déduit donc que S!(F) est reconnaissable si et seulement si le problème de Post n'a pas de solution.❑

Nous aurions pu utiliser pour cette preuve le système de réécriture S défini dans la preuve du problème Q3 et la forêt reconnaissable F utilisée pour la preuve de la première implication dans cette même preuve. Nous avons préféré présenter une construction plus simple.

Q5: *Donnée: système de réécriture S.*
Question: IRR(S) est reconnaissable.
Résultat: Ce problème est ouvert.

Notons que d'après la proposition citée dans la Section 3.1, le problème ne se pose que dans le cas d'un système de réécriture non linéaire à gauche. La résolution de ce problème (difficile) est à faire, car dans le cas où IRR(S) est reconnaissable, on peut en déduire un algorithme pour tester l'inductive réductibilité d'un terme linéaire t.

Q6: *Donnée: homomorphisme ϕ et une forêt reconnaissable F.*
Question: $\phi(F)$ est reconnaissable.
Résultat: Ce problème est ouvert.

Rappelons qu'un homomorphisme linéaire conserve la reconnaissabilité, donc que le problème n'est consistant que pour les homomorphismes non linéaires.

On peut noter le résultat suivant qui lie les questions Q5 et Q6.

Proposition: *La décidabilité de Q6 implique la décidabilité de Q5*

Preuve:

Soit $S=\{ l_i \rightarrow r_i / i \in I, l_i, r_i \in T_\Sigma(X)\}$ un système de réécriture sur T_Σ.

Pour toute règle $l_i \rightarrow r_i$ de S, on associe une nouvelle lettre a_i ayant pour arité le nombre de variables occurant dans l_i.

On considère l'alphabet gradué $\Delta = \Sigma \cup A$ avec $A=\{a_i / i \in I\}$ et le morphisme ϕ de T_Δ dans T_Σ dont la restriction à Σ est l'identité et défini sur A par $\phi(a_i(x_1,\ldots,x_{r(a_i)}))=l_i$.

Soit l'ensemble F des arbres de T_Δ qui contiennent au moins une occurence d'une lettre de A. F est de façon évidente une forêt reconnaissable de T_Δ. De plus $\phi(F)$ est l'ensemble des arbres de T_Σ qui sont réductibles par S.

Si on peut décider Q6, on peut décider si $\phi(F)$ est reconnaissable, on peut donc décider si le complémentaire de $\phi(F)$ dans T_Σ, qui n'est autre que IRR(S), est reconnaissable par S.❑

Il n'est pas surprenant que la plupart des résultats généraux sur la conservation de la reconnaissabilité par des systèmes de réécriture soient indécidables. Mais, si on sait qu'un système de réécriture préserve la reconnaissabilité, on peut en déduire des résultats de décidabilité (par exemple les problèmes d'accessibilité). Nous allons voir une autre illustration de cette remarque dans la section 3.3.

Nous terminons cette section en mentionnant des travaux voisins de cette étude.

Kucherov ([KUCH90]) étudie les systèmes de réécriture dont l'ensemble des termes clos irréductibles est reconnaissable et montre qu'on peut trouver un système de réécriture équivalent linéaire (équivalent au sens du calcul des formes normales).

M.Dauchet et S.Tison ([DATI90b]) étudient les homomorphismes quasi-alphabétiques (les variables sont à la profondeur 1) conservant la reconnaissabilité avec des résultats de réduction de la non-linéarité. Les automates définis par B.Bogaert et S.Tison ([BOGA90]) devraient permettre de décider si un morphisme quasi-alphabétique préserve la reconnaissabilité, avec pour application, pour certaines classes de systèmes de réécriture, un résultat de décision pour la reconnaissabilité de l'ensemble des termes clos irréductibles.

3.3 Fragment de théorie décidable

3.3.1 Définition des formules

Soit Σ un alphabet gradué et X un ensemble de variables.

Les termes sont les constantes (termes de T_Σ) et les termes de $T_\Sigma(X)$.

Les formules atomiques sont de la forme $t \equiv t'$ avec t et t' termes.

Les connecteurs sont les connecteurs usuels $\vee$, $\wedge$, $\neg$.

Les quantificateurs sont les quantificateurs usuels $\forall$ et $\exists$.

Nous considérons le *sous ensemble FLin des formules closes* de cette théorie, défini de la façon suivante:

Une formule de FLin a pour forme prénexe $(Q_1\ x_1)...(Q_N\ x_N)\ F(x_1,...,x_N)$ avec $Q_i=\forall$ ou $Q_i=\exists$ qui satisfait les propriétés (i), (ii) et (iii) suivantes:

(i) Pour tout terme t (non clos) apparaissant dans F, soit toutes les variables occurant dans t sont quantifiées universellement (t est dit universel), soit elles sont quantifiées de façon existentielle (t est dit existentiel).

(ii) Pour toute formule atomique $t\equiv t'$ apparaissant dans F, tous les quantificateurs relatifs aux variables occurant dans t précèdent les quantificateurs relatifs aux variables occurant dans t' ou l'inverse.

(iii) F est linéaire c'est-à-dire que F ne contient pas deux fois la même variable, c'est-à-dire encore que tous les termes sont linéaires, une même variable ne peut apparaître de part et d'autre du prédicat $\equiv$ et ne peut apparaître dans deux formules atomiques distinctes.

Exemple:

Soit $\Sigma=\{0,s,+,*\}$ avec +, * d'arité 2, s d'arité 1, 0 d'arité 0.

$\forall x\ \exists y\ +(x,s(s(0))) \equiv +(y,s(0))$ appartient à FLin.

$\forall x\ \exists y\ \exists z\ *(x,x) \equiv +(*(y,y),*(z,z))$ n'appartient pas à FLin.

Notons que cette définition est légèrement différente de celle utilisée par R.V.Book dans [BOOK83]. En effet, l'auteur partitionne l'ensemble X en deux-sous ensembles X_E et X_U de variables existentielles et universelles, mais dans ce cas, on est obligé de se restreindre à des formules ne faisant pas intervenir la négation.

3.3.2 Interprétation et validité des formules

Etant donné un ensemble d'équations E, nous allons donner une interprétation des formules définies précédemment.

Chaque variable x prend ses valeurs dans son domaine D(x) qui est une forêt reconnaissable.

Le symbole ≡ est interprété comme $\overset{*}{\leftrightarrow}_E$ ou $=_E$.

Les connecteurs sont interprétés de façon usuelle.

Selon cette interprétation, une formule de FLin est vraie ou fausse si pour des substitutions closes associées aux variables dans leur domaine, la propriété est vraie ou fausse pour les termes de T_Σ ainsi obtenus pour $=_E$.

3.3.3 Décidabilité des formules linéaires

Théorème 1: *Soit (P): Pour toute forêt reconnaissable F, l'ensemble S!(F) des formes normales des termes de F pour S est reconnaissable.*
Soit E un ensemble d'équations, si il existe un système de réécriture S tel que:

(*) $\overset{*}{\leftrightarrow}_S = \overset{*}{\leftrightarrow}_E$

(**) S est confluent et noethérien

(***) S satisfait (P)

Alors il existe un algorithme qui, ayant pour donnée une formule de FLin arrête et répond si cette formule est vraie ou fausse pour l'interprétation définie précédemment.

Preuve: Soit $t=t(x_1,\ldots,x_p)$ un terme linéaire de $T_\Sigma(X)$. En effet, les termes apparaissant dans une formule de FLin sont linéaires.

On définit D(t) par $D(t)=\{t(t_1,\ldots,t_p) / t_1 \in D(x_1),\ldots,t_p \in D(x_p)\}$, c'est-à-dire D(t) est l'ensemble des termes obtenus en substituant à chaque variable un terme clos de son domaine. Si t est un terme clos, on pose $D(t)=\{t\}$.

S!(D(t)) désigne, avec nos notations, l'ensemble des formes normales pour S des termes de D(t).

Pour tout terme t, D(t) est une forêt reconnaissable (t est linéaire et clôture par substitution) et donc S!(D(t)) est reconnaissable (hypothèse (***)).

Considérons une formule de FLin. Par les transformations usuelles, on peut faire passer le connecteur $\neg$ le plus à l'intérieur des formules, c'est-à-dire devant les formules atomiques. De plus, comme F est linéaire on peut distribuer, en respectant l'ordre,les quantificateurs $\forall$ et $\exists$ de telle façon que la formule soit équivalente à une conjonction et disjonction de formules $(Q_1\ x_1)\ldots(Q_n\ x_n)\ f$ ou $(Q_1\ x_1)\ldots(Q_n\ x_n)\ \neg f$ vérifiant les hypothèses (i) et (ii) avec f atomique linéaire.

Nous allons considérer les différents cas possibles pour une formule de la forme $(Q_1\ x_1)\ldots(Q_n\ x_n)\ f$ vérifiant les hypothèses (i) et (ii) avec f atomique linéaire:

(1): $t \equiv t'$ où t et t' sont des termes clos.

(2): $(\exists\ y_1)\ldots(\exists y_p)\ t(y_1,\ldots,y_p) \equiv t'$, où t est existentiel et t' est clos.

(3): $(\forall\ x_1)\ldots(\forall x_n)\ t(x_1,\ldots,x_n) \equiv t'$, où t est universel et t' est clos.

(4): $(\exists y_1)\ldots(\exists y_p)\ (\exists y'_1)\ldots(\exists y'_{p'})\ t(y_1,\ldots,y_p) \equiv t'(y'_1,\ldots,y'_{p'})$, t et t' existentiels

(5): $(\forall\ x_1)\ldots(\forall x_n)\ (\forall x'_1)\ldots(\forall x'_{n'})\ t(x_1,\ldots,x_n) \equiv t'(x'_1,\ldots,\forall x'_{n'})$, t et t' universels.

(6): $(\forall\ x_1)\ldots(\forall x_n)\ (\exists y_1)\ldots(\exists y_p)\ t(x_1,\ldots,x_n) \equiv t'(y_1,\ldots,y_p)$, où t est universel et t' existentiel.

(7): $(\exists y_1)\ldots(\exists y_p)\ (\forall\ x_1)\ldots(\forall x_n)\ t(x_1,\ldots,x_n) \equiv t'(y_1,\ldots,y_p)$, où t est universel et t' existentiel.

Cas (1): Il correspond au problème du mot décidable grâce à l'existence de S vérifiant les propriétés (*) et (**).

Cas (2) et (3): Ils peuvent être considérés comme cas particuliers des cas (4) et (5)

Cas (4): la formule est vraie pour l'interprétation considérée si et seulement si il existe un terme de D(t) (c'est-à-dire une substitution close pour t dans son domaine) et un terme de D(t') équivalents pour $\overset{*}{\leftrightarrow}_E$, donc pour $\overset{*}{\leftrightarrow}_S$ (hypothèse (*)) et qui ont donc même forme normale pour S (hypothèse(**)). La formule est donc vraie pour l'interprétation considérée si et seulement si $S!(D(t)) \cap S!(D(t')) \neq \emptyset$. Or la classe des forêts reconnaissables est close par intersection, on peut décider si une forêt reconnaissable est vide.

Cas(5): la formule est vraie pour l'interprétation considérée si et seulement si tout terme de D(t) est équivalent pour $\overset{*}{\leftrightarrow}_E$ à tout terme de D(t') donc, d'après les hypothèses (*) et (**) si et seulement si S!(D(t)) et S!(D(t')) sont égaux et réduits à un singleton. Ces deux ensembles sont reconnaissables, on peut décider de l'égalité de deux forêts reconnaissables, on peut décider si une forêt reconnaissable est réduite à un singleton.

Cas(6): la formule est vraie pour l'interprétation considérée si et seulement si S!(D(t)) est inclus dans S!(D(t')) d'où le résultat, l'inclusion d'une forêt reconnaissable dans une autre étant décidable.

cas(7): la formule est vraie pour l'interprétation considérée si et seulement si S!(D(t)) est inclus dans S!(D(t')) et S!(D(t)) est réduit à un singleton d'où le résultat.

On peut déduire de cette étude le même résultat pour une formule de la forme $Q_1 x_1,\ldots,Q_n x_n, \neg f$ vérifiant les hypothèses (i) et (ii) avec f atomique linéaire en remarquant qu'elles sont équivalentes à des négations de formules étudiées précédemment.❑

Notons que dans les hypothèses de ce théorème 1, la propriété (***), c'est-à-dire la propriété (P) est vérifiée par tout système de réécriture S satisfaisant les propriétés (iv) et (v) suivantes:

(iv) IRR(S) est reconnaissable

(v) S préserve la reconnaissabilité, c'est-à-dire, pour toute forêt reconnaissable F, l'ensemble S(F) est une forêt reconnaissable.

Il suffit, en effet, de remarquer que $S!(F)=S(F)\cap IRR(S)$ et d'utiliser la clôture de REC par intersection. Rappelons également que la propriété (iv) est vérifiée par tout système de réécriture linéaire à gauche (proposition de la section 3.1) et que nous verrons dans les sections 4 et 5 des exemples de systèmes de réécriture qui préservent la reconnaissabilité (propriété (v)), citons les systèmes de réécriture clos, monadiques linéaires à droite et semi-monadiques linéaires à droite.

Notons également que les hypothèses (**) et (***) sont, en général, indécidables pour un système de réécriture mais que, si on les suppose vérifiées, on obtient un résultat de décidabilité.

Théorème 2: *Soit A un ensemble d'équations, si il existe un ensemble d'équations E et un système de réécriture R tels que*

(i) $\overset{*}{\leftrightarrow}_A = \overset{*}{\leftrightarrow}_{R\cup E} = \overset{*}{\rightarrow}_{R/E} \cup \overset{*}{\leftrightarrow}_E$.

(ii) R noetherien et Church-Rosser modulo E.

(iii) Pour toute forêt reconnaissable F, R/E!(F) ensemble des formes normales des termes de F pour R/E est reconnaissable.

(iv) Les classes de congruence modulo E sont reconnaissables.

Alors il existe un algorithme qui, ayant pour donnée une formule de FLin arrête et répond si cette formule est vraie ou fausse pour l'interprétation définie précédemment pour $=_A$.

Preuve: La preuve se calque sur celle du théorème précédent en utilisant R/E en lieu et place de R. Les classes d'équivalence pour E sont reconnaissables (propriété (iv)), ce qui implique la décision de l'égalité entre R/E!(t) avec la classe de congruence d'un terme modulo E.□

Remarque: Cette preuve peut être étendue aux algèbres avec sortes ordonnées. Il suffit d'introduire dans la définition de notre logique les formules atomiques de la forme $t \in s$ pour exprimer la bonne formation des termes. En effet, il suffit alors de considérer, dans la preuve, l'intersection de D(t) avec la forêt reconnaissable des termes bien formés de sorte s (ou son complément pour $t \notin s$) en lieu et place de D(t).

3.3.4 Cas des formules non linéaires

Nous allons montrer que les résultats de décidabilité de la section 3.3.3 ne sont valables que sous l'hypothèse (forte) linéaire pour les formules.

Théorème 3: *Soit (P): Pour toute forêt reconnaissable F, l'ensemble S!(F) des formes normales des termes de F pour S est reconnaissable.*
Soit E un ensemble d'équations, pour lequel il existe un système de réécriture S tel que:

$(*)\ \overset{*}{\leftrightarrow}_S = \overset{*}{\leftrightarrow}_E$

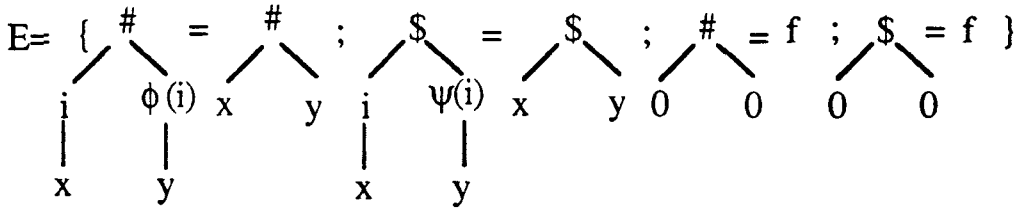
(**) S est confluent et noethérien

(***) S satisfait (P)

Le fragment existentiel de théorie des algèbres de termes clos modulo la congruence engendrée par un ensemble E d'équations satisfaisant les conditions précédentes est indécidable.

Preuve: Soit I={1,...,n} et Σ un alphabet que nous considérons comme des alphabets unaires. Soit $\Delta = I \cup \Sigma \cup \{0,f,\#,\$\}$ où 0 et f sont des lettres d'arité 0 et #,$ d'arité 2 (0,f,#,$ n'appartiennent ni à I, ni à Σ). Soit PCP le problème de Post PCP=(Σ,n,ϕ,ψ). Soit S le système de réécriture sur T_Δ obtenu en orientant les règles de l'ensemble d'équations de la figure 3.3.4 de gauche à droite.

Figure 3.3.4



L'ensemble E satisfait les conditions du théorème 3.
En effet, $\overset{*}{\leftrightarrow}_S = \overset{*}{\leftrightarrow}_E$ donc (*) est vérifiée;
S est noethérien, S est linéaire gauche et sans paire critique donc S est confluent donc (**) est satisfaite;
S est linéaire gauche donc IRR(S) est une forêt reconnaissable (Proposition de la section 3.1) d'où (iv) et S préserve la reconnaissabilité (S est un système de réécriture monadique, voir section 4.6) d'où (v), et donc (***) est également vérifiée.

Soit la formule $\exists x \exists y \bigvee_{i \in I} (\#(i(x),y) \equiv \#(i(x),y))$.

Fait: Cette formule est vraie pour l'interprétation considérée de ≡ si et seulement si le problème de Post PCP a une solution.

Preuvefait: Cette formule est vraie pour l'interprétation considérée si il existe i de I, t et t' de T_Δ tels que $\#(i(t),t') =_E \$(i(t),t')$ et donc tels que ces deux termes aient même forme normale pour S. Si t n'appartient pas à $I^*(0)$ ou si t' n'appartient pas à $\Sigma^*(0)$, on ne pourra pas effacer la racine des termes #(i(t),t') et $(i(t),t'), et donc, ces termes ne pourront pas avoir même forme normale pour S. Donc, cette formule est vraie pour l'interprétation considérée si il existe un mot m de I^+ et un mot u de Σ^* tel que $\#(m(0),u(0)) =_E \$(m(0),u(0))$ et donc tels que #(m(0),u(0)) et $(m(0),u(0)) ont même forme normale pour S. Ceci ne peut être vérifié que si l'on peut appliquer les règles 3 et 4 du système de réécriture S. On en déduit donc que $\#(m(0),u(0)) \overset{*}{\rightarrow}_S \#(0,0)$ et que $\$(m(0),u(0)) \overset{*}{\rightarrow}_S \$(0,0)$ par les règles 1 et 2 et donc que $\phi(m)=u$ et $\psi(m)=u$ c'est à dire encore que PCP a une solution. La réciproque est évidente.❑ *Finpreuvefait.*

Remarque 1: On obtient un résultat identique en utilisant l'interprétation de la section 3.3.3 en associant aux variables x et y des domaines reconnaissables et en considérant la formule:

$\exists x, \exists y, \#(x,y) \equiv \$(x,y)$ avec $D(x)=T_{I\cup\{0\}}-\{0\}$ et $D(y)=T_{\Sigma\cup\{0\}}$.

Remarque 2: On obtient également le même résultat avec le système de réécriture S défini comme précédemment, § une nouvelle lettre d'arité 2, $\Gamma=\Delta\cup\{\S\}$. S est un système de réécriture sur T_Γ qui vérifie les hypothèses du théorème 3. On considère alors la formule:

$\exists x, \exists y, \S(\#(x,y),\$(x,y)) \equiv \S(f,f)$ avec $D(x)=T_{I\cup\{0\}}-\{0\}$ et $D(y)=T_{\Sigma\cup\{0\}}$.

❑ *Finpreuvethéorème3.*

On déduit de ce résultat l'indécidabilité de la E-unification sortée pour une théorie E telle qu'il existe un système de réécriture confluent ,noethérien, adéquat pour E dont les ensembles de formes normales sont reconnaissables. Ce résultat est à mettre en correspondance avec le résultat de A.Bockmayr ([BOCK87]).

4 ETUDE DE CLASSES DE SYSTEMES DE REECRITURE

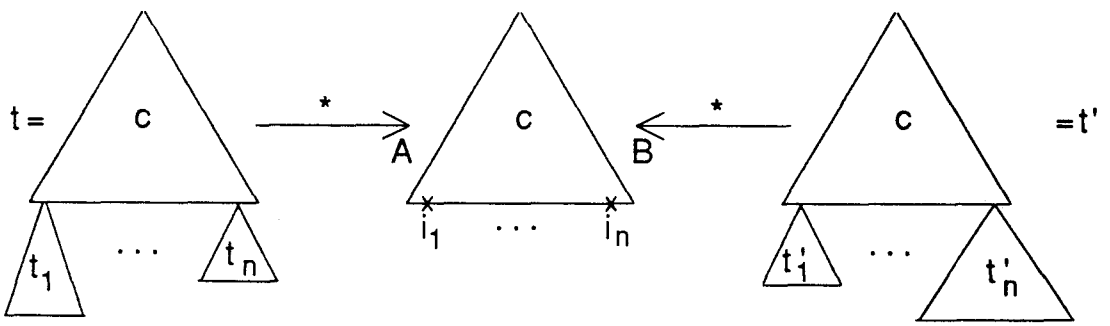
Nous étudions, dans cette partie différents problèmes de décision (accessibilité, terminaison, confluence) pour certaines classes de systèmes de réécriture. Les résultats obtenus sont regroupés dans la section 4.7.

4.1 Systèmes de réécriture clos

4.1.1 Transducteurs d'arbres clos

Définition: Un *transducteur d'arbres clos* (gtt pour ground tree transducer) est un couple V=(A,B) d'automates ascendants d'arbres. Soit $A=(\Sigma,Q_A,Q_{Af},R_A)$ et $B=(\Sigma,Q_B,Q_{Bf},R_B)$, la transformation associée à V est l'ensemble $r(V)=\{(t,t') / t\in T_\Sigma, t'\in T_\Sigma, \exists s\in T_{\Sigma\cup(Q_A\cap Q_B)}$ tel que $t \xrightarrow{*}_A s \xleftarrow{*}_B t'\}$ c'est à dire encore:

r(V) est l'ensemble des couples (t,t') de termes de T_Σ tels que



avec c, contexte commun, appartient à $T_\Sigma(X_n)$ et, pour tout j, t_j, t'_j appartiennent à T_Σ, ij appartient à $Q_A\cap Q_B$.

On peut remarquer que dans cette définition les ensembles Q_{Af} et Q_{Bf} ne jouent aucun rôle et donc on pourra les choisir égaux à l'ensemble vide. Par contre on peut noter l'importance des états de $Q_A\cap Q_B$, ces états sont appelés états interfaces.

Exemple: Un exemple de gtt associé à un système de réécriture clos est donné dans la figure 4.1.

La classe des transformations associées aux gtt est close par inverse, composition, itération. Pour l'union, on a la propriété suivante: soit $V_1=(A_1,B_1)$ et $V_2=(A_2,B_2)$ deux gtt tels que les ensembles d'états utilisés pour V_1 et V_2 soient disjoints et V le gtt obtenu en prenant l'union des états et l'union des règles, on a $r(V)^*=(r(V_1)\cup r(V_2))^*$. Les preuves de ces résultats peuvent être trouvées dans [DATI85].

4.1.2 Compilation d'un système de réécriture clos

***Proposition 1**: Pour tout système de réécriture clos R, il existe un transducteur clos d'arbres V tel que* $r(V) = \overset{*}{\rightarrow}_R$.

Preuve:

Nous allons donc ici prouver que étant donné un système de réécriture clos R, on peut construire un transducteur clos V=(A,B) tel que, quels que soient les termes t et t', $t \overset{*}{\rightarrow}_R t'$ si et seulement si il existe un terme s de $T_{\Sigma\cup(QA\cap QB)}$ tel que $t \overset{*}{\rightarrow}_A s \overset{*}{\leftarrow}_B t'$.

Pour cela, on construit d'abord un gtt U=(G,D) tel que $\rightarrow_R=r(U)$ et donc tel que $\overset{*}{\rightarrow}_R=(r(U))^*$.

Ensuite on construit V tel que $r(V)=(r(U))^*=\overset{*}{\rightarrow}_R$, le principe de cette construction est de créer des ε-transitions qui vont permettre de supprimer les phases de génération-effacement de sous-termes identiques qui se produisent lorsque t se réécrit en t' par R.

Le plan de la preuve est le suivant: dans une première partie, nous donnons la construction de V et prouvons l'arrêt de cette construction; dans une seconde partie, nous prouvons trois lemmes puis l'égalité entre r(V) et $\overset{*}{\rightarrow}_R$ par double inclusion. La construction de V est illustrée par deux exemples dans la figure 4.1.

Preuve de la proposition 1.

Partie 1: Construction de V

Soit $R=\{l_j \rightarrow r_j \;/\; l_j, r_j \in T_\Sigma,\ j\in J,\ J \text{ fini}\}$ un système de réécriture clos.

Pour chaque règle $l_j \rightarrow r_j$ de R, on construit un gtt $U_j=(G_j,D_j)$ avec $G_j=(\Sigma,Q_j,\varnothing,R_j)$ et $D_j=(\Sigma,Q'_j,\varnothing,R'_j)$ automates ascendants déterministes tels que $Q_j \cap Q'_j=\{i_j\}$, $l_j \overset{*}{\rightarrow}_{G_j} i_j$ et $r_j \overset{*}{\rightarrow}_{D_j} i_j$ .On impose, en outre, que les ensembles d'états utilisés pour deux règles distinctes soient disjoints.

On considère alors le gtt $U=(G,D)$ défini par $G=(\Sigma,Q_G,\varnothing,R_G)$, $D=(\Sigma,Q_D,\varnothing,R_D)$ avec $Q_G=\bigcup_{j\in J} Q_j$, $Q_D=\bigcup_{j\in J} Q'_j$, $R_G=\bigcup_{j\in J} R_j$ et $R_D=\bigcup_{j\in J} R'_j$.

Exemple: voir figure 4.1.

On peut maintenant construire le gtt $V=(A,B)$ avec $A=(\Sigma,Q_A,\varnothing,R_A)$ et $B=(\Sigma,Q_B,\varnothing,R_B)$ grâce à l'algorithme suivant:

<u>début</u>

%1ère étape: création de l'ensemble E des ε-transitions%

$E=\varnothing$

<u>tant que</u> on peut trouver des états $q,q',q_1,\ldots,q_n,q'_1,\ldots,q'_n$ tels que $f(q'_1,\ldots,q'_n)\rightarrow_D q'$, $f(q_1,\ldots,q_n)\rightarrow_G q$, $q'_1 \overset{*}{\rightarrow}_E q_1,\ldots, q'_n \overset{*}{\rightarrow}_E q_n$

<u>faire</u> ajouter la règle $q' \rightarrow q$ dans E <u>fait</u>

<u>fin tant que</u>

%2ème étape: construction de V%

$Q_A=Q_G$; $Q_B=Q_D$; $R_A=R_G$; $R_B=R_D$;

<u>pour</u> toute règle $q' \rightarrow q$ dans E

<u>si</u> $q' \in Q_A \cap Q_B$ et $q \in Q_A \cap Q_B$ <u>alors</u> ajouter $q' \rightarrow q$ à R_A et $q \rightarrow q'$ à R_B

<u>finsi</u>

<u>si</u> $q' \in Q_A \cap Q_B$, $q \in Q_A$ et $q \notin Q_B$ <u>alors</u> ajouter $q' \rightarrow q$ à R_A <u>finsi</u>

<u>si</u> $q' \in Q_B$, $q' \notin Q_A$ et $q \in Q_A \cap Q_B$ <u>alors</u> ajouter $q \rightarrow q'$ à R_B <u>finsi</u>

<u>fin pour</u>

<u>fin</u>

Exemple: voir figure 4.1.

