UNIVERSITE DE NANCY I

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

PREUVES PAR COMPLETION

DANS LES VARIETES D'ALGEBRES

THESE DE DOCTORAT D'ETAT EN INFORMATIQUE
PRESENTEE PAR

Hélène Kirchner

SOUTENUE LE 21 JUIN 1985

DEVANT LA COMMISSION D'EXAMEN

PRESIDENT	M. NIVAT
RAPPORTEURS	M.C. GAUDEL
	J. HSIANG
	J.P. JOUANNAUD
EXAMINATEURS	J.P. FINANCE
	G. HUET
	P. LESCANNE
	G. PLOTKIN

UNIVERSITE DE NANCY I

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

PREUVES PAR COMPLETION

DANS LES VARIETES D'ALGEBRES

THESE DE DOCTORAT D'ETAT EN INFORMATIQUE
PRESENTEE PAR

Hélène Kirchner

SOUTENUE LE 21 JUIN 1985

DEVANT LA COMMISSION D'EXAMEN

PRESIDENT	M. NIVAT
RAPPORTEURS	M.C. GAUDEL
	J. HSIANG
	J.P. JOUANNAUD
EXAMINATEURS	J.P. FINANCE
	G. HUET
	P. LESCANNE
	G. PLOTKIN

A Claude, Florent, Johanne, Franz

La manière de démontrer est double:
l'une se fait par l'analyse ou résolution,
et l'autre par la synthèse ou composition.
L'analyse montre la vraie voie par laquelle
une chose a été méthodiquement inventée, et
fait voir comment les effets dépendent des causes.

Descartes

Remerciements

Je remercie sincèrement tous les membres du jury d'avoir accepté de porter un avis sur cette thèse.

Jean-Pierre Finance témoigne de son intérêt pour mon travail en participant à ce jury et je l'en remercie.

Marie-Claude Gaudel a suivi mes recherches depuis mon entrée au CNRS. Je lui suis gré d'être rapporteur et la remercie de ses critiques positives.

Jieh Hsiang m'a fait bénéficier de ses réflexions pertinentes et je lui suis très reconnaissante d'avoir accepté de faire un rapport, malgré l'obstacle présenté par la différence de langue.

Gérard Huet a par ses travaux suscité une grande partie de mes recherches. Je le remercie d'apporter à ce jury ses compétences mondialement reconnues.

Jean-Pierre Jouannaud a dirigé ce travail. Notre collaboration dans le domaine de la réécriture équationnelle a permis l'élaboration de résultats complexes. J'ai aussi apprécié ses conseils tout au long de la réalisation de cette thèse. Je lui adresse ici mes sincères remerciements.

Pierre Lescanne m'a fortement et amicalement encouragée à développer à la fois résultats théoriques et implantations. Je le remercie également des critiques et des idées dont il m'a fait bénéficier.

Maurice Nivat s'est intéressé à ce travail et me fait l'honneur de présider ce jury. Je lui suis extrêmement reconnaissante de son soutien.

Gordon Plotkin est venu d'Edimbourg pour participer à ce jury. Je le remercie sincèrement d'y apporter ses grandes compétences.

Cette thèse a été réalisée au sein de l'équipe EURECA du CRIN et je tiens à remercier tous ceux qui de près ou de loin, ont facilité cette réalisation.

Je remercie plus particulièrement Jean-Luc Remy pour avoir relu attentivement ce travail et m'avoir prodigué encouragements et critiques amicales. Jalel Mzali a participé largement au développement de REVEUR-3 et nous continuerons à collaborer pour en assurer la maintenance. De fructueuses conversations avec Emmanuel Kounalis ont permis de faire progresser mes réflexions dans le domaine de la complétion inductive. Pierre Réty a contribué aux résultats présentés sur la surréduction. Je remercie enfin Françoise Bellegarde, Isabelle Gnaedig, Pierre Marchand, tous les membres de l'équipe et plus généralement du laboratoire, pour leur amical soutien.

Mes derniers remerciements mais non les moindres vont à Claude, autant pour sa collaboration scientifique théorique, que pour son soutien moral et également pour son aide technique précieuse pour la réalisation finale de ce document.

Table des matières

INTRODUCTION

INTRODUCTION

L'objet de cette thèse est l'étude des systèmes de réécriture et de leur procédures de complétion. Elle fournit en cela des outils puissants pour la programmation logique et la démonstration automatique.

Un certain nombre de résultats sur les systèmes de réécriture sont connus, la plupart depuis longtemps. En 1967, Knuth et Bendix, s'appuyant sur les travaux précurseurs de Evans, ont mis en évidence comment réécriture et complétion permettent de résoudre le problème du mot en algèbre universelle, c'est-à-dire de décider si une équation est valide dans une classe d'algèbres [Knuth1970].

- Les systèmes de réécriture sont un outil permettant de prouver dans une théorie équationnelle, des égalités universellement quantifiées du type $t_1=t_2$, où t_1 et t_2 sont des termes du premier ordre avec variables. Un système de réécriture est un ensemble de règles, ou paires de termes orientées, notées g->d. L'opération de réécriture, appliquée à un terme, consiste à remplacer, dans ce terme, une instance d'un membre gauche de règle g par la même instance de son membre droit d. La preuve d'un théorème équationnel $t_1=t_2$ dans la théorie, consiste à vérifier que t_1 et t_2 se réécrivent en un même terme, à l'aide de l'ensemble des règles. Pour cela le système de réécriture doit satisfaire deux propriétés: il doit être bien fondé (ou noethérien), pour assurer qu'un terme ne peut pas se réécrire une infinité de fois, et confluent, pour que le résultat d'un calcul ne dépende pas du chemin suivi.

- La complétion des systèmes de réécriture a été introduite par Knuth

et Bendix dans le but de déduire d'un ensemble d'axiomes équationnels, un système de réécriture confluent et noethérien équivalent, c'est-à-dire ayant la même puissance de déduction. Les deux mécanismes de base sont la normalisation et la superposition. La normalisation d'un terme consiste à calculer une forme irréductible, c'est-à-dire un terme dérivé par réécriture et que l'on ne peut plus réécrire. La superposition est une opération définie sur les règles de réécriture: superposer une règle $g_1 \to d_1$ sur une règle $g_2 \to d_2$ consiste à unifier g_1 et un sous-terme de g_2, avec une substitution σ, puis à réécrire le terme $\sigma(g_2)$ en deux termes qui constituent une paire critique. L'intérêt des paires critiques est qu'elles constituent en un certain sens, les cas les plus simples où un même terme peut se réécrire de deux façons différentes. De plus, assurer la propriété de confluence sur les paires critiques suffit à l'assurer pour des termes quelconques.

La procédure de complétion consiste à orienter les équations à l'aide d'un ordre bien fondé sur les termes, puis à calculer les paires critiques entre les règles obtenues. Si une paire critique se normalise en deux termes différents, la propriété de confluence est mise en défaut et il faut alors compléter le système. La paire critique normalisée devient une nouvelle paire à orienter et le processus s'applique récursivement.

Il est remarquable que la notion de superposition soit aussi à la base de deux autres approches dans des domaines très différents de l'algèbre universelle: celui de la théorie des idéaux de polynômes, développée autour de Buchberger [Buchberger1976], ainsi que la preuve automatique de théorèmes par résolution, qui trouve son origine en 1965 dans les travaux de Robinson[Robinson1965]. Ces approches ont donné lieu depuis à de

nombreuses recherches, dont certaines visent à unifier ces différents domaines.

Parmi les nombreuses applications de la procédure de complétion, cette thèse en privilégie trois.

- La procédure de complétion a été adaptée pour prouver la validité d'un théorème dans l'ensemble des termes sans variables vérifiant les axiomes de la théorie. Le raisonnement équationnel n'est pas toujours suffisant dans ce cas et la preuve d'un tel théorème peut par exemple nécessiter une récurrence structurelle sur les termes sans variables. De telles preuves sont cependant encore réalisées automatiquement par cette adaptation de la procédure de complétion appelée complétion inductive.

- Les mêmes mécanismes de superposition et normalisation sont à la base d'une méthode générale permettant de trouver de façon automatique, des ensembles complets d'unificateurs, dans une théorie équationnelle à laquelle on peut associer un système de réécriture confluent et noethérien. Fay en 1979 a proposé le concept de superréduction (narrowing en anglais) combinant superposition et normalisation, ainsi qu'une procédure implantant cette méthode. Ce concept a été par ailleurs utilisé pour la preuve de théorèmes par résolution dans d'autres travaux.

- Une autre application importante des systèmes de réécriture est de fournir un interprète de langages de programmation qui combinent les avantages de la programmation fonctionnelle et ceux de la programmation logique. En effet les programmes peuvent être écrits comme un ensemble d'équations entre formules ou d'équivalences entre des propositions logiques, puis exécutés à l'aide d'un prouveur de théorèmes spécifique qui

dérive des conséquences à partir des formules données, jusqu'à ce que les valeurs désirées soient obtenues. Cette dernière forme de calcul est similaire à celle utilisée en PROLOG. Le mécanisme de base est encore celui de superposition, la normalisation permettant d'améliorer le processus.

Ces différents résultats conduisent à dégager le concept de preuve par complétion. Une **preuve par complétion** est réalisée à l'aide de deux règles d'inférence qui sont la superposition et la normalisation. A partir d'un ensemble d'équations, ces deux règles permettent d'engendrer soit des conséquences équationnelles (dans la procédure de complétion standard), soit des conséquences inductives (dans la procédure de complétion inductive), soit des systèmes d'équations équivalents (dans la procédure de superréduction). Les résultats obtenus dans chaque cas peuvent être vus comme des résultats de complétude de ces règles d'inférence pour le problème considéré. Cependant, la puissance et la généralité des preuves par complétion sont contrebalancées par le fait qu'elles peuvent ne pas terminer ou au contraire s'arrêter en échec sans que l'on puisse conclure.

Le but de cette thèse est d'étendre le champ d'application de ces preuves par complétion, ceci en poursuivant deux objectifs différents.

● Le premier objectif est l'accroissement de la puissance de la procédure de complétion, afin qu'elle puisse s'appliquer à une classe plus large de théories. Les recherches ont été entreprises dans deux directions, chacune d'elles correspondant à un problème soulevé par les limites de la procédure de Knuth et Bendix.

→ Le premier problème est celui de l'arrêt de la procédure sur une

équation non orientable. C'est la motivation de l'introduction des systèmes de réécriture équationnelle. Ils permettent de prendre en compte des équations qui, si elles étaient orientées en règles, provoqueraient des chaines infinies de réécriture. La présentation de la réécriture équationnelle sur les termes donnée ici, unifie toutes les définitions proposées jusqu'à présent. De même la procédure de **complétion équationnelle** décrite généralise toutes les précédentes et ouvre la voie à de nouvelles applications.

→ Le deuxième problème est de proposer des solutions lorsque la procédure de complétion ne termine pas en succès ou en échec, mais diverge en engendrant des familles infinies de règles. Il est alors souvent possible de schématiser chaque famille infinie par une **méta-règle**. Le but poursuivi est de donner des outils pour assurer que les méta-règles ont la même puissance de déduction que l'ensemble infini de règles théoriquement engendré.

● Le deuxième objectif est d'étudier les types de problèmes auxquels s'appliquent ces preuves par complétion. Deux d'entre eux sont étudiées dans cette thèse et un troisième y est évoqué.

→ La complétion des systèmes de réécriture est connue pour être un outil de certification dans les types abstraits algébriques. Elle permet de faire des preuves par consistance. Rappelons en brièvement le principe: pour prouver qu'une équation e est valide dans l'algèbre initiale d'une spécification définie par un ensemble d'équations A, il s'agit de prouver que l'adjonction de e ne modifie pas la congruence engendrée par A sur les termes de l'algèbre initiale. Nous présentons et prouvons une version de la procédure de complétion équationnelle, adaptée aux preuves par consistance

dans des spécifications construites de façon structurée. Cette procédure de **complétion inductive hiérarchique** exploite la construction de la spécification pour structurer ses preuves. Elle complète ainsi la spécification par des conséquences inductives qui apparaissent comme des lemmes dans les sous-spécifications, permettant de prouver le théorème donné. Cette procédure est aussi utilisée pour valider l'ajout dans une spécification hiérarchique d'un ou plusieurs nouveaux opérateurs et des axiomes les définissant.

→ La procédure de complétion équationnelle nécessite l'existence pour la théorie E, d'un algorithme de filtrage et d'un algorithme d'unification complet (i.e. retournant un ensemble générateur de l'ensemble des unificateurs). Ce problème a motivé la description et l'étude d'une méthode générale et incrémentale de construction d'algorithmes d'unification dans les théories équationnelles. La méthode de Fay, basée sur la superréduction est généralisée aux théories possédant un système de réécriture équationnelle (R,E) associé. La technique obtenue est puissante, puisqu'elle permet de construire des algorithmes d'unification de facon incrémentale: elle utilise un algorithme complet d'unification dans la théorie E pour construire un algorithme complet dans la théorie E augmentée des règles de R. Malheureusement elle présente aussi le sérieux inconvénient de ne pas terminer dans de nombreux cas. Cela conduit à étudier des améliorations de la **procédure de superréduction**. Celles qui sont proposées consistent à

- se limiter à certaines superpositions,
- éliminer certaines branches dans l'arbre de recherche des solutions,
- détecter les bouclages,

- schématiser certaines familles de superréductions.

La procédure de superréduction entre dans le cadre des preuves par complétion, puisqu'elle permet, quand elle termine, de prouver la satisfiabilité d'une équation. Elle est présentée comme une **restriction de la complétion**, en ce sens qu'on peut l'implanter comme une procédure de complétion qui se limite à certaines superpositions. Cette implantation est décrite et prouvée.

→ Enfin les preuves par complétion trouvent également des perspectives d'application en **programmation logique**. La programmation logique est née de l'idée d'utiliser la logique des prédicats du premier ordre comme un language de programmation. Le terme "langage de programmation logique" désigne tout langage possédant une sémantique basée sur un système logique, que ce soit la logique des prédicats du premier ordre, la logique équationnelle, la logique des clauses de Horn ou la logique des clauses de Horn avec égalité. En logique équationnelle, la superréduction est une restriction de la complétion analogue à la résolution linéaire qui permet de satisfaire des requêtes exprimées à l'aide d'équations. La conclusion évoque en quoi les preuves par complétion constituent des outils intéressants dans un language de programmation logique.

CHAPITRE 1:

RAPPELS D'ALGEBRE UNIVERSELLE

CHAPITRE 1: RAPPELS D'ALGEBRE UNIVERSELLE

Nous donnons dans ce chapitre les bases théoriques préliminaires aux thèmes abordés dans cette thèse. Nous y introduisons les termes du premier ordre et donnons divers résultats classiques obtenus en munissant cet ensemble de la structure d'algèbre. Nous nous sommes autorisés de nombreux emprunts aux travaux de G. Huet et D.Oppen [Huet1980a], J.M.Hullot [Hullot1980a], J.Goguen et J.Meseguer [Meseguer1983] et renvoyons à ces références le lecteur intéressé par les preuves des résultats cités. Nous citons également plusieurs résultats dus à Birkhoff qui peuvent être trouvés dans [Birkhoff1935]. Ce chapitre a également pour but de fixer les notations et la terminologie utilisées dans cette thèse.

1. Algèbres homogènes

Nous définissons dans un premier temps les algèbres homogènes. C'est le cadre dans lequel se place la plus grande partie de cette thèse.

1.1. F-algèbres ou algèbres de type F

Soit F un ensemble dénombrable de symboles de fonctions; à chaque symbole f de F est associé un entier appelé arité de f et noté ar(f). On partitionne F en une réunion de sous-ensembles F_i, où F_i est l'ensemble des symboles de fonctions d'arité i. Par convention, les symboles de fonctions d'arité 0 sont appelés constantes et notés a,b,c... Ceux d'arité non nulle sont désignés par les lettres f,g,h...