Cet algorithme termine car les ensembles R_G, R_D et $Q_D \times Q_G$ sont des ensembles finis et que l'ensemble E est inclus dans $Q_D \times Q_G$. L'algorithme a été décomposé en deux étapes pour une plus grande lisibilité de la preuve.

Partie 2: Justification de l'égalité $\stackrel{*}{\rightarrow}_R = r(V)$

Lemme 1: $\stackrel{*}{\rightarrow}_R = (r(U))^* = (\bigcup_{j \in J} r(U_j))^*$

preuve lemme 1: Ce résultat est une conséquence de la propriété des transformations associées aux gtt sur l'union, des définitions de U_j et U. La preuve se fait sans difficulté par doubles inclusions en utilisant les propriétés des clôtures réflexive et transitive.❑*Fin preuve lemme 1.*

Lemme 2: Si il existe t de T_Σ tel que $t \stackrel{*}{\rightarrow}_G q$ et $t \stackrel{*}{\rightarrow}_D q'$ alors $q' \rightarrow q$ appartient à E.

preuve lemme 2: La preuve se fait par induction sur la structure des termes
Si $t=a \in \Sigma_0$(lettres de Σ d'arité 0) et si $a \stackrel{*}{\rightarrow}_G q$ et $a \stackrel{*}{\rightarrow}_D q'$(hypothèse) alors $a \rightarrow_G q$ et $a \rightarrow_D q'$ car G et D sont sans ε-transitions, de plus $a \stackrel{*}{\rightarrow}_E a$ donc $q' \rightarrow q$ appartient à E.
Si $t=f(t_1,\ldots,t_n)$ et $f(t_1,\ldots,t_n) \stackrel{*}{\rightarrow}_G q$ et $f(t_1,\ldots,t_n) \stackrel{*}{\rightarrow}_D q'$ (hypothèse) alors $f(t_1,\ldots,t_n) \stackrel{*}{\rightarrow}_G f(q_1,\ldots,q_n) \rightarrow_G q$ et $f(t_1,\ldots,t_n) \stackrel{*}{\rightarrow}_D f(q'_1,\ldots,q'_n) \rightarrow_D q'$.
De plus, pour tout i, $t_i \stackrel{*}{\rightarrow}_G q_i$ et $t_i \stackrel{*}{\rightarrow}_D q'_i$ donc par hypothèse d'induction $q'_i \rightarrow q_i$ appartient à E.
donc $f(q_1,\ldots,q_n) \rightarrow_G q$,$f(q'_1,\ldots,q'_n) \rightarrow_D q'$ et pour tout i, $q'_i \stackrel{*}{\rightarrow}_E q_i$ et donc d'après les règles de construction de E, la règle $q' \rightarrow q$ appartient à E. ❑*Fin preuve lemme 2.*

Lemme 3: Pour toute règle $q' \rightarrow q$ de E, on a $q' \ (_D\!\stackrel{*}{\leftarrow} \stackrel{*}{\rightarrow}_G)^* \ q$

Preuve lemme 3: Preuve par récurrence sur le nombre de règles dans E
Si $E=\varnothing$, la première règle créée dans E l'est nécessairement par l'existence d'un terme a de Σ_0 tel que $a \rightarrow_G q$ et $a \rightarrow_D q'$ et donc $q' \ _D\!\leftarrow a \rightarrow_G q$ et donc $q' \ (_D\!\stackrel{*}{\leftarrow} \stackrel{*}{\rightarrow}_G)^* \ q$.

Si la propriété est vraie pour les n règles de E, montrons que la propriété est encore vraie pour la $n+1^{ème}$. Cette règle est créée par $f(q_1,\ldots,q_n) \rightarrow_G q$, $f(q'_1,\ldots,q'_n) \rightarrow_D q'$ et pour tout i, $q'_i \stackrel{*}{\rightarrow}_E q_i$ donc en utilisant l'hypothèse de récurrence et la propriété des clôtures réflexives et transitives, pour tout i, $q'_i \ (_D\!\stackrel{*}{\leftarrow} \stackrel{*}{\rightarrow}_G)^* \ q_i$ de plus $q' \ _D\!\leftarrow f(q'_1,\ldots,q'_n)$, $f(q_1,\ldots,q_n) \rightarrow_G q$, et donc $q' \ (_D\!\stackrel{*}{\leftarrow} \stackrel{*}{\rightarrow}_G)^* \ q$. ❑*Fin preuve lemme 3.*

Preuve de l'inclusion $\stackrel{*}{\to}_R \subset r(V)$:

nous noterons $\stackrel{n}{\longrightarrow}_R$ la réécriture en n pas par R.
$r(V) \supset \stackrel{0}{\longrightarrow}_R$ par définition de r(V)
$r(V) \supset \stackrel{1}{\longrightarrow}_R = r(U)$ par définition de U et construction de V
Supposons que $\stackrel{n}{\longrightarrow}_R \subset r(V)$, soit t et t' tels que $t \stackrel{n\pm1}{\longrightarrow}_R t'$, soit u de T_Σ tel que $t \stackrel{n}{\longrightarrow}_R u \to_R t'$, par hypothèse de récurrence, il existe un contexte c tel que
$t=c(t_1,...,t_n) \stackrel{*}{\to}_A s=c(i_1,...,i_n)\ {}_B\!\stackrel{*}{\leftarrow} u=c(u_1,...,u_n)=u'(l) \to_R t'=u'(r)$ avec pour tout j, i_j état de $Q_A \cap Q_B$ et $l \to r$ la règle utilisée pour la réécriture de u en t'. Plusieurs cas sont à étudier:

Cas 1: Si l n'a aucun des u_j comme sous-terme et si l n'est sous-terme d'aucun des u_j, le résultat est immédiat car $\to_R \subset r(V)$, c'est-à-dire on peut déterminer un contexte c' tel que
$t=c'(t_1,...,t_n,t_{n+1}) \stackrel{*}{\to}_A s'=c'(i_1,...,i_n,i_{n+1})\ {}_B\!\stackrel{*}{\leftarrow} t'=c'(t'_1,...,t'_{n+1})$.

Cas 2: l possède l'un des u_j comme sous-terme (la preuve s'étend facilement au cas ou l possède plusieurs u_j comme sous-termes)
$t=c(t_1,..,t_j,..,t_n) \stackrel{*}{\to}_A s=c(i_1,..,i_j,..,i_n)\ {}_B\!\stackrel{*}{\leftarrow} u=c(u_1,..,u_j,..,u_n)=c'(u_1,..,l,..,u_n) \to_R c'(u_1,..,r,..,u_n)=t'$ avec $l=l'(u_j)$. La règle $l \to r$ est une règle de R donc par construction de V, il existe un état interface i tel que $l \stackrel{*}{\to}_A i\ {}_B\!\stackrel{*}{\leftarrow} r$, de plus $l=l'(u_j)$ donc $u_j \stackrel{*}{\to}_A q$. Or nous avons aussi $u_j \stackrel{*}{\to}_B i_j$ (en utilisant la réécriture de t en u) donc en utilisant le lemme 2, il existe une règle $i_j \to q$ dans E et donc dans R_A. On peut en déduire que $l'(i_j) \stackrel{*}{\to}_A i$ et
$t=c(t_1,..,t_j,..,t_n) \stackrel{*}{\to}_A s=c(i_1,..,i_j,..,i_n) \stackrel{*}{\to}_A c'(i_1,..,i,..,i_n)\ {}_B\!\stackrel{*}{\leftarrow} c'(u_1,..,r,..,u_n)=t'$ donc $(t,t') \in r(V)$.

Cas 3: l est sous terme de l'un des u_j, la preuve se fait de façon identique en utilisant une règle de la forme $i \to q'$ dans R_B.
Ce qui prouve donc par récurrence que $\stackrel{*}{\to}_R \subset r(V)$.

❑ *Fin preuve $1^{ère}$ inclusion.*

Preuve de l'inclusion $r(V) \subset \overset{*}{\rightarrow}_R$:

Nous savons que $\overset{*}{\rightarrow}_R = (r(U))^*$ (lemme 1), nous allons donc démontrer que $r(V) \subset (r(U))^*$, c'est à dire démontrer que si il existe s de $T_{\Sigma\cup(Q_A\cap Q_B)}$ tel que $t \overset{*}{\rightarrow}_A s \overset{*}{\leftarrow}_B t'$ alors $t\ (\rightarrow_G D \overset{*}{\leftarrow})^*\ t'$.

Le problème se pose pour les règles de R_A qui ne sont pas dans R_G et pour les règles de R_B qui ne sont pas dans R_D, c'est-à-dire pour les ε-transitions obtenues dans E.

La preuve se fait par récurrence sur le nombre d'utilisations des ε-transitions et utilise le lemme 3, en effet pour une règle de la forme $i \rightarrow q$ dans R_A, on a $i\ (_D\overset{*}{\leftarrow} \overset{*}{\rightarrow}_G)^*\ q$ et pour une règle de la forme $i \rightarrow q'$ dans R_B on a $q'\ (_D\overset{*}{\leftarrow} \overset{*}{\rightarrow}_G)^*\ i$, soit encore, $i\ (_G\overset{*}{\leftarrow} \overset{*}{\rightarrow}_D)^*\ q'$.

❑ *Fin preuve de la 2ème inclusion.*

❑ *Fin preuve proposition 1.*

Figure 4.1

Exemple 1: Soit Σ={a,b,c,f,g} avec a,b,c,f,g d'arités respectives 0,0,0,1,1
Soit le système de réécriture clos
R={1: a $\rightarrow$ f(b); 2: b$\rightarrow$ g(c); 3: f(g(c)) $\rightarrow$ g(a) }.

G_1: a $\rightarrow$ i_1 $\qquad$ D_1: b $\rightarrow$ q'$_1$, f(q'$_1$)$\rightarrow$ i_1
G_2: b $\rightarrow$ i_2 $\qquad$ D_2: c $\rightarrow$ q'$_2$, g(q'$_2$)$\rightarrow i_2$
G_3: c $\rightarrow q_1$, g(q_1)$\rightarrow$ q_2 , f(q_2)$\rightarrow$ i_3 $\qquad$ D_3: a $\rightarrow$ q'$_3$, g(q'$_3$)$\rightarrow i_3$
On a ainsi construit U=(G,D) avec $R_G=R_{G1}\cup R_{G2}\cup R_{G3}$ et $R_D=R_{D1}\cup R_{D2}\cup R_{D3}$.

On obtient E={ q'$_3\rightarrow$ i_1 (a$\rightarrow_G$ i_1 , a$\rightarrow_D$ q'$_3$)
q'$_1\rightarrow$ i_2 (b$\rightarrow_G$ i_2 , b$\rightarrow_D$ q'$_1$)
q'$_2\rightarrow$ q_1 (c$\rightarrow_G$ q_1 , c$\rightarrow_D$ q'$_2$)
$i_2\rightarrow$ q_2 (g(q_1)$\rightarrow_G$ q_2 , g(q'$_2$)$\rightarrow_D$ i_2 , q'$_2\rightarrow_E$ q_1)
$i_1\rightarrow$ i_3 (f(q_2)$\rightarrow_G$ i_3 , f(q'$_1$)$\rightarrow_D$ i_1 , q'$_1\rightarrow_E$ $i_2\rightarrow_E$ q_2) }.

et on définit V=(A,B) en posant:
$R_A=R_G\cup$ { $i_2\rightarrow$ q_2 , $i_1\rightarrow$ i_3} et $R_B=R_D\cup$ { $i_1\rightarrow$ q'$_3$, $i_2\rightarrow$ q'$_1$, $i_3\rightarrow$ i_1}.
Avec R , a $\stackrel{*}{\rightarrow}_R$ g(a).
Avec U, a$\stackrel{*}{\rightarrow}_G$ i_1 $_D\!\stackrel{*}{\leftarrow}$ f(b) $\stackrel{*}{\rightarrow}_G$ f(i_2) $_D\!\stackrel{*}{\leftarrow}$ f(g(c)) $\stackrel{*}{\rightarrow}_G$ i_3 $_D\!\stackrel{*}{\leftarrow}$ g(a).
Avec V , a $\stackrel{*}{\rightarrow}_A$ i_3 $_B\!\stackrel{*}{\leftarrow}$ g(a). Les nouvelles règles créées permettent de supprimer les phases de génération-effacement.

Exemple 2: Soit Σ={a,b,f,g} avec a,b,f,g d'arités respectives 0,0,1,2.
Soit le système de réécriture clos
R={1: g(a,b)$\rightarrow$ g(b,a); 2: a$\rightarrow$ f(a); 3: f(a)$\rightarrow$ a}.

G_1: a$\rightarrow$ q_1 , b$\rightarrow$ q_2 , g(q_1,q_2)$\rightarrow$ i_1 $\qquad$ D_1: b$\rightarrow$ q'$_1$, a$\rightarrow$q'$_2$, g(q'$_1$,q'$_2$)$\rightarrow i_1$
G_2: a$\rightarrow$ i_2 $\qquad$ D_2: a$\rightarrow$ q'$_3$, f(q'$_3$)$\rightarrow$ i_2
G_3: a$\rightarrow$ q_3 , f(q_3)$\rightarrow$ i_3 $\qquad$ D_3: a$\rightarrow$ i_3

On obtient alors $R_A=R_G\cup$ { $i_3\rightarrow$ q_1, $i_3\rightarrow$ q_3, $i_3\rightarrow$ i_2, $i_2\rightarrow$ i_3 } avec $R_G=R_{G1}\cup R_{G2}\cup R_{G3}$ et $R_B=R_D\cup$ { $i_2\rightarrow$ q'$_2$, $i_2\rightarrow$ q'$_3$, $i_3\rightarrow$ i_2 , $i_2\rightarrow$ i_3 } avec $R_D=R_{D1}\cup R_{D2}\cup R_{D3}$.

Avec R , g(f^3(a),b) $\stackrel{*}{\rightarrow}_R$ g(b,f^2(a))
Avec V, g(f^3(a),b)$\stackrel{*}{\rightarrow}_A$ i_1 $_B\!\stackrel{*}{\leftarrow}$ g(b,f^2(a)).

4.1.3 Les systèmes de réécriture clos préservent la reconnaissabilité

Proposition 2: *Quel que soit le système de réécriture clos R et la forêt reconnaissable F, l'ensemble R(F) des transformés des termes de F par R est une forêt reconnaissable.*

Preuve proposition 2:

Soit R un système de réécriture clos et soit V le gtt tel que $\stackrel{*}{\rightarrow}_R$ = r(V), soit F une forêt reconnaissable, nous avons R(F)={t'∈ T_Σ / ∃ t∈ F, t$\stackrel{*}{\rightarrow}_R$ t'} soit encore R(F)={t'∈ T_Σ / ∃t∈ F, (t,t')∈ r(V)}. Soit $A=(\Sigma,Q_A,\varnothing,R_A)$, $B=(\Sigma,Q_B,\varnothing,R_B)$ tels que V=(A,B) et $M=(\Sigma,Q_M,Q_{Mf},R_M)$ un automate déterministe reconnaissant F. On suppose que les ensembles Q_B et Q_M sont disjoints.

Soit $C=(\Sigma,Q_C,Q_{Cf},R_C)$ défini par $Q_C=Q_B\cup Q_M$, $Q_{Cf}=Q_{Mf}$, $R_C=R_B\cup R_M\cup R'$ où $R'=\{i\rightarrow q$ / $i\in Q_A\cap Q_B$, $q\in Q_M$, $\exists s\in T_\Sigma$ t.q $s\stackrel{*}{\rightarrow}_A i$ et $s\stackrel{*}{\rightarrow}_M q\}$. On peut construire R_C car les ensembles $\{s/\ s\stackrel{*}{\rightarrow}_A i\}$ et $\{s/\ s\stackrel{*}{\rightarrow}_M q\}$ sont reconnaissables donc on peut décider si leur intersection est non vide.

Fait: *L(C)=R(F).*

Preuve de ⊂*:*

Montrons d'abord que {t'∈ T_Σ / ∃t∈ F, (t,t')∈ r(V)} = R(F) ⊂ L(C).

Soit t' tel que (t,t')∈ r(V) et t∈ F

Soit encore F∋t=$c(t_1,\ldots,t_n)\stackrel{*}{\rightarrow}_A c(i_1,\ldots,i_n)\ {}_B\stackrel{*}{\leftarrow} c(t'_1,\ldots,t'_n)=t'$

On a $t'=c(t'_1,\ldots,t'_n)\stackrel{*}{\rightarrow}_C c(i_1,\ldots,i_n)$ en utilisant les règles de R_B

$c(i_1,\ldots,i_n)\stackrel{*}{\rightarrow}_C c(q_1,\ldots,q_n)$ en utilisant les règles de R'(en effet pour tout j, $t_j\stackrel{*}{\rightarrow}_A i_j$ par hypothèse et $t_j\stackrel{*}{\rightarrow}_M q_j$ car t appartient à F et est donc reconnu par M et donc la règle $i_j\rightarrow q_j$ appartient à R')

$c(q_1,\ldots,q_n)\stackrel{*}{\rightarrow}_C q$ et $q\in Q_{Cf}=Q_{Mf}$ par les règles de R_M (en effet, pour tout j, $t_j\stackrel{*}{\rightarrow}_M q_j$ et $t=c(t_1,\ldots,t_n)$ appartient à F donc est reconnu par M).

t' est reconnu par C, c'est à dire t'∈ L(C).

Preuve de ⊃:

Soit $t' \in L(C)$, soit encore $t' \xrightarrow{*}_C q \in Q_{Cf}$.
Si tous les états sont dans Q_M, alors $t' \in F$ et donc $t' \in R(F)$.
Si il existe des états de Q_B, alors nécessairement, par construction de R_C, il faut utiliser des règles de R' et donc nécessairement la reconnaissance peut se mettre sous la forme
$t'=c(t'_1,...,t'_n) \xrightarrow{*}_C c(i_1,...,i_n)$ en utilisant les règles de R_B,
$c(i_1,...,i_n) \xrightarrow{*}_C c(q_1,...,q_n)$ en utilisant les règles de R',
$c(q_1,...,q_n) \xrightarrow{*}_C q$ et $q \in Q_{Cf}=Q_{Mf}$ par les règles de R_M.
donc $t'=c(t'_1,...,t'_n) \xrightarrow{*}_B c(i_1,...,i_n)$ *(1)*, de plus,
il existe $t_1,...,t_n$ tels que $c(t_1,...,t_n) \in F$ et $c(t_1,...,t_n) \xrightarrow{*}_A c(i_1,...,i_n)$ *(2)*, en effet, pour tout j, on a une règle $i_j \rightarrow q_j$ dans R', donc par définition de R', il existe t_j tel que $t_j \xrightarrow{*}_M q_j$ et $t_j \xrightarrow{*}_A q_j$ donc $c(t_1,...,t_n) \xrightarrow{*}_A c(i_1,...,i_n)$ et $c(t_1,...,t_n) \in F$ car $c(t_1,...,t_n) \xrightarrow{*}_M c(q_1,...,q_n) \xrightarrow{*}_M q$ avec q appartenant à $Q_{Cf}=Q_{Mf}$.
Par (1) et (2), nous déduisons que (t,t')appartient à r(V) avec t appartenant à F, donc t' appartient à R(F).
❑ *Fin preuve proposition 2.*

4.1.4 Applications et conclusion

Une autre preuve de la proposition 1 peut être trouvée dans [DATI85]. L'intérêt de la construction donnée dans le paragraphe 4.1.2 est que *l'algorithme présenté est celui qui est implanté dans VALERIAN* ([DADE89]). La compilation du système de réécriture clos et sa complexité sont étudiées dans [DEGI89].

Le résultat donné dans la proposition 2 peut, lui aussi, être obtenu comme une conséquence des résultats généraux de reconnaissabilité obtenus dans [DATI85]. Nous en donnons ici une preuve directe, en effet, ayant pour donnée le gtt associé à R et un automate reconnaissant F, nous construisons un automate reconnaissant R(F). Cette construction fournit donc un *algorithme pour la résolution des problèmes d'accessibilité du premier et du second ordre pour les systèmes de réécriture clos.* Ces algorithmes sont implantés dans VALERIAN et étudiés dans [DEGI89], [COGI90].

L'étude des systèmes de réécriture clos est close. En effet, après l'obtention de nombreux résultats partiels (confluence [DATI85], [DHTL87], [OYAM87], problème du mot [KOZE77], problème d'accessibilité [DEGI89], [OYAM86]), M.Dauchet et S.Tison ([DATI90]) ont prouvé que la théorie de la réécriture close est décidable. De plus, la terminaison ([HULA78]), la terminaison équitable ([TISO89]) sont décidables.

D'autre part, la complétion d'un système de réécriture clos est étudiée dans [SNYD89]. Dans cet article, E.W.Snyder fournit un algorithme en nlogn que permet de générer un système de réécriture convergent à partir d'un ensemble d'équations closes.

La réécriture close peut avoir des applications intéressantes car on peut parfois ramener des preuves à des équations closes [GRS87].

Il nous reste à poursuivre et parfaire l'implantation de VALERIAN et à étudier les applications éventuelles de ce logiciel en remarquant que du résultat sur la décidabilité de la réécriture close, on peut déduire le même résultat pour la théorie des systèmes de réécriture séparés multisortés. Un système de réécriture est séparé si les ensembles de variables occurant dans les membres gauches et droits d'un même règle sont distincts et sorté si toutes les variables sont sortées.

4.2 Systèmes de réécriture clos modulo la commutativité

Soit R un système de réécriture clos, soit f une lettre d'arité 2 de Σ, soit C={ f(x,y) → f(y,x)} et soit S le système de réécriture R∪C.

Proposition 1:

Il existe un système de réécriture clos R^C tel que $\xrightarrow{*}_S = \xrightarrow{*}_{Rc} o \xrightarrow{*}_C$.

Preuve:

Soit R = {l → r / l, r ∈ T_Σ }. Soit R^c le système de réécriture clos défini par R^c = { u → r / l→r ∈ R et u =C l} donc obtenu à partir de R en prenant pour membre gauche u de règle tout terme équivalent à l modulo C. On peut remarquer que la relation de réécriture $\rightarrow_{Rc}$ est égale à la relation $\rightarrow_{R,C}$. L'inclusion de la relation $\xrightarrow{*}_{Rc} o \xrightarrow{*}_C$ dans $\xrightarrow{*}_S$ est évidente.

Montrons par induction sur la longueur de la réécriture l'inclusion inverse.

Le résultat est évident si la réécriture par S est de longueur 0 ou 1.

Supposons le résultat vrai pour une réécriture de longueur n et considérons deux termes t et t' de T_Σ tel que $t \xrightarrow{*}_S t'$ par une dérivation de longueur n+1.

Par hypothèse d'induction, nous avons: $t \xrightarrow{*}_{Rc} u \xrightarrow{*}_C v \rightarrow_S t'$.

Si la réécriture de v en t' se fait à l'aide d'un règle de C, nous obtenons immédiatement la décomposition souhaitée.

Si la réécriture de v en t' se fait par une règle de R, nous avons $t \xrightarrow{*}_{Rc} u \xrightarrow{*}_C v \rightarrow_R t'$. Soit l→r la règle de R utilisée, il existe un contexte c tel que v=c(l) et t'=c(r). $u \xrightarrow{*}_C v=c(l)$ donc $c(l)=v \xrightarrow{*}_C c'(l')=u$ avec c(x) et c'(x), l et l' équivalents modulo C. Il existe, par construction, la règle l'→r dans R^c et par conséquent, nous avons $c'(l')=u \rightarrow_{Rc} c'(r)$. Les contextes c et c' étant équivalents modulo C, $c'(r) \xrightarrow{*}_C c(r)=t'$ et donc $t \xrightarrow{*}_{Rc} c'(r) \xrightarrow{*}_C t'$ ce qui nous donne la décomposition souhaitée.❑

Proposition 2: *S préserve la reconnaissabilité.*

Preuve: Ce résultat est une conséquence de la proposition 1. En effet, de cette proposition, nous déduisons l'égalité entre S(F) et $C(R^c(F))$. De plus, les systèmes de réécriture R^c et C préservent la reconnaissabilité car R^c est un système de réécriture clos et C est un système de réécriture monadique (voir section 4.6, on pourrait également construire un transducteur linéaire ascendant pour $\xrightarrow{*}_C$).❑

Une application de ce résultat est que les différents problèmes d'accessibilité pour les systèmes de réécriture clos modulo la commutativité sont décidables. On pourrait réaliser la compilation d'un tel système de réécriture en composant un transducteur d'arbre clos (compilation de R^c) et un transducteur linéaire ascendant (compilation de C).

Proposition 3: *La terminaison d'un système de réécriture clos modulo la commutativité est décidable. La confluence est décidable quand le système de réécriture clos termine modulo C.*

Preuve:

R termine modulo C si et seulement si R^c termine d'après la décomposition donnée dans la proposition 1. R^c est un système de réécriture clos et la terminaison d'un système de réécriture clos est décidable ([HULA78]).

Le système de réécriture $S=R \cup C$ est confluent si et seulement si R^c est Church-Rosser modulo C. On prouve ce résultat en utilisant la décomposition de la proposition 1.

Si R termine modulo C, on peut alors utiliser les résultats connus de la réécriture équationnelle pour la décidabilité de la confluence. Il suffit, en effet, de vérifier les conditions données dans le théorème 2 de la section 2.3.2. On peut également utiliser les résultats obtenus par G.Huet ([HUET80]) car R est un système de réécriture clos et donc est linéaire à gauche et C est linéaire.❑

Tous les résultats donnés dans ce paragraphe 4.2 s'étendent sans difficulté aux cas de *plusieurs opérateurs commutatifs* et également au cas où l'on considère des *règles de permutation sur un opérateur*, c'est-à-dire aux cas de règles de la forme $f(x_1,\ldots,x_n)\rightarrow f(x_{\sigma(1)},\ldots,x_{\sigma(n)})$ où σ est une permutation de [1,n].

Le problème de la décidabilté de la confluence de tels systèmes, sans l'hypothèse de terminaison, reste posé, et, de façon plus générale, le problème de la décidabilité de la réécriture close commutative, au sens de M.Dauchet et S.Tison dans [DATI90] pour la réécriture close, reste posé. c'est-à-dire la décidabilité de toute formule du premier ordre où les variables sont les termes et les prédicats $\rightarrow_S$, $\stackrel{*}{\rightarrow}_S$ pour tout S système de réécriture clos modulo la commutativité. On peut remarquer que les méthodes utilisées (codage canonique et reconnaissable de la transformation) ne s'appliquent plus dans le cas considéré ici. Ce problème reste donc à étudier.

4.3 Systèmes de réécriture clos modulo l'associativité

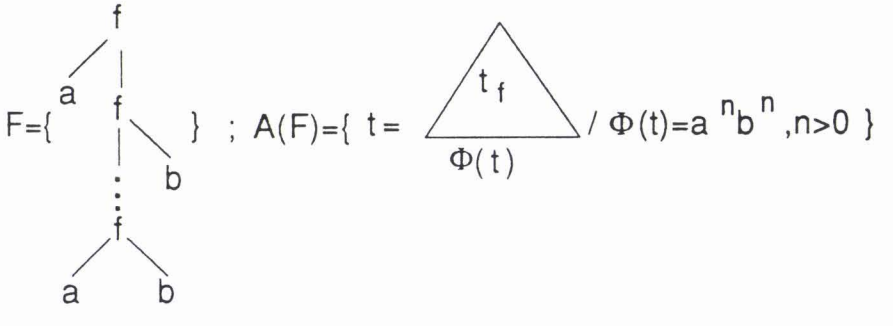
Soit R un système de réécriture clos, soit f une lettre d'arité 2 de Σ, soit $A=\{f(f(x,y),z)\rightarrow f(x,f(y,z));\ f(x,f(y,z))\rightarrow f(f(x,y),z)\}$ et soit S le système de réécriture $R\cup A$.