Définition 1 : Une **F-algèbre** est un couple $M=(D_M,F_M)$ où D_M est un ensemble non vide appelé domaine de M, et F_M une famille d'opérations sur D_M indexées par l'ensemble des symboles de fonctions de F. Il existe une application de F dans F_M qui à f associe f_M. Si ar(f)=k, f_M est une

application de D_M^k dans D_M. On notera $F_M = \{f_M \mid f \text{ appartient à } F\}$. ○

Exemple 1 : Soit F = {0, S, +}. Choisissons pour domaine l'ensemble **N** des entiers naturels et pour opérations: 0_N qui est l'entier 0, S_N qui est la fonction successeur, $+_N$ qui est l'addition usuelle. Le couple $(N,\{0_N,S_N,+_N\})$ est une F-algèbre.

1.2. Variétés d'algèbres

Définissons tout d'abord un certain nombre d'opérations sur les algèbres. Celles qui respectent la structure de F-algèbre sont les homomorphismes.

Définition 2 : Soient $M=(D_M,F_M)$ et $N=(D_N,F_N)$ deux F-algèbres. Un **homomorphisme** h de M dans N est une application de D_M dans D_N telle que pour tout symbole de fonction f d'arité n dans F et pour tous éléments $d_1, d_2, \ldots, d_n$ dans D_M, on ait :

$$h(f_M(d_1,d_2,\ldots,d_n)) = f_N(h(d_1),h(d_2),\ldots,h(d_n)).$$

Un homomorphisme d'une F-algèbre dans elle même est appelé **endomorphisme.** Si de plus il est bijectif, il est appelé **isomorphisme.**

Une algèbre $N=(D_N,F_N)$ est une **sous-algèbre** de $M=(D_M,F_M)$ si et seulement si D_N est inclus dans D_M et si pour tout f de F, la restriction de f_M à D_N coincide avec f_N.

Le **produit** d'une famille $(M_i=(D_{M_i},F_{M_i})$ de F-algèbres est l'algèbre $M=(D_M,F_M)$ telle que

- D_M est le produit ensembliste des D_{M_i}
- pour tout f de F d'arité k, pour tous $a_1,\ldots,a_k$ dans D_M

$$\pi_i(f_M(a_1,\ldots,a_k)) = f_{M_i}(\pi_i(a_1),\ldots,\pi_i(a_k))$$

où π_i est l'application ieme projection. ○

Définition 3 : Une classe K non vide d'algèbres est une **variété** si et seulement si elle est fermée pour les opérations de sous-algèbres, image homomorphe et produit. ○

1.3. Algèbre libre

Définition 4 : Soit K une classe quelconque de F-algèbres et X un ensemble. On appelle **F-algèbre K-libre engendrée par X** (on dit aussi sur X) toute F-algèbre $M=(D_M,F_M)$ telle que

1) M appartient à K

2) D_M contient l'ensemble X

3) pour toute F-algèbre $N=(D_N,F_N)$ de K et toute application h_0 de X dans D_N, il existe un unique homomorphisme h de M dans N dont la restriction à X soit égale à h_0. ○

Si M_1 et M_2 sont deux F-algèbres K-libres sur X, il est facile de montrer qu'il existe un isomorphisme unique entre M_1 et M_2 qui est l'identité sur X. On peut donc sans ambiguité parler de la F-algèbre K-libre engendrée par X.

Théorème 1 : (Birkhoff)

Si K est une classe de F-algèbres stable par produit et sous-algèbre, l'algèbre K-libre existe pour tout ensemble X.

Notation : Par la suite, les éléments de X sont appelés variables et notés x,y,z...

Le résultat suivant, cas particulier du précédent théorème de Birkhoff, nous permet de définir l'ensemble des termes.

Théorème 2 : Soit K la classe de toutes les F-algèbres pour F fixé et X un ensemble de variables. On supposera toujours que soit X soit l'ensemble des constantes est non vide. Alors la F-algèbre K-libre (ou plus simplement la F-algèbre libre) engendrée par X existe. Elle est notée T(F,X).

Définition 5 : Dans les conditions du théorème précédent, on appelle ensemble des termes du premier ordre, ou plus simplement **termes**, les éléments de la F-algèbre libre engendrée par X. ○

Exemple 2 : Si K est l'ensemble des groupes, on obtient la définition du groupe libre engendré par un ensemble X. ▽

Notation : Les termes seront désignés dans la suite par les lettres t,t',..,g, d,..

Cette définition des termes étant particulièrement peu explicite et peu parlante à notre intuition, nous allons en donner une représentation sous forme d'arbres étiquetés.

Soit $\mathbf{N}_+$ l'ensemble des entiers strictement positifs , $\mathbf{N}_+^*$ le monoide libre engendré, ε le mot vide et . l'opération de concaténation (parfois omise quand il n'y a pas d'ambiguité).

Définition 6 : Soit une application t de $\mathbf{N}_+^*$ dans FUX et dom(t) son domaine de définition, c'est-à-dire l'ensemble des u appartenant à $\mathbf{N}_+^*$ tels que t(u) soit défini. Alors t est un arbre sur FUX si et seulement si

1) ε est élément de dom(t)

2) dom(t) est clos par préfixe i.e.: u appartient à dom(t) si u.u' appartient à dom(t).

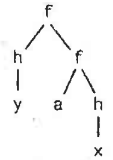
3) pour tout u dans dom(t), u.i appartient à dom(t) si et seulement si i est compris entre 1 et l'arité de t(u). ○

Notation : Le domaine d'un arbre t est aussi appelé ensemble des **occurrences** de t. Il sera noté dom(t). Les éléments de dom(t) sont les noeuds de l'arbre, le noeud ε est la racine (ou sommet) et u.i est le i-ème fils du noeud u.

Exemple 3 : Soit F = {f, g, h, a} avec ar(f) = 2, ar(g) = ar(h) = 1 et ar(a) = 0 et X = {x,y}. L'application t définie sur {ε,1,2,11,21,22,221} par:

t(ε)=f , t(1)=h , t(2)=f , t(11)=y , t(21)=a , t(22)=h , t(221)=x

est un arbre que l'on représentera graphiquement ainsi:



et qui correspond au terme bien formé f(h(y),f(a,h(x))). ▽

Des occurrences sont disjointes si elles ne sont pas préfixes l'une de l'autre:

Définition 7 : Deux occurrences u et u' sont **disjointes** si et seulement si il n'existe pas de u" dans $\mathbf{N}_+^*$ tels que u=u'.u" ou u'=u.u". ○

Nous désignerons aussi par T(F,X) l'ensemble des arbres construits sur l'ensemble des symboles F et des variables X. Cela se justifie par le lemme suivant:

Lemme 1 : L'ensemble des arbres construits sur FUX peut être muni d'une structure de F-algèbre, et c'est la F-algèbre libre engendrée par X.

Nous parlerons donc maintenant indifférement de termes ou d'arbres.

Outre les opérations de la F-algèbre, nous utiliserons deux opérations sur les termes qui sont l'opération de sous-terme et le remplacement d'un sous-terme par un terme.

Définition 8 : Soit t un terme et u un élément de dom(t). Le **sous-terme** de t à l'occurrence u, noté $t_{|u}$ est le terme t' défini par t'(u')=t(u.u') pour tout u' appartenant à N_+^*. ○

Définition 9 : Soient t et t' deux termes et u une occurrence de t. L'opération de **remplacement** dans t **du sous-terme** $t_{|u}$ par t', consiste à définir un nouveau terme t" noté t[u←t'] tel que

$dom(t") = dom(t) - dom(t_{|u}) + u.dom(t')$

$t"(u') = t(u')$ si u' appartient à $dom(t)-dom(t_{|u})$

$= t'(u")$ si $u'=u.u"$ ○

Définition 10 : L'ensemble V(t) des **variables d'un terme** t est défini inductivement par:

- si t est une constante alors $V(t) = \emptyset$
- si t est une variable x alors $V(t) = \{x\}$
- si $t = f(t_1,\ldots,t_k)$ alors $V(t) = V(t_1) \cup \ldots \cup V(t_k)$. ○

Notation : Nous désignerons par G(t) l'ensemble des occurrences de t ayant une image dans F, c'est-à-dire l'ensemble des occurrences de non variables.

Exemple 4 : si $t = f(f(f(x,a),g(x)),g(y))$, $V(t) = \{x, y\}$ et $G(t) = \{\varepsilon,1,11,112,12,2\}$. ▽

Les endomorphismes de l'ensemble des termes sont les substitutions.

Définition 11 : Une **substitution** est une application σ de X dans T(F,X). Le domaine de σ, noté D(σ), est le sous-ensemble de X défini par

$$D(\sigma) = \{x \mid \sigma(x) \neq x\}$$

L'ensemble des variables introduites par σ est noté I(σ), il est défini par:

$$I(\sigma) = \bigcup_{x \in D(\sigma)} V(\sigma(x))$$

○

Le caractère libre de la F-algèbre permet de dire qu'une telle application se prolonge de facon unique en un endomorphisme de T(F,X). Il vérifiera donc pour tout f d'arité k de F, pour tous $t_1,\ldots,t_k$ de T(F,X) : $\sigma(f(t_1,\ldots,t_k)) = f(\sigma(t_1),\ldots,\sigma(t_k))$.

Notation : Les substitutions seront désormais désignées par les lettres α, β, γ,... et représentées par leurs graphes, c'est-à-dire par des ensembles de couples $\{(x,\sigma(x))\}$ pour x dans le domaine de σ, ou bien sous la forme $(x \leftarrow \sigma(x))$.

Définition 12 : La composée de deux substitutions α et β est la substitution notée α.β et définie pour toute variable x par

$$\alpha.\beta(x) = \alpha(\beta(x)) \quad ○$$

Définition 13 : La restriction d'une substitution σ à un sous-ensemble V de X est notée σ_V et définie par ;

$\sigma_V(x) = \sigma(x)$ si x appartient à V

$= x$ sinon.

1.4. Algèbre initiale

On a vu que l'algèbre libre sur un ensemble de variables X est unique à un isomorphisme près. Mais on peut d'autre part faire varier X et prendre en particulier X = ø. On obtient alors la notion d'algèbre initiale.

Définition 14 : Si K est la classe de toutes les F-algèbres, l'algèbre K-libre sur l'ensemble ø est appelée **algèbre initiale** et notée I(F).
L'ensemble des termes T(F,ø) sans variables est appelé ensemble des termes clos et noté T(F). ○

1.5. Equations et variété équationnelle d'algèbres

Définition 15 : Une **équation** est une paire de termes $\{t_1,t_2\}$ notée $(t_1=t_2)$. ○

Définition 16 : Une F-algèbre $M = (D_M,F_M)$ **satisfait** l'équation $(t_1 = t_2)$ si et seulement si pour tout homomorphisme h de T(F,X) dans D_M, $h(t_1) = h(t_2)$. On dit aussi que l'équation $(t_1 = t_2)$ est **valide** dans M, et l'on note

$$M \models (t_1 = t_2).$$

○

Pour déterminer h il suffit de se donner sa restriction h_0 aux variables de X. De plus, comme on ne s'intéresse qu'aux termes t_1 et t_2, il suffit de se donner h_0 sur $V(t_1) \cup V(t_2)$.

Définition 17 : Une **variété équationnelle** de F-algèbres est une classe K de F-algèbres pour laquelle il existe un ensemble d'équations A tel que K soit la classe de toutes les algèbres validant les équations de A. K est aussi appelée **classe des modèles de A** et notée M(A). ○

G.Birkhoff a prouvé que si une classe de F-algèbres est une variété, alors la F-algèbre M(A)-libre engendrée par X existe pour tout ensemble X. Nous allons maintenant montrer comment construire cette F-algèbre M(A)-libre à partir de l'ensemble des termes et de la congruence engendrée par A.

Définition 18 : Soit $M=(D_M,F_M)$ une F-algèbre. Une relation R sur D_M est une **congruence** sur M si et seulement si pour tout symbole de fonction f dans F d'arité n et pour tous éléments $t_1,\ldots, t_n, t'_1, \ldots, t'_n$ dans D_M:

$$t_1 \; R \; t'_1 ,\ldots, t_n \; R \; t'_n \text{ implique } f(t_1,\ldots,t_n) \; R \; f(t'_1,\ldots,t'_n).$$

Les opérations de F_M passent au quotient et on peut ainsi définir une **algèbre quotient** dont le domaine est l'ensemble quotient D_M/R et dont les opérations s'identifient à celle de F_M. ○

1.6. Problème de validité dans une théorie équationnelle

Etant donné un ensemble d'équations A, le problème consistant à décider si une équation $(t_1=t_2)$ est valide dans toute F-algèbre de M(A) est appelé **problème de validité** dans M(A). Nous allons voir que ce problème peut s'énoncer en termes de théorie équationnelle.

Définition 19 : Soit A un ensemble d'équations. La plus petite congruence sur T(F,X) contenant toutes les paires $\{\sigma(t_1),\sigma(t_2)\}$, où $(t_1=t_2)$ est une équation quelconque de A et σ une substitution, est appelée **A-égalité** ou congruence engendrée par A et notée $\sim^A$.
Deux termes tels que $t_1 \sim^A t_2$ sont dit A-égaux.
On notera $t_1 \vdash_A t_2$ une étape de A-égalité, appelée aussi un **remplacement en une étape.**

On parlera aussi de la **théorie équationnelle** engendrée par A pour désigner l'algèbre quotient $(T(F,X)/\sim^A, F)$ et par abus de langage la congruence elle-même. Lorsque A est l'ensemble vide, on parlera parfois de la théorie vide. ○

Théorème 3 : La F-algèbre M(A)-libre sur X est l'algèbre quotient $(T(F,X)/\sim^A,F)$.

Le théorème suivant est le fondement de la logique équationnelle. Il ramène le problème sémantique de la validité d'une équation dans une variété équationnelle au problème syntaxique de la A-égalité, c'est-à-dire à un raisonnement équationnel.

Théorème 4 : Théorème de complétude du raisonnement équationnel (Birkhoff) Etant donné un ensemble d'équations A, une équation $(t_1 = t_2)$ est valide dans la variété M(A) si et seulement si $t_1 \sim^A t_2$.

Reprenons le cas particulier important où X est l'ensemble vide.

Définition 20 : Si A est un ensemble d'équations, l'algèbre M(A)-libre sur ø est l'**algèbre initiale de la variété équationnelle** engendrée par A. On la notera I(F,A). Son domaine est l'ensemble quotient $T(F)/\sim^A$. ○

En appliquant la définition 16 à l'algèbre initiale, on obtient:

Définition 21 : Une équation est **valide dans l'algèbre initiale** I(F) (resp.I(F,A)) si et seulement si pour toute substitution close σ (c'est-à-dire à valeurs dans T(F)), $\sigma(t) = \sigma(t')$ (resp. $\sigma(t) \sim^A \sigma(t')$. ○

1.7. Satisfiabilité dans une théorie équationnelle

Nous allons nous poser maintenant le problème dual du problème de validité: étant donnés deux termes t_1 et t_2 dans T(F,X) et une algèbre M, existe-t-il un homomorphisme h de T(F,X) dans D_M vérifiant $h(t_1)=h(t_2)$? On dira alors que l'équation $(t_1=t_2)$ est **satisfiable** dans l'algèbre M. Comme précédemment il suffit de trouver une application h_0 de $V(t_1) \cup V(t_2)$ dans D_M vérifiant la condition requise. De plus, on se restreindra ici au cas où M est l'algèbre M(A)-libre sur X, A étant un ensemble d'équations.

Pour pouvoir décrire l'ensemble des homomorphismes répondant à la question, nous allons définir un système générateur minimal de cet ensemble; cela nécessite l'utilisation d'un préordre sur les termes et sur les substitutions, obtenu en utilisant la notion de filtrage.

1.7.1. Filtre et préordre de filtrage

Définition 22 : La relation $\leq_A$ est définie sur les termes de T(F,X) par: $t_1 \leq_A t_2$ si et seulement si il existe une substitution σ telle que $\sigma(t_1) \sim^A t_2$. Si σ existe, σ restreinte à $V(t_1)$ est appelé **A-filtre** ou **filtre modulo A** de t_1 vers t_2. ○

La relation $\leq_A$ est un préordre sur T(F,X).

Dans le cas où A est vide, on notera le préordre de filtrage par $\leq_\emptyset$. Si $t_1 \leq_\emptyset t_2$ on dira que t_1 généralise t_2. Dans ce cas, le filtre de t_1 vers t_2, s'il existe, est unique et peut être trouvé par un algorithme récursif simple [Huet1976].

Un préordre et une relation d'équivalence sur les substitutions se déduisent de la relation $\leq_A$.

Définition 23 : Soit V un ensemble de variables, σ et μ deux substitutions.

$$\sigma \leq_A \mu \ [V]$$

si et seulement si il existe une substitution ρ telle que pour tout x appartenant à V, $\rho(\sigma(x)) \sim^A \mu(x)$.

On note $\sigma \equiv_A \mu \ [V]$ si et seulement si $\sigma \leq_A \mu$ et $\sigma \leq_A \mu$ sur V. ○

Exemple 5 : Dans le cas où A est vide et où V = {x},

$\sigma = \{(x \leftarrow f(a,f(y,b)))\} \leq \mu = \{(x \leftarrow f(a,f(a,b)))\}$ en prenant $\rho = \{(y \leftarrow a)\}$ et

$\sigma \equiv_{\emptyset} \alpha = \{(x \leftarrow f(a,f(a,z)))\}$.

Si A = {f(a,f(a,x)) = x} et si V = {x},

$\sigma = \{(x \leftarrow f(a,f(y,b)))\} \leq_A \mu = \{(x \leftarrow b)\}$ en prenant $\rho = \{(y \leftarrow a)\}$. ∇

Contrairement à ce qui se passe dans la théorie vide, il n'y a pas unicité du A-filtre d'un terme t_1 vers un terme t_2, quand il existe. On cherche alors à engendrer cet ensemble, noté $F_A(t_1,t_2)$ éventuellement infini, par une partie génératrice que nous définissons maintenant.

Définition 24 : Soient t_1 et t_2 deux termes et $V = V(t_1)$. Un **ensemble complet de A-filtres** t_1 vers t_2 est un ensemble de substitutions S tel que

1) pour toute $\sigma \in S$, $D(\sigma)$ est inclus dans V,

2) S est inclus dans $F_A(t_1,t_2)$,

3) pour toute $\alpha \in F_A(t_1,t_2)$, il existe $\sigma \in S$ tel que $\sigma \leq_A \alpha \ [V]$

On dira que S est un ensemble complet **minimal** de A-filtres de t_1 vers t_2 s'il vérifie de plus la condition suivante:

4) pour toutes substitutions $\sigma, \sigma' \in S$, si $\sigma \leq_A \sigma' \ [V]$, alors $\sigma = \sigma'$. ○

1.7.2. Unificateurs dans une théorie équationnelle

Le problème de la satisfiabilité d'une équation est celui de la recherche de ses solutions.

Définition 25 : Un **A-unificateur** ou **unificateur modulo A** de deux termes t_1 et t_2 est une substitution σ telle que $\sigma(t_1) \sim^A \sigma(t_2)$. Si σ existe, t_1 et t_2 sont dits A-unifiables et σ est une A-solution de l'équation $(t_1 = t_2)$. On note $SU_A(t_1,t_2)$ l'ensemble des A-unificateurs de t_1 et t_2. ○

Exemple 6 : Soit A l'ensemble d'équations {x+x = x}. Les termes (a+y) et z admettent pour A-unificateurs les substitutions

$\sigma = (z \leftarrow a+y)$ et $\sigma' = (y \leftarrow a)(z \leftarrow a)$. ∇

1.7.3. Ensemble minimal complet de A-unificateurs

La relation de préordre $\leq_A$ permet de définir des parties génératrices de l'ensemble $SU_A(t_1,t_2)$, c'est-à-dire un ensemble éventuellement minimal de A-unificateurs à partir desquels peuvent se déduire tous les autres. Plus précisément:

Définition 26 : Soient t_1 et t_2 deux termes, $V = V(t_1) \cup V(t_2)$ et W un ensemble fini de variables contenant V. Un **ensemble complet de A-unificateurs** de t_1 et t_2 en dehors de W est un ensemble de substitutions S tel que:

1) pour toute $\sigma \in S$, $D(\sigma)$ est inclus dans V et $I(\sigma) \cap W = \emptyset$

2) S est inclus dans $SU_A(t_1,t_2)$

3) pour tout A-unificateur α, il existe σ dans S tel que $\sigma \leq_A \alpha \ [V]$.

On dira que S est un ensemble complet **minimal** de A-unificateurs de t_1 et t_2 s'il vérifie de plus la condition suivante:

4) pour toutes σ et ρ appartenant à S, $\sigma \leq_A \rho$ [V] implique $\sigma = \rho$. ○

La condition 2 traduit la cohérence, la condition 3 la complétude. La condition 1 n'est qu'une condition technique. Elle permet d'éviter les conflits entre les "anciennes" variables et les variables introduites par σ.

Notation : Un ensemble complet de A-unificateurs de t_1 et t_2 sera noté $ECU_A(t_1,t_2)$. Si de plus il est minimal, on le notera $ECUM_A(t_1,t_2)$.

Remarque : Pour deux termes donnés, un ECUM, quand il existe, n'est pas unique (voir par exemple [Fages1983a].)

Rappelons quelques résultats. Dans le cas où A est réduit à l'axiome d'associativité pour un symbole de fonction f et F={f}, [Plotkin1972] décrit un algorithme permettant d'engendrer un ensemble complet minimal de A-unificateurs pour tous termes t et t'. [Stickel1981] donne un algorithme construisant un ensemble minimal complet de A-unificateurs dans le cas où A est constitué des axiomes de commutativité et d'associativité pour un seul symbole de fonction f et pour F={f}. [Siekmann1978] traite le cas de F quelconque, A étant composé des axiomes de commutativité pour un nombre fini de symboles de fonctions. Dans d'autres cas, par exemple pour la structure de groupe abélien [Lankford1979], on sait calculer un ensemble complet et fini de A-unificateurs. Nous développerons dans un chapitre ultérieur une méthode générale, étudiée précédemment dans [Lankford1979a,Fay1979,Hullot1980], pour construire des ensembles complets de A-unificateurs dans certaines théories équationnelles. En général ces ensembles ne seront pas minimaux.

Le cas particulier où il n'y a pas d'équations sur les termes est particulièrement important: en effet dans ce cas, soit les deux termes donnés ne sont pas unifiables, soit il existe un ensemble complet minimal d'unificateurs réduit à un seul élément appelé unificateur minimum ou quelquefois principal. De nombreux algorithmes ont été écrits pour trouver cet unificateur minimum. Citons entre autres [Robinson1965,Huet1976,Martelli1982,Paterson1978,Corbin1983].

L'état de l'art sur le sujet de l'unification dans les théories équationnelles est présenté dans [Kirchner1985].

2. Algèbres hétérogènes

L'étude des types abstraits algébriques nécessite l'introduction d'algèbres hétérogènes, c'est-à-dire d'algèbres dont le domaine est une réunion d'ensembles distincts [Birkhoff1970].

2.1. Définition des algèbres hétérogènes

Soit S un ensemble de noms de domaines appelés sortes. On lui associe un language S^* qui est l'ensemble des mots construits sur S. Un mot $s_1....s_n.s$ sera noté s si n=0 ou $s_1,...,s_n;s$ sinon.

Définition 27 : Une **signature** Σ sur S est la donnée d'un ensemble F de noms d'opérateurs et d'une fonction α de F dans S^* qui à f associe $\alpha(f)=s_1....s_n;s$. α(f) est le **profil** de f, s le **codomaine**, n l'arité de f. Un opérateur d'arité 0 est une constante.

$\Sigma_{w;s}$ désigne la signature composée de l'ensemble des opérateurs de profil w;s avec w appartenant à S^*. ○

Définition 28 : Une **Σ-algèbre** M est définie par une famille indexée par S

d'ensembles non vides $(D_M)_s$ et une famille indexée par Σ de fonctions notée F_M vérifiant la condition suivante pour tout opérateur f de F:

- si $a(f) = s$ alors f_M appartient à $(D_M)_s$
- si $a(f)=s_1,\ldots,s_n;s$ alors $f_M : (D_M)_{s_1} x \ldots x (D_M)_{s_n} \rightarrow (D_M)_s$.

On note

$$D_M = \bigcup_{s\in S} (D_M)_s$$

○

Les fonctions qui préservent la structure de Σ-algèbre sont les Σ-homomorphismes.

Définition 29 : Soient $M=(D_M,F_M)$ et $N=(D_N,F_N)$ deux Σ-algèbres. Un Σ-**homomorphisme** h de M dans N est une application indexée par S de D_M dans D_N telle que pour tout symbole de fonction f dans F de profil $s_1\ldots s_n;s$ et pour tous éléments $d_1,\ldots, d_n$ dans $(D_M)_{s_1} x \ldots x (D_M)_{s_n}$, on ait :

$$h_s(f_M(d_1,\ldots,d_n)) = f_N(h_{s_1}(d_1),\ldots,h_{s_n}(d_n)).$$ ○

La notion de variété de F-algèbres s'étend facilement au cas des Σ-algèbres (voir par exemple [Meseguer1983].)

2.2. Algèbre initiale et algèbre libre

La construction de l'algèbre libre et de l'algèbre initiale de la classe de toutes les Σ-algèbres peut se faire comme dans le paragraphe précédent. Nous en présentons cependant une autre, très classique elle aussi, qui consiste à construire l'algèbre libre à partir de l'algèbre initiale.

Définition 30 : Soit K une classe de Σ-algèbres. Une Σ-algèbre $M=(D_M,F_M)$ est **initiale** dans K si et seulement si

1) M appartient à K

2) pour toute Σ-algèbre $N=(D_N,F_N)$ de K il existe un unique Σ-homomorphisme h de M dans N. ○

Rappelons que si K la classe de toutes les Σ-algèbres, K possède une algèbre initiale notée $I(\Sigma)$. Son domaine est noté $T(\Sigma)$ et ses éléments sont appelés termes clos. Le domaine est une réunion d'ensembles appelés ensembles des termes clos de sorte s.

$$T(\Sigma) = \bigcup_{s\in S} (T(\Sigma))_s$$

Afin d'étendre la notion d'algèbre libre, il faut introduire la notion de variable typée.

Définition 31 : Un ensemble de variables typées est une famille indexée par S, $X=\{X_s \mid s\in S\}$ d'ensembles disjoints.

Etant donnée une signature Σ et un ensemble de variables typées X disjoint de Σ, on définit $\Sigma(X)$ par

$$\Sigma(X)_s = \Sigma_s \cup X_s$$

$$\Sigma(X)_{w;s} = \Sigma_{w;s} \text{ pour } w\neq\varepsilon,\ w\in S^*.$$

La **Σ-algèbre libre** engendrée par l'ensemble X, notée $I(\Sigma,X)$ est la Σ-algèbre $I(\Sigma(X))$. Son domaine, noté $T(\Sigma,X)$ est l'ensemble des termes $T(\Sigma(X))$ et est la réunion des termes de sorte s pour toute s dans S.

$$T(\Sigma,X) = \bigcup_{s\in S} (T(\Sigma(X))_s$$

○

2.3. Equations et variétés équationnelles

Pour définir correctement la notion de validité d'une équation, il est important dans un premier temps de quantifier explicitement les équations en introduisant des ensembles de variables.

Définition 32 : Une **Σ-équation** est un triplet (X,t,t') où X est un ensemble de variables, t et t' deux termes de $T(\Sigma(X))_s$ pour une sorte s de S. On la notera (X)(t=t').
Si X=V(t)UV(t'), la référence à l'ensemble X peut être omise sans ambiguité et nous utiliserons la forme non quantifiée t=t'. ○

Définition 33 : Une **Σ-équation** (X)(t=t') est **valide** dans une Σ algèbre M si et seulement si toute application h:X→M s'étend en un Σ-homomorphisme h# de T(Σ,X) dans M tel que h#(t)=h#(t'). On dit aussi que M **satisfait l'équation** (t=t'). ○

La notion de **variété équationnelle** de Σ-algèbres satisfaisant un ensemble de Σ-équations A, est alors définie comme dans le cas homogène.

Théorème 5 : Si K est une variété équationnelle de Σ-algèbres satisfaisant un ensemble d'équations A, K possède une **algèbre initiale** notée I(Σ,A). Son domaine est l'ensemble quotient $T(\Sigma)/\sim^A$.

2.4. Déduction équationnelle dans les algèbres hétérogènes

La généralisation aux algèbres hétérogènes du théorème 4 (théorème de complétude du raisonnement équationnel de Birkhoff) nécessite quelques précautions. Un exemple de déduction non valide est présenté dans[Goguen1982]. La correction et la complétude du raisonnement équationnel dans le cadre des algèbres hétérogènes y sont également

étudiées. Des conditions suffisantes sont proposées pour pouvoir utiliser les règles de déduction équationnelle standard (réflexivité, symétrie, transitivité, stabilité par substitution). En particulier, il suffit de supposer que toute sorte s considérée est non vide, c'est-à-dire qu'il existe au moins soit une constante de sorte s, soit une opération de profil $s_1 \ldots s_n;s$ telle que $s_1,\ldots,s_n$ ne soient pas toutes des sortes vides. Plus brièvement la sorte s est non vide si et seulement si il existe au moins un terme de cette sorte. Une telle hypothèse peut paraitre trop restrictive quand on définit par exemple des types abstraits paramétrés. C'est pourquoi une autre approche a été développée dans[Goguen1985]. En se plaçant dans le cadre des systèmes de réécriture équationnels, ils donnent des conditions suffisantes simples garantissant la complétude du raisonnement équationnel dans les classes d'algèbres hétérogènes.

Cependant dans la mesure où dans cette thèse, nous ne considérons pas de spécifications paramétrées, nous supposons simplement que les algèbres hétérogènes considérées ont toutes leur sortes non vides.

3. Induction multi-ensemble

Pour terminer ce chapitre sur les généralités, nous allons préciser une technique de preuve largement utilisée par la suite, la preuve par induction noethérienne sur les multi-ensembles ou plus brièvement preuve par induction multi-ensemble.

Rappelons d'abord brièvement le **principe d'induction noethérienne** [Cohn1965]. Etant donnée une relation noethérienne sur un ensemble H (i.e. une relation de préordre sans chaine infinie décroissante), la propriété P est vraie sur H si pour tout t dans H, P(t) se déduit de l'hypothèse que P

est vraie sur tous les éléments de H qui sont des fils stricts de t pour la relation noethérienne donnée.

Définition 34 : Un **multi-ensemble** sur un ensemble H est une collection d'éléments de H avec répétitions éventuelles. ○

Toute relation $\rightarrow$ sur H s'étend en une relation $\rightarrow_{mult}$ sur l'ensemble M(H) des multi-ensembles de H, de la façon suivante:

Définition 35 : Pour tous multi-ensembles M et N de M(H), $M \rightarrow_{mult} N$ si et seulement si N est obtenu à partir de M en enlevant l'un de ses éléments t et en ajoutant un nombre fini d'éléments t', éventuellement nul et avec répétitions, tels que $t \rightarrow t'$. ○

Par application du lemme de Koenig, la relation $\rightarrow_{mult}$ est noethérienne si et seulement si $\rightarrow$ l'est [Dershowitz1979]. Une preuve par **induction multi-ensemble sur** $\rightarrow$ est une preuve par induction noethérienne sur $\rightarrow_{mult}$. Pour plus de détails, nous renvoyons le lecteur à [Dershowitz1979] et [Jouannaud1982].

CHAPITRE 2:

COMPLETION EQUATIONNELLE

CHAPITRE 2: COMPLETION EQUATIONNELLE

1. Introduction

A l'origine, la procédure de complétion de Knuth et Bendix a pour but de calculer un système de réécriture R qui engendre la même équivalence sur les termes qu'un ensemble donné d'axiomes A. De plus le système résultant a la propriété de Church-Rosser, qui permet de décider de l'égalité de deux termes à l'aide de réécritures uniquement, à condition que la relation de réécriture soit noethérienne. Une méthode de preuve de la A-égalité de deux termes consiste alors à calculer leurs formes irréductibles et à vérifier leur identité syntaxiquement.

Cette méthode a été étendue au cas où certains axiomes de A ne sont pas orientables en règles de réécriture sous peine de produire des chaines infinies de réécritures. C'est le cas d'axiomes permutatifs tel celui exprimant la commutativité d'un symbole de fonction, ou encore le cas, très important en pratique, de théories comportant des symboles associatifs et commutatifs.