Considérons sur l'alphabet $\Sigma=\{f,a,b\}$ avec f d'arité 2, a et b d'arité 0 le cas où le système de réécriture R est vide et donc le cas S=A. Soit la forêt reconnaissable définie dans la figure 4.3, S(F) est alors l'ensemble des arbres de T_Σ tels que le feuillage soit égal à l'ensemble des mots de la forme a^nb^n avec n strictement positif. F engendrée par une grammaire régulière est une forêt reconnaissable, par contre S(F) n'est pas reconnaissable. Nous pouvons donc conclure que, en général, la *reconnaissabilité n'est pas préservée pour un système de réécriture clos modulo l'associativité.*

FIGURE 4.3

Soit la forêt F engendrée par la grammaire régulière

$G=\{\ \sigma \longrightarrow f(a,f(\sigma,b)) \mid f(a,b)\ \}$



Soit Γ un alphabet fini et W un système de réécriture (fini) sur Γ^* tels que les membres gauches et droits de toute règle de W ne soient pas réduits au mot vide. Soit Σ l'alphabet gradué obtenu en ajoutant à Γ un opérateur binaire f. A tout mot u de Γ^+, on associe le terme t_u de T_Σ défini par: si $|u|=1$, $t_u=u$; si $|u|>1$ et $u=a_1...a_n$, $t_u=f(a_1,f(...,f(a_{n-1},a_n)...))$. Soit R le système de réécriture clos défini de la façon suivante: $R=\{t_u \rightarrow t_v / u \rightarrow v \in W\}$. Soit $S=R \cup A$. Soit t et t' deux termes de T_Σ.

Fait: $(\Phi(t) \xrightarrow{*}_W \Phi(t')) \Leftrightarrow (t \xrightarrow{*}_S t')$.

Preuve: Notons que $t \xrightarrow{*}_A t'$ si et seulement si $\Phi(t)=\Phi(t')$. Ce résultat est une conséquence directe de l'associativité de l'opérateur f et de la définition de Σ (f est le seul symbole d'arité non nulle de Σ).

Preuve de $\Leftarrow$*:* Si $t \rightarrow_S t'$ par une règle de A, nous avons $\Phi(t)=\Phi(t')$.
Si $t \rightarrow_S t'$ par la règle $t_u \rightarrow t_v$ de R, alors $t=c(t_u)$, $t'=c(t_v)$, il existe deux mots w et w' de Γ^* tels que $\Phi(t)=wuw'$ et $\Phi(t')=wvw'$ et, par conséquent, nous obtenons $\Phi(t) \rightarrow_W \Phi(t')$. Nous en déduisons par récurence la preuve de cette implication.

Preuve de $\Rightarrow$*:* Si $\Phi(t) \rightarrow_W \Phi(t')$ par la règle $u \rightarrow v$ de W, il existe deux termes t_1 et t'_1 de T_Σ et un contexte c tels que $t_1=c(t_u)$, $t'_1=c(t_v)$, $\Phi(t_1)=\Phi(t)$, $\Phi(t'_1)=\Phi(t')$. En utilisant la remarque préliminaire, nous avons $t \xrightarrow{*}_A t_1$ et $t'_1 \xrightarrow{*}_A t'$ (t et t_1, t' et t'_1 ont même feuillage). Nous obtenons donc que $t \xrightarrow{*}_A t_1 \rightarrow_R t'_1 \xrightarrow{*}_A t'$, soit encore $t \xrightarrow{*}_S t'$. Par récurrence, nous en déduisons donc la seconde implication.❑

De ce résultat, nous pouvons déduire:

Proposition: *Les différents problèmes d'accessibilité, la terminaison, la confluence des systèmes de réécriture clos modulo l'associativité sont indécidables.*

Preuve: Les problèmes correspondants sont en effet indécidables pour les systèmes de réécriture de mots (la restriction sur le mot vide ne changeant pas le résultat).❑

4.4 Systèmes de réécriture clos modulo la commutativité et l'associativité

Soit R un système de réécriture clos, soit f une lettre d'arité 2 de Σ, soit $AC=\{f(x,y)\rightarrow f(y,x);\ f(f(x,y),z)\rightarrow f(x,f(y,z));\ f(x,f(y,z))\rightarrow f(f(x,y),z)\}$ et soit S le système de réécriture $R\cup AC$.

Soit F la forêt reconnaissable définie dans la figure 4.3, soit R vide et donc S=AC, on a alors $S(F)=\{\ t\in T_\Sigma\ /\ |\Phi(t)|_a = |\Phi(t)|_b\ \}$ qui n'est pas une forêt reconnaissable et, donc, en général:

Un système de réécriture clos modulo l'associativité et la commutativité ne préserve pas la reconnaissabilité.

Nous avons étudié le problème d'accessibilité du premier ordre pour les systèmes de réécriture clos modulo AC. Nous avons obtenu des résultats partiels de décidabilité en nous limitant à des termes tels que l'opérateur associatif et commutatif f n'apparaît qu'au sommet de l'arbre, ceci pour les termes t et t' pour lesquels on veut résoudre le problème d'accessibilité, mais aussi pour les règles du système de réécriture dans lesquelles l'opérateur f apparait. Formellement:

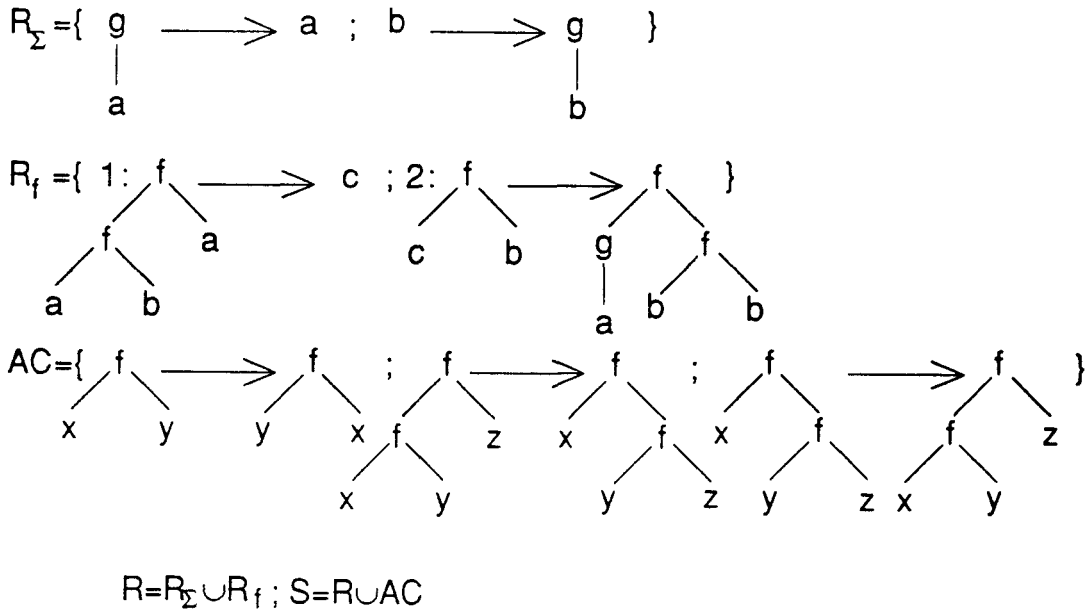
soit $T_{f\Sigma}=\{t=t_f(t_1,\ldots,t_n)\ /\ t_f\in T_{\{f\}}(X)$ et $\forall i\in[1,n],\ t_i\in T_\Sigma\}$.
soit $R= R_\Sigma\cup R_f$ avec $R_\Sigma=\{\ l\rightarrow r\ /\ l,r\in T_{\Sigma-\{f\}}\}$ et $R_f=\{l\rightarrow r\ /l\in T_{f\Sigma},\ l\notin T_\Sigma,\ r\in T_{f\Sigma}\}$.

R est donc partitionné en deux systèmes, l'un R_Σ correspond à l'ensemble des règles de R où l'opérateur f n'apparaît pas, l'autre R_f est constitué des règles où f apparaît.

On peut également remarquer que la condition l n'appartient pas à T_Σ pour les règles de R_f a pour but d'éviter de générer des termes contenant l'opérateur f à partir de termes n'en contenant pas.

Exemple: Voir figure 4.4.a.

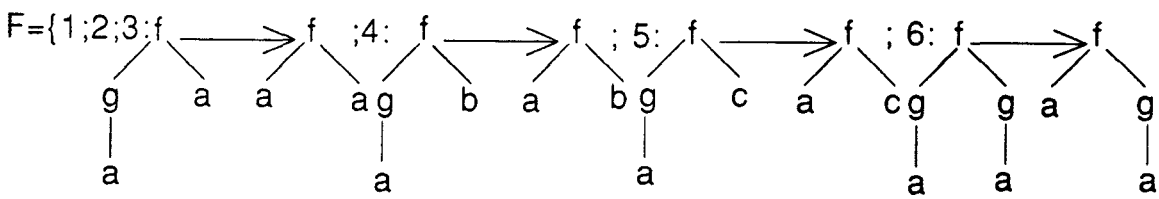
Figure 4.4.a



$R=R_\Sigma \cup R_f$; $S=R \cup AC$

On définit G={a,b,c}; D={a,b,g(a)}; Sub={a,b,c,g(a)}.

On construit le système de réécriture F, on obtient:



Lemme 1: Il existe un système de réécriture clos F sur $T_{f\Sigma}$ tel que:

$$\xrightarrow{*}_S = \xrightarrow{*}_{R_\Sigma} o \xrightarrow{*}_{F \cup AC} o \xrightarrow{*}_{R_\Sigma}.$$

Preuve: L'intérêt de ce lemme est de décomposer la réécriture par S en séparant les réécritures closes par R_Σ sur les sous-termes clos ne contenant pas l'opérateur f des réécritures sur des sous-termes contenant l'opérateur f.

Construction de F:

Soit $t=t_f(t_1,\ldots,t_n)$ un terme de $T_{f\Sigma}-T_\Sigma$, on lui associe $Sub(t)=\{t_1,\ldots,t_n\}$ et $Sub(t)=\{t\}$ si t appartient à T_Σ.

On définit G comme l'union des Sub(l) et D comme l'union des Sub(r) pour toute règle $l \to r$ de R_f, rappelons que nous ne considérons que des systèmes de réécriture finis.

On définit $F = \{ f(u,w) \to f(v,w) \ / \ w \in G \cup D, u \in D, v \in G \text{ et } u \xrightarrow{*}_{R_\Sigma} v \} \cup R_f$. Le sous-terme w n'est introduit que dans le but d'obtenir des règles dans F qui ont toutes la même configuration (toujours un opérateur f au sommet). La construction de F est effective car les ensembles G et D sont des ensembles finis et le problème d'accessibilité pour le système de réécriture clos R_Σ est décidable.

Preuve de $\supset$*:* les réécritures par les règles créées peuvent, par définition être simulées par des réécritures par R_Σ, d'où le résultat.

Preuve de $\subset$*:* La preuve se fait par récurrence sur la longueur de la réécriture. Le résultat est évident pour une réécriture de longueur 0 et 1. Supposons le résultat vrai pour toute dérivation de longueur au plus n. Considérons une dérivation d'un terme t en un terme t' de longueur n+1 par S. En utilisant l'hypothèse de récurrence, nous obtenons:

$$t \xrightarrow{*}_{R_\Sigma} t_1 \xrightarrow{*}_{F \cup AC} t_2 \xrightarrow{*}_{R_\Sigma} t_3 \to_S t'.$$

Premier cas: La réécriture de t_3 en t' se fait à l'aide d'une règle de R_Σ, le résultat est immédiat.

Deuxième cas: La réécriture de t_3 en t' se fait à l'aide d'une règle de AC. Par définition de R_Σ (règles closes ne faisant pas intervenir l'opérateur f), les règles de AC commutent avec les règles de R_Σ, par conséquent, la réécriture peut se décomposer sous la forme souhaitée.

Troisième cas: La réécriture de t_3 en t' se fait à l'aide d'une règle de R_f. Nous avons donc: $t \xrightarrow{*}_{R_\Sigma} t_1 \xrightarrow{*}_{F \cup AC} t_2 \xrightarrow{*}_{R_\Sigma} t_3 \to_{R_f} t'$.

Soit $l \to r$ la règle de R_f utilisée et c le contexte associé à la réécriture de t_3 en t'. Nous avons donc $t_3 = c(l)$, $t' = c(r)$, nous avons aussi $t_2 = c(u)$, le contexte ne pouvant pas être modifié dans la réécriture de t_2 en t_3 par des règles de R_Σ.

l contient l'opérateur f par définition de R_f, u et l sont donc de la forme $u=l_f(u_1,...,u_n)$, $l=l_f(l_1,...,l_n)$ avec u_i et l_i dans T_Σ pour tout i. Les règles de R_Σ ne contenant pas f, nous avons, pour tout i, $u_i \xrightarrow{*}_{R_\Sigma} l_i$.

Si u_i est un élément de D (sous-terme de T_Σ d'un membre droit de règle), la réécriture de u_i en l_i peut se faire à partir de t_2 en utilisant des règles de F-R_f, par construction de F.

Si u_i n'est pas un élément de D, u_i n'a pas été modifié par la réécriture de t_1 en t_2, alors, il existe un sous-terme t_i de T_Σ de $t=t_f(t_1,...,t_n)$ tel que $t_i \rightarrow_{R_\Sigma} u_i$ et comme $u_i \xrightarrow{*}_{R_\Sigma} l_i$, nous avons $t_i \xrightarrow{*}_{R_\Sigma} l_i$.

La réécriture de t_i en l_i peut donc être faite avant toute utilisation de règle de F∪AC. Il existe donc des termes t'_1, t'_2 et t'_3 tels que nous ayons: $t \xrightarrow{*}_{R_\Sigma} t'_1 \xrightarrow{*}_{F\cup AC} t'_2 \rightarrow_{R_f} t'_3 \xrightarrow{*}_{R_\Sigma} t'$ d'où nous déduisons la décomposition souhaitée.❑

Nous allons utiliser cette décomposition pour prouver la décidabilité du problème d'accessibilité du premier ordre pour S dans $T_{f\Sigma}$. Nous allons, dans un premier temps nous intéresser au problème d'accessibilité pour le système de réécriture F∪AC qui sera noté FAC et prouver ce problème équivalent au problème d'accessibilité pour un réseau de Pétri en codant la réécriture par FAC sur des termes de $T_{f\Sigma}$ en une réécriture dans les mots modulo la commutation totale.

Soit $Sub=G\cup D=\{u_1,...,u_m\}$ (Sub est l'ensemble des sous-termes de T_Σ qui apparaissent dans les règles de R_f), soit $\Gamma=\{x_1,...,x_m\}$ un alphabet en bijection avec Sub.

Sur Γ^*, on définit la relation ≡ par m≡m' si et seulement si, pour toute lettre x de Γ, $|m|_x=|m'|_x$ où $|m|_x$ désigne le nombre d'occurences de la lettre x dans m. La relation ≡ est une relation d'équivalence sur Γ^* compatible avec la concaténation.

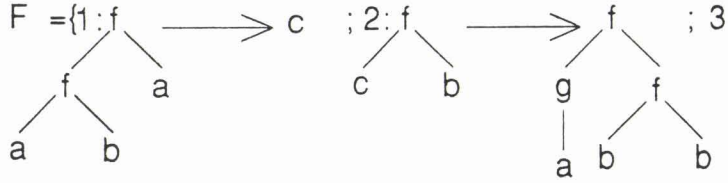
Un représentant canonique d'une classe d'équivalence est de la forme $x_1^{y1}...x_m^{ym}$ où pour tout i, y_i désigne le nombre d'occurences de la lettre x_i.

A tout terme $t=t_f(t_1,\ldots,t_n)$ de $T_{f\Sigma}$, on associe la classe d'équivalence C(t) de $\Gamma^*/_{\equiv}$ de représentant canonique $x^{y1}_1 \ldots x^{ym}_m$ où y_i désigne le nombre d'occurences du terme u_i de Sub dans $(t_1,\ldots,t_n)$ et le multi-ensemble N(t) constitué des t_i qui ne sont pas dans Sub. A toute règle $l \to r$ de F, on associe la règle $C(l) \to C(r)$ et à F correspond donc un système de réécriture C(F) sur $\Gamma^*/_{\equiv}$.

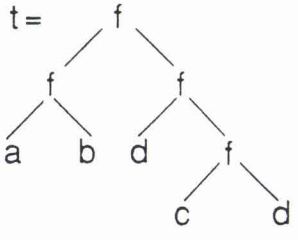
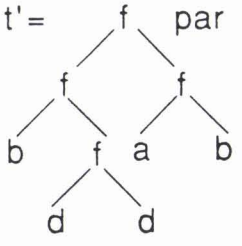
Exemple: Voir figure 4.4.b.

Figure 4.4.b

F ={1 : f → c ; 2: f → f ; 3; 4; 5; 6 }



Les règles 3, 4, 5, 6 permettent de réécrire g(a) en a.

t =  $\xrightarrow[FAC]{*}$ t' =  par les règles 2,4,1,2,4 et les règles de AC

Soit $\Gamma=\{x,y,z,t\}$ en bijection avec Sub={a,b,c,g(a)}

On a C(t)=xyz, N(t)={d,d}, $C(t')=xy^2$, N(t')={d,d}

On construit $C(F)=\{1:x^2y \to z;2:yz \to y^2t;3:xt \to x^2;4:yt \to xy;5:zt \to xz;6:t^2 \to xt\}$

On a $C(t) \xrightarrow[C(F)]{*} C(t')$ (par les règles 2,4,1,2,4 de C(F)) et N(t)=N(t').

Lemme 2: Pour tous termes t et t' de $T_{f\Sigma}$, on a:

(i): $(t \xrightarrow{*}_{FAC} t') \Leftrightarrow (\ C(t) \xrightarrow{*}_{C(F)} C(t')$ et N(t)=N(t'))

(ii): On peut décider si $t \xrightarrow{*}_{FAC} t'$, c'est-à-dire le problème d'accessibilité du premier ordre pour FAC dans $T_{f\Sigma}$ est décidable.

Preuve:

Preuve de (i): Remarquons, tout d'abord, que $t \stackrel{*}{\rightarrow}_{AC} t'$ si et seulement si $C(t)=C(t')$ et $N(t)=N(t')$.

Si t se réécrit en t' par la règle $l \rightarrow r$ de F, nous avons $N(t)=N(t')$ et par construction de C(t), C(t') et C(F), C(t) se réécrit en C(t') par la règle $C(l) \rightarrow C(r)$. Nous en déduisons, par récurrence la première implication.

Si t et t' sont tels que $C(t) \rightarrow_{C(F)} C(t')$ par la régle $C(l) \rightarrow C(r)$ et $N(t)=N(t')$, il existe un terme $t_1=c(l)$, un terme $t'_1=c(r)$ qui vérifient de plus $C(t)=C(t_1)$ et $N(t)=N(t_1)$, $C(t')=C(t'_1)$ et $N(t')=N(t'_1)$ et donc d'après la remarque initiale $t \stackrel{*}{\rightarrow}_{AC} t_1=c(l)$ et $c(r)=t'_1 \stackrel{*}{\rightarrow}_{AC} t'$ et donc $t \stackrel{*}{\rightarrow}_{FAC} t'$. Par récurrence, nous pouvons déduire de ce résultat la seconde implication.❑ *Finpreuve(i).*

Preuve(ii): Etant donné deux termes t et t' de $T_{f\Sigma}$, on peut construire C(t), C(t'), N(t) et N(t'). N(t) et N(t') sont finis donc l'égalité entre N(t) et N(t') est décidable. Le problème se ramène donc à la décidabilité du problème d'accessibilité pour C(F) dans $\Gamma^*/_{\equiv}$. Au système de réécriture C(F), on associe le réseau de Pétri défini de la façon suivante:
Ensemble de places: $P=\{p_1,...,p_m\}$ en bijection avec Γ.
Ensemble de transitions: $T=\{t_1,...,t_q\}$ en bijection avec C(F).
Matrices de transition Pré et Post définies par: $Pré(p_j,t_i)=l_{ij}$ et $Post(p_j,t_i)=r_{ij}$ avec $l_i \rightarrow r_i$ règle de C(F) associée à la transition t_i de T, $l_i = x_1^{li1}...x_m^{lim}$, $r_i = x_1^{ri1}...x_m^{rim}$.

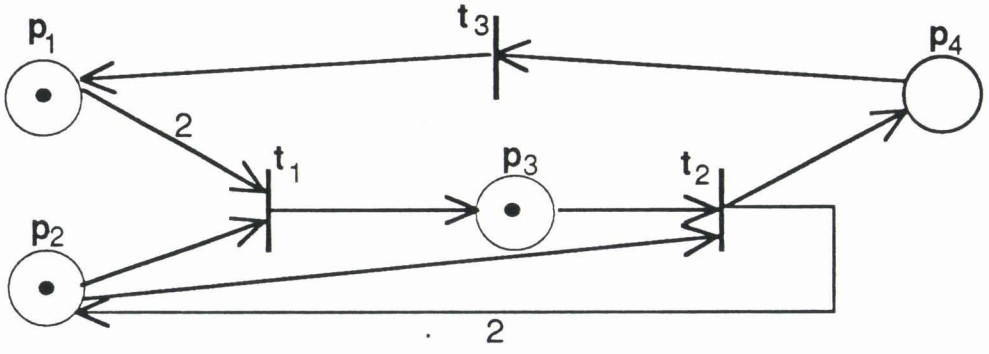
Exemple: Voir figure 4.4.c.

A chaque classe d'équivalence de représentant canonique $w=x_1^{y1}...x_m^{ym}$, on associe le vecteur v(w) de N^m défini par $v(w)(i)=y_i$.

Décider si la classe d'équivalence de représentant canonique w peut se réécrire en la classe d'équivalence de représentant w' est équivalent au problème de décider si le vecteur v(w') est accessible pour le réseau de Pétri P avec le marquage initial v(w). Le problème d'accessibilité pour les réseaux de Pétri est décidable ([KOSA82], [MAYR84]), on en déduit donc la décidabilité pour C(F) dans $\Gamma^*/_{\equiv}$ et donc la décidabilité du problème d'accessibilité pour FAC dans $T_{f\Sigma}$.

Figure 4.4.c.

Soit $\Gamma=\{x,y,z,t\}$.
Soit $C(F)=\{1:x^2y\rightarrow z;\ 2:yz\rightarrow y^2t;\ 3:xt\rightarrow x^2;\ 4:yt\rightarrow xy;\ 5:zt\rightarrow xz;\ 6:t^2\rightarrow xt\}$.
Soit $P=\{p_1,p_2,p_3,p_4\}$ en bijection avec Γ.
On peut simplifier les règles 3,4,5,6 en $3:t\rightarrow x$. Soit t_1,t_2,t_3 associées aux règles 1,2 et 3 de C(F). Nous obtenons le réseau de Pétri de marquage initial une marque dans les places p_1,p_2 et p_3 correspondant à la classe de représentant xyz associée à l'arbre t de la figure 4.4.b.



On peut remarquer que les deux problèmes d'accessibilité (pour les réseaux de Pétri et pour un système de réécriture dans $\Gamma^*/_{\equiv}$) sont équivalents (on peut associer à tout réseau de Pétri un système de réécriture de cette forme, [SCHW85]).❑ *Fin preuve(ii).*
❑ *Fin preuve lemme 2.*

Proposition: *Le problème d'accessibilité pour un système de réécriture clos R modulo la commutativité et l'associativité d'un opérateur f est décidable dans* $T_{f\Sigma}$.

Preuve: Soit t et t' deux termes de $T_{f\Sigma}$, soit $S=R\cup AC$ avec $R=R_f\cup R_\Sigma$, le problème est donc de décider si t peut se réécrire en t' par S.

Premier cas: $t\in T_\Sigma$. Si $t'\in T_\Sigma$, alors le problème se ramène à décider si t peut se réécrire en t' par le système de réécriture clos R, problème connu décidable. Si $t'\notin T_\Sigma$, t ne peut se réécrire en t' par S, en effet, par définition de R_f, nous interdisons la génération de termes contenant l'opérateur f à partir de termes n'en contenant pas.

Deuxième cas: $t=t_f(t_1,...,t_n)\in T_{f\Sigma}-T_\Sigma$ avec $t_f\in T_{\{f\}}(X)$ et, pour tout i, $t_i\in T_\Sigma$. Nous utilisons la décomposition trouvée dans le lemme 1, par conséquent, t se réécrit en t' par S si et seulement si il existe deux termes u et v de $T_{f\Sigma}$ tels que: $t \xrightarrow{*}_{R_\Sigma} u \xrightarrow{*}_{FAC} v \xrightarrow{*}_{R_\Sigma} t'$.

Soit $t=t_f(t_1,...,t_n)\in T_{f\Sigma}$, nous définissons l'ensemble F(t) par:

$F(t)=\{t_f(u_1,...,u_n)\in T_{f\Sigma}$ / si $R_\Sigma(t_i)\cap Sub=\emptyset$, $u_i=t_i$ }

si $R_\Sigma(t_i)\cap Sub=\{m_1,...,m_j\}$, $u_i\in\{t_i,m_1,...,m_j\}$ }.

L'idée de cette définition est que chaque t_i peut se réécrire en des sous-termes de T_Σ qui pourront, ou pas, être réécrits par FAC, rappelons, en effet, que Sub est l'ensemble des sous-termes de T_Σ qui apparaissent dans les règles de F. F(t) est donc défini de façon à décrire toutes ces possibilités. R_Σ est un système de réécriture clos, par conséquent, pour tout i, $R_\Sigma(t_i)$ est une forêt reconnaissable, de plus, Sub est un ensemble fini, on peut donc effectivement construire, pour tout i, l'ensemble $R_\Sigma(t_i)\cap Sub$. La construction de F(t) est donc effective. De façon symétrique, nous pouvons définir $F^{-1}(t)$ en utilisant la relation $_{R_\Sigma}\leftarrow$.

Fait: $(t \xrightarrow{*}_S t') \Leftrightarrow$ ($\exists T\in F(t)$, $\exists T'\in F^{-1}(t')$, $T \xrightarrow{*}_{FAC} T'$ et il existe une bijection f de N(T) dans N(T') telle que, pour tout u de N(T), $u \xrightarrow{*}_{R_\Sigma} f(u)$)

La preuve du fait est sans difficultés, elle utilise la décomposition du lemme 1 et la définition des ensembles F(t) et $F^{-1}(t')$. Ces ensembles sont finis, le problème d'accessibilité pour FAC est décidable (lemme 2), de plus, pour tout T et T', les ensembles N(T) et N(T') sont finis, le problème d'accessibilité pour R_Σ est décidable, par conséquent, l'existence de la bijection f est elle aussi décidable, d'où le résultat.❑

4.5 Systèmes de réécriture clos modulo la commutativité, l'associativité et l'idempotence

Soit R un système de réécriture clos, soit f une lettre d'arité 2 de Σ, soit ACI=$\{f(x,y)\rightarrow f(y,x);\ f(f(x,y),z)\rightarrow f(x,f(y,z));\ f(x,f(y,z))\rightarrow f(f(x,y),z);\ f(x,x)\rightarrow x;\ f(x,y)\rightarrow f(f(x,x),y)\}$ et soit S le système de réécriture $R\cup ACI$.

Nous utilisons les mêmes notations que dans le cas AC. Soit $T_{f\Sigma}=\{t=t_f(t_1,\ldots,t_n) / t_f\in T_{\{f\}}(X)$ et $\forall i\in[1,n],\ t_i\in T_\Sigma\}$ et soit $R= R_\Sigma\cup R_f$ avec $R_\Sigma=\{ l\rightarrow r / l,r\in T_{\Sigma-\{f\}}\}$ et $R_f=\{l\rightarrow r / l\in T_{f\Sigma},\ l\notin T_\Sigma,\ r\in T_{f\Sigma}\}$.