L'idée est alors de chercher un système de réécriture équationnelle (c'est-à-dire un couple (R,E) d'un ensemble de règles R et d'un ensemble d'équations E) ayant la même puissance de déduction que A. Citons quelques approches particulières de ce problème: celle de Huet [Huet1980], restreinte à des ensembles de règles linéaires à gauche, celle de Peterson et Stickel [Peterson1981], pour laquelle l'ensemble des axiomes non orientables E doit être composé d'équations linéaires, enfin celle de Jouannaud [Jouannaud1983] qui permet de s'affranchir de toutes ces conditions de linéarité, généralisant par là les deux approches précédentes qui en

deviennent des cas particuliers. Néanmoins ces trois méthodes diffèrent par la définition de la réécriture de termes utilisée. Ainsi Huet utilise la réécriture standard, tandis que Peterson et Stickel définissent la réécriture modulo E basée sur le E-filtrage. Jouannaud autorise l'emploi des deux types de réécriture précédents. Une autre approche a été introduite récemment par Pedersen [Pedersen1985]: les restrictions apparaissent non pas dans le type de règles ou d'équations, mais dans la relation de filtrage utilisée.

Ces quatre approches ont en commun de ne pas travailler sur les classes d'équivalence engendrées par les axiomes non orientables E, mais plutôt sur les termes. Plus précisément, au lieu de définir une relation de réécriture sur les classes d'équivalence de termes modulo E, comme le font Lankford et Ballantyne [Lankford1977,Lankford1977a,Lankford1977b], on définit une réécriture sur les termes tenant compte des équations de E. Nous proposons dans ce chapitre une vision unifiée de ces différentes réécritures sur les termes, qui permet d'exprimer des résultats très généraux. Une propriété de Church-Rosser modulo E, paramétrée par la relation de réécriture, est définie. Elle s'énonce de la façon suivante: deux termes sont égaux modulo A si et seulement si ils se réécrivent avec la relation paramètre, en deux termes égaux modulo E. L'assurance que la réécriture sur les termes simule bien celle induite par l'ensemble de règles sur les classes d'équivalence modulo E, se traduit par le fait que pour un terme t d'une classe T, toutes ses formes irréductibles $t\downarrow$ appartiennent à la même classe $T\downarrow$; celle-ci est justement celle obtenue par réécriture dans les classes, à condition que celle-ci termine.

Cette assurance est donnée dès que deux propriétés abstraites de la relation de réécriture, la cohérence et la confluence modulo E, sont prouvées. Une procédure de complétion équationnelle a pour objectif de calculer un système de règles possédant ces deux propriétés et ayant la même puissance de déduction que l'ensemble des axiomes A.

Huet d'une part, Peterson et Stickel d'autre part avaient proposé des procédures de complétion adaptées aux théories associatives et commutatives. Une procédure de complétion équationnelle généralisant leurs travaux est décrite et prouvée. Pour une notion donnée de réécriture sur les termes, elle permet d'engendrer à partir d'un ensemble d'équations E et d'un ensemble de paires de termes, un système de réécriture équationnelle possédant la propriété de Church-Rosser associée à la relation de réécriture choisie. REVEUR-3 est l'implantation de cette procédure de complétion généralisée dont toutes les précédentes (celle de Knuth et Bendix, de Huet, de Peterson et Stickel) sont des instances. Les deux caractéristiques essentielles de ce système sont les suivantes:

- On autorise des symboles de fonctions possédant des propriétés définies par des équations non orientables E, à condition de disposer d'algorithmes de E-égalité, de E-filtrage et de E-unification. L'ensemble de ces algorithmes et des équations E constitue une théorie prédéfinie de REVEUR-3.
- Le système permet de faire des expérimentations, en ce sens qu'une complétion est paramétrée par le type de réécriture choisi par l'utilisateur. Celui-ci peut donc essayer de compléter sa théorie en utilisant à son gré la méthode de Knuth et Bendix, de Huet, de Peterson et Stickel ou de Jouannaud. Outre la relation de réécriture, deux autres paramètres ont été introduits: l'un porte sur la façon d'assurer la

propriété de cohérence mentionnée plus haut: il est possible soit d'ajouter des extensions de règles systématiquement, évitant ainsi certains calculs de paires critiques, soit de les ajouter uniquement en cas de nécessité. L'autre paramètre porte sur l'ordre dans lequel on choisit de superposer deux règles entre elles et permet par exemple de traiter les plus petites d'abord.

Ce chapitre présente les fondements de la réécriture équationnelle et la procédure de complétion équationnelle. En partie à cause de la complexité du sujet, il est très dense et très technique. Néanmoins, il nous a paru essentiel d'y inclure les détails des preuves et les précisions techniques, afin de servir de référence aux travaux ultérieurs sur le sujet.

- La première partie de ce chapitre est constituée d'un article accepté à la revue SIAM Journal of Computing et écrit en collaboration avec Jean-Pierre Jouannaud. La théorie de la réécriture équationnelle et de la complétion équationnelle y est présentée en détail. La relation de réécriture équationnelle utilisée prend en compte la linéarité des membres gauches de règles, pour des raisons d'efficacité. Les preuves des résultats énoncés sont données in extenso.

- La deuxième partie est constituée d'un article écrit en collaboration avec Claude Kirchner et qui décrit l'implantation de la procédure de complétion équationnelle dans le système REVEUR-3. Conjointement à la présentation du système, quelques expérimentations sont décrites pour mettre en évidence l'importance du choix de la réécriture, à la fois pour la terminaison de la procédure de complétion et pour la rapidité de celle-ci.

- La troisième partie du chapitre est une généralisation récente de la première, qui permet de prendre en compte des mélanges de réécritures.

Completion of a Set of Rules Modulo a Set of Equations[1]

by

Jean-Pierre Jouannaud and Helene Kirchner[2]

Centre de Recherche en Informatique de NANCY and GRECO-Programmation
Campus Scientifique, BP 239, 54506 Vandoeuvre les Nancy CEDEX, FRANCE.

Abstract: Abstract Church-Rosser properties are first presented, depending on an arbitrary relation $\rightarrow^R$, an equivalence relation $\sim^E$ and a reduction relation $\rightarrow^{R'}$ used to compute normal forms of $\rightarrow^R$ modulo $\sim^E$.
Terminating Rewriting Systems operating on equational congruence classes of terms of a free algebra are then considered. In this framework, the Church-Rosser property is proved decidable for a very general reduction relation which may take into account the left linearity of rules for efficiency reasons, under the only assumption of existence of a complete and finite unification algorithm for the underlying equational theory, whose congruence classes are assumed to be finite. This extends previous results by [50, 67, 30, 37]. A general Completion Procedure for mixed sets of rules and equations is then presented that generalizes and improves [67]. In addition to computing a Church-Rosser set of rules when it terminates, it yields a semi-decision procedure for testing equality when it runs forever. A post-processing is finally described that yields a Church-Rosser set of rules in normal form.
All proofs, including the correctness proof of our completion algorithm, are based on a powerful proof technique called multiset induction.

Keywords and Phrases: Church-Rosser Property, Confluence, Coherence, Equational Theories, Equational Deduction, Term Rewriting Systems, Reduction Systems, Critical Pairs, Complete sets of Unifiers, Completion Procedure.

---

[1] A preliminary version of this paper was presented at the 11th ACM Conference on Principles of Programming Languages, Salt Lake City, January 1984.
This research was supported in part by Agence pour le Developpement de l'Informatique under contract 82/767 and for another part by Office of Naval Research under contract N00014-82-0333.

[2] Part of this work was done while the first author was visiting the Computer Science Laboratory of SRI-International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA.

1 Introduction

Term Rewriting Systems (TRS for short) are known to be a major tool for expressi nondeterministic computations, because they are based on directed equalities, with no expli control. In addition, they have a very simple semantics, whenever the result of a computati does not depend on the choice of the rules to be applied (the so-called *Confluence* propert When a TRS is not confluent, it may be transformed into an equivalent confluent one, using t Knuth and Bendix *Completion Procedure* [49]. This procedure can be seen as a way to comp equational specifications into confluent sets of rules, providing *Rewrite Programs*, which can used in a similar way as PROLOG ones [11].

During the past several years, confluent TRS and the Knuth and Bendix completion procedu were shown to be a major tool for a wide class of problems, mainly the word problem universal algebra [15, 16, 49, 3, 5, 74, 46], the word problem for finitely presented algebr [2, 4, 54], equivalence proofs of sets of axioms in algebra [55, 56], unification in equation theories [20] [35] [75] [42], software validation [26], inductive proofs in data typ [62, 23, 32, 48], theorem proving in various logics [66, 27, 28, 22], program synthesis fro specification [29, 11], computing with rewrite programs [11] or with Horn clauses with equali [24], describing PROLOG's semantics [8], constructing initial algebras and executing equation specifications as in OBJ [25], inferring types in the presence of polymorphism [61], co optimization in compilers [1] or even code generation thought of as tree rewriting [73].

The Knuth and Bendix completion procedure is based on using *Equations* as *Rewrite Rules* a computing *Critical Pairs* when left hand sides of rules overlap. If a critical pair has distin *Irreducible Forms*, a new rule must be added and the procedure recursively applies until maybe stops. This procedure requires the *Termination* property of the set of rules, which ca be proved by various tools, such as the recursive path ordering [9], the generalized recursi path ordering [44], the recursive decomposition ordering [43] or the recursive decompositio ordering with status [57]. A full implementation of these techniques is described in [55].

The method was extended to handle the case of an *Equational Term Rewriting System* (ETR for short), i.e., a set A of axioms split into a first subset R whose axioms are used as rules and second subset E whose axioms are used as equations. This allows axioms such as commutativi that cannot be used as rules without loosing the termination property. A first approach [51, 50, 52] handles the case of commutative or more generally permutative axioms th generate finite E-congruence classes. The case of infinite E-congruence classes is studied [67, 30, 37]. Huet's approach is restricted to sets R of left linear rules, while Peterson a Stickel's one is restricted to linear theories E for which a finite and complete unificati algorithm is known. Both [52] and [67] results yield an efficient associative-commutati completion procedure, with a few more restrictions needed for applying the first. The comple unification algorithm is required to compute complete sets of E-overlappings, providi complete sets of E-critical pairs. These results are unified in [37] by describing at a mo abstract level the underlying model of computation used in both approaches: it makes cle that two properties, namely *Confluence* and *Coherence*, are necessary and sufficient conditio for Church-Rosser properties of this model, assuming the so called E-*Termination* property the rewriting relation modulo the set of equations. In addition to Confluence, Coherence required to enable computations in E-congruence classes. Applying then this abstract model

ETRS, Huet's results are easily obtained as well as a more general version of Peterson and Stickel's ones: nonlinear equational theories can be handled, which was known as a hard problem of the theory of ETRS.

In this paper, we refine our abstract approach and provide simpler definitions. In addition to classical Church-Rosser and Confluence notions using E-congruence classes, we also introduce new Church-Rosser, Confluence and Coherence notions which are parameterized by the reduction relation $\rightarrow^{R'}$ used to reduce terms. This allows us to deal with efficient reduction relations, whereas $\rightarrow^{R/E}$, the reduction relation induced by the rules in E-congruence classes, requires that one searches the congruence class for a reducible term. As in [37], the Coherence property is the necessary and sufficient condition for $\rightarrow^{R/E}$ and $\rightarrow^{R'}$ to have the same normal forms. We then study a very general reduction relation, which is a mixture of the usual reduction relation for linear rules and the Peterson and Stickel's reduction relation using matching modulo E for the others. This was already done in [37] and allows us to avoid useless computations of complete sets of E-critical pairs when left linear rules are involved. This is a major improvement in practice, because using E-unification or E-matching algorithms (such as the associative-commutative ones of [76] and [34]) is known to be time consuming. On the other hand, the more powerful the rewriting relation, the weaker the associated Church-Rosser property, and therefore the easier to ensure. As a consequence, given a set E of equations, the same set R of rules can be Church-Rosser for some reduction relation and not for a weaker one.

For handling our definitions, a new powerful proof technique is required, based on *Multiset Induction*, which permits us to prove our Church-Rosser results in a very elegant way. In addition, this new proof technique allows us to solve a last open problem of [67] and [37]: assume that the reduction relation induced in E-congruence classes is terminating. Then the Church-Rosser property is shown equivalent to the E-equality of normal forms of all critical pairs and is thus decidable when these pairs are finitely many, which is the case for finite ETRS. Such a result was already known when R' is the usual rewriting relation, assuming the rules are left linear [30]. We prove it here for the case where R' is the mixed rewriting relation, assuming a complete and finite unification algorithm is known for E, whose congruence classes must also be finite. Our proof holds for the particular case of the Peterson and Stickel's rewriting relation, without any linearity hypothesis on rules or on equations. However, the case of infinite congruence classes remains as the last open problem of the theory of ETRS.

A new completion procedure is then derived from these Church-Rosser results. It is written in a very elegant recursive form, which makes both understanding and proof easier. This procedure improves and generalizes Peterson and Stickel's one for associative commutative theories. Two kinds of critical pairs are to be distinguished: *Confluence Pairs* are processed in a classical way whereas *Coherence Pairs* are processed in a more complex way. In addition, *Extended Rules* are sometimes used instead of coherence pairs to ensure the Coherence property. These rules generalize to an arbitrary theory E the so called *Associative-Commutative Extensions* of [67]. Following [31], a complete proof of our algorithm is then given using multiset induction. The case where the algorithm stops without adding any rule provides in addition another proof of our Church-Rosser results. We finally define *complete sets of reductions in normal form*, i.e., Church-Rosser sets of rules which have a normal form property and show how to transform a given Church-Rosser set of rules into a complete set of rules in normal form. These normal

forms can be used to check whether two specifications are actually different presentations of the same theory.

2 Abstract Church-Rosser Results

The presentation of this section is strongly influenced by that in [30]: we deal with bi... relations on an arbitrary set S. As in [30], most of our proofs will use the powerful *noether... induction principle* (See [6] for a precise introduction): Given a noetherian relation on S, i... relation without any infinite decreasing chain, we prove a property P on S by showing that... any t in S, P(t) follows from the assumption that P is true for all elements in S which are pr... sons of t for the given noetherian relation. We use here what we call *Multiset Inducti...* particular case of noetherian induction, that we introduce in the following.

2.1 Preliminary Definitions and Notations

Let $\vdash\!\dashv^E$ be a symmetric relation on S and $\sim^E$ its reflexive transitive closure, called E-equa... which is obviously an equivalence relation. For simplicity, the E subscript will sometime... omitted if no ambiguity arises.

Let $\rightarrow^R$ (R for short) be any relation on S, called **reduction**, $\stackrel{+}{\rightarrow}{}^R$ and $\stackrel{*}{\rightarrow}{}^R$ its transitive... reflexive transitive closures respectively, and $\rightarrow^{R/E}$ (R/E for short) the relation $\sim^E \circ \rightarrow^R \circ \sim^E$, which simulates the relation induced by $\rightarrow^R$ in E-equivalence classes.

Let $\vdash\!\dashv^A$ be the union of $\vdash\!\dashv^E$ with the symmetric closure of $\rightarrow^R$, that is the smallest symme... relation containing $\vdash\!\dashv^E$ and $\rightarrow^R$. Let $\sim^A$ be the reflexive <transitive closure of $\vdash\!\dashv^A$, ca... A-equality, which is an equivalence relation. The A superscript will never be omitted in th... notations.

In the following, t (with maybe a subscript or a superscript) denotes an arbitrary element i... and $\rightarrow$ an arbitrary reduction on S.

We say that t is **$\rightarrow$-reducible** iff $t \rightarrow t'$ for some t'. Else t is said to be **$\rightarrow$-irreducible**... **normal form** of t, denoted $t\downarrow$, is an irreducible term t' such that $t \stackrel{*}{\rightarrow} t'$. We write $t\downarrow_R$ if... want to make the reduction relation $\rightarrow^R$ precise.

The relation R is **terminating modulo** E or **E-terminating** iff the relation $\rightarrow^{R/E}$... noetherian, i.e., there is no sequence of the form $t_0 \sim^E t'_0 \rightarrow^R t_1 \ldots t_n \sim^E t'_n \rightarrow^R t_{n+1} \ldots$. ... simply said to be **terminating** or **noetherian** or **well-founded** if E is empty. Our particu... interest in terminating relations is twofold: first, they provide normal forms for every eleme... second, they allow proofs by noetherian induction.

Multiset Induction: A **multiset** on S is an unordered collection of elements of S, possi... with repetitions. For instance, {1 2 1 3 15 15} is a multiset on $\mathcal{N}$, the set of natural numbe... Operations on sets extend in a natural way to multisets, also called **bags**. Specifical... removing an element from a multiset amounts to removing one of its occurrences in ... multiset. In the same way, adding an element to a multiset increases by one its numbe... occurrences. We can extend any relation $\rightarrow$ on S to a relation $\rightarrow_{mult}$ on $\mathcal{M}(S)$, the set of... multisets on S, in the following way:

Given two multisets M and N in $\mathcal{M}(S)$, $M \rightarrow_{mult} N$ iff N is obtained from M by removing one... its elements, say t, and adding a finite number of elements t' (possibly with repetitions) su...

that $t \rightarrow t'$. As an easy consequence of Koenig's lemma, the transitive closure of $\rightarrow_{mult}$ is noetherian iff $\rightarrow$ is noetherian (see [12] for a precise proof, and [38] for a discussion about multiset orderings). What we call **Multiset Induction** on $\rightarrow$ is simply noetherian induction on the transitive closure of $\rightarrow_{mult}$. All our proofs are based on this technique that will appear to be the well-suited one for proving Church-Rosser properties. □

2.2 Church-Rosser Definitions

We now begin studying the Church-Rosser property for such reductions on S. Our goal is to decide A-equality of two terms t and t' by computing their normal forms using some reduction relation related to $\rightarrow^R$ and $\sim^E$, then checking these normal forms for E-equality (that we implicitly assume decidable for practical use). A very natural idea is to compute in the quotient set of S by $\sim^E$, thus to use $\rightarrow^{R/E}$ as the reduction relation. Let us recall what the Church-Rosser and Confluence properties are in the context of E-equivalence classes on S:

Definition 1:

R is **Church-Rosser modulo** E iff $\forall t_1,t_2 \;\; t_1 \sim^A t_2 \Rightarrow \exists t' \;\; t_1 \stackrel{*}{\rightarrow}{}^{R/E} t'$ and $t_2 \stackrel{*}{\rightarrow}{}^{R/E} t'$.

R is **confluent modulo** E iff $\forall t,t_1,t_2 \;\; t \stackrel{*}{\rightarrow}{}^{R/E} t_1$ and $t \stackrel{*}{\rightarrow}{}^{R/E} t_2 \Rightarrow \exists t' \;\; t_1 \stackrel{*}{\rightarrow}{}^{R/E} t'$ and $t_2 \stackrel{*}{\rightarrow}{}^{R/E} t'$. □

These two properties are nothing but the classical Church-Rosser and Confluence properties in the quotient set of S by $\sim^{\mathcal{E}}$ and are known to be equivalent.

However, R/E-reducibility may be undecidable if E-equivalence classes are infinite and is at least very inefficient if large classes must be traversed to find a reducible term. This problem can be overcome by choosing an element for representing each equivalence class, as in OBJ [25]. This can be done in a canonical way for many practical cases of reductions on terms modulo associativity and commutativity. A more general idea, the key one actually, is to compute using another relation $\rightarrow^{R'}$. This yields R'-dependent definitions that have to be linked with definitions using R/E by introducing a new property called Coherence. As a matter of fact, the first point of view is an instance of the second: starting from an element of S, we first compute its canonical form and then reduce it. This is clearly a reduction on S. As the second point of view allows better understanding and more general results, we adopt it here.

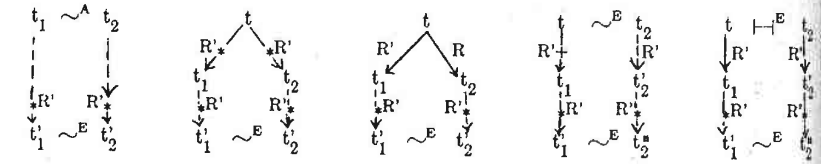
Fundamental Assumption: In the following, $\rightarrow^{R'}$ ($\rightarrow$ or R' can also be used for short if no ambiguity arises) is any reduction that satisfies: $\rightarrow^R \subset \rightarrow^{R'} \subset \rightarrow^{R/E}$.

Definition 2:

- A pair (t_1,t_2) is R'-**confluent modulo** E, written $t_1 \downarrow^*_{R'} t_2$, iff $\exists t'_1,t'_2 \;\; t_1 \stackrel{*}{\rightarrow}{}^{R'} t'_1$, $t_2 \stackrel{*}{\rightarrow}{}^{R'} t'_2$ and $t'_1 \sim^{\mathcal{E}} t'_2$. For R'=R/E, we can assume $t'_1 = t'_2$.
- R is R'-**Church-Rosser modulo** E iff $\forall t_1,t_2 \;\; t_1 \sim^A t_2 \Rightarrow t_1 \downarrow^*_{R'} t_2$.
- R' is **confluent modulo** E iff $\forall t,t_1,t_2 \;\; t \stackrel{*}{\rightarrow}{}^{R'} t_1$ and $t \stackrel{*}{\rightarrow}{}^{R'} t_2 \Rightarrow t_1 \downarrow^*_{R'} t_2$.
- R' is **locally confluent modulo** E **with** R iff $\forall t,t_1,t_2 \;\; t \rightarrow^{R'} t_1$ and $t \rightarrow^{R} t_2 \Rightarrow t_1 \downarrow^*_{R'} t_2$.
- R' is **coherent modulo** E iff $\forall t,t_1,t_2 \;\; t \stackrel{+}{\rightarrow}{}^{R'} t_1$ and $t \sim^{\mathcal{E}} t_2 \Rightarrow \exists t'_2 \;\; t_2 \rightarrow^{R'} t'_2$ and $t_1 \downarrow^*_{R'} t'_2$.

6. R' is **locally coherent modulo** E iff $\forall t,t_1,t_2 \quad t \to^{R'} t_1$ and $t \vdash\!\!\!\dashv^{E} t_2 \Rightarrow \exists t'_2 \quad t_2 \to^{R'} t'_2$ and $t_1 \downarrow\!\!\!\sim t'_2$.

Our main definitions are pictured below. Full arrows denote universal hypotheses while dashed arrows denote existential conclusions:



R'-*Church-Rosser Modulo* E | *confluence modulo* E *of* R' | *local confluence modulo* E *of* R' *with* R | *coherence modulo* E *of* R' | *local coherence modulo* E *of* R'

Figure 1: *Main Definitions*

Before developing our results, let us emphasize some important points:

- Since $\to^{R} \subset \to^{R'} \subset \to^{R/E}$, the relations $\sim^{\mathcal{E}} \circ \to^{R'} \circ \sim^{\mathcal{E}}$, $\sim^{\mathcal{E}} \circ \to^{R/E} \circ \sim^{\mathcal{E}}$ and $\to^{R/E}$ are t same relation.

- Church-Rosser modulo E and R'-Church-Rosser modulo E are two different properties if R and R/E are different. R'-Church-Rosser is actually a stronger one as shown by t following example: Let $t \to^{R'} t' \sim^{E} t'_1 \to^{R'} t_1$. As a matter of fact, t' and t_1 are in R'-norm form, however not E-equal; therefore the R'-Church-Rosser property is not satisfi although the R/E-Church-Rosser one is.
  The R'-Church-Rosser property is actually *monotonic* with respect to the inclusion relations: this is a straightforward consequence of its definition. Therefo R-Church-Rosser $\Rightarrow$ R'-Church-Rosser $\Rightarrow$ R/E-Church-Rosser, this last property bei precisely the *intrinsic* Church-Rosser property first defined. Choosing the adequate will thus be a relevant practical problem.

- Different R'-normal forms of a term are E-equal, by the E-confluence property. W therefore will refer to *the* R'-normal form of a term.

- (Local) Confluence and (local) Coherence are two instances of the same general concep which expresses that two (maybe different) relations have a diamond property.

- Local Confluence involves both R and R'. Using only R would give a too weak relatio On the other hand, using only R' gives one that is too strong for application to ETRS.

- Coherence and Local Coherence differ from the corresponding definitions in [30] b replacing R by R', which is a key point for applications, and by requiring a first R' ste from t_2 in the Coherence definition, which will be justified next. It differs from th

Compatibility Property in [67] by allowing t_1 to be reduced into t'_1. This will permit a more powerful result stating an equivalence between Church-Rosser and both Confluence and Coherence. It differs also from the *Commutation* property in [39] by allowing t_1 to be reduced to t'_1. Handling Coherence, however, requires a more powerful termination property than handling commutation. Note finally, that property P1 in [72] is a common ancestor to all of these definitions.

- The first $\to^{R'}$ step required in the Coherence definition plays a crucial role: If t is in R'-normal form, then any t' E-equal to t must also be in R'-normal form else a R'-step would apply to t by Coherence. For the same reason, t is also in R/E-normal form, else $t \sim^{E} t' \to^{R} t^{*}$ for some t' and t^{*} and the same applies. Therefore the two sets of normal forms with respect to $\to^{R'}$ and $\to^{R/E}$ must be the same. This remark will be used freely in the rest of the paper.
  This first reduction step arises in a very natural way: If we don't require it, then we can have the following cycle for the relation $\to^{R/E}$:
  $t_2 \sim^{E} t \not\to^{R'} t_1 \to^{R'} t'_1 \sim^{E} t_2$, therefore $t_2 \to^{R/E} t_2$ which contradicts the E-termination property of R that is required anyway in the following.

- It is even possible to give a slightly different form to our definitions (without changing at all either the results or the proofs) by using the relation R/E itself to compute from t_1 and t_2 in each of our definitions, except for coherence and local coherence where a first R' step from t_2 to t'_2 is required for having a nonvacuous definition. This was actually done in [37], despite some tedious proofs stemming from a proof technique that was not so powerful as the one used here. The confluence step from the pairs (t_1,t_2) or (t_1,t'_2) of the definitions can even be performed with an arbitrary reduction provided it contains R' and is contained in R/E. This last remark is very important in practice, as we will see in sections 3 and 4.

2.3 Role of Coherence

We now show that coherence of R' modulo E is the property that enables one to compute in E-congruence classes with the relation R'. This was already the case with the definitions of [37], which are actually equivalent to ours, provided R is E-terminating.

Proposition 3: Assume R is E-terminating and R' is confluent modulo E. Then $t\downarrow R/E \sim^{\mathcal{E}} t\downarrow R'$ iff R' is coherent modulo E.

Proof: - Let us first prove the only if part.
Assume $t_2 \sim^{\mathcal{E}} t \not\to^{R'} t_1$ and let $t_1\downarrow R'$ be any R'-normal form of t_1, which is also an R/E-normal form of t_1 as it was already remarked, thus also of t_2. Using the hypothesis, it follows that $t_2\downarrow R' \sim^{\mathcal{E}} t_1\downarrow R'$. As R is E-terminating, $t_2\downarrow R'$ is different from t_2, otherwise $t_2 \sim^{\mathcal{E}} t \not\to^{R'} t_1\downarrow R' \sim^{\mathcal{E}} t_2$ and there is a cycle for R/E. Therefore, $t_2 \not\to^{R'} t_2\downarrow R'$ and we are done.
- Let us now prove the if part by noetherian induction on $\to^{R/E}$.
Let t be any element of $\mathcal{S}$. First, the result is, true if t is in R/E normal form, thus in R'-normal form. Else $t \sim^{\mathcal{E}} t_1 \to^{R} t_2 \not\to^{R/E} t\downarrow R/E$ and $t \not\to^{R'} t\downarrow R'$.

By the coherence property applied to $(t, t{\downarrow}R', t_1)$, $\exists t'_1, t''_1\ t_1 \to^{R'} t'_1$ and $t'_1 \stackrel{*}{\to}^{R'} t''_1 \sim^{\mathcal{E}} t{\downarrow}$
Note that t''_1 must be in R'-normal form since it is E-equal to a normal form.

By the confluence property applied to (t_1, t''_1, t_2), $\exists t''_2$ $t_2 \stackrel{*}{\to}^{R'} t''_2 \sim^{\mathcal{E}} t''_1$ and t''_2 must be R'-normal form, thus it is a R'-normal form of t_2. We can now finish the proof noetherian induction applied to t_2 which is a proper son of t for $\to^{R/E}$.

Note that we could assume R locally R'-confluent with R modulo E instead of R'-conflue modulo E: a simple extra induction step is to be added to the proof for that.

This result states that we can use R'-computations instead of R/E-computations, if we a interested in normal forms of terms and not in the intermediate steps of computation. This usually the case in practice. Therefore coherence is exactly the property needed.
As a consequence, confluence and coherence modulo E are both needed to compute with R', th is to decide the A-equality by computing R'-normal forms and checking for their E-equality.

2.4 The Abstract Church-Rosser Theorem

We see now that Confluence and Coherence modulo E can be restricted together to their loc properties, that generalizes the theorem of [63] which corresponds to the case where E is t empty set, since both coherence and local coherence become vacuous in that case. By the wa we also use normal form computations, since normal forms are the link betwe R'-computations and R/E-computations.

Theorem 4: If R is E-terminating, the following properties are equivalent:
(1) R is R'-Church-Rosser modulo E.
(2) R' is confluent modulo E and R' is coherent modulo E.
(3) R' is locally confluent with R modulo E and locally coherent modulo E.
(4) $\forall t,t'\quad t \sim^{A} t' \Leftrightarrow t{\downarrow}R' \sim^{\mathcal{E}} t'{\downarrow}R'$.

Proof: Note that property (4) asserts that the R'-Church-Rosser property of R modulo E true when computing R'-normal forms of t and t', provided R is E-terminating. This perfectly similar with the usual case of an empty set E of equations: In that case t Church-Rosser property can be checked on normal forms, provided R is terminating.
(2) $\Rightarrow$ (3) and (4) $\Rightarrow$ (1) are straightforward.
(1) $\Rightarrow$ (2): the confluence proof is straightforward, but the coherence proof requires t use of E-termination to exhibit the first R'-step from t_2 in the definition of coherence.

Let $t \stackrel{+}{\to}^{R'} t_1$ and $t \sim^{\mathcal{E}} t_2$. Then $t_1 \sim^{A} t_2$ and by R'-Church-Rosser property, $\exists t'_1$ $t_1 \stackrel{*}{\to}^{R'} t'_1$, $t_2 \stackrel{*}{\to}^{R'} t'_2$ and $t'_1 \sim^{\mathcal{E}} t'_2$. Assume now that $t_2 = t'_2$. The $t'_1 \sim^{\mathcal{E}} t'_2 = t_2 \sim^{\mathcal{E}} t \stackrel{+}{\to}^{R'} t_1 \stackrel{*}{\to}^{R'} t'_1$ and thus $t'_1 \stackrel{+}{\to}^{R/E} t'_1$ which contradicts E-termination.
(3) $\Rightarrow$ (4): That the right part of the equivalence implies the left is straightforward. Let prove the converse by multiset induction on $\to^{R/E}$:
Let $M=\{t_0, t_1, \ldots, t_{n+1}\}$ be a multiset of at least two elements (the one element case straightforward) s.t. $t = t_0 \vdash^{A} t_1 \ldots \vdash^{A} t_{n+1} = t'$. The basic case being obvious, consider the general one. Three cases are to be distinguished according to the la equality step $t_n \vdash^{A} t_{n+1}$.

- $t_{n+1} \to^{R} t_n$: Since $t_n{\downarrow}$ is a R'-normal form of t_{n+1}, the result follows from the induction hypothesis applied to the multiset $M-\{t_{n+1}\}$.

- $t_n \vdash^{E} t_{n+1}$: The result follows from either the induction hypothesis applied to the multiset $M-\{t_{n+1}\}$ if t_n, thus t_{n+1}, is R'-irreducible, or else it follows from the local coherence of R' and from the induction hypothesis applied to the multiset $\{t_0 \ldots t_n\ t''_n \ldots t'_n \ldots t'_{n+1}\}$ which is strictly smaller than M, since its terms are all proper sons of t_{n+1} for $\to^{R/E}$. The second case is shown on the left part of Figure 2, where circled numbers stand for successive steps in the proof.