La seconde règle permettant de définir l'idempotence de l'opérateur f a été modifiée ($f(x,y)\rightarrow f(f(x,x),y)$ à la place de $x\rightarrow f(x,x)$) pour préserver la configuration des arbres de $T_{f\Sigma}$.
Nous noterons i_1: $f(x,x)\rightarrow x$ et i_2: $f(x,y)\rightarrow f(f(x,x),y)$.

Soit $\Sigma=\{a,b,g,f\}$ avec a et b d'arité 0, g d'arité 1 et f d'arité 2.
Soit $F=\{ f(g^n(a),b) / n\geq 0 \}$, soit R réduit à l'ensemble vide et donc on considère S=ACI. F est une forêt reconnaissable, la forêt S(F) est :
$S(F) = \{ t_f(t_1,\ldots,t_p) / t_f\in T_{\{f\}}(X),\ \exists n\geq 0,\ (\forall i, t_i=b$ ou $t_i=g^n(a))$ et
$(\exists j,k, t_j=b, t_k=g^n(a)\ \}$.
La forêt S(F) ainsi obtenue n'est pas reconnaissable. En général:
Un système de réécriture clos modulo l'associativité, la commutativité et l'idempotence ne préserve pas la reconnaissabilité.

Nous avons étudié le problème d'accessibilité du premier ordre pour ces systèmes de réécriture et avons obtenu des résultats partiels de décidabilité en considérant les mêmes restrictions que pour le cas AC.

Nous avons également prouvé que si les règles d'associativité, de commutativité et d'idempotence ne peuvent s'appliquer qu'à un nombre fini de termes (plus précisément, on ne peut dupliquer, commuter que les termes de l'ensemble Sub), la reconnaissabilité est préservée en se limitant à des termes où l'opérateur f apparaît au sommet de l'arbre.

Exemple: Voir figure 4.5.a.

Lemme1: Il existe un système de réécriture clos F sur $T_{f\Sigma}$ tel que
$$\xrightarrow{*}_S = \xrightarrow{*}_{i_2} \circ \xrightarrow{*}_{R_\Sigma} \circ \xrightarrow{*}_{F\cup ACI} \circ \xrightarrow{*}_{R_\Sigma} \circ \xrightarrow{*}_{i_1}.$$

Le but de ce lemme, comme dans le cas AC, est de décomposer la réécriture par S en séparant les réécritures closes par R_Σ sur les sous-termes clos ne contenant pas l'opérateur f des réécritures sur des sous-termes contenant l'opérateur f pour pouvoir coder la réécriture par F∪ACI en une réécriture dans les mots.

Preuve: La construction de F est identique à la construction faite dans le cas AC. Par la même démonstration que dans le cas AC, nous obtenons: $t \xrightarrow{*}_S t'$ si et seulement si il existe des arbres t_1 et t_2 tels que $t \xrightarrow{*}_{R_\Sigma\cup\{i_1,i_2\}} t_1 \xrightarrow{*}_{F\cup ACI} t_2 \xrightarrow{*}_{R_\Sigma\cup\{i_1,i_2\}} t'$. Le système de réécriture $R_\Sigma\cup\{i_1,i_2\}$ vérifie: $R_\Sigma\cup\{i_1,i_2\} = \xrightarrow{*}_{i_2} \circ \xrightarrow{*}_{R_\Sigma} \circ \xrightarrow{*}_{i_1}$. En effet, si u se réécrit en u' par R_Σ et u' est copié par i_2, on peut d'abord copier u et réécrire ensuite chacune des copies en u' par R_Σ, la propriété symétrique pour l'effacement par i_1 et les propriétés de i_1 et i_2 donnent le résultat. Donc il existe des arbres t'_1, t'_2, t_1, t_2, t''_1, t''_2 de $T_{f\Sigma}$ tels que:
$t \xrightarrow{*}_{i_2} t'_1 \xrightarrow{*}_{R_\Sigma} t'_2 \xrightarrow{*}_{i_1} t_1 \xrightarrow{*}_{F\cup ACI} t_2 \xrightarrow{*}_{i_2} t''_1 \xrightarrow{*}_{R_\Sigma} t''_2 \xrightarrow{*}_{i_1} t'$ d'où la décomposition annoncée.❑

Nous allons coder la réécriture par le système de réécriture F∪ACI (noté FACI). Nous utilisons les mêmes notations pour les ensembles M et Γ que dans le cas AC.

Sur Γ^*, on définit la relation ≡ par: m≡m' si et seulement si, pour toute lettre x de Γ, $|m|_x>0$ si et seulement si $|m'|_x>0$. La relation ≡ est une relation d'équivalence compatible avec la concaténation. Un représentant canonique d'une classe d'équivalence est de la forme $x_1^{y_1}\ldots x_m^{y_m}$ où pour tout i, y_i vaut 0 (x_i n'occure dans aucun mot de la classe) ou 1 (x_i occure dans tout mot de la classe avec un nombre quelconque d'occurences).

A tout terme $t=t_f(t_1,\ldots,t_n)$ de $T_{f\Sigma}$, on associe la classe d'équivalence C(t) de $\Gamma^*/_\equiv$ de représentant canonique $x_1^{y_1}\ldots x_m^{y_m}$ où y_i vaut 0 si le terme u_i de Sub n'apparaît pas dans $(t_1,\ldots,t_n)$ et 1 sinon et l'ensemble N(t) constitué des t_i qui ne sont pas dans Sub.

A toute règle $l \to r$ de F, on associe l'ensemble de règles défini de la façon suivante: soit $l=l_f(l_1,\ldots,l_n)$ et $r=r_f(r_1,\ldots,r_p)$, on définit les règles $l_\Gamma \to r_\Gamma$ avec $l_\Gamma=C(l)$ et $r_\Gamma = x_1^{y_1}\ldots x_m^{y_m}$ où $y_i=1$ si $u_i \in \{r_1,\ldots,r_p\}$, $y_i=0$ si $u_i \notin \{r_1,\ldots,r_p\} \cup \{l_1,\ldots,l_n\}$, $y_i=0$ ou 1 si $u_i \notin \{r_1,\ldots,r_p\}$ et $u_i \in \{l_1,\ldots,l_n\}$. La définition de r_Γ est telle que si un terme u de Sub apparaît en membre gauche d'une règle et pas en membre droit (troisième cas), on peut, soit appliquer la règle et effacer u (cas $y_i=0$), soit copier u puis appliquer la règle (cas $y_i=1$). A F correspond donc un système de réécriture C(F) sur $\Gamma^*/_{\equiv}$.

Exemple: Voir figure 4.5.a.

Lemme 2: Pour tous termes t et t' de $T_{f\Sigma}$, on a:
(i): $(t \xrightarrow{*}_{FACI} t') \Leftrightarrow (C(t) \xrightarrow{*}_{C(F)} C(t')$ et $N(t)=N(t'))$
(ii): On peut décider si $t \xrightarrow{*}_{FACI} t'$, c'est-à-dire le problème d'accessibilité du premier ordre pour FACI dans $T_{f\Sigma}$ est décidable.

Preuve:

Preuve de (i): Remarquons, tout d'abord, que, pour t et t' dans $T_{f\Sigma}$, on a:
$(t \xrightarrow{*}_{ACI} t') \Leftrightarrow (C(t)=C(t')$ et $N(t)=N(t'))$.

Si t se réécrit en t' par la règle $l \to r$ de F telle que tout terme de Sub apparaissant dans l apparaît aussi dans R, alors C(t) se réécrit en C(t') par la règle $C(l) \to C(r)$ dans C(F).

Si t se réécrit en t' par la règle $l \to r$ de F, C(t) se réécrit en C(t') par la règle $l_\Gamma \to r_\Gamma$ de C(F) définie de la façon suivante: pour tout terme u_i de Sub qui apparaît dans l et pas dans R, si $t=c(l) \to_F t'=c(r)$ avec un contexte c tel que u_i apparaît dans c, on prend la règle telle que $y_i=1$ (u_i apparaît encore dans t') et si u_i n'apparaît pas dans c, on prend la règle telle que $y_i=0$ (u_i n'occure plus dans t'). On peut en déduire, par récurence, la première implication.

Si t et t' sont tels que $C(t) \to C(t')$ par la règle $l_\Gamma \to r_\Gamma$ de C(F) et $N(t)=N(t')$ alors il existe un terme $t_1=c(l)$, un terme $t'_1=c(r)$ tels que $C(t_1)=C(t)$, $C(t'_1)=C(t')$, $N(t_1)=N(t)$, $N(t'_1)=N(t')$ et donc tels que:
$t \xrightarrow{*}_{ACI} t_1=c(l) \to_F t'_1=c(r) \xrightarrow{*}_{ACI} t'$. On en déduit par récurence la seconde implication d'où le résultat annoncé.❑ *finpreuve(i)*

Figure 4.5.a

Exemple de sdr S:

R_Σ={ 1:g(a)→ a }

R_f={ 2:f(a,b)→ c }
3:f(c,b)→ g(g(a)) }

ACI={ 4:f(x,y)→ f(y,x) }
5:f(f(x,y),z)→ f(x,f(y,z)) }
6:f(x,f(y,z))→ f(f(x,y),z) }
7:f(x,x)→ x }
8:f(x,y)→ f(f(x,x),y) }

$R=R_f \cup R_\Sigma$ et $S=R \cup ACI$

Ensemble Sub et sdr F : Sub={a,b,c,g(g(a))}, on a $g(g(a)) \stackrel{*}{\rightarrow}_R a$ et donc $F=R_f \cup \{f(u,g(g(a)))\rightarrow f(u,a) / u \in Sub\}$.

Ensemble X et sdr C(F) : $X=\{x_1,x_2,x_3,x_4\}$ alphabet en bijection avec M et on définit C(F) comme l'ensemble de règles:

$x_1x_2\rightarrow x_3|x_1x_3|x_2x_3|x_1x_2x_3$ (règle 2)
$x_2x_3\rightarrow x_4|x_2x_4|x_3x_4|x_2x_3x_4$ (règle 3)
$x_1x_4\rightarrow x_1$ (règle de F)
$x_2x_4\rightarrow x_2x_1|x_2x_1x_4$ (" ")
$x_3x_4\rightarrow x_3x_1|x_3x_1x_4$ (" ")
$x_4 \rightarrow x_4x_1$ (" ").

Illustration du lemme 2:
Soit t_1=f(f(a,b), f(d, f(g(g(a)),d))). On a $C(t_1)=x_1x_2x_4$, $N(t_1)=\{d\}$.
Soit t_2=f(f(c,c),f(d,d)). On a $C(t_2)=x_3$, $N(t_2)=\{d\}$.

On peut remarquer que $t_1 \stackrel{*}{\rightarrow}_{F_{ACI}} t_2$ et vérifier que $C(t_1) \stackrel{*}{\rightarrow}_{C(F)} C(t_2)$ et $N(t_1)=N(t_2)=\{d\}$.

Preuve de (ii): Etant donné t et t', on peut construire C(t), C(t'), N(t) et N(t'). L'égalité entre les ensembles finis N(t) et N(t') est décidable. On peut décider si C(t) se réécrit en C(t') par C(F), il suffit de construire un automate d'états fini dont les états sont les représentants canoniques de $X^*/_{\equiv}$, les transitions représentent l'action des règles de C(F) sur ces représentants et l'état initial est C(t). Il suffit alors de décider si l'état C(t') est accessible pour l'automate ainsi construit.❑

Proposition 1 : *Le problème d'accessibilité pour un système de réécriture clos modulo la commutativité, l'associativité et l'idempotence d'un opérateur f dans $T_{f\Sigma}$ est décidable.*

Preuve: Soit t et t' deux termes de $T_{f\Sigma}$, d'après le lemme 1, t se réécrit en t' par S si et seulement si il existe t'_1, t'_2, t_1, t_2, t''_1 et t''_2 de $T_{f\Sigma}$ tels que: $t \xrightarrow{*}_{i_2} t'_1 \xrightarrow{*}_{R_\Sigma} t'_2 \xrightarrow{*}_{i_1} t_1 \xrightarrow{*}_{F\cup ACI} t_2 \xrightarrow{*}_{i_2} t''_1 \xrightarrow{*}_{R_\Sigma} t''_2 \xrightarrow{*}_{i_1} t'$.

Soit $t=t_f(t_1,\ldots,t_n)\in T_{f\Sigma}$, nous définissons l'ensemble F(t) comme l'ensemble des arbres $T_f(u_1,\ldots,u_N)$ de $T_{f\Sigma}$ où l'ensemble des u_i est construit de la façon suivante: si $R_\Sigma(t_i)\cap Sub=\varnothing$, on ajoute $u_k=t_i$ à l'ensemble, si $R_\Sigma(t_i)\cap Sub=\{m_1,\ldots,m_j\}$, on ajoute un sous-ensemble non vide de $\{m_1,\ldots,m_j\}\cup\{t_i\}$ à l'ensemble. On définit de façon symétrique l'ensemble $F^{-1}(t)$ en utilisant la relation ${}_{R_\Sigma}\leftarrow$.

Fait: $(t \xrightarrow{*}_S t') \Leftrightarrow$ (il existe T de F(t) et T' de $F^{-1}(t')$ tels que
(i) $C(T)\xrightarrow{*}_{C(F)} C(T')$ et
(ii) il existe une relation Φ dans N(T)xN(T') telle que
$Dom(\Phi)=N(T)$, $Im(\Phi)=N(T')$ et $\forall(u,v)\in\Phi$, $u \xrightarrow{*}_{R_\Sigma} v$).

Preuve du fait: La condition (i) est nécessaire pour les termes de Sub obtenus après les réécritures par i_2 et par R_Σ, la condition (ii) est nécessaire pour les termes qui ne sont pas dans Sub.

Preuve de $\Leftarrow$*:* Conséquence du lemme 2 et de la définition de F(T) et $F^{-1}(T')$.

Preuve de $\Rightarrow$*:* Soit $t=t_f(t_1,\ldots,t_n)$, pour chaque sous terme t_i, en utilisant la décomposition du lemme 1, on peut le copier par la règle i_2 puis réécrire chacune des copies en utilisant R_Σ, on peut alors obtenir des termes de Sub ou non.

Pour ceux de Sub, en utilisant le lemme 2, il suffit de considérer les sous ensembles de $R_\Sigma(t_i) \cap Sub$ (ensembles finis car Sub est fini).

Pour ceux qui ne sont pas dans Sub, par construction de F(t), il existe un terme T qui possède t_i comme sous terme. Si t_i n'appartient pas à Sub, t_i appartient à N(t) et on peut alors vérifier que t_i peut se réécrire par i_2 et R_Σ en certains sous termes t'_i de t' en utilisant (ii). Si t_i appartient à Sub, par définition de C(F), il existe une réécriture par C(F) telle que $y_i=1$ et donc t_i est un sous terme d'un T' de $F^{-1}(t')$, la construction de $F^{-1}(t')$ permet de vérifier que t_i se réécrit en certains sous termes t'_i de t'. On a les mêmes propriétés pour t' et $F^{-1}(t')$, on peut alors vérifier cas par cas que l'on peut construire T et T' vérifiant les propriétés (i) et (ii) à partir des termes t et t'.❑ *Fin preuve fait.*

F(t) et $F^{-1}(t')$ sont des ensembles finis, il existe un nombre fini de relations dans N(T)xN(T') car N(T) et N(T') sont finis, le problème d'accessibilité pour R_Σ est décidable nous permettent de conclure au résultat annoncé.❑

Nous allons maintenant prouver que, en se restreignant toujours à l'ensemble $T_{f\Sigma}$, la reconnaissabilité est préservée si on considère *l'associativité, commutativité et idempotence sur un nombre fini de termes.* L'idée étant d'éliminer le problème des sous termes de la forme $g^n(a)$ dans l'exemple donné au début de cette section 4.5. On peut également remarquer que cette démarche est sans intérêt dans le cas associatif commutatif. Nous reprenons les notations utilisées pour R_Σ, R_f, R et $Sub=\{u_1,...,u_m\}$.

Nous définissons un nouveau système de réécriture ACI comme suit: $ACI=\bigcup_i \{r_1(u_i),...,r_8(u_i)\}$ avec, pour tout u_i de M,

$r_1(u_i)$: $\sigma(u_i,x)\rightarrow\sigma(x,u_i)$ $\quad$ $r_2(u_i)$: $\sigma(x,u_i)\rightarrow\sigma(u_i,x)$

$r_3(u_i)$: $\sigma(\sigma(x,u_i),y)\rightarrow\sigma(x,\sigma(u_i,y))$ $\quad$ $r_4(u_i)$: $\sigma(x,\sigma(u_i,y))\rightarrow\sigma(\sigma(x,u_i),y)$

$r_5(u_i)$: $\sigma(\sigma(x,y),u_i)\rightarrow\sigma(x,\sigma(y,u_i))$ $\quad$ $r_6(u_i)$: $\sigma(x,\sigma(y,u_i))\rightarrow\sigma(\sigma(x,y),u_i)$

$r_7(u_i)$: $\sigma(u_i,u_i)\rightarrow u_i$ $\quad$ $r_8(u_i)$: $\sigma(x,u_i)\rightarrow\sigma(x,\sigma(u_i,u_i))$

Soit S le système de réécriture $R \cup ACI$. Nous allons comme précédemment décomposer la réécriture par S.

Lemme1': Il existe un système de réécriture clos F sur $T_{f\Sigma}$ tel que
$\xrightarrow{*}_S = \xrightarrow{*}_{R_\Sigma} o \xrightarrow{*}_{F\cup ACI} o \xrightarrow{*}_{R_\Sigma}$.

Preuve: Identique à la preuve du lemme 1 de cette section 4.5. La décomposition de la réécriture est ici simplifiée car, par définition de S, on ne peut dupliquer que les termes de Sub.❑

Lemme 2': Pour tous termes t et t' de $T_{f\Sigma}$, on a:
(i): $(t \xrightarrow{*}_{FACI} t') \Leftrightarrow (C(t) \xrightarrow{*}_{C(F)} C(t')$ et $N'(t)=N'(t'))$.
(ii): On peut décider si $C(t) \xrightarrow{*}_{C(F)} C(t')$.

Preuve: $\Gamma,\equiv,\Gamma^*/_\equiv$ et C sont définis comme précédemment.

Pour tout t de $T_\sigma(\Sigma)$, N'(t) est le p-uplet de sous termes de t qui ne sont pas dans Sub, i.e si $t=t_\sigma(t_1,\ldots,t_n)$, $g'(t)=(t_{i_1},\ldots,t_{i_p})$ avec $(t_1,\ldots,t_n)=(t_1,..,t_{i_1},..,t_{i_2},\ldots,t_{i_p},..,t_n)$et $\forall i\in\{i_1,\ldots,i_p\}$, $t_i\notin$ Sub et $\forall i\notin\{i_1,\ldots,i_p\}$, $t_i\in$ Sub.

L'égalité $N'(t_1)=N'(t_2)$ est donc ici une égalité de n-uplets ordonnés en effet les termes de T_Σ-Sub ne peuvent commuter entre eux. La preuve est identique à la preuve du lemme 2. De même, pour (ii).❑

Lemme 3 : Si F est une forêt reconnaissable incluse dans $T_{f\Sigma}$ alors FACI(F) est reconnaissable.

Preuve: Soit F une forêt reconnaissable incluse dans $T_{f\Sigma}$ et A un automate ascendant déterministe reconnaissant F.

On construit un automate ascendant déterministe B qui reconnait F et tel que, pour tout t de F, si $t=t_f(t_1,\ldots,t_n)$ et $\{t_1,\ldots,t_n\}\cap Sub=U$ alors $t\xrightarrow{*}_B (q,U)$ avec (q,U) final dans B. La construction est sans difficultés, il suffit de particulariser les termes de Sub (Sub est fini) et de mémoriser dans la reconnaissance l'ensemble des termes de M rencontrés. Pour celà, on construit d'abord les règles sur Σ-{f} de telle façon que pour tout u de Sub, $u\xrightarrow{*}_B (q,\{u\})$ avec $u\xrightarrow{*}_A q$ et pour tout t de T_Σ-M, $t\xrightarrow{*}_B (q,\varnothing)$ avec $t\xrightarrow{*}_A q$,puis les règles avec σ en prenant $\sigma((q_1,U_1),(q_2,U_2))\rightarrow\sigma(q,U_1\cup U_2)$ avec $\sigma(q_1,q_2)\rightarrow q$ dans A. Les états finaux sont de la forme (q,U) avec $U\in P(M)$ et q final dans A.

On a alors $F=_{U\in\wp(M)} F_U$ avec F_U forêt reconnaissable reconnue par l'automate B_U dont les règles sont celles de B et les états finaux tous les états finaux de B de la forme (q,U) (F_U est l'ensemble des termes de F qui ont comme sous termes dans M l'ensemble U).

On a FACI(F) = FACI($_{U\in\wp(M)}F_U$) = $_{U\in\wp(M)}$ FACI(F_U). Il suffit donc de démontrer que, pour tout U de P(M), la forêt F_U est reconnaissable (la classe des forêts reconnaissables est close par union).

Soit U un sous-ensemble de M, on peut déterminer tous les sous-ensembles V de M tels que $C(U)\stackrel{*}{\rightarrow}_{C(F)} C(V)$. On a alors FACI($F_U$) qui est égal à la réunion des F_V pour tous les sous-ensembles V de Sub tels que $C(U)\stackrel{*}{\rightarrow}_{C(F)}C(V)$ avec $F_V=\{t'\in T_{f\Sigma}/\ C(t')=C(V)$ et $\exists t\in F_U$ t.q $N'(t)=N'(t')\}$. Ceci se prouve de façon immédiate en utilisant le lemme 2. Il suffit donc de démontrer que pour tout V sous-ensemble de Sub, l'ensemble F_V est reconnaissable.

Soit donc U un sous ensemble de M, F_U la forêt reconnaissable correspondante, V un sous ensemble de M tel que $C(U)\stackrel{*}{\rightarrow}_{C(F)}C(V)$ avec $F_V=\{t'\in T_{f\Sigma}/\ C(t')=C(V)$ et $\exists t\in F_U$ t.q $N'(t)=N'(t')\}$.

Pour démontrer que F_V est reconnaissable, nous allons démontrer que F_V peut être obtenue à partir de F_U par composition de transductions linéaires et d' inverse de transductions. Intuitivement, par une première transduction, nous allons effacer les termes de U, par une seconde générer tous les termes de V une fois et une seule, puis générer un nombre quelconque d'éléments de V (ceci est illustré dans la figure 4.5.b).

Figure 4.5.b

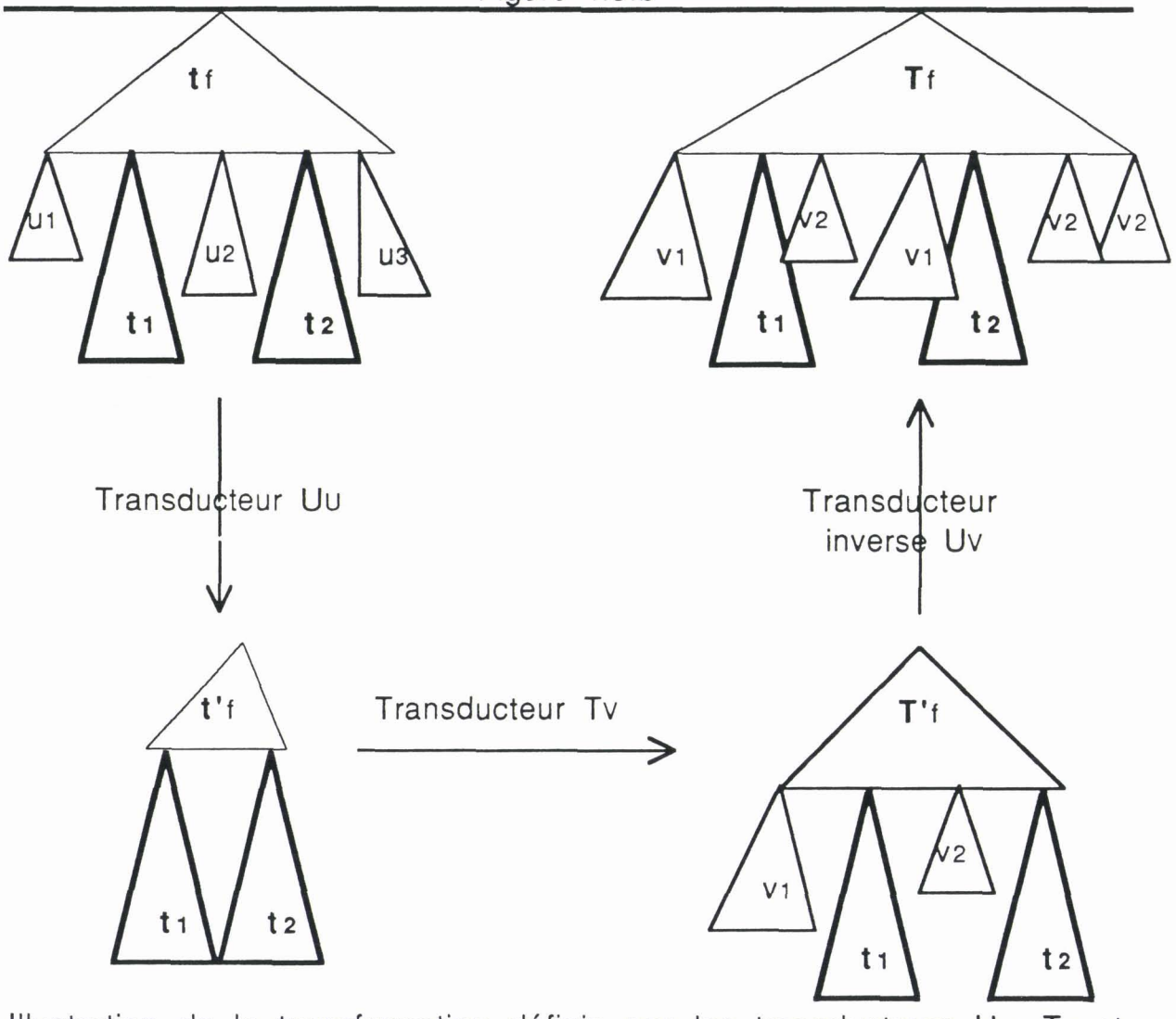


Illustration de la transformation définie par les transducteurs U_U, T_V et de l'inverse de U_U avec $U=\{u_1,u_2,u_3\}$, $V=\{v_1,v_2\}$, $N'(t)=N'(t')=(t_1,t_2)$.

Soit u un élément de M.

Soit U_u le transducteur linéaire ascendant défini par:

est une lettre d'arité 0 n'appartenant pas à Σ (le terme t est effacé).

Les règles de l'automate ascendant déterministe Sub tel que $u \xrightarrow{*}_M q_u$ et pour tout t de T_Σ, distinct de u, $t \xrightarrow{*}_M q$ avec $q \neq q_u$.

$f(q_u[x_1],q[x_2]) \rightarrow q_f[x_2]$ pour tout q distinct de q_u et $q_\#$

$f(q[x_1],q_u[x_2]) \rightarrow q_f[x_1]$ pour tout q distinct de q_u et $q_\#$

$f(q[x_1],q'[x_2]) \rightarrow q_f[\sigma(x_1,x_2)]$ pour q et q' distincts de q_u et $q_\#$

$f(q_u[x_1],q_u[x_2]) \rightarrow q_u[x_1] \mid q_\#[\#]$

Les états finaux sont q_f et $q_\#$.

Soit T_u le transducteur linéaire complet descendant défini de la façon suivante:
est une lettre d'arité 0 n'appartenant pas à Σ.
$q_u[f(x_1,x_2)] \rightarrow f(q_u[x_1],q_u[x_2]) \mid f(q[x_1],q_u[x_2])$.
$q_u[\#] \rightarrow u$.
$q_u[l(x_1,\ldots,x_n)] \rightarrow \sigma(u,l(q[x_1],\ldots,q[x_n])) \mid \sigma(l(q[x_1],\ldots,q[x_n]),u), \forall l \in \Sigma$.
$q[f(x_1,x_2)] \rightarrow f(q[x_1],q[x_2])$.
$q[l(x_1,\ldots,x_n)] \rightarrow l(q[x_1],\ldots,q[x_n]), \forall l \in \Sigma$.
L'état initial est q_u.

Soit $W=\{u_1,\ldots,u_p\}$. Soit $U_W=U_{u1}\circ\ldots\circ U_{up}$, La classe des transducteurs ascendants linéaires est close par composition donc U_W est un tranducteur ascendant linéaire. Soit $T_W=T_{u1}\circ\ldots\circ T_{up}$, La classe des transducteurs descendants linéaires complets est close par composition donc T_W est un transducteur descendant linéaire.

On a alors $F_V=U_V^{-1}(T_V(U_U(F)))$. La preuve se fait par double inclusion. Les transducteurs (ascendants et descendants) linéaires préservent la reconnaissabilité, la classe des forêts reconnaissables est close par inverse de transducteur, donc F_V est une forêt reconnaissable.❑

Proposition 4 : *La reconnaissabilité est préservée par S dans* $T_{f\Sigma}$.

Preuve proposition 4 : D'après le résultat du lemme 1, on a, pour toute forêt reconnaissable F incluse dans $T_{f\Sigma}$, $S(F)=R_\Sigma(FACI(R_\Sigma(F)))$. R_Σ est un système de réécriture clos donc la reconnaissabilité est préservée, de même FACI préserve la reconnaissabilité, d'après le lemme 3, d'où le résultat.❑

4.6 Systèmes de réécriture monadiques

Définition: Un *système de réécriture monadique de mots S* est tel que le membre droit de toute règle de S est réduit à une lettre ou égal au mot vide.

Les systèmes de réécriture monadiques de mots ont été largement étudiés (dans le cadre des systèmes de Thue, se reporter à [BOOK87] pour un article de synthèse). Nous mettons ici simplement en évidence le résultat qui nous intéresse, dans le cadre de cette étude.

Proposition 1: *([BENO69], [BERS79], [BJW81]) Pour tout système de réécriture monadique (de mots) et tout langage régulier L, l'ensemble S*(L) des réductions des mots de L par S est régulier.*

R.V.Book et J.H.Gallier ont défini une notion de système de réécriture monadique d'arbre de la façon suivante: tout membre gauche de règle est de hauteur supérieure ou égale à 1 et tout membre droit de hauteur inférieure ou égale à 1. Nous préférons adopter la définition plus restrictive suivante:

Définition: Un *système de réécriture monadique* est un système de réécriture S tel que pour toute règle $l \rightarrow r$ de S, la hauteur h(l) du membre gauche est supérieure ou égale à 1 et le membre droit r est une constante, ou une variable, ou de la forme $b(y_1,...,y_n)$ avec, pour tout i, y_i est une variable.

Nous définissons également la notion de système de réécriture semi-monadique par:

Définition: Un *système de réécriture semi-monadique* est un système de réécriture S tel que pour toute règle $l \rightarrow r$ de S, la hauteur h(l) du membre gauche est supérieure ou égale à 1 et le membre droit r est une constante, ou une variable, ou de la forme $b(y_1,...,y_n)$ avec, pour tout i, y_i est une variable ou un terme clos.

Les systèmes de réécriture monadiques (au sens de Book et Gallier) sont alors un cas particulier de systèmes semi-monadiques.

Dans [BOGA85], les auteurs prouvent que l'on peut calculer les formes normales d'un système de réécriture monadique linéaire et confluent à l'aide d'un automate à pile d'arbres. Nous verrons d'ailleurs une généralisation de ce résultat à une classe plus large de systèmes de réécriture dans la partie 5.

Un résultat identique à celui de la proposition 1 a été montré pour les systèmes de réécriture monadiques d'arbres par K.Salomaa.

Proposition 2: *([SALO88]) Les systèmes de réécriture monadiques linéaires à droite préservent la reconnaissabilité.*

Nous déduisons de ce résultat que *les différents problèmes d'accessibilité pour les systèmes de réécriture monadiques linéaires à droite sont décidables.*

Notons également que ce résultat est étendu aux systèmes de réécriture semi-monadiques linéaires à droite dans la partie 5.

Nous allons, dans cette section, étudier la décidabilité de la terminaison et de la confluence des systèmes de réécriture monadiques.

Nous prouvons que la terminaison et la confluence sont décidables pour les systèmes de réécriture monadiques linéaires à droite. Ces résultats ont été prouvés indépendamment par K.Salomaa qui a de plus prouvé que la terminaison des systèmes de réécriture monadiques non linéaires à droite est indécidable. Le problème de la confluence reste posé dans le cas non linéaire droit.

Proposition 3: *La terminaison des systèmes de réécriture monadiques linéaires à droite est décidable.*

Preuve:

Soit S un système de réécriture monadique linéaire à droite sur $T_\Sigma(X)$.

Pour toute règle $l \to r$ de S telle que $h(l)>1$ ou $h(l)=1$ et $h(r)=0$, la hauteur de l'arbre diminue strictement lorsqu'on réduit en utilisant une telle règle.

Soit $l \to r$ une règle de S telle que $h(l)=h(r)=1$. La règle est donc de la forme $l=a(x_1,\ldots,x_n) \to b(x_{i_1},\ldots,x_{i_p})=r$ avec $a \in \Sigma_n$, $b \in \Sigma_p$ et $V(r) \subset V(l)$. La règle considérée est linéaire à droite (hypothèse) donc la taille de l'arbre diminue strictement si $V(r) \subset V(l)$ et $V(r) \neq V(l)$ ou si $V(r)=V(l)$ et l est non linéaire lorsqu'on réduit par S en utilisant une telle règle.

Soit T l'ensemble des règles du système de réécriture S ne vérifiant aucune des conditions définies précédemment. S est monadique et linéaire à droite donc T est l'ensemble des règles de la forme $l=a(x_1,\ldots,x_n) \to b(x_{\sigma(1)},\ldots,x_{\sigma(n)})=r$ avec $a \in \Sigma_n$, $b \in \Sigma_n$ et σ est une permutation de [1,n].

Fait: (S ne termine pas) $\Leftrightarrow$ ($\exists (l \to r) \in T$, $r \xrightarrow{*}_T l$).

Preuve du fait:

$\Leftarrow$ est évident.

preuve de $\Rightarrow$: Supposons donc que le système de réécriture S ne termine pas. Il existe donc une réduction infinie $t_0 \to_S t_1 \to_S t_2 \to_S \ldots$ à partir d'un terme t_0 de $T_\Sigma(X)$.

D'après l'étude préliminaire, il existe donc un entier k tel que, à partir de cet entier k, toutes les règles utilisées appartiennent à l'ensemble T. Soit $t=t_k \to_T t_{k+1} \to_T \ldots$, pour tout i supérieur ou égal à k, la taille de t_i est égale à la taille de t.

Supposons que, pour toute règle $l \to r$ de t, r ne peut être réécrit en l par T. On montre, facilement, par induction sur la structure des termes, que, pour tout terme t, il ne peut exister de calcul infini par T.❑ *fin preuve fait.*

La propriété ($\exists (l \to r) \in T$, $r \xrightarrow{*}_T l$) est décidable donc la terminaison est décidable.

❑*fin preuve proposition 3.*

Proposition 4: *La confluence des systèmes de réécriture monadiques linéaires à droite est décidable.*

Preuve: Nous allons pour démontrer ce résultat utiliser les résultats sur la confluence pour les systèmes de réécriture modulo des ensembles d'équations. Les résultats utilisés peuvent être trouvés dans [JOKI86] et [PEST81]. Ils sont également rappelés dans la Section 2.3.2.

Soit S un système de réécriture monadique linéaire à droite sur $T_\Sigma(X)$. Soit R et SE les systèmes de réécriture définis par:
$SE=\{ l\to r / (l\to r)\in T \text{ et } r \overset{*}{\to}_T l\}$ et R=S-SE. Soit E l'ensemble d'équations défini de la façon suivante: $E=\{ l=r / (l\to r)\in SE \}$.

Lemme 1: (i): $\overset{*}{\leftrightarrow}_S = \overset{*}{\leftrightarrow}_{R\cup E}$.
(ii): $\overset{*}{\to}_S = \overset{*}{\to}_{R/E} \cup \overset{*}{\leftrightarrow}_E$.
(iii): $\overset{*}{\leftrightarrow}_E = \overset{*}{\to}_{SE}$.

Preuve lemme 1: Les trois propriétés (i), (ii) et (iii) se vérifient de façon immédiate en utilisant les définitions de R, SE, E et de la relation $\to_{R/E}$.❑

Lemme 2: Le système de réécriture R termine modulo E.

Preuve lemme 2: La preuve se déduit facilement de la preuve de la décidabilité de la terminaison d'un système de réécriture monadique linéaire à droite, de la définition de SE et E et du lemme 1. En effet, SE contient toutes les règles de S pour lesquelles on pouvait avoir un calcul infini.❑

Lemme 3: S est confluent si et seulement si R,E est Church-Rosser modulo E.

Preuvelemme3: On montre que S est confluent si et seulement si R est Church-Rosser modulo E de façon immédiate en utilisant les définitions et le lemme 1. Il nous faut donc montrer que R est Church-Rosser modulo E si et seulement si R,E est Church-Rosser modulo E.

Rappelons les définitions:
(R est Church-Rosser modulo E) $\Leftrightarrow$ ($\overset{*}{\leftrightarrow}_{R\cup E} \subset \overset{*}{\to}_{R/E} \circ \overset{*}{\leftrightarrow}_E \circ \overset{*}{\leftarrow}_{R/E}$).
(R,E est Church-Rosser modulo E) $\Leftrightarrow$ ($\overset{*}{\leftrightarrow}_{R\cup E} \subset \overset{*}{\to}_{R,E} \circ \overset{*}{\leftrightarrow}_E \circ \overset{*}{\leftarrow}_{R,E}$).

Par définition de la relation $\to_{R,E}$ (incluse dans $\to_{R/E}$), on a toujours l'implication: R,E Church-Rosser modulo E implique R Church-Rosser modulo E. Il nous reste donc à prouver que, dans le cas particulier qui nous intéresse, l'implication inverse est vraie.

Pour celà , nous allons démontrer le fait suivant:

Fait: $\overset{*}{\rightarrow}_{R/E} = \overset{*}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E$.

Preuvefait:

L'inclusion de $\overset{*}{\rightarrow}_{R,E} \circ \overset{*}{\leftrightarrow}_E$ dans $\overset{*}{\rightarrow}_{R/E}$ est évidente.

Démontrons donc l'inclusion inverse. Soit t et t' tels que $t \rightarrow_{R/E} t'$, nous avons donc $t \overset{*}{\leftrightarrow}_E t_1 \rightarrow_R t_2 \overset{*}{\leftrightarrow}_E t'$. On a donc $t_1 = c_1(\sigma(l))$, $t_2 = c_1(\sigma(r))$ pour une règle $l \rightarrow r$ de R, une occurence p de t_1 et c_1 contexte associé à p dans t_1.

Les termes t et t_1 sont équivalents modulo E, les règles de E sont des règles de permutation, on a donc $O(t_1)=O(t)$, on peut donc montrer qu'il existe un contexte c tel que $t=c(u)$ avec $c \overset{*}{\leftrightarrow}_E c_1$ et $u \overset{*}{\leftrightarrow}_E \sigma(l)$. On peut donc en déduire que $t=c(u) \rightarrow_{R,E} c(\sigma(r))$ et donc nous obtenons la décomposition: $t=c(u) \rightarrow_{R,E} c(\sigma(r)) \overset{*}{\leftrightarrow}_E c_1(\sigma(r))=t_2 \overset{*}{\leftrightarrow}_E t'$.

On en déduit par récurrence le résultat annoncé.❑ *finpreuvefait.*

Remarque: R est E-compatible au sens de [PEST81].

Pour terminer la preuve de la proposition 4, il nous reste à remarquer qu'il existe un algorithme fini et complet d'unification pour la théorie E, E étant un ensemble d'équations de la forme $a(x_1,\ldots,x_n)=b(x_{\sigma(1)},\ldots,x_{\sigma(n)})$ ([HUET80], [PEST81]) et d'utiliser le théorème 2 ([JOKI86]) rappelé dans la section 2.3.2., le nombre de paires critiques étant fini.

❑ *Finpreuveproposition4.*

Signalons enfin que la confluence close des systèmes de réécriture monadiques a été étudiée par D. Kapur, P. Narendran et F. Otto dans [KNO90]. Les auteurs prouvent que la confluence close est indécidable dans les cas non linéaire droit et non linéaire gauche. Le problème n'est pas résolu dans le cas linéaire (droit et gauche).

4.7 Résultats de cette section

Nous présentons ici un résumé des différents résultats obtenus. Présentons, tout d'abord, les différents résultats pour les problèmes d'accessibilité et la préservation de la reconnaissabilité.

Système	REC préservé	Accessibilité 1 ordre	Accessibilité 2 ordre
clos	oui	oui	oui
clos+C	oui	oui	oui
clos+A	non	non	non
clos+AC	non	partiellement	?
clos+ACI	non	partiellement	?

La terminaison des systèmes clos modulo la commutativité est décidable, la confluence est décidable, sous cette hypothèse de terminaison.

La terminaison des systèmes clos modulo l'associativité est indécidable.

La terminaison et la confluence des systèmes de réécriture monadiques linéaires à droite sont décidables.

5 AUTOMATES A PILES ET CALCUL DE FORMES NORMALES

5.1 Automates à piles d'arbres

Les automates à piles d'arbres sont une généralisation, au cas des arbres, des automates à piles de mots. Une classe d'automates à piles descendants a été définie et étudiée par I. Guessarian dans [GUES83] et ascendants par K.M.Schimpf et J.H. Gallier dans [GASC85] et [SCHI82].

Dans les deux cas, les auteurs ont défini des classes d'automates à piles d'arbres telles que la classe des langages reconnus soit exactement la classe des langages algébriques d'arbres. Nous ne nous intéressons ici qu'aux automates à piles ascendants d'arbres.

Nous comparons dans cette partie les différentes définitions possibles pour les automates ascendants à piles d'arbres.

Une définition d'automate à piles d'arbres doit correspondre, dans le cas des arbres filiformes, à la définition des automates à piles dans les mots. Lorsqu'on veut généraliser la définition utilisée dans les mots au cas des arbres, deux problèmes se posent: le premier concerne la hauteur des termes dépilés, le second concerne l'arité des états.

Une règle de transition, dans le cas des mots, est de la forme: $(q,a,\alpha)\rightarrow(q',\beta)$ où q, q' sont des états, a est une lettre de l'alphabet d'entrée ou le mot vide, α est une lettre de l'alphabet de pile et β un mot sur l'alphabet de pile. Dans le mouvement associé à cette transition, a est lu, le sommet de pile α est dépilé et on empile le mot β. Il est facile de voir que l'on peut généraliser cette définition en prenant pour α un mot de l'alphabet de pile (c'est-à-dire encore que l'on peut lire dans la pile à une profondeur bornée) sans modifier la puissance de l'automate à pile ainsi défini. L'idée de la construction est de dépiler lettre par lettre à l'aide d' ε-règles en mémorisant, dans les états, le mot dépilé.

La situation est toute différente dans le cas des arbres comme l'illustre l'exemple suivant:

Soit l'ε-règle $q(b(b(x,y),z))\rightarrow q'(c(x,y,z)$ associée à un automate à piles d'arbres où q, q' sont des états, b, c des lettres de l'alphabet (gradué) de pile et x, y et z des variables. Si on impose que les états soient d'arité 1, cette règle ne pourra être simulée par des règles d'un automate à piles pour lequel on ne peut dépiler que le sommet de pile (c'est à dire à la profondeur 1). En effet, lorsque l'on dépile le premier b, les états ayant pour arité 1, on perd l'information sur l'un de ses deux sous-arbres. L'idée est donc d'introduire des états avec arité. En effet, reprenons l'exemple précédent et introduisons un nouvel état q" d'arité 2, la règle pourra alors être simulée à l'aide des deux règles suivantes: $q(b(x,y))\rightarrow q"(x,y)$ et $q"(b(x,y),z)\rightarrow q'(c(x,y,z))$.

Nous donnons donc ici les différentes définitions possibles en fonction de la profondeur des termes dépilés et de l'arité des états.

Nous montrons l'équivalence des différentes définitions: automates à piles d'arbres avec arité quelconque pour les états et profondeur quelconque pour les termes dépilés, avec arité 1 et profondeur quelconque, arité quelconque et profondeur 1 pour les termes dépilés.

Une autre option est d'utiliser la définition de R.V. Book et J.H. Gallier, J.H. Gallier et K.M. Schimpf, K.Salomaa (voir [BOGA85], [GASC85], [SALO88]).

Cette définition utilise deux types de règles: des règles de lecture (read-rules) qui permettent de lire une lettre de l'alphabet d'entrée en empilant une lettre de l'alphabet de pile au sommet de l'arbre et des règles de réécriture de la pile (reduce-rules) plutôt que des règles qui lisent et réduisent en même temps.

Nous prouvons que cette définition est équivalente aux précédentes comme l'illustre l'exemple suivant:

la règle $a(q_1(u_1),\ldots,q_n(u_n))\rightarrow q(u)$ avec a d'arité n, pour tout i, u_i terme de $T_\Gamma(X)$ où Γ est l'alphabet de pile sera simulée par la règle de lecture (read-rule) $a(q_1(x_1),\ldots,q_n(x_n))\rightarrow q'(a'(x_1,\ldots,x_n))$ avec q' nouvel état et a' nouvelle lettre de Γ et la règle de réduction (reduce-rule) $q'(a'(u_1,\ldots,u_n))\rightarrow q(u)$.

Nous définissons également la notion de déterminisme associé à un automate à piles. Un automate à piles est déterministe si l'ensemble des règles de réécriture associé est linéaire à gauche et sans paires critiques. En effet, le système de réécriture associé est alors confluent ([HUET80]), ceci n'étant vrai que dans le cas linéaire à gauche. Nous démontrons que les différentes définitions sont encore équivalentes dans le cas déterministe.

5.1.1 Définitions des automates à piles

Définition 1: Un *automate à piles ascendant d'arbres* (tpda pour tree pushdown automaton) est un quintuplet $T=(\Sigma, \Gamma, Q, Q_f, R)$ où:

Σ est un alphabet gradué fini d'entrée.

Γ est un alphabet gradué fini de pile.

Q est un ensemble gradué fini d'états.

Q_f, inclus dans Q_1 (états d'arité 1), ensemble d'états finaux.

R est un système de réécriture fini sur $T_{\Sigma\cup\Gamma\cup Q}$ dont les règles ont l'une des deux configurations suivantes:

règles standard:

$u= \alpha(q_1(u_{11},...,u_{1n_1}),...,q_m(u_{m1},...,u_{mn_m})) \rightarrow q(u_1,...,u_n)$ avec $\alpha\in\Sigma_m$, $q_i\in Q$, n_i est l'arité de q_i, n est l'arité de q, $u_{ij},u_k\in T_\Gamma(X)$ pour $j=1,...,n_i$, $i=1,...,m$, $k=1,...,n$, les ensembles de variables $V(u_{ij})$ sont disjoints.

ε-règles:

$q(u_1,...,u_n) \rightarrow q'(u'_1,...,u'_{n'})$ avec q, q' dans Q d'arité n et n', $u_i,u'_j\in T_\Gamma(X)$ pour $i=1,...,n$, $j=1,...,n'$.

Soit T un automate à piles d'arbres, on définit:

Une *configuration* est un terme c de $T_{\Sigma\cup\Gamma\cup Q}$ tel que chemin(c) est inclus dans $\Sigma^*Q\Gamma^+\cup\Sigma^*$.

La *relation* $\rightarrow_T$ est la relation de réécriture $\rightarrow_R$ restreinte à l'ensemble des configurations de T. Un mouvement élémentaire de l'automate à pile T est donc défini par $c\rightarrow_T c'$ si c, c' sont des configurations de T et $c\rightarrow_R c'$.

La *relation* $\overset{*}{\rightarrow}_T$ est la clôture réflexive et transitive de $\rightarrow_T$.

Une *configuration initiale* est un terme c=t de T_Σ.

Soit $c=t_0(q_1(u_{11},...,u_{1n_1}),...,q_m(u_{m1},...,u_{mn_m}))$ une configuration de T obtenue à partir de la configuration initiale $t=t_0(t_1,...,t_m)$, alors $t_1,...,t_m$ ont été lus, la $i^{ème}$ tête de lecture est dans l'état q_i d'arité n_i avec pour arbres de pile associés $u_{i1},...,u_{in_i}$.

Une *configuration finale* est un terme c=q(v) où q est un état final et v un arbre de pile dans T_Γ.

La *tranformation associée à T* est l'ensemble $\tau(T)$ défini par:
$\tau(T)=\{t,t')\in T_\Sigma x T_\Gamma / \ t \overset{*}{\rightarrow}_T q(t'),\ q\in Q_f\ \}$.

T et T' sont *équivalents* (noté T≡T') si et seulement si $\tau(T) = \tau(T')$.

L'arité maximale des états de Q est appelée le rang de T et notée rang(T). La hauteur maximale des arbres de piles (appartenant à $T_\Gamma(X)$) apparaissant dans les membres gauches de règles de R est appelée la profondeur de T et notée prof(T).

Nous définissons et notons $TPDA^{**}$, $TPDA_{n^*}$, $TPDA^{*}_{k}$, $TPDA_{nk}$ les classes d'automates à piles ascendants d'arbres vérifiant respectivement: rang et profondeur quelconques, rang n et profondeur quelconque, rang quelconque et profondeur k, rang n et profondeur k.

Exemple: Nous illustrons dans la figure 5.1 les définitions précédentes dans le cas d'un automate à piles d'arbres appartenant à la classe $TPDA_{1^*}$ définie ci dessus.

Figure 5.1.

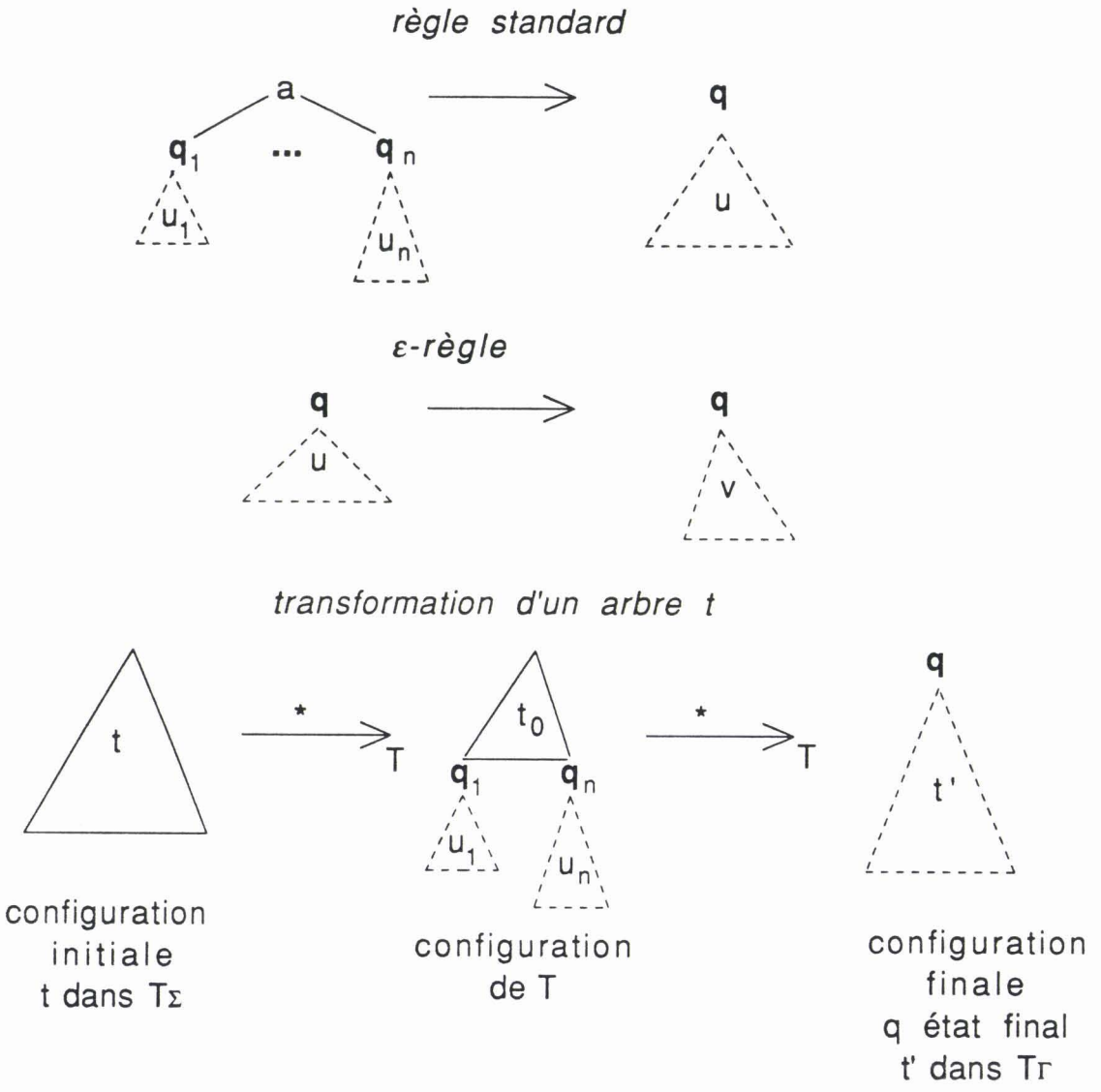


Illustration des définitions pour un automate de $TPDA_{1^*}$

Proposition 1: *Pour tout automate à piles T dans $TPDA_{**}$, il existe un automate T' dans $TPDA_{1^*}$ tel que $T \equiv T'$.*

Preuve: Soit $T=(\Sigma,\Gamma,Q,Q_f,R)$ un automate à piles ascendant d'arbres appartenant à la classe $TPDA_{**}$. Nous construisons un automate à piles T' équivalent appartenant à la classe $TPDA_{1^*}$ comme suit:

A tout état q d'arité n supérieure ou égale à 2, on associe un nouvel état qn d'arité 1. On obtient ainsi un ensemble d'états Q'_1. Nous posons $Q'=Q_0 \cup Q_1 \cup Q'_1$.

On ajoute à l'alphabet de pile Γ de nouvelles lettres d'arité k, pour tout k supérieur ou égal à 2 et inférieur ou égal au rang de T. Formellement, $\Gamma'=\Gamma \cup \{<\#,k> / 2 \leq k \leq rang(T)\}$.

Pour tout terme $q(u_1,...,u_n)$ avec n supérieur ou égal à 2 apparaissant dans un membre gauche ou droit de règle de R, on lui substitue $qn(<\#,n>(u_1,...,u_n))$. On obtient ainsi un nouvel ensemble de règles de réécriture R'.

Soit $T'=(\Sigma,\Gamma',Q',Q_f,R')$. Par construction, T' est un automate qui appartient à la classe $TPDA_{1^*}$ (tous les états sont maintenant d'arité maximale 1). La preuve de l'équivalence des automates T et T', c'est-à-dire de l'égalité $\tau(T)=\tau(T')$, est sans difficultés:

A toute configuration $c=t_0(q_1(u_{11},...,u_{1n_1}),...,q_m(u_{m1},...,u_{mn_m}))$, on associe la configuration c' de T' obtenue en substituant à tout terme $q_i(u_{i1},...,u_{in_i})$ le terme $q_in_i(<\#,n_i>(u_{i1},...,u_{in_i}))$ si n_i est supérieur ou égal à 2.