- $t_n \to^{R} t_{n+1}$: We first apply the induction hypothesis to the multiset $M-\{t_{n+1}\}$. As t_n is reducible by R (therefore by R') we can apply local confluence modulo E of R' with R. We end the proof by applying the induction hypothesis to the multiset $\{t_n{\downarrow} \ldots t''_n \ldots t'_n \ldots t'_{n+1} \ldots t_{n+1}{\downarrow}\}$ as shown on the right part of Figure 2:

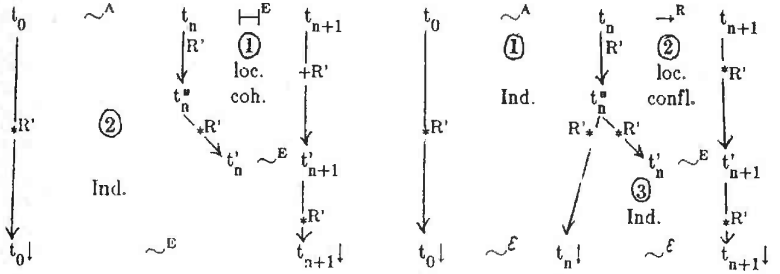


Figure 2: *Church-Rosser Proof using Multiset Induction*

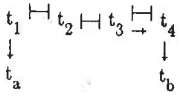
□

Theorem 4 is false if termination of R' is assumed in place of E-termination of R, as proved by the examples of Figure 3, where local properties are true but global ones are not. Note that these examples are similar to [30], yet more complex in order to contradict our more sophisticated coherence property.

This counterexample is based on contradicting the Coherence property. We could also contradict the Confluence property by adding new arrows to Figure 3:
For the cycle case, this can be done by adding a new element s and replacing the single step $t_2 \vdash t_3$ by $t_2 \vdash s \vdash t_3$, which allows us now to add two new arrows from s to t_2 and t_3. An alternative is to add these two arrows from s to t_1 and t_4.
For the infinite chain case, we can expand this new diagram as previously. We can also add two arrows in the previous diagram from t_2 to t'_3 and from t_3 to t'_2, but no more.

As indicated in [72], abstract Church-Rosser results like these apply in a wide range of areas in Computer Science as well as in Mathematics. An interesting example dealing with cyclic

cycle case:



infinite chain case (obtained by folding the cycle):

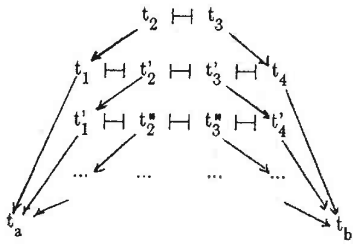


Figure 3: *Crucial role of* E*-termination*

covering matrices is given in [72]. A more complex example in formal language theory is g in [60]. In the following we are interested in those applications which deal with terms and rewriting systems for which we give decision procedures for testing for the Church-R property.

3 Application to Equational Term Rewriting Systems

This section is devoted to the study of Church-Rosser results for ETRS, i.e., mixed sets of axioms E and R on a Term Algebra, such that axioms in E are used as equations and define an equational theory $\sim^E$, whereas axioms in R are used as rules and define a *Rewriting Relation* $\rightarrow^R$. However, rewriting with rules in presence of equations is somewhat more complicated than rewriting with rules only: we will introduce other rewriting relations that will play the role of the reduction relation R' of Section 2.
In practice, E contains those axioms that cannot be directed without loosing the termination property for the rewriting relation. It can also contain other axioms causing the divergence of the completion process when used as rules.
The notation used in this Section is consistent with that of Section 2 and will not be reintroduced. This section is divided into two different parts: the first one recalls the more or less classical background on TRS (see [33] for a more complete presentation); the second states and proves our Church-Rosser result.

3.1 Terms

Definition 5: Given a set $\mathcal{X}$ of variables and a graded set $\mathcal{F}$ of function symbols, $\mathcal{T}(\mathcal{F},\mathcal{X})$ or $\mathcal{T}$ for simplicity denotes the **free algebra** over $\mathcal{X}$ also called the **Term Algebra**. Variables have arity 0 by convention. Elements of $\mathcal{T}(\mathcal{F},\mathcal{X})$, called **terms**, are viewed as *labelled trees* in the following way: A term t is a partial application of $\mathcal{N}_+^*$ into $\mathcal{F} \cup \mathcal{X}$ such that its domain $\mathcal{D}(t)$ satisfies: the empty word $\epsilon \in \mathcal{D}(t)$, and $\upsilon i \in \mathcal{D}(t)$ iff $\upsilon \in \mathcal{D}(t)$ and $i \in [1,\text{arity}(t(\upsilon))]$.
$\mathcal{D}(t)$ is the set of **occurrences** of t, $\mathcal{G}(t)$ the subset of **ground occurrences** (i.e. $t(\upsilon)$ is not a variable for $\upsilon \in \mathcal{G}(t)$), $\mathcal{V}(t)$ the set of variables of t, t/υ the **subterm** of t at occurrence υ said to be strict if $\upsilon \neq \epsilon$, $t \rightarrow^{\text{(strict) subterm}} t'$ the relation t' is a (strict) subterm of t, $t[\upsilon \leftarrow t']$ the term obtained by replacing t/υ by t' in t and $\#(x,t)$ the number of occurrences in t of $x \in \mathcal{X}$. A term t is **linear** iff $\#(x,t) = 1$ for any x in $\mathcal{V}(t)$. □

In the following, i, j, k denote natural numbers, x, y, z denote variable symbols, a, b, c and e denote constant symbols, f and h denote function symbols, s and t denote terms, υ, ν, ω, o and π denote occurrences, ϵ denotes the empty occurrence, and $\upsilon\nu$ denotes the concatenation of occurrences υ and ν.

Example:
Let t=(x+e)+x. Then $\mathcal{D}(t)=\{\epsilon,1,11,12,2\}$, $t(\epsilon)=+$ and $t(12)=e$, $\mathcal{G}(t)=\{\epsilon,1,12\}$ and $\#(x,t)=2$.

A term with top function symbol f will often be written f(... t ...) indicating that we are interested in its first level subterm t. By convention, t and t' occur at the same place in f(... t ...) and f(... t' ...).

Definition 6: A relation ϕ on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is **compatible** iff $\forall t,t' \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and $\forall f \in \mathcal{F}$, $t \phi t' \Rightarrow f(... t ...)\ \phi\ f(... t' ...)$. A compatible equivalence on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is called a **Congruence**. □

Compatibility can easily be extended by induction to the case where t and t' are plugged into the same arbitrary context (a term having a free place to be filled). This remark will be used implicitly in the rest of the paper.

3.2 Substitutions, Axioms, Unifiers

Definition 7: **Substitutions** σ are defined to be endomorphisms of $\mathcal{T}(\mathcal{F},\mathcal{X})$ with a fin domain $\mathcal{D}(\sigma) = \{x \mid \sigma(x) \neq x\}$. We denote by Σ the set of substitutions on $\mathcal{T}(\mathcal{F},\mathcal{X})$ and by σ $\sigma(t)$ the result of applying the substitution σ to the term t. Composition of substitutions σ and is simply written $\sigma\tau$. If $t'=\sigma t$ (resp. $\tau'=\sigma\tau$), we say that t (resp. τ) is **more general** than (resp. τ'), that t' (resp. τ') is an **instance** of t (resp. τ), and that σ is the **match** from t to (resp. τ to τ'). The **subsumption** preordering $\preceq$ on $\mathcal{T}$ (resp. Σ) is defined by: $t \preceq t'$ (resp. $\tau' \preceq$ iff t' is an instance of t (resp. τ' of τ). We write $\tau' \preceq \tau\ [\mathcal{V}]$ if it holds for the restrictions of τ an τ' to a subset $\mathcal{V}$ of $\mathcal{X}$. The equivalence associated with this preordering is **variable renamin** i.e., bijection between two sets of variables.

Remember that substitutions are homomorphisms: we will make intensive (and implicit) use homomorphic properties of substitutions that are often given as lemmas in the literature. W will also use the fact that the strict ordering associated with the subsumption preordering well founded [70].
In the following, greek letters σ, θ and τ denote substitutions and $(x\backslash t,\ y\backslash t',\ \ldots)$ is t substitution σ such that $\mathcal{D}(\sigma)=\{x,y,\ldots\}$ and $\sigma x=t$, $\sigma y=t'$, ...

Definition 8: We call an **axiom** or **equation** any pair (t,t') of terms and write it t=t'. It said to be **linear** if both t and t' are linear.
Given a set E of equations, the **one step E-equality** is defined by:
$t \vdash^{E}_{[\upsilon,\sigma,l=r]} t'$ with $\upsilon \in \mathcal{D}(t)$, $\sigma \in \Sigma$ and $l=r \in E$ iff $t/\upsilon=\sigma l$ and $t'=t[\upsilon \leftarrow \sigma r]$ or $t/\upsilon=\sigma r$ an $t'=t[\upsilon \leftarrow \sigma l]$.
Any of these subscripts may be omitted for the sake of simplicity without any ambiguity, sinc different notations range over disjoint sets of characters. We write $l \rightarrow r$ (resp. $r \rightarrow l$) if we wa to make precise that the equation l=r is used from left to right (resp. right to left).
The reflexive-transitive closure of $\vdash^{E}$, denoted $\sim^{E}$, is called **E-equality** or **equational theor** E.

Note that $\vdash^{E}$ is actually the smallest relation on $\mathcal{T}$ which is symmetric, compatible, clos under instantiation and contains all pairs (l,r) for every $l=r \in E$. As a consequence, $\sim^{E}$ is th smallest congruence relation on $\mathcal{T}$ closed under instantiation and containing all pairs (l,r).

All our definitions concerning matching extend straightforwardly to the case of an equation theory $\sim^{E}$. For instance, an **E-match** from t to t' is a substitution σ such that $t' \sim^{E} \sigma($ Note that the **E-subsumption** preorderings defined on $\mathcal{T}$ and Σ as previously are not alwa well founded if E is not empty [19]. Their associated equivalence is the composition of $\sim^{E}$ wit variable renaming. This will be used in section 4.

Example: $t' = e+(y+z)$ is an instance of $t = x+x'$ with the match $\sigma = (x\backslash e,\ x'\backslash y+z)$. If contains a commutativity axiom for +, then $\sigma' = (x\backslash z+y,\ x'\backslash e)$ is an E-match from t to These two matches do not compare using $\preceq$. Now $\sigma'' = (x\backslash y+z,\ x'\backslash e)$ is another E-match an $\sigma' \sim^{E} \sigma''$. Note that $\mathcal{D}(\sigma) = \mathcal{D}(\sigma') = \mathcal{D}(\sigma'') = \{x,x'\}$.

Our next definitions handle the general case of a non empty set E of equations and a specialized for the standard empty case.

Definition 9: A **E-unifier** (or simply **unifier**) of two terms t and t' is a substitution σ such that $\sigma(t) \sim^{E} \sigma(t')$. If E is empty there exists a **most general unifier**, called **mgu**, for any pair of terms that has unifiers: all other unifiers of these two terms are actually instances of the mgu. If E is not empty, a basis for generating the set of unifiers by instantiation, whenever one exists, is called a **complete set of unifiers**. A **complete unification algorithm** computes such a basis for any two given terms. The algorithm is said **finite** if the basis is finite and **minimal** if any two unifiers in the basis do not compare using the E-subsumption preordering.

Standard unification is known to be solvable in linear time [65] but non linear algorithms are usually more efficient for many practical applications [7] [17]. Unification in equational theories is a much more difficult problem and the complexity of usual algorithms is not precisely known for most of them. However, complete and finite unification algorithms are known for a lot of practical theories including commutativity [70], associativity and commutativity [76] [58] [18], permutativity [36] and minus theory [45]. Some of these are not minimal. However, given such a complete set of unifiers that is finite, a minimal one with respect to the subsumption preordering can always be computed by eliminating those that are instances of others. Minimal sets are usually prefered for matter of efficiency. Most of our results however carry over to complete sets of unifiers that are infinite, except the decidability results. This enables us to apply some of our results to the associative case, since a complete but not finite algorithm is known [70].
Finally, we remark that the E-subsumption ordering is well founded for all theories of practical interest that have a complete finite unification algorithm. For the theory given in [19], the E-subsumption preordering is not well founded, but the unification problem in this theory is not finite.

We write **SU(t,t'), mgu(t,t'), SU(t,t',E), CSU(t,t',E)** for (respectively) the set of unifiers of t and t', their most general unifier, the set of E-unifiers of t and t' and a complete set of unifiers of t and t'. This last notation can be abbreviated as CSU(t,t') if no ambiguity arises.

Example:
Let $t=(x+y)+z$ and $t'=(c+x')$. Then $mgu(t,t')=\emptyset$, but $CSU(t,t',\{x+y=y+x\}) = (x'\backslash x+y,\ z\backslash e)$.

3.3 Rewrite Rules

Many theorical problems arise in equational theories that can be approached by the use of *rewrite rules* i.e. directed equations, or more generally by the use of *mixed sets of rules* R *and equations* E.

Definition 10: A **rewrite rule** is a pair of terms, denoted $l \rightarrow r$, such that $\mathcal{V}(r)$ is a subset of $\mathcal{V}(l)$. The **rewriting relation** $\rightarrow^{R,E}$ (or more simply $\rightarrow^{R}$ if E is empty) is defined as follows:
$t \rightarrow^{R,E} t'$ iff there exist an occurrence υ of $\mathcal{G}(t)$, a rule $l \rightarrow r$ of the set R of rewrite rules and a substitution σ such that $t/\upsilon \sim^{E} \sigma l$ and $t' = t[\upsilon \leftarrow \sigma r]$. t/υ is called a **redex**. We write $t \rightarrow^{R,E}_{[\upsilon,\sigma,l \rightarrow r]}$ if we want to make precise the occurrence, the substitution and the rule involved in the rewriting. Any of these subscripts may be omitted for simplicity.
$\stackrel{*}{\rightarrow}^{R,E}$ (resp. $\stackrel{+}{\rightarrow}^{R,E}$) denotes the reflexive transitive (resp. transitive) closure of $\rightarrow^{R,E}$ and is called the **derivation relation**. □

Example: Let (e+x')→x' be a rule and x+y=y+x be an equation. Then x+(y+e) is
R-normal form, but x+(y+e) $\rightarrow^{R,E}_{[1]}$ x+y.

As previously, $\rightarrow^R$ is the smallest relation which is compatible, closed under instantiation a
contains all pairs (l,r) for every l→r ∈ R. $\dot{\rightarrow}^R$ and $\dot{\rightarrow}^R$ are in addition transitive, and $\dot{\rightarrow}^R$ is al
reflexive. But they are generally not symmetric. In the following we will be interested in the
that are well founded.

Notice that R,E-reducibility is *decidable* whenever the matching problem is decidable in t
theory E. This is always the case if the theory E is empty. When the theory E is not empty, w
will either use the relation $\rightarrow^R$ or the relation $\rightarrow^{R,E}$. We will speak about **rewritings** for →
and E-**rewritings** for $\rightarrow^{R,E}$.

It must be well understood that the two relations $\rightarrow^{R,E}$ and $\sim^E\!\circ\rightarrow^R$ are different: If t $\rightarrow^{R,E}$
at occurrence υ, the E-equality steps may apply in t only at occurrences that are suffixes of
If t $\sim^E\!\circ\rightarrow^R$ t', then the E-equality steps may apply at any place in t. The two relations a
therefore the same only if $\rightarrow^{R,E}$ rewrites at the top.

Example: Let + be a binary infix operator with the properties:
associativity used as an equation: (x+y)+z = x+(y+z), *left identity* used as a rule: e+x→x.
Let us now consider the term t = (y+e)+z. By associativity, t $\sim^E$ y+(e+z) and this last ter
rewrites to y+z. However, t is in normal form for $\rightarrow^{R,E}$ because its subterm (y+e) is obvious
in normal form and the whole term t itself cannot be matched with the left member of the ru
using associativity since neither (y+e)+z nor y+(e+z) are instances of e+x.