On démontre alors que, pour toutes configurations c_1 et c_2 de T et leurs configurations associées c'_1 et c'_2 de T', on a l'équivalence suivante: $c_1 \rightarrow_T c_2$ si et seulement si $c'_1 \rightarrow_{T'} c'_2$.

De plus, à toute configuration initiale t de T, on associe la configuration initiale t de T' et les configurations finales de T et T' sont les mêmes car $Q_f=Q'_f$ et, dans la construction, les états d'arité 1 ont été laissés inchangés.❑

On peut remarquer, dans la construction utilisée dans la preuve de la proposition 1, que si T appartient à la classe $TPDA_{nk}$, alors T' appartient à la classe $TPDA_{1k+1}$.

***Proposition** 2: Pour tout automate à piles T dans $TPDA_{*1}$, il existe un automate T' dans $TPDA_{12}$ tel que $T \equiv T'$.*

Preuve: Ce résultat est une conséquence de la proposition 1.

Proposition 3: *Pour tout automate à piles T dans $TPDA_{1*}$, il existe un automate T' dans $TPDA_{*1}$ tel que $T \equiv T'$.*

Preuve: Soit $T=(\Sigma,\Gamma,Q,Q_f,R)$ un automate à piles ascendant d'arbres appartenant à la classe $TPDA_{1*}$. Nous construisons un automate à piles $T'=(\Sigma,\Gamma,Q',Q'_f,R')$ équivalent de la classe $TPDA_{*1}$ comme suit:

L'ensemble des états Q' de T' contient Q. L'ensemble des états finaux Q'_f de T' est égal à Q_f.

Nous construisons de nouveaux états dans l'ensemble Q' et le système de réécriture R' de la façon suivante:

Soit un terme q(u) apparaissant dans un membre gauche de règle de R tel que la profondeur de u soit supérieure ou égale à 2. Illustrons, tout d'abord, sur un exemple, la construction faite. L'idée intuitive étant de descendre niveau par niveau, en vérifiant l'égalité du terme avec u et en mémorisant tous les sous-arbres utiles.

Exemple: Soit le terme $q(u)=q(c(x,b(y,b(a,z)),t)$ apparaissant dans un membre gauche de règle de R, on définit les nouveaux états q_1, q_2, q_3 et q_4 et les règles suivantes:

$q(c(x_1,x_2,x_3)) \rightarrow q_1(x_1,x_2,x_3)$
$q_1(x_1,b(x_2,x_3),x_4) \rightarrow q_2(x_1,x_2,x_3,x_4)$
$q_2(x_1,x_2,b(x_3,x_4),x_5) \rightarrow q_3(x_1,x_2,x_3,x_4,x_5)$
$q_3(x_1,x_2,a,x_3,x_4) \rightarrow q_4(x_1,x_2,x_3,x_4)$

Formellement, pour q(u) vérifiant les conditions précédentes, on introduit les états et les règles suivantes:

Si $u=l(u_1,\ldots,u_m)$, on ajoute à R' la règle $q(l(x_1,\ldots,x_m))\rightarrow q_1(x_1,\ldots,x_m)$.

Soit k un entier supérieur ou égal à 1 et soit $tronc(u,k)=u_k$, la troncature à la profondeur k de u dont nous rappelons la définition: Pour toute occurrence p dans $O(u_k)$, soit $|p|<k$ et les symboles occurant à l'occurrence p de u_k et u sont égaux, soit $|p|=k$ et $u_{k/p}\in X$, de plus si u_k appartient à $T_\Sigma(X_m)$, $fr(u_k)\in \Gamma^* x_1 \Gamma^* \ldots x_m \Gamma^*$.

Pour tout entier k supérieur ou égal à 1 et inférieur à la profondeur de u, considérons les termes u_k et u_{k+1} avec $u_{k+1}=u_k(l_1(x_{11},\ldots,x_{1n_1}),\ldots,l_p(x_{p1},\ldots,x_{pn_p})$ dans $T_\Sigma(X_m)$ (remarquons que l_j peut être une lettre d'arité 0). On définit alors la règle $q_k(y_1,\ldots,y_q)\rightarrow q_{k+1}(x_1,\ldots,x_m)$ telle que $y_i=l_j(x_{j1},\ldots,x_{jn_j})$ ou $y_i=x_n$ (x_n est une variable occurant dans u_k). Une borne supérieure pour l'arité des nouveaux états créés est la largeur de u.

Cette construction est faite pour tous les termes q(u) occurant dans un membre gauche avec la profondeur de u supérieure ou égale à 2. Pour deux termes distincts, on introduit de nouveaux ensembles d'états disjoints.

Pour chaque ε-règle $q(u)\rightarrow q'(u')$ de R, si la profondeur de u est inférieure ou égale à 1, on ajoute la règle $q(u)\rightarrow q'(u')$ dans R', sinon, on ajoute la règle $q_{h(u)}(x_{i_1},\ldots,x_{i_n})\rightarrow q'(u')$ dans R' , où $(x_{i_1},\ldots,x_{i_n})$ désigne le n-uplet de variables obtenu en renommant les variables de u dans l'ordre du feuillage et en respectant les égalités éventuelles. Une telle règle sera dite de type (*).

Exemple: Soit l'ε-règle q(c(x, b(y,x), a(y))) $\rightarrow$q'(u'), on définit, comme précédemment, les nouveaux états q_1 et q_2 et les règles:

$q(c(x_1,x_2,x_3)) \rightarrow q_1(x_1,x_2,x_3)$,

$q_1(x_1,b(x_2,x_3),a(x_4)) \rightarrow q_2(x_1,x_2,x_3,x_4)$.

On définit alors la règle de type (*) par:

$q_2(x_1,x_2,x_1,x_2) \rightarrow q'(u')$.

De plus, soit $\alpha(q_1(u_1),\ldots,q_m(u_m))\rightarrow q'(u')$ une règle standard de T, on substitue à chaque sous-terme $q_i(u_i)$ tel que la profondeur $h(u_i)$ de u_i est supérieure ou égale à 2 le sous-terme $q_{h(ui)}(x_{i_1},\ldots,x_{i_{ni}})$ défini comme dans le cas précédent. La règle ainsi définie est ajoutée à R'. Une telle règle sera dite de type (**).

La construction de T' étant faite, il nous reste à prouver que les automates obtenus sont équivalents, soit encore l'égalité $\tau(T)=\tau(T')$.

Si c et c' sont deux configurations de T telles que $c \rightarrow_T c'$, c et c' sont deux configurations de T' et on montre, sans difficultés, que $c \overset{*}{\rightarrow}_{T'} c'$, par construction de R'. On peut donc en déduire, par récurrence, que $\tau(T)$ est inclus dans $\tau(T')$.

Réciproquement, montrons que $\tau(T')$ est inclus dans $\tau(T)$. Considérons la réduction par T' d'un terme t de T_Σ (configuration initiale) en un terme q(t') avec q état final de T' et t' terme de $T_{\Gamma'}$ (configuration finale). t est, de façon évidente, une configuration initiale de T, q(t') est une configuration finale de T, en effet, les alphabets de piles de T et T' sont les mêmes, et, par construction de R', les membres droits de règles de R' de type (*) ou (**) ne contiennent que des états de Q, donc lors de l'application de la dernière règle, q est bien un état de T.

Nous allons, pour conclure, prouver le lemme technique suivant:

Lemme: Soit c et c' deux configurations de T, si $c \overset{*}{\rightarrow}_{T'} c'$ en utilisant n règles de type (*) ou (**) alors $c \overset{*}{\rightarrow}_T c'$ en utilisant n règles de R.

Preuvelemme: La preuve se fait par récurrence sur n.

Pour n=1, comme l'ensemble des états créés est disjoint de l'ensemble Q et comme c' est une configuration de T, on ne peut que simuler une règle de R et donc, $c \rightarrow_T c'$.

Soit $n \geq 2$ et supposons la propriété vraie jusqu'à n-1. Considérons deux configurations c et c' de T telles que $c \overset{*}{\rightarrow}_{T'} c'$ avec n utilisations de règles de type (*) ou (**). Considérons la première application d'une règle r de type (*) ou (**), nous avons donc: $c \overset{*}{\rightarrow}_{T'} c_1 \overset{r}{\rightarrow}_{T'} c_2 \overset{*}{\rightarrow}_{T'} c'$.

Cas 1: Si c_2 est une configuration de T, alors $c \rightarrow_T c_2$ (cas n=1) et il suffit d'appliquer l'hypothèse de récurrence à la réduction de c_2 en c'.

Cas 2: Si c_2 n'est pas une configuration de T, les applications des règles de T' permettant d'appliquer la règle r sont indépendantes des applications des autres règles, ceci par construction de T', en effet, les ensembles d'états créés pour deux termes distincts sont disjoints. Donc, il existe une configuration c'_3 de T' et une configuration c_3 de T telles que: $c \overset{*}{\rightarrow}_{T'} c'_3 \overset{r}{\rightarrow}_{T'} c_3 \overset{*}{\rightarrow}_{T'} c_2$. Il suffit alors d'utiliser le même raisonnement que dans le cas 1.*Finpreuvelemme.*❑

Finpreuveproposition❑

Nous avons donc prouvé l'équivalence entre les classes TPDA**, TPDA$_1{}^*$, TPDA$^*{}_1$ et TPDA$_{12}$. La classe la plus utilisée est la classe TPDA$_1{}^*$. Nous allons maintenant rappeler la définition des automates à piles ascendants d'arbres de R.V.Book, J.H.Gallier et K.M.Schimpf. Dans cette définition, les règles de lecture ("read-rules") n'autorisent pas de modification des arbres de pile.

Définition 2: *Un automate à piles ascendant d'arbres read-reduce* (noté tpda^{rr}) est un quintuplet $A=(\Sigma,\Gamma,Q,Q_f,R)$ où:

Σ est un alphabet gradué fini d'entrée.

Γ est un alphabet gradué fini de pile.

Q est un ensemble fini d'états d'arité 1.

Q_f, inclus dans Q, ensemble d'états finaux.

R est un système de réécriture fini sur $T_{\Sigma\cup\Gamma\cup Q}$ dont les règles ont l'une des deux configurations suivantes:

Règles de lecture ("read rules"):

$\alpha(q_1(x_1),...,q_k(x_k)) \rightarrow q(\beta(x_1,...,x_k))$ avec $\alpha\in\Sigma_k$, $k\geq 0$, $q_i\in Q$, $x_i\in X$, $i=1,...,k$, $\beta\in\Gamma_k$.

Règles de réduction ("reduce rules", "tree stack update rules"):

$q(l) \rightarrow q'(r)$ avec $q,q'\in Q$, $l,r\in T_\Gamma(X)$.

Nous notons TPDA^{rr} la classe des automates à piles ascendants read-reduce. On peut remarquer que cette classe est une sous-classe de la classe TPDA$_1{}^*$.

Les définitions précédentes de l'ensemble des configurations, de configuration initiale, de configuration finale, de mouvement élémentaire, de transformation associée à A s'appliquent donc à tout automate A de la classe TPDA^{rr}.

Proposition 4: *Pour tout automate à piles T de la classe TPDA$_1{}^*$, il existe un automate à piles A de la classe TPDA^{rr} tel que $T\equiv A$.*

Preuve: Soit $T=(\Sigma,\Gamma,Q,Q_f,R)$ un automate de la classe $TPDA_1{}^*$. Nous construisons l'automate équivalent $A=(\Sigma,\Gamma',Q',Q'_f,R')$ de la classe $TPDA^{rr}$ comme suit:

Remarquons, tout d'abord, que Q peut contenir des états d'arité 0. On considère donc un nouveau symbole $\perp$ (correspondant au symbole de pile vide), et, à tout état q d'arité 0, on associe un nouvel état q^1 d'arité 1. On remplace alors toute occurrence d'un état q d'arité 0 par le terme $q^1(\perp)$.

Pour toute règle standard r: $\alpha(q_1(u_1),\ldots,q_m(u_m)) \rightarrow q(u)$ de T, on définit un nouvel état qr, une nouvelle lettre $\bar{\alpha}$ et les nouvelles règles suivantes: $\alpha(q_1(x_1),\ldots,q_m(x_m)) \rightarrow qr(\bar{\alpha}(x_1,\ldots,x_m))$ (règle de lecture, "read-rule") et $qr(\bar{\alpha}(u_1,\ldots,u_m)) \rightarrow q(u)$ (règle de réduction, "reduce-rule").

On définit Γ' comme l'union de Γ et des nouvelles lettres créées, Q' comme l'union de Q et de l'ensemble des nouveaux états, Q'_f est égal à Q_f et R' est la réunion de l'ensemble des ε-règles de R et de l'ensemble des nouvelles règles. A appartient bien à la classe $TPDA^{rr}$ et on vérifie facilement l'égalité entre $\tau(T)$ et $\tau(A)$. ❑

5.1.2 Automates à piles déterministes

Nous allons maintenant définir la notion de déterminisme associé à un automate à piles ascendant d'arbres.

Nous restreignons notre définition au cas d'automates à piles dont l'ensemble R de règles de réécriture est linéaire à gauche. En effet, nous dirons qu'un automate est déterministe si l'ensemble R est sans paires critiques et, dans le cas où R est linéaire à gauche, on peut en déduire que la relation $\rightarrow_R$ est confluente (voir [HUET80] pour le résultat et pour un exemple de système de réécriture sans paires critiques et non confluent).

Définition 3: Un automate à piles $T=(\Sigma,\Gamma,Q,Q_f,R)$ ascendant d'arbres est *déterministe* si le système de réécriture R est linéaire à gauche et sans paires critiques.

Nous notons $DTPDA^{**}$, $DTPDA_1{}^*$, $DTPDA^*{}_1$ et $DTPDA^{rr}$ les classes d'automates à piles ascendants déterministes d'arbres associées aux classes définies précédemment. Nous prouvons que l'équivalence des différentes classes est encore vraie pour les classes déterministes correspondantes.

Proposition1': *Pour tout automate à piles T dans $DTPDA^{**}$, il existe un automate T' dans $DTPDA_1{}^*$ tel que $T\equiv T'$.*

Preuve: Il est facile de vérifier que, dans la construction de T' faite dans la preuve de la proposition 1, si R est linéaire à gauche et sans paires critiques, R' vérifie les mêmes propriétés.❑

Proposition 2': *Pour tout automate à piles T dans $DTPDA^*{}_1$, il existe un automate T' dans $DTPDA_{12}$ tel que $T\equiv T'$.*

Preuve: Ce résultat est une conséquence de la proposition 1'.❑

Pour les preuves des deux dernières propositions, nous démontrons tout d'abord un lemme technique.

Lemme: *Soit T un automate à piles déterministe ascendant d'arbres de la classe $DTPDA_1{}^*$, il existe un automate T' de $DTPDA_1{}^*$ équivalent à T et vérifiant la propriété (P) suivante:*

(P): pour tous états q et q', pour tous arbres de piles u et v de $T_\Gamma(X)$, si q(u) apparaît dans un membre gauche d'une règle standard et q'(v) apparaît dans un membre gauche de règle (standard ou ε-), alors on a l'équivalence $(q=q')\Leftrightarrow(u=v)$.

Preuve:

Remarquons, tout d'abord, que la propriété (P) n'est pas en général vérifiée par un automate déterministe.

En effet, considérons deux règles standard dont les membres gauches respectifs sont $b(q(u),q_1(u_1))$ et $b(q_2(u_2),q(v))$. Ces deux règles ne définissent pas de paire critique pour le système de réécriture même dans le cas où u et v sont unifiables.

Un autre exemple est de considérer une règle standard et une ε-règle de membres gauches respectifs $b(q(u),q_1(u_1))$ et $q(v)$, si u et v sont distincts et non unifiables, ces deux règles n'induisent pas de paire critique.

Nous verrons dans les démonstrations des propositions 3' et 4' que nous avons besoin de cette propriété (P) pour que les constructions des automates équivalents préservent le déterminisme.

Soit un état q et soit L(q) l'ensemble des arbres de piles u de $T_\Gamma(X)$ tels que q(u) occure dans un membre gauche de règle standard ou d'ε-règle. Soit $k(q) = \max\{ h(u) / u \in L(q) \} + 1$. Soit $K = \max\{k(q) / q \in Q\}$.

Pour tout arbre K-normalisé t, on définit un nouvel état q_t et on définit l'ensemble de règles $q(t) \rightarrow q_t(t)$, pour tout q de Q.

Pour toute ε-règle $q(u) \rightarrow q'(u')$, pour tout arbre K-normalisé t tel qu'il existe une substitution σ vérifiant $\sigma(u)=t$, on définit la règle $q_t(t) \rightarrow q'(t')$, avec $t'=\sigma(u')$.

Pour toute règle standard $\alpha(q_1(u_1),\ldots,q_m(u_m)) \rightarrow q(u)$ de R, pour tous arbres K-normalisés $t_1,\ldots,t_m$ tels qu'il existe une substitution σ vérifiant $\sigma(u_i) = t_i$, pour tout i, on définit la règle $\alpha(q_{t1}(t_1),\ldots,q_{tm}(t_m)) \rightarrow q(t)$, avec $t=\sigma(u)$.

Q' est la réunion de Q et de l'ensemble des états q_t associés aux arbres de $T_\Gamma(X,K)$. Q'_f est égal à Q_f. R' est la réunion des ensembles de règles ainsi construites. On vérifie aisément que l'automate T' appartient à la classe $TPDA_1$* et que le déterminisme de T est préservé dans la construction de T'. De plus, T' satisfait la propriété (P), en effet, considérant la propriété (P), le seul cas à considérer est $q(u)=q_t(t)$ et $q'(v)=q_{t'}(t')$ et l'équivalence demandée est bien vérifiée. En effet, $q_t=q_{t'}$ si et seulement si t=t'. L'équivalence des automates T et T' se vérifie par double inclusion.❑

Proposition 3': *Pour tout automate à piles T dans $DTPDA_{1*}$, il existe un automate T' dans $DTPDA_{*1}$ tel que $T \equiv T'$.*

Preuve: Soit $T=(\Sigma,\Gamma,Q,Q_f,R)$ un automate à piles ascendant d'arbres appartenant à la classe $DTPDA_{1*}$. Nous pouvons supposer, d'après le lemme 2, que T satisfait la propriété (P).

Nous construisons un automate à piles $T'=(\Sigma,\Gamma,Q',Q'_f,R')$ équivalent de la classe $DTPDA_{*1}$ par la même construction que celle utilisée dans la preuve de la proposition 3.

On peut vérifier que cette construction n'induit pas de nouvelles paires critiques à l'exception du cas des règles de la forme $q(l(x_1,\ldots,x_m))\rightarrow q_1(x_1,\ldots,x_m)$ avec $u=l(u_1,\ldots,u_m)$, q(u) occure dans un membre gauche de règle de R et la profondeur h(u) de u est supérieure ou égale à 2.

Soit q un état et soit L(q) l'ensemble des arbres de pile u tel que q(u) apparaît en membre gauche d'une règle de R. En utilisant la propriété (P), on ne peut avoir deux termes q(u) et q(v), avec u et v distincts, que dans le cas où ils apparaissent tous les deux en membre gauche d'ε-règles de R. Mais, dans ce cas, l'automate T étant déterministe, deux termes distincts de L(q) ne sont pas unifiables (on aurait en effet une paire critique entre deux ε-règles de T).

On simule les règles de la même façon que dans la preuve de la proposition 3, mais, on introduit de plus un ordre total sur les éléments de L(q). D'après les remarques précédentes, une et une seule de ces simulations peut mener à un succès. On ajoute donc de nouvelles règles de façon à ce que les simulations se fassent en respectant l'ordre total imposé à l'ensemble L(q) et, de plus, que lorsqu'une simulation échoue, on réinitialise l'arbre en cours de traitement pour essayer de reconnaître l'arbre suivant en respectant l'ordre sur L(q). Nous ne détaillons pas ici les règles correspondant à cette partie de la construction.

Par cette construction, nous obtenons un automate déterministe de la classe $DTPDA_{*1}$. La preuve de l'équivalence entre T et T' est identique à la preuve de la proposition 3.❑

Proposition 4': *Pour tout automate à piles T de la classe $DTPDA_{1^*}$, il existe un automate à piles A de la classe $DTPDA^{rr}$ tel que $T \equiv A$.*

Preuve: Soit $T=(\Sigma,\Gamma,Q,Q_f,R)$ un automate de la classe $DTPDA_{1^*}$ possédant la propriété (P) (lemme). Nous construisons l'automate $A=(\Sigma,\Gamma',Q',Q'_f,R')$ équivalent à T de la classe $TPDA^{rr}$ comme dans la proposition 4. Montrons que A est déterministe.

Il est clair que R' est linéaire à gauche car R est linéaire à gauche. Montrons donc que R' est sans paire critique.

Cas 1: Il est évident que la construction de A n'induit pas de nouvelles paires critiques entre deux règles de réduction ("reduce-rules").

Cas 2: Considérons maintenant le cas d'un règle de réduction ("reduce-rule") et d'une règle de lecture ("read-rule").

Cas 2.1: Soit $\alpha(q_1(x_1),\ldots,q_m(x_m)) \rightarrow qr(\tilde{\alpha}(x_1,\ldots,x_m))$ une règle de lecture de A (déduite de la règle standard $\alpha(q_1(u_1),\ldots,q_m(u_m)) \rightarrow q'(u)$ de T) et $q(v)\rightarrow q''(v')$ une règle de réduction de A. T est déterministe donc, pour tout i, $q(v)$ et $q_i(u_i)$ ne sont pas unifiables, et donc, pour tout i, on a soit $q\neq q_i$, soit $q=q_i$ avec v et u_i non unifiables. En utilisant la propriété (P), le deuxième cas ne peut se produire donc, pour tout i, q est différent de q_i, donc on n'a pas de paires critiques entre deux règles de ce type.

Cas 2.2: Si on considère maintenant une règle de lecture et une règle de réduction de la forme $qr(\tilde{\alpha}(u_1,\ldots,u_m))\rightarrow q(u)$. Ces deux règles ne peuvent induire de paire critique car la nouvelle lettre $\tilde{\alpha}$ ne peut, par construction de A, apparaître en membre gauche d'une règle de lecture.

Cas 3: Le cas de deux règles de lecture se résoud exactement comme le cas 2.1 en utilisant le déterminisme de T et la propriété (P).❑

La définition du déterminisme pour le cas des automates de la classe $TPDA^{rr}$ correspond exactement à la définition du déterminisme donné par R.V.Book et J.H.Gallier dans [BOGA85]. Dans la suite de cette section 5, sauf mention contraire, nous considérerons toujours des automates à piles appartenant aux classes $TPDA^{rr}$ ou $DTPDA^{rr}$.

5.2 Calcul de formes normales

Nous définissons dans cette partie la notion de système de réécriture trf (pour tail reduction free).

Dans la section 5.2.1, nous donnons la définition de la propriété trf et montrons que la classe des systèmes de réécriture trf est large. Elle contient, en particulier, les systèmes de réécriture monadiques.

Nous prouvons que pour tout système de réécriture semi-monadique convergent, on peut trouver un système de réécriture semi-monadique équivalent ayant la propriété trf et démontrons que l'on peut simuler une machine de Türing par un système de réécriture trf.

Dans la section 5.2.2, nous démontrons que cette propriété est décidable. Pour prouver ce résultat, nous utilisons la décidabilité de l'inductive réductibilité (D.A.Plaisted).

Dans la section 5.2.3, le théorème 1 permet d'associer, à tout système de réécriture trf et convergent, un automate à piles ascendant d'arbres avec priorité à la réduction (on ne peut appliquer une règle de lecture que lorsqu'on ne peut utiliser aucune règle de réduction) qui calcule les formes normales. Dans le cas où le système de réécriture, vérifiant les mêmes hypothèses, est, de plus, linéaire à gauche, la notion de priorité devient inutile et le théorème 2 permet d'associer à tout système de réécriture trf, convergent et linéaire à gauche un automate à piles ascendant d'arbres et déterministe qui calcule les formes normales.

Nous rappelons que $\rightarrow_{i,S,<}$ est la relation de réécriture IO induite par S respectant <. Par cette relation, on réécrit un terme à une occurence o si aucune règle n'est applicable en dessous de cette occurence et s'il n'existe pas de règle plus grande selon < applicable à cette occurence o.

On étend cette définition aux automates à piles d'arbres et nous notons $\rightarrow_{<A}$ *la relation de réécriture IO induite par R selon un ordre < sur R*, où R désigne l'ensemble des règles de A.

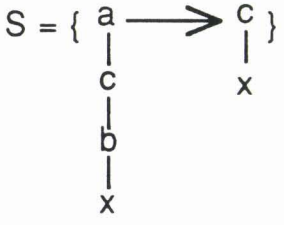
Cette définition nous permet de décrire formellement la transformation respectant la stratégie *"priorité à la réduction"* pour un automate à piles ascendant d'arbres.

5.2.1 La propriété trf

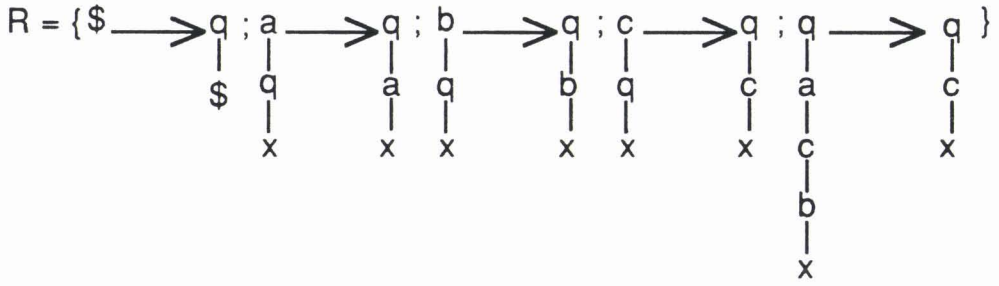
Nous introduisons la propriété trf à l'aide des deux exemples suivants.

Exemple 1: Soit Σ={a,b,c,$} avec a,b et c d'arité 1 et $ d'arité 0.

Soit le système de réécriture S réduit à une seule règle défini par:



Soit l'automate à piles A=(Σ,Γ,Q,Q_f,R) tel que Σ=Γ, Q=Q_f={q} et



A appartient à la classe $TPDA^{rr}$. Considérons le terme t=aaacbb$ de T_Σ (les parenthèses ont été omises). Les différentes réductions du terme t par l'automate A (non déterministe) ainsi défini sont:

$t \xrightarrow{*}_A q(t)$ (on lit sans réduire),

$t \xrightarrow{*}_A q(aacb\$)$ (on applique une fois la règle de réduction),

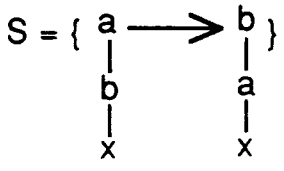
$t \xrightarrow{*}_A q(ac\$)$ (priorité à la réduction sur la lecture).

Les arbres de piles ainsi obtenus sont donc les différentes réductions du terme t par le système de réécriture S.

Si on impose, de plus, que la transformation par A se fasse en respectant la *priorité à la réduction* (on ne peut appliquer une règle de lecture que lorsqu'aucune règle de réduction n'est applicable, c'est-à-dire que les quatre premières règles sont inférieures à la cinquième pour <), on obtient alors $t \xrightarrow{*}_{<A} q(ac\$)$ et donc *l'arbre de pile obtenu dans ce cas est la forme normale de t pour le système de réécriture S.*

Exemple 2: Soit $\Sigma=\{a,b,\$\}$ avec a et b d'arités 1 et \$ d'arité 0.

Soit le système de réécriture S défini par:



Soit l'automate à piles $A=(\Sigma,\Gamma,Q,Q_f,R)$ tel que $\Sigma=\Gamma$, $Q=Q_f=\{q\}$ et R est défini comme précédemment par les règles de lecture de \$, a et b et la règle de type "reduce": $q(a(b(x)))\rightarrow q(b(a(x)))$.

On impose, en outre, la *stratégie priorité à la réduction*, si on considère le terme t=aabb\$, on obtient $t \xrightarrow{*}_{<A} aq(abb\$)$ par les règles de lecture, $aq(abb\$)\rightarrow_{<A} aq(b\mathbf{ab}\$)$ par la règle de type "reduce". On peut alors remarquer que l'*on ne pourra pas calculer la forme normale du terme t pour le système de réécriture S.*

En effet, on obtient $t \xrightarrow{*}_{<A} q(ba\mathbf{ab}\$)$ et l'arbre de pile ba**ab**\$ n'est pas la forme normale de t, cette forme normale étant baba\$.

La différence entre les systèmes de réécriture considérés dans ces deux exemples est que, si l'on considère le membre droit b(a(x)) de la règle de l'exemple 2, il existe une substitution close irréductible σ telle que $\sigma(a(x))$ soit réductible par S. Il suffit, en effet, de considérer la substitution $\sigma=\{x\rightarrow b\}$.

Définition: Soit S un système de réécriture, *S possède la propriété trf* (pour tail reduction free) si, pour toute règle $l\rightarrow r$ de S avec $r=b(r_1,...,r_n)$ et b lettre d'arité n, pour toute substitution close irréductible σ, alors, pour tout i, $1\leq i\leq n$, on a $\sigma(r_i)$ irréductible pour S.

Remarque: Nous ne considérons que des systèmes de réécriture pour lesquels il existe au moins une substitution close irréductible. Par exemple, nous interdisons que le membre gauche d'une règle de S soit réduit à une variable.

Exemples: Le système de réécriture défini dans l'exemple 1 de la section 5.2.1 possède de façon évidente la propriété trf. Par contre, celui de l'exemple 2 de la même section ne possède pas la propriété trf en considérant la substitution close irréductible $\sigma=\{x\rightarrow b\}$.

5.2.2 Exemples de systèmes trf

Exemple 1: Systèmes de réécriture monadiques.

Définition: Un système de réécriture est *monadique* si, pour toute règle $l\rightarrow r$ de S, la hauteur du membre gauche l est supérieure ou égale à 1 et le membre droit r est une constante, une variable, ou de la forme $b(y_1,...,y_n)$ avec b d'arité n et, pour tout i, y_i est une variable.

Proposition 1: *tout système de réécriture monadique a la propriété trf.*

Preuve: La preuve est évidente car $h(r)\leq 1$.❑

Exemple 2: Systèmes de réécriture semi-monadiques.

Définition: Un système de réécriture est *semi-monadique* si, pour toute règle $l\rightarrow r$ de S, la hauteur du membre gauche est supérieure ou égale à 1 et le membre droit r est, soit une variable, soit de la forme $b(y_1,...,y_n)$ avec b d'arité n et, pour tout i , y_i est une variable ou un terme clos.

On remarquera que les systèmes de réécriture monadiques et les systèmes de réécriture clos sont des cas particuliers de systèmes de réécriture semi-monadiques.

Proposition 2: *Considérons un système de réécriture S semi-monadique et convergent. Il existe un système de réécriture semi-monadique et convergent S' équivalent à S tel que S' posséde la propriété trf.*

Preuve: Nous remplaçons chaque terme clos apparaissant dans une partie droite de règle par sa forme normale pour S.❑

Exemple 3: Simulation d'une machine de Turing.

Nous pouvons identifier toute machine de Turing à un automate à piles de la classe $TPDA_{1,2}$, que l'on peut également considérer comme un système de réécriture trf, de la façon suivante:

Nous mettons en correspondance les chaînes de caractères et les arbres monadiques et utilisons les nouveaux symboles $ et #(x) pour délimiter les réécritures.

Le mot d'entrée t de Σ^* est identifié à l'arbre #t'$ sur un alphabet Σ'. L'alphabet de pile Σ est distinct de Σ', Σ' est une copie de Σ (nous notons a' la copie de a). Le mot de sortie u est identifié à #u$. Nous notons ^t le miroir de t; s, s' sont des états spéciaux, s_f est l'état final.

"règles de lecture" :

$\$ \rightarrow s(\$)$;

$a's(x) \rightarrow s(ax)$ pour toute lettre a' de Σ';

$\#s(x) \rightarrow q_i(T(\#(x),\$))$ (q_i est l'état initial de la machine de Turing).

"règles de réécriture" :

A chaque état q de la machine de Turing on associe l'état q d'arité 1 du tpda.

A chaque règle r de la machine de Turing (q,a) → (q',b,droite) nous associons les règles du tpda:

$q(T(a(x),y)) \rightarrow q(u_r(b(x),y))$,

$q(u_r(x,c(y))) \rightarrow q'(T(c(x),y))$, pour tout c de Σ;

A chaque règle de la machine de Turing (q,*) → (q',b*) (extension de la bande), nous associons les règles du tpda $q(T(\#(x), y)) \rightarrow q'(T(b(x),\#(y)))$;

A chaque règle de la machine de Turing (q,a) → (q',b,gauche), nous associons la règle du tpda $q(T(a(x),y)) \rightarrow q'(T(x,b(y)))$;

A chaque état final q_f de la machine de Turing, nous associons un mouvement final du tpda qui remonte l'état final en utilisant les règles suivantes:

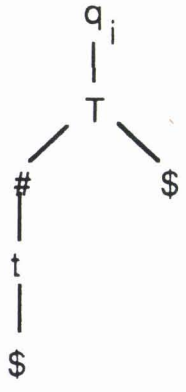
$q_f(T(a(x),y) \rightarrow s'(T(x,a(y)))$, pour tout a de Σ,

$s'(T(a(x),y) \rightarrow s'(T(x,a(y)))$, pour tout a de Σ,

$s'(T(\$,y) \rightarrow s_f(\#(y))$.

Théorème 1: *Nous pouvons simuler toute machine de Turing par un système de réécriture trf associé à un* $TPDA_{12}$.

Preuve:. Grâce aux règles de lecture, l'arbre #t'$ est transformé en l'arbre



La simulation ne peut commencer qu'après l'apparition de l'état q_i au sommet de l'arbre; les règles de réécriture données ci-dessus simulent les différents mouvements d'une machine de Turing; à tout état final de la machine correspondent des réécritures amenant dans l'état final s_f. Aussi on vérifie qu'à partir d'une donnée ^t , nous obtenons ^u par la machine de Turing si et seulement si $\#t'\$ \overset{*}{\rightarrow} s_f(\#u\$)$ pour le tpda.

De plus, l'ensemble des règles que nous avons défini est un système de réécriture trf.❑

5.2.3 La propriété trf est décidable

Théorème: *La propriété trf est décidable.*

Preuve:

Nous rappelons, tout d'abord, quelques définitions.

Un *terme* t de $T_\Sigma(X)$ est *inductivement réductible* pour S si et seulement si toutes ses instances closes $\sigma(t)$ dans T_Σ sont réductibles pour S.

Une *substitution* σ est *inductivement réductible* pour S si et seulement si $\$(\sigma(x_1),...,\sigma(x_n))$ est inductivement réductible pour S avec $D(\sigma)=\{x_1,...,x_n\}$ et \$ comme nouveau symbole (c'est à dire si et seulement si pour toute substitution close δ, il existe x dans $D(\sigma)$ tel que $\delta(\sigma(x))$ soit réductible pour S).

Soit W un sous-ensemble de V, nous notons $\sigma_{/W}$ la substitution dont le domaine est l'intersection du domaine $D(\sigma)$ et de W, elle sera appelée restriction de σ à W.

La décidabilité de la propriété trf pour un système de réécriture fini S correspond à la décidabilité de la *propriété P(t)*: "pour toute substitution close irréductible σ, $\sigma(t)$ est irréductible", étant donné que le nombre de règles et donc de termes r_i est fini. Pour prouver la décidabilité de P(t), il est équivalent de prouver la décidabilité de P'(t), la négation de P(t). *Propriété P'(t):* "il existe une substitution close irréductible σ telle que $\sigma(t)$ est réductible".

Pour celà, nous allons prouver le résultat suivant:

Lemme: *Il existe une substitution close irréductible σ pour laquelle $\sigma(t)$ est réductible si et seulement s'il existe une partie gauche l de règle de R et un sous-terme t' (différent d'une variable) de t tels que l et t' soient unifiables par un unificateur le plus général σ', $\sigma'/V(t')$ n'étant pas inductivement réductible.*

Preuve:

Preuve de $\Rightarrow$*:*

Supposons que $\sigma(t)$ soit réductible pour la substitution close irréductible σ. Il existe donc une décomposition $t = t''.t'$ avec t' non réduit à une variable telle que $\sigma(t') = \sigma_2(l)$ pour une partie gauche l de règle de R et pour une substitution σ_2. t' et l étant unifiables, considérons l'unificateur le plus général σ' de t' et l vérifiant donc $\sigma'(t')=\sigma'(l)$.

Nous pouvons alors distinguer deux cas:

Premier cas: σ' est une substitution close.

Dans ce cas $\sigma_{/V(t')}=\sigma'_{/V(t')}$ (i.e. $\sigma(t')=\sigma'(t')$). Si $\sigma'_{/V(t')}$ était réductible alors σ le serait aussi ce qui contredit l'affirmation que σ est irréductible.

Deuxième cas: σ' est une substitution non close.

Considérons la substitution σ'' telle que $\sigma(t') = \sigma''(\sigma'(t'))$ et $\sigma_{/V(t')}= \sigma''(\sigma'_{/V(t')})$.

Si $\sigma'_{/V(t')}$ était inductivement réductible, alors pour la substitution close σ'', $\sigma''(\sigma'_{/V(t')})$ serait réductible ce qui contredit l'affirmation que σ est irréductible. Donc $\sigma'_{/V(t')}$ n'est pas inductivement réductible.

Preuve de $\Leftarrow$*:*

Inversement, supposons qu'il existe une partie gauche l de règle de R et un sous-terme t' (différent d'une variable) de t tels que l et t' soient unifiables par un unificateur le plus général σ', $\sigma'_{/V(t')}$ n'étant pas inductivement réductible.

Nous distinguons à nouveau deux cas:

Premier cas: σ' est une substitution close.

Comme $\sigma'_{/V(t')}$ est irréductible, pour toute variable x_i de $V(t')$, $\sigma'(x_i)$ est irréductible.

Deuxième cas: σ' est une substitution non close.

Il existe une substitution close σ'' telle que $\sigma''(\sigma'_{/V(t')})$ soit irréductible. Pour toute variable x_i de $V(t')$, $\sigma''(\sigma'(x_i))$ est irréductible.

Dans les deux cas, nous définissons la substitution σ de la façon suivante: $\sigma(x_i) = \sigma''(\sigma'(x_i))$ ($\sigma'(x_i)$ dans le premier cas) pour tout $x_i \in V(t')$ et $\sigma(y) = t_y$ pour tout y de $V(t)-V(t')$ où t_y est un terme clos irréductible arbitraire. Il est évident que σ est une substitution close et irréductible et que $\sigma(t)$ est réductible.❑

En utilisant ce lemme et la décidabilité de l'inductive réductibilité ([PLAI85]), nous en déduisons la décidabilité de P'(t), d'où la décidabilité de P(t) et donc la décidabilité de la propriété trf.❑

5.2.4 Calcul des formes normales d'un système trf

Dans cette partie, nous allons associer, à tout système de réécriture possédant la propriété trf, un automate à piles ascendant d'arbres.

Dans les propositions 1 et 2, nous démontrons qu'il existe un automate à piles avec priorité (cas général), déterministe (cas linéaire à gauche) qui calcule les formes normales pour le système de réécriture pour la relation de réécriture IO induite par S.

Avec l'hypothèse supplémentaire que S est convergent, on déduit de ces résultats les deux théorèmes principaux de cette section relatifs au calcul des formes normales d'un système de réécriture trf convergent.

Proposition 1: *Pour tout système de réécriture S possédant la propriété trf, il existe un automate à piles ascendant d'arbres* $A=(\Sigma,\Gamma,Q,Q_f,R)$ *et un ordre* $<$ *sur R tels que l'équivalence (*) suivante soit vérifiée:*

$$(*): (\ t \xrightarrow{*}_{i,S} t' \text{ et } t' \in IRR(S)\ ) \Leftrightarrow (\ t \xrightarrow{*}_{<A} q(t') \text{ et } q \in Q_f\ ).$$

Preuve: Soit S un système de réécriture sur $T_\Sigma(X)$ possédant la propriété trf. Nous définissons, tout d'abord, A et $<$.

$A=(\Sigma,\Gamma,Q,Q_f,R)$ avec $\Gamma=\Sigma$, $Q=\{q, q'\}$, $Q_f=\{q\}$ et R constitué des règles des trois types suivants:

(i): $q(l) \rightarrow q(r)$, pour toute règle $l \rightarrow r$ de S.

(ii): $q(x) \rightarrow q'(x)$.

(iii): $b(q'(x_1),\ldots,q'(x_k)) \rightarrow q(b(x_1,\ldots,x_k))$, pour tout b de Σ_k.

Les règles de type (i) et (ii) sont les règles de réduction ("reduce-rules") de A, les règles de type (iii) sont les règles de lecture ("read-rules") de A.

La relation d'ordre $<$ sur R est une relation d'ordre partiel définie par: $(q(x) \rightarrow q'(x)) < (q(l) \rightarrow q(r))$, pour toute règle $q(l) \rightarrow q(r)$ de R.

Intuitivement, l'ordre < est défini pour que, dans la transformation par A, on n'applique une règle de lecture (de type (iii)) que lorsque les arbres de piles correspondants sont irréductibles pour les règles de réduction (de type (i) et (ii)), c'est-à-dire encore lorsque ces arbres de pile sont irréductibles pour le système de réécriture S.

Preuve de $\Rightarrow$*:* Nous prouvons pour celà l'implication suivante:

(**): Pour t_0 dans $T_\Sigma(X_n)$, $t_1,\ldots, t_n$, $u_1,\ldots,u_n$ dans T_Σ,
si (1) $t=t_0(t_1,\ldots,t_n) \xrightarrow{*}_{i,S} u=t_0(u_1,\ldots,u_n)$, et
(2) Pour tout i de [1,n], $t_i \xrightarrow{*}_{i,S} u_i$ et la racine de t_i est réécrite,
alors il existe une configuration c de A vérifiant:
(a) $t \xrightarrow{*}_{<A} c=t_0(q_1(u_1),\ldots,q_n(u_n))$,
(b) si u_i est irréductible pour S, alors $q_i=q'$,
(c) si u_i est réductible pour S, alors $q_i=q$,
(d) si u_i est réductible pour S et $u_i=b(v_1,\ldots,v_k)$ avec b dans Σ_k et $v_1,\ldots,v_k$ dans T_Σ, alors, pour tout j de [1,k], v_j est irréductible pour S.

Preuve de (**): La preuve se fait par induction sur la longueur p de la dérivation $t \xrightarrow{*}_{i,S} u$.

Pour p=0, c=t est la configuration vérifiant les propriétés.

Soit p un entier supérieur ou égal à 1, et supposons (**) prouvée pour des dérivations de longueur inférieure à p. Considérons une dérivation: $t=t_0(t_1,\ldots,t_n) \xrightarrow{(p-1)}_{i,S} u=t_0(u_1,\ldots,u_n) \rightarrow_{i,S} u'$. Par hypothèse d'induction, il existe une configuration c de A associée à t et u vérifiant les hypothèses (a), (b), (c) et (d). Nous distinguons deux cas:

Premier cas: La réécriture de u en u' se fait à une occurence o n'appartenant pas à $O(t_0)$. Donc, nous réécrivons un sous-terme de l'un des u_i: $u=t_0(u_1,\ldots,u_i,\ldots,u_n) \rightarrow_{i,S} u'=t_0(u_1,\ldots,u_i',\ldots,u_n)$ avec la règle $l\rightarrow r$ de R. Soit $c=t_0(q_1(u_1),\ldots,q_i(u_i),\ldots,q_n(u_n))$ la configuration de A associée à t et u. Comme u_i est réductible pour S, $q_i=q$ et, de plus, si $u_i = b(v_1,\ldots,v_k)$, alors, pour tout j de [1,k], v_j est irréductible pour S. Donc la réécriture se fait à une occurence o telle que $u_{/o}=u_i$. Nous avons donc $u_{/o}=u_i=\sigma(l)$, $u'_{/o}=u_i'=\sigma(r)$ avec σ qui est une substitution close et irréductible pour S. Par définition de A, il existe une règle $q(l)\rightarrow q(r)$ dans R (associée à la règle $l\rightarrow r$ de S) et donc $c \rightarrow_{<A} c'=t_0(q_1(u_1),\ldots,q(u_i'),\ldots,q_n(u_n))$ par définition de $\rightarrow_{<A}$.

Si u'_i est réductible pour S, l'état correspondant est bien l'état q et, de plus, si $u'_i = b'(v'_1,...,v'_s)$, alors, pour tout j de [1,s], v'_j est irréductible pour S car σ est une substitution close et irréductible pour S et S possède la propriété trf. Dans ce cas, la configuration c' vérifie les propriétés (a), (b), (c) et (d).

Si u'_i est irréductible pour S, alors, aucune règle de type (i) n'est applicable à l'occurence o pour la configuration c', donc on peut appliquer la règle de type (ii), donc $c' \rightarrow_{<A} c''=t_0(q_1(u_1),...,q'(u'_i),..., q_n(u_n))$ et c" vérifie les conditions (a), (b), (c) et (d).

Deuxième cas: La réécriture de u en u' par S se fait à une occurence o appartenant à $O(t_0)$. Nous avons donc:
$u=t_0(u_1,...,u_n) = u_0(u_1,...,u_{i-1},u',u_{j+1},...,u_n)$ avec $u' = u'_0(u_i,...,u_j)$, $u \rightarrow_{i,S} v$, $v=u_0(u_1,...,u_{i-1},v',u_{j+1},...,u_n)$ et $u' = \sigma(l)$, $v' = \sigma(r)$ pour une règle $l \rightarrow r$ de R.

Par définition de la relation de réécriture IO engendrée par S, tout sous terme propre de u' est irréductible pour S et donc, en particulier, tous les états associés aux arbres de pile $u_i,...,u_j$, pour la configuration c, sont égaux à q'. Soit le sous terme de c dont la racine est à l'occurence o, nous avons $c_{/o}= u'_0(q'(u_i),..,q'(u_j))$. Pour tout symbole dans u'_0 distinct de la racine de u'_0, on peut appliquer une règle de lecture, on obtient alors q(w) avec w sous-terme propre de u', et donc, w est irréductible pour S, donc, on ne peut, à chaque étape, qu'appliquer la règle de type (ii). Donc, par induction sur la structure du terme, on peut montrer que:
$c_{/o} \xrightarrow{*}_{<A} q(u')$ et $c \xrightarrow{*}_{<A} c'=u_0(q_1(u_1),...,q_{i-1}(u_{i-1}),q(v'),q_{j+1}(u_{j+1}),...,q_n(u_n))$.

Par le même raisonnement que dans le premier cas, la configuration c' ou la configuration c" (selon que v' soit réductible par S ou pas) vérifie les propriétés (a), (b), (c) et (d).

❑ *finpreuve(**)*

Supposons maintenant que $t \xrightarrow{*}_{i,S} t'$ and $t' \in IRR(S)$. Il existe une décomposition $t=t_0(t_1,...,t_n)$ avec t_0 dans $T_\Sigma(X_n)$ telle que, pour tout i de [1,n], $t_i \xrightarrow{*}_{i,S} u_i$ et $t'=t_0(u_1,...,u_n)$. En utilisant l'implication (**), il existe une configuration $c=t_0(q'(u_1),...,q'(u_n))$ de A telle que $t \xrightarrow{*}_{<A} c$ et $c \xrightarrow{*}_{<A} q'(t')$.

❑ *finpreuve de* $\Rightarrow$.

Preuve de $\Leftarrow$: Pour celà, nous prouvons l'implication suivante:

(***): Pour t_0 dans $T_\Sigma(X_n)$, $t_1,...,t_n$, $\pi_1,...,\pi_n$ dans T_Σ, si
$t=t_0(t_1,...,t_n) \overset{*}{\rightarrow}_{<A} c=t_0(q_1(\pi_1),...,q_n(\pi_n))$, alors
(a) si $q_i=q'$, alors π_i est irréductible pour S,
(b) si $q_i=q$ et $\pi_i=b(\upsilon_1,...,\upsilon_k)$ avec b dans Σ_k et $v_1,...,v_k$ dans T_Σ, alors, pour tout j de [1,k], υ_j est irréductible pour S,
(c) $t \overset{*}{\rightarrow}_{i,S} t_0(\pi_1,...,\pi_n)$.

Preuve de (***): La preuve se fait par induction sur la longueur p du calcul.

Pour p=0, la propriété est vérifiée.

Soit p un entier supérieur ou égal à 1, et supposons (***) vérifiée pour tout calcul de longueur inférieure à p. Considérons un calcul:
$t \overset{(p-1)}{\rightarrow}_{<A} c=t_0(q_1(\pi_1),...,q_n(\pi_n)) \rightarrow_{<A} c'$. Nous distinguons trois cas:

Premier cas: La règle appliquée à l'étape p est de type (i).
Donc un sous-terme $q_i(\pi_i)$ est réécrit en $q_i(\pi'_i)$ avec $q_i=q$ pour un certain i et une règle $q(l)\rightarrow q(r)$ (associée à la règle $l\rightarrow r$ de S) de R.
Donc, dans cette étape du calcul, nous ne modifions que le $i^{ème}$ arbre de pile et n'introduisons pas de nouvelle occurence de l'état q'. Il suffit donc de prouver que si $\pi'_i=b'(\upsilon'_1,...,\upsilon'_m)$ alors, pour tout j de [1,m], υ'_j est irréductible pour S.

Soit σ la substitution utilisée pour réécrire $q_i(\pi_i)$ en $q_i(\pi'_i)$, nous avons $\sigma(l) = \pi_i$ et donc, par hypothèse d'induction, σ est une substitution close irréductible pour S (tout sous-terme propre de π_i est irréductible pour S). De plus $\sigma(r)= \pi'_i$ et donc, S possédant la propriété trf, nous en déduisons que pour tout j de [1,m], υ'_j est irréductible pour S.

Deuxième cas: La règle appliquée à l'étape p est de type (ii).
Il existe donc i tel que $q(\pi_i)$ est transformé en $q'(\pi_i)$. Par définition de $\rightarrow_{<A}$ et de <, aucune règle de type (i) ne peut être appliquée au sous-terme $q(\pi_i)$, ce qui signifie que π_i est irréductible pour S.

Troisième cas: La règle appliquée à l'étape p est de type (iii). Donc, un sous-terme de la forme $b(q'(\pi_i),...,q'(\pi_{i+k}))$ avec b dans Σ_k est réécrit en $q(b(\pi_i,...,\pi_{i+k}))$. Par hypothèse d'induction, tous les sous-termes $\pi_i,...,\pi_{i+k}$ sont irréductibles pour S.

Nous avons donc prouvé, par induction, que la configuration c' vérifie (a) et (b). La démonstration de (c) par induction ne présente aucune difficulté.❑ *finpreuve⇐.*
❑ *finpreuveproposition1.*

Nous allons maintenant considérer le cas d'un système de réécriture S linéaire à gauche possédant la propriété trf tel que S soit muni d'une relation d'ordre total et prouver que, dans ce cas, on peut calculer les formes normales pour la relation de réécriture IO induite par S en respectant < à l'aide d'un automate à piles ascendant et déterministe.

***Proposition 2**: Pour tout système de réécriture S linéaire à gauche possédant la propriété trf et pour tout ordre total < sur S, il existe un automate à piles déterministe ascendant d'arbres $B=(\Sigma,\Gamma,Q,Q_f,R)$ tel que:*

$$(t \xrightarrow{*}_{i,S,<} t' \text{ et } t' \in IRR(S)) \Leftrightarrow (t \xrightarrow{*}_B q(t') \text{ et } q \in Q_f).$$

Preuve: Soit S un système de réécriture linéaire à gauche possédant la propriété trf. Soit < un ordre total sur S. Soit $k = \max\{h(l)/\ l \to r \in S\}+1$ et $T_\Sigma(X,k)$ l'ensemble des arbres k-normalisés.

Nous définissons $B=(\Sigma,\Gamma,Q,Q_f,R)$ comme suit:
$\Gamma=\Sigma$,
$Q=\{q\} \cup \{q_t / t \in T_\Sigma(X,k)\}$,
$Q_f=\{q_t$ / t n'est pas unifiable avec un membre gauche de règle de S$\}$,
R est l'ensemble des règles des trois types suivants:

(i): $q(t) \to q_t(t)$, pour tout t de $T_\Sigma(X,k)$.

(ii): $q_t(l) \to q(r)$, pour tout q_t n'appartenant pas à $Q_f \cup \{q\}$, avec $l \to r$ la plus grande règle pour < telle que l filtre t.

(iii): $b(q_1(x_1),...,q_n(x_n)) \to q(b(x_1,...,x_n))$ avec b dans Σ_k et, pour tout i de [1,n], q_i appartient à Q_f.

Les règles de type (i) et (ii) sont les règles de réduction (reduce-rules") de B, les règles de type (iii) sont les règles de lecture ("read-rules") de B.

B est déterministe car R est linéaire à gauche et sans paires critiques. La preuve de l'équivalence annoncée se fait exactement de la même façon que pour la proposition 1.

Intuitivement,

les règles de type (i) permettent de mémoriser dans l'état le sous-arbre de pile à une profondeur k;

les règles de type (ii) permettent de réécrire les arbres de pile par S en respectant l'ordre < sur S. Les états de Q_f correspondent à l'état q' de la proposition 1, c'est-à-dire que les sous-arbres de piles correspondants sont irréductibles pour S;

les règles de type (iii) permettent de lire un nouveau symbole de l'alphabet d'entrée tout en vérifiant que les sous arbres de piles correspondants sont irréductibles pour S.❑

Remarque importante: La construction faite *n'est valable que pour un système de réécriture linéaire à gauche.* En effet, dans le cas non linéaire à gauche, on ne peut pas exprimer avec un nombre fini d'états ou un nombre fini d'arbres (les états q_t ou les arbres t de $T_\Sigma(X,k)$) qu'un terme est unifiable ou pas avec un membre gauche de règle, en effet, on doit tester une égalité de sous-termes à une profondeur non bornée.

Nous ajoutons maintenant l'hypothèse convergent (noethérien et confluent) et donc la stratégie de réécriture utilisée pour le système de réécriture S n'importe pas sur le résultat, la forme normale d'un terme étant unique. Les deux théorèmes principaux sont donc:

Théorème 1: *Pour tout système de réécriture S convergent possédant la propriété trf, il existe un automate à piles ascendant d'arbres* $A=(\Sigma,\Gamma,Q,Q_f,R)$ *et un ordre* $<$ *sur R tel que:* $(t \xrightarrow{*}_S t' \text{ et } t' \in IRR(S)) \Leftrightarrow (t \xrightarrow{*}_{<A} q(t') \text{ et } q \in Q_f)$.

Théorème 2: *Pour tout système de réécriture S linéaire à gauche convergent possédant la propriété trf, il existe un automate à piles déterministe ascendant d'arbres* $B=(\Sigma,\Gamma,Q,Q_f,R)$ *t. q:* $(t \xrightarrow{*}_S t' \text{ et } t' \in IRR(S)) \Leftrightarrow (t \xrightarrow{*}_B q(t') \text{ et } q \in Q_f)$.

5.3 Automates à piles d'arbres et langages d'arbres

5.3.1 Systèmes de réécriture semi-monadiques et langages reconnaissables

Nous prouvons dans cette partie que la reconnaissabilité est préservée pour les systèmes de réécriture semi-monadiques. Ce résultat généralise le résultat de K. Salomaa ([SALO88]) obtenu pour les systèmes de réécriture monadiques et le même résultat pour les systémes de réécriture clos ([DATI85],[COGI90]).