3.4 Critical Pairs

Two reductions applied to a same term can sometimes overlap, yielding critical pairs:

Definition 11: A non variable term t' E-**overlaps** a term t at occurrence υ in $\mathcal{G}$(t) with
complete set Θ of E-**overlappings** iff Θ is a CSU(t/υ,t',E). The classical **overlappin**
associated with mgu(t/υ,t') is obtained for E empty. Given two rules g→d and l→r such tha
$\mathcal{V}(g)\cap\mathcal{V}(l) = \emptyset$ and l overlaps g at occurrence υ with the complete set of E-overlappings Θ, the
the set {≪p,q≫ | p=θd and q=θ(g[υ←r]) for any θ in Θ} is called a **complete set of E-critica**
pairs of the rule l→r on the rule g→d at occurrence υ (a trivial one if υ=ε, l=g and r=d
The complete set of critical pairs is simply a **critical pair** if E is empty. With eac
(E-)overlapping θ is associated the (E)-**overlapped term** θg which produces the (E-)critical pa
≪p=θd,q=θ(g[υ←r])≫ by rewriting with the rules g→d and l→r.

Notice that l→r and g→d do not play symmetric roles in this definition.
Let us point out that a term t may be considered as a **variable overlapping** of any variable
such that #(x,t)=0 on itself at any occurrence υ∈$\mathcal{G}$(t), υ≠ε, with the mgu σ =(x\ t/υ). Suc
overlappings produce **variable critical pairs** of an equation g=x on the rule l→r b
overlapping the variable x with any non variable strict-subterm of l. These critical pairs a
needed to handle linear rules together with equations of the form g=x. We want to point ou
that they allow a generalization of results in [30] and [37] to the case where such equations a

allowed. However, they will not be considered in this framework where the main interest is in
decidability results that do not hold if axioms like g=x can be used as equations.

Let **SCP**(R,R), **SCP**(E,R) and **SCP**(R,E) be the sets of non trivial critical pairs for respectively:
all l→r and g→d belonging both to R, all l→r and r→l for l=r in E on all g→d in R, all l→r in
R on all g→d and d→g for g=d in E. Note that **top critical pairs**, i.e. critical pairs coming
from an overlapping at occurrence ε, are not to be considered for SCP(R,E) since they already
belong to SCP(E,R).
Let also **CSECP**(R,R) and **CSECP**(R,E) be the complete sets of non trivial E-critical pairs for
respectively all l→r in R on all g→d in R, and all l→r in R on all g→d and d→g for g=d in E.
As previously, but for a slightly different reason that will become clear later, top critical pairs
are not to be considered for CSECP(R,E).
The notations **SCP**(l→r,g→d) and **CSECP**(l→r,g→d) will also be used to make the rules or
equations precise.

Example:
Let (x+y)+z→x+(y+z) and (e+x')→x' be rules and (y'+e)=y' be an equation. Then:
≪p=e+(y+z),q=y+z≫ is a critical pair of the second rule on the first at occurrence 1,
associated with the overlapped term (e+y)+z. Note that p→^{R}q.
≪p=e+(e+z),q=z≫ is a E-critical pair of the second rule on the first at occurrence ε,
associated with the E-overlapped term (e+e)+z $\sim^E$ e+z. Note that p$\dot{\rightarrow}^R$q.
≪p=x+(y+e),q=x+y≫ is a critical pair of the equation on the first rule at occurrence ε,
associated with the overlapped term (x+y)+e. Note that p $\sim^E$ q.
≪p=x+(y+z),q=((x+y)+e)+z≫ is a variable critical pair of the equation on the first rule at
occurrence 1, associated with the overlapped term (x+y)+z. Now q reduces to p in two
different ways: q $\dot{\rightarrow}^R$ x+(y+(e+z)) $\rightarrow^R_{[2,2,e+x'\rightarrow x']}$ p or q $\rightarrow^{R,E}_{[\epsilon,(x+y)+z\rightarrow x+(y+z)]}$ p.

In the following, we use R-reductions as well as R,E-reductions or more generally combinations
of both of them. As a consequence, we will have *critical pairs* as well as complete sets of
critical pairs, according to the kind of reduction associated with each rule.
R-reductions yield critical pairs as indicated by the critical pairs lemma:

Critical pairs lemma: [30] Assume t $\rightarrow^R_{[\epsilon,\sigma,l\rightarrow r]}$ t_1 or t $\vdash\!\!\!\dot{\rightarrow}^E_{[\epsilon,\sigma,l\rightarrow r]}$ t_1 and t $\rightarrow^R_{[\nu,\sigma,g\rightarrow d]}$ t_2 or t
$\vdash\!\!\!\dot{\rightarrow}^E_{[\nu,\sigma,g\rightarrow d]}$ t_2 with ν in $\mathcal{G}$(l). Then there exists a critical pair ≪p=τr,q=θ(l[ν←d])≫ of the rule or
the equation g→d on the rule or the equation l→r at occurrence ν and a substitution τ such that
θ=mgu(l/ν,g) and σ=τθ [$\mathcal{V}$(g)∪$\mathcal{V}$(l)], therefore $t_1 = \tau p$ and $t_2 = \tau q$. □

Note that we use the same substitution σ for both redexes t and t/ν. This is legal, since we can
always assume that $\mathcal{V}(l)\cap\mathcal{V}(g)=\emptyset$ without lost of generality: just rename the variables when
needed. Although it was given for standard critical pairs, the lemma remains true for the case
of variable ones with the same proof. The verification is left to the reader.

R,E-reductions yield complete sets of E-critical pairs:

E-Critical pairs lemma: [37]

Assume $t \rightarrow^{R}_{[\epsilon,\sigma,l\rightarrow r]} t_1$ or $t \vdash\!\!\!\rightarrow^{E}_{[\epsilon,\sigma,l\rightarrow r]} t_1$ and $t \rightarrow^{R,E}_{[\nu,\sigma,g\rightarrow d]} t_2$ with ν in $\mathcal{G}(l)$ and $\nu \neq \epsilon$. Then there exists a critical pair $\ll p=\theta r, q=\theta(l[\nu \leftarrow d])\gg$ in a complete set of E-critical pairs of the rule $g \rightarrow d$ on the rule $l \rightarrow r$ at occurrence ν and a substitution τ such that $\theta \in CSU(l/\nu, g, E)$ and $\sigma \sim^{E} \tau\theta\ [\mathcal{V}(g) \cup \mathcal{V}(g)]$, therefore $t_1 \sim^{E} \tau p$ and $t_2 \sim^{E} \tau q$.

Note that we overlap R,E-reductions with R-reductions or with a one step E-equality. No other case has to be considered, as shown by a careful examination of our two local definitions of coherence and confluence given in section 2. In addition, we don't consider the case where $\rightarrow^{R}$ applies at an occurrence ν and $\rightarrow^{R,E}$ at occurrence ϵ: the E-critical pairs lemma would be false for such cases! We will see in our main theorem how to treat this case without using the lemma.

The E-critical pairs lemma is proved in a way similar to Huet's, but we need to make precise where E-equality steps can take place from t_1 to τp: since $t_1 = \sigma r$ and $\tau p = \tau\theta r$, they actually take place in the substitution part of t_1 and not in $\mathcal{G}(l)$. This remark is essential for carrying out the proofs in [67] and in [37]. Here we will only need to assume that it does not take place at the top of t_1, thanks to multiset induction.

3.5 Decidability of the R'-Church-Rosser Property for ETRS

Our next goal is to restrict local confluence modulo E of $\rightarrow^{R'}$ with $\rightarrow^{R}$ and local coherence modulo E of $\rightarrow^{R'}$ to a convergence check on a finite number of critical pairs or E-critical pairs for a suitable relation $\rightarrow^{R'}$.
The two cases where $\rightarrow^{R'} = \rightarrow^{R}$ or $\rightarrow^{R'} = \rightarrow^{R,E}$ are already studied in [37]. The first one requires R to be left linear and yields Huet's classical results on *confluence modulo*, while the second requires the computation of E-*critical* pairs using a complete E-unification algorithm and yields a generalized version of Peterson and Stickel's results.

A first subgoal is to generalize these two cases by splitting the set R of rules into disjoint sets Rl and Rnl with left linear rules only in Rl and use for $\rightarrow^{R'}$ the relation $\rightarrow^{Rl \cup Rnl,E}$ defined to be $\rightarrow^{Rl} \cup \rightarrow^{Rnl,E}$. The relation $\rightarrow^{Rl \cup Rnl,E}$ was introduced and also studied in [37], with a less powerful proof technique however, yielding more restrictive results. Note that linear rules can be in Rnl which allows us to consider $\rightarrow^{R}$ and $\rightarrow^{R,E}$ as two particular cases of $\rightarrow^{Rl \cup Rnl,E}$. Making Rl=∅ yields the classical Peterson and Stickel relation and Rnl=∅, provided all rules are linear, yields the standard rewriting relation $\rightarrow^{R}$.
In the following, R' is used instead of the full notation Rl∪Rnl,E for simplicity.

A second subgoal is to prove the decidability of the R'-Church-Rosser property modulo E, provided R is E-terminating. This is done by proving that the R'-Church-Rosser modulo E property is equivalent to checking that the normal forms of finitely many critical pairs are E-equal. Such a result was already known for the case where Rnl is empty and rules in R are left linear. We will prove it for $\rightarrow^{Rl \cup Rnl,E}$, provided the theory E has a complete and finite unification algorithm, and also finite congruence classes if Rnl is not empty.

Since $\rightarrow^{Rl \cup Rnl,E}$ is included in $\rightarrow^{R,E}$, the Rl∪Rnl,E-Church-Rosser modulo E property implies the R,E-Church-Rosser modulo E property. As already noticed before, we must be aware of the fact that R can be R,E-Church-Rosser modulo E and not Rl∪Rnl,E-Church-Rosser modulo E for a non empty set Rl of left linear rules. Examples will be given at the end of the section.

The basic tool of the proof is multiset induction on the reduction relation $\rightarrow^{R/E} \cup \rightarrow^{\text{strict-subterm/E}}$, denoted $\mapsto$ in the following. The termination of $\mapsto$ is simply based first on the fact that $\rightarrow^{R/E}$ and $\rightarrow^{\text{strict-subterm}}$ are both noetherian and second on the following commutation lemmas, whose use as many times as needed permits to the transformation of a $\mapsto$-derivation into a $\rightarrow^{R/E}$-derivation followed by a $\rightarrow^{\text{strict-subterm}}$-derivation, as stated in the next corollary.

Lemma 12: $s \stackrel{+}{\rightarrow}^{\text{strict-subterm/E}} t \stackrel{+}{\rightarrow}^{R/E} t' \Rightarrow \exists s' \in \mathcal{T}$ such that $s \stackrel{+}{\rightarrow}^{R/E} s' \stackrel{+}{\rightarrow}^{\text{strict-subterm}} t'$.

Proof: By induction on the number n of $\rightarrow^{\text{strict-subterm}}$-reductions.
The basic n=0 case is straightforward.
Else, let $s \stackrel{+}{\rightarrow}^{\text{strict-subterm/E}} s_1 \rightarrow^{\text{strict-subterm/E}} t \stackrel{+}{\rightarrow}^{R/E} t'$. By definition, $\exists s_1'', t_1'' \in \mathcal{T}$ such that $s_1 \sim^{E} s_1'' \rightarrow^{\text{strict-subterm}} t_1'' \sim^{E} t \stackrel{+}{\rightarrow}^{R/E} t'$, therefore $s_1 \sim^{E} s_1'' \rightarrow^{\text{strict-subterm}} t_1'' \stackrel{+}{\rightarrow}^{R/E} t'$.
Since $\rightarrow^{\text{strict-subterm}}$ can be performed after the reductions instead of before, we get $s_1 \sim^{E} s_1'' \stackrel{+}{\rightarrow}^{R/E} s_1' \rightarrow^{\text{strict-subterm}} t'$ for some s_1', therefore $s_1 \stackrel{+}{\rightarrow}^{R/E} s_1' \rightarrow^{\text{strict-subterm}} t'$.
The result is then achieved by applying the induction hypothesis to $s \stackrel{+}{\rightarrow}^{\text{strict-subterm/E}} s_1 \stackrel{+}{\rightarrow}^{R/E} s_1'$. □

Corollary $\mapsto^{+} = \stackrel{+}{\rightarrow}^{R/E} \circ \stackrel{+}{\rightarrow}^{\text{strict-subterm}} \circ \stackrel{+}{\rightarrow}^{\text{strict-subterm/E}}$ □

Infinite chains can thus occur iff strict-subterm/E is not well-founded.

Lemma 13: $t \rightarrow^{\text{strict-subterm/E}} t' \Rightarrow \exists t''$ such that $t \sim^{E} t'' \rightarrow^{\text{strict-subterm}} t'$.

Proof: Straightforward induction on the number of derivations. □

As a consequence of these lemmas, we get the following termination result:

Proposition 14: Assume R is E-terminating and E-conguence classes are finite. Then $\mapsto$ is well founded.

Proof: After the previous corollary, we have to prove that strict-subterm/E is well founded. From the previous lemma, this must be the case if E-congruence classes are finite, since the size of the terms in a given congruence class becomes bounded. □

Note that the problem of deciding whether a set of equations generates finite congruence classes only is undecidable in general [71]. On the other hand, many theories of interest have this property. Some interesting ones do not however, like equipotency --x = x or even identity. These kind of axioms do not fall into the scope of our decision results.

Let us now make some comments about the termination of R/E : Assume that E contains an equation g=d which is erasing i.e. there exists a variable x of $\mathcal{V}(g)$ which is not in $\mathcal{V}(d)$. Let $l \rightarrow r$ be a rule of R, σ and σ' substitutions such that $\sigma(x)=l$ and $\sigma'(x)=r$. Then

$d=\sigma d \sim^{\mathcal{E}} \sigma g \not\rightarrow^{R} \sigma' g \sim^{\mathcal{E}} \sigma' d=d$ and the E-termination property is not satisfied. Therefore
will assume in the following that equations in E are *non erasing* i.e. $\mathcal{V}(g) = \mathcal{V}(d)$ for
equation g=d in E.
Assume now that E contains an equation t=x where x has at least two occurrences in t. N
that such an equation makes strict-subterm/E non well-founded. The same is true for R/E,
we now see: Let l→r be any rule. Then l is E-equal to a term with several occurrences of l. O
can rewrite one of them and start the process again with another occurrence of l. For instan
in the case of the idempotency equation x = x+x, for any rule l→r:
$l \sim^{\mathcal{E}} l+l \rightarrow^{R} r+l \sim^{\mathcal{E}} r+l+l \rightarrow^{R} r+r+l \ldots$. The same problem actually arises with any instan
of such an axiom where x is replaced by any term with variables. As a consequence, the o
allowed axioms of the form g=x, with respect to E-termination of R, satisfy #(x,g)=1. Not
however that these axioms are forbidden for strict-subterm/E to be well-founded, at least if g
not x itself.

Let us now give our main result:

Theorem 15: Let R = Rl∪Rnl be an E-terminating set of rules such that Rl is left linear,
complete and finite unification algorithm exists for the theory $\sim^{\mathcal{E}}$, and E-congruence classes a
finite. Then R is Rl∪Rnl,E-Church-Rosser modulo E iff:
All *confluence pairs* ≪p,q≫ in SCP(Rl,Rl) ∪ SCP(Rl,Rnl) ∪ CSECP(Rnl,Rnl) ∪ CSECP(Rnl,
are R/E-confluent modulo E.
For any *coherence pair* ≪p,q≫ in SCP(Rl,E) ∪ CSECP(Rnl,E), ∃p' such that $p \rightarrow^{R'} p'$ and
pair (p',q) is R/E-confluent modulo E.
For any *coherence pair* ≪p,q≫ in SCP(E,Rl), ∃q' such that $q \rightarrow^{R'} q'$ and the pair (p,q')
R/E-confluent modulo E.