Nous considérons un système de réécriture S et un langage reconnaissable d'arbres L; nous rappelons que S(L) désigne l'ensemble des réductions des termes de l'ensemble L par le système de réécriture S, i.e $S(L)=\{ t' / t \xrightarrow{*}_S t' \text{ et } t \in L\}$.

Théorème: *Pour tout système de réécriture semi-monadique linéaire S et tout langage reconnaissable d'arbres L, S(L) est reconnaissable.*

Preuve: Considérons un automate ascendant d'arbres $A = (\Sigma, Q, Q_f, R)$ complètement spécifié tel que $L(A) = L$.

Soit S un système de réécriture semi-monadique linéaire, c'est à dire tel que pour toute règle $l \to r$ de S, $h(l) \geq 1$ et r est soit une variable ($r \in X$) soit un arbre du type $b(y_1,\ldots,y_k)$ où b est un élément de Σ_k, y_i est une variable ($y_i \in X$) ou un terme clos ($y_i \in T_\Sigma$), pour tout i de [1,k].

Nous notons Sub l'ensemble de tous les sous-termes clos des parties droites de règles de S et des termes clos y_i.

Nous utilisons le nouveau symbole # lors de la définition des états permettant de reconnaître S(L) lorsque nous rencontrons un arbre n'appartenant pas à l'ensemble Sub.

Nous notons $\tilde{Q}$ l'ensemble des états utilisés lors de la reconnaissance de $S^*(L)$. Il est défini par:

$\tilde{Q}=\{(q,p) \in Q \times (Sub \cup \{\#\}) \text{ / soit } p=\# \text{ et } (\exists r \in T_\Sigma): r \xrightarrow{*}_A q \text{ avec } r \notin Sub$
soit $p \in Sub$ et $p \xrightarrow{*}_A q$ $\}$

Nous définissons, par induction, les ensembles R_i de règles de la façon suivante:

$$R_0=\{\sigma((q_1,p_1),\ldots,(q_n,p_n))\to(q,p) \;/\; \sigma\in\Sigma_n,\ (q_1,p_1),\ldots,(q_n,p_n),(q,p)\in \tilde{Q},$$
$$\sigma(q_1,\ldots,q_n)\to q\in R,$$
$$p=\sigma(p_1,\ldots,p_n) \text{ si } \sigma(p_1,\ldots,p_n)\in Sub$$
$$\#\quad \text{sinon}$$
$$\}$$

Pour tout $i, i\geq 1$, $R_i=R_{i-1}\cup N_i$ avec

$$N_i=\{\sigma((q_1',p_1'),\ldots,(q_n',p_n'))\to(q,p)/\ (q_1',p_1'),\ldots,\ (q_n',p_n'),\ (q,p)\in \tilde{Q},$$
$$\sigma\in\Sigma_n,\ l(x_1,\ldots,x_k)\to\sigma(y_1,\ldots,y_n)\in S,$$
$$l((q_1,p_1),\ldots,(q_k,p_k))\xrightarrow{*}R_{i-1}\ (q,p),\ (q_1,p_1),\ldots,\ (q_n,p_n)\in \tilde{Q},$$
$$\text{Si } y_j\in X,\ y_j\in V(l) \text{ alors } (q_j,p_j)=(q_j',p_j'),$$
$$\text{Si } y_j\in T_\Sigma \text{ alors } (q_j',p_j')=(q_j,y_j) \text{ avec } y_j\xrightarrow{*}_A q_j$$
$$\}$$
$$\cup\{\sigma((q_1',p_1'),\ldots,(q_n',p_n'))\to(q,p)/\ (q_1',p_1'),\ldots,(q_n',p_n'),(q,p)\in \tilde{Q},$$
$$\sigma\in\Sigma_n,\ l(x_1,\ldots,x_k)\to x_i\in S,$$
$$\sigma((q_1',p_1'),\ldots,(q_n',p_n'))\xrightarrow{*}R_{i-1}\ (q_i,p_i),$$
$$l((q_1,p_1),\ldots,(q_k,p_k))\xrightarrow{*}R_{i-1}\ (q,p),\ (q_1,p_1),\ldots,(q_n,p_n)\in \tilde{Q}\}.$$

Comme $Qx(Sub\cup\{\#\})$ et R sont des ensembles finis il existe un entier naturel k_0 tel que $R_k = R_{k_0}$ pour tout $k \geq k_0$.

Nous pouvons maintenant définir l'automate $B = (\Sigma, \tilde{Q}_f, \tilde{Q}_f, R_{k_0})$ tel que $\tilde{Q}_f = \tilde{Q}\cap(Q_f x(Sub\cup\{\#\}))$. Nous allons démontrer que $L(B) = S(L)$.

Considérons un terme t de T_Σ et u un élément de $dom(t)$; nous dirons que le noeud u est réécrit par utilisation d'une R_i-règle propre avec $0 \leq i \leq k_0$, si la règle appliquée en u est un élément de R_i-R_{i-1}.

Si j, $0\leq j\leq k_0$, est le maximum des i tels que des R_i-règles propres ont été appliquées lors du calcul C et si $w > 0$ est le nombre d'applications de R_j-règles propres lors de C alors nous dirons que C a un (j,w)-calcul.

De plus nous dirons que C est un j-calcul, si C est un (j,w)-calcul pour un $w>0$ donné.

Soit t un élément de L(B) et C un (j,w)-calcul de B sur t, $0 \leq j \leq k_0$, $w > 0$.

(i) si $j = 0$ alors $t \in L$

(ii) Prenons $j > 0$ et $w \geq 2$. De par la définition de N_i nous savons qu'il existe un arbre t' tel que $t' \rightarrow_S t$ et il existe un (j,w-1)-calcul de B sur t'.

(iii) Prenons $j > 0$ et $w = 1$. De par la définition de N_i nous savons qu'il existe un arbre t' tel que $t' \rightarrow_S t$ et il existe un (j-1,w')-calcul de B sur t'.

Par conséquent il existe un arbre t_0 et un entier naturel w_0 (≥ 1) tels que $t_0 \overset{*}{\rightarrow}_S t$ et il existe un $(0,w_0)$-calcul de B sur t_0. Nous avons $t_0 \in L$ et donc $t \in S(L)$.

Réciproquement, s'il existe un i-calcul sur t et si $t \rightarrow_S t'$ alors il existe un j-calcul sur t' avec $j \leq \sup(i+1,k_0)$. Puisque L est inclus dans L(B), S(L) est inclus dans L(B).

Nous avons donc L(B) = S(L) et, par conséquent, S(L) est un langage reconnaissable.❑

Remarque: Ce résultat est évidemment faux si S n'est pas linéaire à droite; il suffit de considérer le système de réécriture réduit à la règle $f(x) \rightarrow g(x,x)$ et le langage reconnaissable d'arbres L= fa*$.

Nous conjecturons que le résultat reste valable dans le cas d'un système de réécriture linéaire à droite. La preuve précédente ne s'étend pas de façon immédiate au cas non linéaire à gauche, en effet, il faut dans la construction de B déterminiser les automates définis à chaque étape, ce qui complique singulièrement la preuve de terminaison de la construction.

5.3.2 Automates à piles d'arbres et langages algébriques

Soit $A=(\Sigma,\Gamma,Q,Q_f,R)$ un automate à piles, on peut définir le langage reconnu par A par états finaux ou par pile vide.

$L_f(A) = \{ t \in T_\Sigma / t \xrightarrow{*}_A q(t') \text{ et } t \in Q_f \}$ est le langage reconnu par états finaux de A.

$L_e(A)=\{ t \in T_\Sigma / t \xrightarrow{*}_A q(\perp) \}$ où $\perp$ est le symbole de pile vide, symbole distingué de Γ.

On peut montrer que ces deux définitions sont équivalentes pour les classes considérées. Nous avons montré (théorème 1, section 5.2.2) que toute machine de Turing pouvait être simulée par un automate à piles de la classe $TPDA_{1,2}$, on montrerait facilement le même résultat pour la classe $TPDA_{2,1}$.

Considérons maintenant la classe $TPDA^{rr}$. K.M.Schimpf et J.H.Gallier ont défini une sous-classe de $TPDA^{rr}$ associée à la classe des langages algébriques ([GASC85], [SCHI82]). Le principe de la construction est le suivant: l'automate possède un seul état q, le système de réécriture R est obtenu de la façon suivante :

A toute règle $l \rightarrow r$ de la grammaire algébrique G, avec $l=B(x_1,...,x_n)$ où B appartenant à V est d'arité n, et, r est un terme de $T_{\Sigma \cup V}(X_n)$, on associe la règle de réécriture monadique $q(r) \rightarrow q(l)$.

Notons que pour la règle $q(r) \rightarrow q(l)$ ainsi définie, l'ensemble des variables du membre droit n'est pas nécessairement inclus dans l'ensemble des variables du membre gauche. Nous avons toujours supposé cette condition vérifiée, donc, par le même raisonnement, nous obtiendrions le même résultat mais uniquement en nous limitant au cas complet (toute variable apparaissant dans l apparaît dans r) et strict (r n'est pas réduit à une variable). Ce cas correspond à un algorithme non déterministe linéaire d'analyse.

Une classe intéressante à étudier, du point de vue théorie des langages d'arbres, est la classe $TPDA_{11}$ des automates à piles d'arbres dont les états ont pour arité 1 et tels que la profondeur des arbres de piles dépilés est au maximum 1.

Nous conjecturons que la classe des langages reconnus est une sous-classe des langages algébriques et que dans cette classe, un arbre d'entrée peut être analysé en temps linéaire.

6 BIBLIOGRAPHIE

[ARDA76]: A. Arnold and M. Dauchet, Un théorème de duplication pour les forêts algébriques, *J. Computer Systems Science*, 13, pp 223-244, (1976).

[ARDA78]: A. Arnold and M. Dauchet, Forêts algébriques et homomorphismes inverses, *Inform. and Control*, 37, pp182-196, (1978).

[BACH88]: L.Bachmair, Proof by consistency in equational theories, in *Proc. 3rd IEEE Symp. Logic in Computer Science*, (1988).

[BADE86]: L.Bachmair and N.Dershowitz, Commutation, transformation and termination, *Lec. Notes in Comp. Sci.*, 230, (1986).

[BADE87]: L.Bachmair and N.Dershowitz, Completion for rewriting modulo a congruence, *Lec. Notes in Comp. Sci.*, 256, pp 192-203, (1987).

[BENO69]: M.Benois, Paries rationnelles du groupe libre, C.R.Acad.Sci Paris, A 269, (1969).

[BERS79]: J.Berstel, Transductions and context free languages, in *Teubnen Studienbücher Informatik*, (1979).

[BJW81]: R.V. Book, M. Jantzen and C. Wrathall, Monadic Thue Systems, *Theoret. Comput. Sci. 19* , 3, pp 231-252, (1981).

[BOCK87]: A.Bockmayr, A note on a canonical theory with undecidable unification and matching problem, *Journal of Automated reasoning*, 3, pp 379-381, (1987).

[BOGA85]: R.V. Book and J.H. Gallier, Reductions in Tree Replacement Systems, *Theoret. Comput. Sci.*, 37, pp 123-150, (1985).

[BOGA90]: B.Bogaert, Automates à tests d'égalités, Thèse de l'université de Lille, (1990).

[BOOK83]: R.V. Book, Decidable sentences of Church-Rosser congruences, *Theoret. Comput. Sci.*, 24, pp 301-312, (1983).

[BOOK87]: R.V. Book, Thue systems as Rewriting Systems, *J. of Symbolic Computation*, 3, pp 39-68, (1987).

[BRAI69]: W.S. Brainerd, Tree generating regular systems, *Inf. and Control*, 14 , pp 217-231, (1969).

[CDGV90]: J.L.Coquidé, M.Dauchet, R.Gilleron and S.Vagvölgyi, Tree pushdown automata and Rewrite Systems, to appear in *Proc. RTA91*, *Technical Report I.T.190*, Université Lille, (1990).

[CDT89]: J.L. Coquidé, M.Dauchet and S.Tison, About connections between syntactical and computational complexity, FCT'89, *Lec.Notes.comp.Sci 380*, (1989).

[COGI90]: J.L.Coquidé and R.Gilleron, Proofs and Reachability problems for Rewrite Systems, IMYCS 90, to appear in *Lec.Notes.Comp.Sci*, (1990).

[COLE89]: H. Comon and P. Lescanne, Equational problems and disunification, *J.Symbolic.Computation*, 7, pp 371-425, (1989).

[COMO88]: H. Comon, Unification et désunification: Théorie et Applications, Ph.D., Grenoble, (1988).

[COMO89]: H. Comon, Inductive Proofs by Specification Transformations, Rewriting Technics and Applications, *Lec. Notes Comp. Sci.*, 355, (1989).

[COMO90]: H. Comon, Equational formulas in order-sorted algebras,in *Proc. ICALP 90*, Springer-Verlag, (1990).

[COQU90]: J.L. Coquidé, Preuves et Contrôles dans les systèmes de réécriture clos, Automates à piles et Calculs de formes normales, thèse de l'université de Lille 1, (1990).

[COUR89]: B. Courcelle,On recognizable sets and tree automata, Resolution of Equations in Algebraic Structures, *Academic Press*, M.Nivat & H. Ait-Kaci edts, (1989).

[DAUC89]: M.Dauchet, Simulation of Turing machines by a left-linear rewrite-rule, Proc 3rd R.T.A, *Lect. Notes Comput. Sci. 355*, (1989).

[DADE89]: M. Dauchet and A. Deruyver, "VALERIANN":Compilation of Ground Term Rewriting Systems and Applications, Proc 3rd R.T.A, *Lect. Notes Comput. Sci. 355*, (1989).

[DATI85]: M. Dauchet and S.Tison, Decidability of Confluence in Ground Term Rewriting Systems, F.C.T 85, *Lec. Notes Comp. Sci.*, 199, (1985).

[DATI90]: M.Dauchet and S. Tison, The theory of Ground Rewrite System is Decidable, in *Proc 5th I.E.E.E symp. on Logic in Computer Science*, (1990).

[DATI90b]: M.Dauchet and S.Tison, Réduction de la non-linéarité des morphismes d'arbres, Rapport technique IT 196, Université de Lille 1, (1990).

[DEGI89]: A. Deruyver and R. Gilleron, The reachability problem for ground rewrite systems and some extensions, CAAP 89, *Lec. Notes. Comp. Sci*, 351, (1989).

[DEJO90]: N.Dershowitz and J.P. Jouannaud, Rewrite systems, Handbook of Theoretical Computer Science, J.V.Leeuwen editor, North-Holland,.(1990).

[DERS87]: N.Dershowitz, Termination of Rewriting, J. Symbolic Computation 3,(1987).

[DERS89]: N.Dershowitz, Completion and its applications, in Ait-Kaci, M.Nivat eds, *Resolution of Equations in Algebraic Structures*, Vol II: Rewriting Techniques, pp 31-86, (1989).

[DEVI90]: P.Devienne, Weighted graphs, a tool for studying the halting problem and time complexity in logic programming, TCS 75, (1990).

[DHTL87]: M. Dauchet,P. Heuillard, P. Lescanne and S. Tison, Decidability of the Confluence of Ground Term Rewriting Systems, 2nd Symposium on Logic in Computer Science, New-York, IEEE Computer Society Press, (1987).

[ENGE75]: J. Engelfriet, Bottom-up and Top-down Tree Transformations, a Comparison, *Math. Systems Theory*, 9, (1975).

[FAGE84]: F.Fages, Le système KB: manuel de référence: présentation et bibliographie, mise en oeuvre, Report R.G.10.84, Greco de Programmation, Bordeaux, (1984).

[FUVA89]: Z. Fülöp and S. Vàgvölgyi, Ground Term Rewriting rules for the Word Problem of Ground Term Equations, submitted paper, (1989).

[FUVA90]: Z. Fülöp and S. Vàgvölgyi, A characterization of irreducible sets modulo left-linear rewriting systems by tree automata, *Fundamenta Informaticae XIII*, (1990).

[GASC85]: J.H. Gallier and K.M. Schimpf, Parsing Tree Languages using Bottom-up Tree Automata with Tree Pushdown Stores, *J. Computer Science*, (1985).

[GEST84]: F. Gecseg and M. Steinby, Tree automata, Akademiai Kiado, (1984).

[GILL91]: R.Gilleron, Decision problems for term rewriting systems and recognizable tree languages, to appear in Proc. STACS 91, (1991).

[GOGU80]:J.A.Goguen, How to prove inductive hypothesis without induction, *Lec. Notes in Comp. Sci.*, 87, (1980).

[GRS87]: J.H.Gallier, S.Raatz and E.W.Snyder, Theorem proving using rigid E-unification: Equational mating, 2nd Symposium on Logic in Computer Science, IEEE Computer Society Press, pp338-346, (1987).

[GTWW77]: J. Goguen, J.W. Thatcher, E. Wagner and E. Wright, An initial algebra approach to the specification correctness, and implementation of abstract data types, in: *Current Trends in Programming Methodology* Vol. 4 (Prentice-Hall, Englewood Cliffs, NJ), pp 80-149, (1977).

[GUES83]: I. Guessarian, Pushdown Tree Automata, *Math. Systems Theory*, 16, (1983).

[HODO79]: C.M. Hoffmann and M.J. O'Donnell, Interpreter Generation using Tree Pattern Matching, *Proc. of the Sixth ACM Symp. on Principles of Programming Languages*, (1979).

[HODO82]: C.M. Hoffmann and M.J. O'Donnell, Programming with Equations, *TOPLAS*, 4, (1982).

[HUET80]: G. Huet, Confluent Reductions: Abstract properties and applications to Term Rewriting Systems, *J.A.C.M.*, 27, (1980).

[HUHU82]: G.Huet and J.M.Hullot, Proofs by induction in equational theories with constructors, *J. of Comp. and Syst. Sciences*, 25, (1982).

[HULA78]: G.Huet and D.S.Lankford, On the uniform halting problem for term rewriting systems, Rapport Laboria 283, Institut de recherche en Informatique et en Automatique, Le Chesnay, France, (1978).

[HULE79]: G.Huet and J-J. Levy, Call by need Computation in non-ambiguous linear Term Rewriting Systems, *TR359,* I.N.R.I.A., Le Chesnay, France, (1979).

[HUOP80]: G.Huet and D.C. Oppen, Equations and Rewrite Rules: A survey, in R.V.Book, ed., New York, *Academic Press*, Formal Language Theory: Perspectives and Open Problems, (1980).

[JANT88]: M.Jantzen, Confluent string rewriting, EATCS Monographs on Theoretical Computer Science 14, Springer Verlag, (1988).

[JOKI86]: J.P.Jouannaud and H.Kirchner, Completion of a set of rules modulo a set of equations, *SIAM J. Comput.* 15, (1986).

[JOKO86]:J.P.Jouannaud and E.Kounalis, Automatic proofs by induction in equational theories without constructors, in *Proc. 1st IEEE Symp. on Logic in Computer Science*, (1986).

[KANA85]: D.Kapur and P.Narendran, An equational approach to theorem proving in first-order predicate calculus, in *Proc. of the 9th International conference on Artificial Intelligence*, pp 1146-1153, (1985).

[KAZH88]: D.Kapur and H.Zhang, An overview of Rewrite Rule Laboratory (RRL), in *Proc of the 8th Conference on Automated deduction*, pp 559-563, (1988).

[KIRC85]: C.Kirchner, Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles, Thèse de Doctorat d'Etat, Nancy, (1985).

[KIRC85]: H.Kirchner, Preuves par complétion dans les variétés d'algèbres, Thèse de Doctorat d'Etat, Nancy, (1985).

[KNBE70]: D.E.Knuth and P.B.Bendix, Simple word problems in universal algebras, *Computational Problems in Abstract Algebra*, pp263-297, (1970).

[KNO90]: D.Kapur, P.Narendran and F.Otto, On ground confluence of term rewriting systems, *Information and Computation* 86, pp 14-31, (1990).

[KNZ87]: D. Kapur, P. Narendran and H. Zhang, On sufficient completeness and related properties of term rewriting systems, *Acta Informatica*, 24, (1987).

[KOUN90]: E.Kounalis, Testing for inductive (co)-reducibility, *in Proc.CAAP 90*, (1990).

[KOSA82]: S.R.Kosaraju, Decidability of reachability in vector addition systems, *Proc 14th Symposium on theory of Computing*, pp 267-281, (1982).

[KOZE77]: D.Kozen, Complexity of finitely presented algebras, 9th ACM Symposium on theory of computing, Boulder, Colorado, pp 164-177, (1977).

[KRUS60]: J.B.Kruskal, Well-quasi-Ordering, the Tree Theorem and Vazsonyi's conjecture, *Transactions of the American Mathematical Society*, 95, pp 210-225, (1960).

[KUCH90]: G.A.Kucherov, On relationship between term rewriting systems and regular tree languages, to appear, (1990).

[LABA77]: D.S.Lankford and A.M.Ballantyne, Decision procedures for simple equational theories with permutative axioms: complete sets of permutative reductions, *ATP37*, Dept. of Mathematics and Computer Science, Austin, Texas, (1977).

[LABA77]: D.S.Lankford and A.M.Ballantyne, Decision procedures for simple equational theories with commutative-associative axioms: complete sets of commutative-associative reductions, *ATP39*, Dept. of Mathematics and Computer Science, Austin, Texas, (1977).

[LEBE88]: P.Lebègue, Towards the study of logic programming with weighted graphs, Thèse de l'université de Lille, (1988).

[LEGU80]: B.Leguy, Reductions, transformations et classification des grammaires algébriques d'arbres, Thèse de l'université de Lille, (1976).

[LESC83]:P.Lescanne, Computer experiments with the REVE term rewriting system generator, in *Proc. of the 10th ACM Symp. on Principles of Programming Languages*, pp 99-108, (1983).

[MAHE88]: M.J.Maher, Complete axiomatizations of the algebras of finite, rational and infinite trees, *Proc 3rd IEEE Symp. Logic in Computer Science*, (1988).

[MALC71]: A.I.Mal'cev, Axiomatizable classes of locally free algebras of various types, the *Metamathematics of Algebraic systems*, North-Holland, (1971).

[MAYR85]: E.W.Mayr, An algorithm for the general Petri net reachability problem, *Siam J. Comput. 13*, pp 441-460, (1985).

[MUSS80]: D.Musser, Proving inductive properties of abstract data types, in Proc. 7th ACM Symp. Principles of Programming Languages, (1980).

[NAOT90]: P.Narendran and F.Otto, Some results on equational unification, Lec. Notes in Comp. Sci., 449, pp 276-291, (1990).

[NEOP80]: G. Nelson and D.C. Oppen, Fast Decision Procedures Based on Congruence Closure, *J.A.C.M.*, 27, (1980).

[NEWM42]: M.H.A.Newman, On theories with a combinatorial definition of equivalence, *Annals of Mathematics*, 43, pp 223-243, (1942).

[ODON77]: M.J. O'Donnell, Computing in Systems described by Equations, *Lec. Notes in Comp. Sci.*, 58, (1977).

[OYAM86]: M. Oyamaguchi, The reachability Problem for Quasi-ground Term Rewriting Systems, *Journal of Information Processing*, 9-4, (1986).

[OYAM87]: M.Oyamaguchi, The Church-Rosser property for ground term rewriting systems is decidable, *TCS* 49, pp43-79, (1987).

[OYAM90]: M. Oyamaguchi, The reachability and joinability Problems for right-ground Term Rewriting Systems, to appear in *J. Inf. Process*, (1990).

[OYAM90]: M.Oyamaguchi, On the word problem for right-ground term rewriting systems, *Trans of the IEICE*, E73, pp718-723, (1990).

[PEST81]: G.Peterson and M.Stickel, Complete sets of reductions for some equational theories, *J. Assoc. Comput. Mach.* 28, (1981).

[PLAI85]: D.A. Plaisted, Semantic Confluence Tests and Completion Methods, *Information and Control*, 65, (1985).

[PLOT72]: G.Plotkin, Building in equational theories, *Machine Intelligence*, 7, pp 73-90, (1972).

[RABI69]: M.O.Rabin, Decidability of second order theories and automata on infinite trees, *Amer. Math. Soc.*, (1969).

[RABI77]: M.O.Rabin, Decidable theories, *Handbook of Mathematical Logic*, North Holland eds, pp 595-627, (1977).

[RAOU81]: J-C. Raoult, Finiteness results on rewriting systems, *R.A.I.R.O.* Theoretical Informatics 15, (1981).

[RAVU80]: J-C. Raoult and J. Vuillemin, Operational and Semantic Equivalence between Recursive Programs, *J.A.C.M.* , 27, (1980).

[ROSE73]: B.K. Rosen, Tree-manipulating Systems and Church Rosser Theorems, *J.A.C.M.*, 20, (1973).

[SALO88]: K. Salomaa, Deterministic Tree Pushdown Automata and Monadic Tree Rewriting Systems, *Journal of Comput. and Syst. Sci.*, 37, (1988).

[SCHI82]: K.M. Schimpf, A Parsing Method for Context-free Tree Languages, Ph.D., Univ. of Pennsylvania, (1982).

[SCHW85]: S.R.Schwer, Décidabilité de l'algébricité des langages associés aux réseaux de Pétri, Thèse Paris VII, (1985).

[SNYD89]: E.W.Snyder, Efficient ground completion: An O(nlogn) algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations, RTA 89, *Lec. Notes in Comp. Sci.* 355, (1989).

[STIC86]: M.E.Stickel, The Klaus automated deduction system, in *Proc. of the 8th International Conference on Automated deduction*, pp 703-704, (1986).

[THAT67]: J.W.Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, J. Comput. System Sci., 1, (1967).

[THOM90]: W.Thomas, Automata on infinite objects, to appear in Handbook of Theoretical and Computer Science, (1990).

[TISO89]: S. Tison, The Fair Termination is decidable for Ground Systems, Rewriting Technics and Applications, Chapel Hill, *Lec. Notes Comp. Sci.*, 355, (1989).

[TREI90]: R.Treinen, A new method for undecidability proofs of first order theories, technical report TR A-09/90, Universität des Saarlandes, Saarbrücken, (1990).

Titre de la thèse: Reconnaissabilité et fragments décidables en réécriture; automates à piles et calcul de formes normales.

Résumé en français:

La réécriture est, à plusieurs titres (programmation fonctionnelle, logico-fonctionnelle, équationnelle), un paradigme de la programmation. Notre approche se situe en amont de la programmation et a pour but de contribuer à une meilleure connaissance des arbres et de la réécriture dans les arbres. Nous utilisons, en particulier, la théorie des automates d'arbres. Les résultats principaux de notre travail sont les suivants:

1) Nous étudions les problèmes de décidabilité sur les systèmes de réécriture et les forêts reconnaissables. En particulier, nous étudions la propriété (P): Pour toute forêt reconnaissable, l'ensemble des formes normales des termes de F pour S est reconnaissable. Nous prouvons l'indécidabilité de cette propriété.

2) Nous prouvons l' indécidabilité du fragment existentiel de théorie des algèbres de termes clos quotientée par une relation de congruence définie par un ensemble d'équations E, lorsqu'il existe un système de réécriture S complet pour E et satisfaisant (P). Nous prouvons, cependant, la décidabilité pour une classe de formules linéaires closes.

2) Nous montrons que le problème d'accessibilité est décidable pour les systèmes de réécriture clos modulo la commutativité (complexité polynomiale), est indécidable modulo l'associativité et décidable (résultat partiel) dans le cas associatif et commutatif (équivalent au problème d'accessibilité pour les réseaux de Pétri).

3) Nous introduisons une notion qui permet d'éclairer et de généraliser les études antérieures sur les liens entre systèmes de réécriture et automates à piles.

Mots clés: Automates finis, reconnaissabilité, réécriture, accessibilité, termes clos, théories égalitaires, automates à piles.