Proof:
For the only if part, we use theorem 4 to show that local properties are satisfied, thus
satisfied for the particular case of critical pairs. Since these pairs are R'-confluent modu
E, they are R/E-confluent modulo E by inclusion of $\rightarrow^{R'}$ into $\rightarrow^{R/E}$.
The if part starts as the proof of theorem 4 (part (3) ⇒ (4)) and we assume it done (noti
that it implies the use of theorem 4 because we prove statement (4) instead of stateme
(1) of this theorem). The difference is that we have to prove the two properties of lo
coherence and local confluence using the sets of critical pairs. This is done by multi
induction on ↦. Notations are the same as in theorem 4. In particular, t_n is the o
introduced in its proof , t↓ is a R'-normal form of t and $p \overset{*}{\downarrow} q$ iff (p,q) is R'-conflue
modulo E.
Let us first prove local coherence, specifically:
$\forall t,t',t''$ such that $t_n \sim^{\mathcal{E}} t$, $t \vdash^{E}_{[\nu,\sigma,g\rightarrow d]} t'$ and $t \rightarrow^{R'}_{[\nu,\sigma,l\rightarrow r]} t''$, $\exists t'_1$ such that $t' \rightarrow^{R'} t'_1$ an
$t'_1 \overset{*}{\downarrow} t''$.
As in the critical pairs lemmas, we use the same substitution for both redexes. Note al
that we prove a slightly more general local coherence property than the one needed in t
proof of theorem 4, because it will be needed in the rest of the proof.
Let us now discuss different cases according to the respective positions of υ and ν:

- *case 1:* υ and ν are disjoint occurrences. Then the $\rightarrow^{R'}$ and $\vdash^{E}$ steps commute.

- *case 2:* For the remaining cases, one occurrence is a prefix of the other. The result is now straightforward if this occurrence (assume it is υ) is not the empty occurrence ϵ, since we can apply multiset induction to the multiset $\{t'/\upsilon, t/\upsilon\ t''/\upsilon\}$, yielding $t'/\upsilon\downarrow \sim^{E} t''/\upsilon\downarrow$, with $t'/\upsilon \neq t'/\upsilon\downarrow$, since R is E-terminating. By the way, this is exactly the coherence property and it can be now easily lifted to (t', t, t'') by adding the missing context.
  We therefore assume in the following that the prefix occurrence is ϵ.

- *case 3:* ν is a prefix of υ, therefore we assume $\nu=\epsilon$, with l→r in Rnl.
  It follows that $t' \vdash^{E} t$, thus $t' \rightarrow^{Rnl,E}_{[\epsilon,l\rightarrow r]} t''$. This is the reason why critical pairs of equations on rules of Rnl are not to be considered.

- *case 4:* ν is a prefix of υ, therefore we assume $\nu=\epsilon$ with l→r in Rl, and $\upsilon \notin \mathcal{G}(l)$.
  Then the result follows in the same way as in case 6 below.

- *case 5:* ν is a prefix of υ, therefore we assume $\nu=\epsilon$, with l→r in Rl, and $\upsilon \in \mathcal{G}(l)$.
  Then the result follows classically from the critical pairs lemma, using a critical pair of SCP(E,Rl). Note that g=d can be the reflexivity axiom and that no critical pair is needed for this particular case.

- *case 6:* υ is a prefix of ν, therefore we assume $\upsilon=\epsilon$, and $\nu \notin \mathcal{G}(g)$, thus $\nu \neq \epsilon$.
  Because no overlapping occurs in this case, there exists an occurrence ω which is a prefix of $\nu=\omega\nu'$ and such that $g(\omega)$ is a variable x. Let us now define a substitution θ such that $\theta(y) = \sigma(y)$ for any y distinct from x and $\theta(x) = \sigma(x)[\nu' \leftarrow \sigma r]$. As equations are non erasing, x occurs at least once in d. It is then easy to check that $t' \not\rightarrow^{R'} \theta d$ and $t'' \not\rightarrow^{R'} \theta g$ which ends this case, since $g=d \in E$. Note that our definition for coherence allows rewriting from t'', which was not allowed by Peterson and Stickel's, and this was the reason why equations were required to be linear in their work.

- *case 7:* υ is a strict prefix of ν, therefore we assume $\upsilon=\epsilon$, $\nu\neq\epsilon$, and $\nu \in \mathcal{G}(g)$ with l→r in Rl. Then the result follows classically from the critical pair lemma, using a critical pair of SCP(Rl,E). Top critical pairs are useless here since $\nu\neq\epsilon$.

- *case 8:* υ is a strict prefix of ν, therefore we assume $\upsilon=\epsilon$, and $\nu\neq\epsilon$ and $\nu \in \mathcal{G}(g)$ with l→r in Rnl. As a consequence, $\mathcal{G}(g) \neq \emptyset$.
  Once more, there will be no need of top critical pairs since $\nu\neq\epsilon$.
  By the E-critical pairs lemma, there is a pair ≪p=θd, $q=\theta(g[\nu\leftarrow\theta r])$≫ in CSECP(Rnl,E) and a substitution τ such that $\sigma \sim^{\mathcal{E}} \tau\theta\ [\mathcal{V}(g)\cup\mathcal{V}(l)]$, therefore $t' \sim^{\mathcal{E}} \tau p$ with E-equality steps taking place out of $\mathcal{G}(d)$ and $t'' \sim^{\mathcal{E}} \tau q$. By hypothesis, $p \rightarrow^{R'}_{[\omega]} p'$ and the pair (p',q) is R/E-confluent modulo E, therefore $\tau p \rightarrow^{R'}_{[\omega]} \tau p'$ and the pair $(\tau p', \tau q)$ is R/E-confluent modulo E, which implies $\tau p' \sim^{A} \tau q$ with a proof containing terms that are all smaller than $\tau p'$ or τq for ↦. This is step 1 on Figure 4.
  Since the multiset $\{t'' \ldots \tau q \ldots \tau p' \ldots t'\}$ contains terms like τ and t' which are not smaller than t for ↦, the induction hypothesis cannot be applied to this multiset.

To overcome this difficulty, we now show that t' is R'-reducible to some t'_2 related $\tau p'$ by a multiset of A-equal terms all smaller than t:
As already noted, we know from the E-critical pairs lemma that the equality st between τp and t' take place out of $\mathcal{G}(d)$, thus at some occurrences $o_1 \ldots o_n$ different from ϵ. As a consequence, we may apply the already proved l coherence property (cases 1 to 6), starting from $(\tau p'\ \tau p\ t'_1)$ until we reach t'. Th are steps 2 and 3 on Figure 4.
The conclusion then follows by applying the induction hypothesis to the multi $\{t'' \ldots \tau q \ldots \tau p' \ldots t''_1 \ldots t'_2\}$.

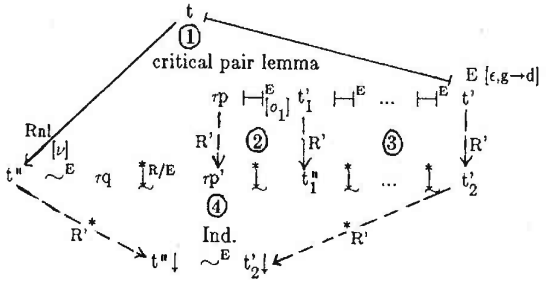


Figure 4: *Local Coherence Proof, case 8*

Let now prove local confluence, specifically:
For any t' and t'' such that $t_n \to^{R'}_{[\upsilon,\sigma,g\to d]} t'$ and $t_n \to^{R}_{[\nu,\sigma,l\to r]} t''$, $t' \downarrow t''$. As before, discuss by cases according to the respective positions of υ and ν.

- *case 1:* υ and ν are disjoint. Then the diagram commutes and we are done.

- *case 2:* neither υ nor ν is ϵ. This case works as case 2 in the proof of local coherenc

- *case 3:* $\nu = \epsilon$. The subcase where there is no overlapping works as usual. The oth subcase is proved by the critical pairs lemma if $\to^{R'}$ is $\to^{Rl}$, and by the E-criti pairs lemma if $\to^{R'}$ is $\to^{Rnl,E}$, followed (in both cases) by multiset induction. T will use, in the first case a critical pair in SCP(Rl,Rl) or SCP(Rl,Rnl), and in t second case an E-critical pair in CSECP(Rnl,Rnl) or CSECP(Rnl,Rl). Note that t critical pairs are needed here.

- *case 4:* $\upsilon = \epsilon$ and $\nu \neq \epsilon$. The subcase without overlapping works as usual. T other subcase is proved using the critical pair lemma if $\to^{R'}$ is $\to^{Rl}$. If $\to^{R'}$ is $\to^{Rn}$ the E-critical pair lemma does not apply and we need the use of the already prov local coherence property. Since $t_n \to^{R'} t'$, therefore $t_n \to^{R/E} t'$, there exists t su that $t_n \sim^E t \to^{R}_{[\epsilon,g\to d]} t'$. On the other hand, $t_n \to^{R}_{[\nu,l\to r]} t''$. We therefore can ap local coherence n times starting from t_n until we reach t with the following situati $t'' \downarrow t''_n \downarrow \ldots \downarrow t''_1$, $t \to^{R'} t''_1$ and $t \to^{R}_{[\epsilon,g\to d]} t'$.

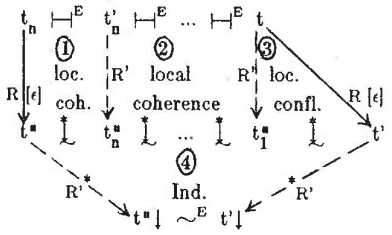


Figure 5: *Local Confluence Proof, case 4*

These are steps 1 and 2 on Figure 5.
Since the R-reduction applies now at the top of t, we can apply the already proved case 3 to close the diagram, thus $t''_1 \downarrow t'$. This is step 3 on Figure 5. The proof is then completed by using the Church-Rosser property applied to the multiset $\{t'' \ldots t''_1 \ldots t'\}$. This is step 4 on Figure 5.
The reader is encouraged to check that the critical pairs used all along the proof are exactly those defined previously. □

The proof makes clear why left linear rules are easier to handle: what happens at a variable place in an instance of rule happens once which makes the coherence diagrams commute by simple rewriting. This is not the case for non left linear rules: If an equality is applied at a variable place, the first rewriting needed in the coherence property must incorporate the same equality at the other variable places (of the same variable of course), since an equality step is not allowed here.

A weaker result was given in [37], where it was shown that the R'-Church-Rosser property was true under the stronger assumption that coherence pairs of rules in Rnl on equations have the property that p is always reducible to p' at an occurrence in $\mathcal{G}(d)$. As a matter of fact, this result does not require the equational theory E to have finite congruence classes. The reason is that we do not need any induction with strict-subterm/E in this case, since case 8 in the proof of local coherence becomes vacuous and cases 2 in both local proofs are used only for convenience. On the other hand, the requirement on these critical pairs forbids an equivalence result, because this strong assumption on the way the coherence pairs become convergent cannot be a consequence of the R'-Church-Rosser property. The question arises now whether our equivalence result is true for those theories that have infinite congruence classes. We conjecture it is. A way to prove it would be to show that we do not need the full power of the relation $\mapsto$ in the proof, i.e. the infinite subterm/E-chains cannot occur. For doing that, a useful assumption could be that the critical pairs are of minimal size, which can be achieved by choosing an adequate E-unification algorithm. Note that such an algorithm can always be obtained from a standard complete one by checking each unifier for minimality with respect to its size.

As a corollary of the previous result we can decide the R'-Church-Rosser property of a set R of

rules modulo a set E of equations by checking for E-equality the R"-normal forms of the pair (p,q) or (p',q) or (p,q') of the previous theorem, for any reduction relation R" ranging between R' and R/E.

Theorem 16: Let R = RlURnl be an E-terminating set of rules such that Rl is left linear, a complete and finite unification algorithm exists for the theory $\sim^{E}$, and E-congruence classes are finite. Then the R'-Church-Rosser property is decidable.

Proof: As a matter of fact, the E-equality of the previous confluence and coherence pairs implies the R'-Church-Rosser property. Conversely, the R'-Church-Rosser property implies the desired property with R'-normal forms instead of R"-normal forms. But we know from theorem 3 that R/E-normal forms and R'-normal forms are the same, thus R"-normal forms too, since $\rightarrow^{R"}$ is included in $\rightarrow^{R/E}$ and contains $\rightarrow^{R'}$. □

We want to emphasize that we allow normal forms of critical pairs to be computed with any reduction relation $\rightarrow^{R"}$ ranging between $\rightarrow^{R'}$ and $\rightarrow^{R/E}$ instead of $\rightarrow^{R'}$. This is very important in practice, because it allows E-equality steps to be performed during the reduction process by $\rightarrow^{R'}$. This arises in a natural way in many practical cases: For computing associative-commutative (AC) critical pairs for instance, *flattened terms* can be used to make the unification process easier. Flattening results in applying the rule f(x,f(y),z)→f(x,y,z), where x, y and z stand for (maybe empty) vectors of variables, for any AC-symbol f, whose arity is now varying. This can result in modifications of the starting terms within a AC-equivalence class. Our result proves that it does not alter the soundness of the whole process. Of course, this is not completely true for coherence pairs, since a first R' step is needed here: only then can terms be flattened!

Let us now come back to a main practical problem: which relation R' is the best one for efficiency of computations? Clearly, the more rules there are in Rnl, the more inefficient the rewritings are. On the other hand, the more rules there are in Rnl, the more powerful the rewriting. We therefore want to find the most efficient rewriting relation $\rightarrow^{R'}$ that provides a R'-Church-Rosser ETRS.
The previous proof makes clear that the relation R' is linked to the rules that are used to reduce p to p' or q to q' for the coherence pairs (p,q). As a consequence, if a linear rule of Rnl is never used with the full power of the relation $\rightarrow^{Rnl,E}$ (recall that $\rightarrow^{Rnl} \subset \rightarrow^{Rnl,E}$) to reduce such a p or a q, it can maybe be dropped to Rl. But this is only a hint: new critical pairs of the equations on this rule have to be computed now and they must be confluent modulo E.
This discussion suggests that all rules should be first in Rnl, trying then to construct an Rl as indicated. Clearly, some work will be needed to find good stategies for that. Let us now consider an example.

Example[3]*:* Let 0, 1, -1 be constants and + a binary infix operator.
Let 0 + x' → x', 1 + -1 → 0 and (x' + 1) + -1 → x' be the rules of R
and (x+y)+z = x+(y+z) and x+y = y+x the equations of E.

[3]This example is due to Etienne Paul, CNET, Issy-les-Moulineaux, FRANCE, who pointed out the previous problem.

It can be verified (using a Knuth and Bendix implementation, for example the FORMEL system of G. HUET at INRIA or the REVE system [55, 47]) that this set of rules is R,E-Church-Rosser. As these rules are all left linear, they can all be dropped to Rl. In that case, the set of rules is not Church-Rosser anymore. Moreover, its completion generates an infinite set of rules, as we will see in the next section. Which rules do we really need to have in Rnl? First of all, we need the first rule in Rnl to make the following coherence E-critical pair convergent: ≪p=(x+0)+z,q=x+z≫ obtained from the superposition of 0+x' on x+(y+z) at occurrence 1 with the unifier (y\0, x'\z). Then p reduces to q using the rule 0+x' → x' modulo commutativity. Checking the other rules is left to the reader or her Knuth and Bendix implementation.
Notice that only commutativity was used in this example: the question arises whether our results can be extended to the case where each rule l→r has an associated subset E' of E that can be used for rewriting with the relation $\rightarrow^{l\rightarrow r,E'}$. Clearly, the framework used here is powerful enough to solve this problem and see what are exactly the critical pairs to be computed for such a rewrite relation. In that case, a linear rule can be considered to have both non-linear and linear status according to the respective sets E' and E-E'. Non linear status will require the computation of complete sets of critical pairs whereas linear status will require the computation of classical critical pairs. In this case, we need complete sets of unifiers for a subtheory E' of the whole theory E. For the last example, the subtheory is the commutative one. However a theory can have a complete and finite unification algorithm whereas at the same time one of its subtheories does not have one. For instance, the AC-theory has one although the associative one does not have a finite one. This remark limits the practical interest of such a method. It can however be interesting for speeding up reductions in some cases.

Example: Let us consider an interesting algebraic theory having constants, a unary prefix operator - and a binary infix operator + with the following properties:
-x = x, -(x + y) = -y + -x, -x + (x + y) = y and (y + x) + -x = y.
These axioms can be used as rules. Applying then the standard Knuth and Bendix completion procedure carries on a divergence for any possible orientation of the generated rules. However, the infinite set of rules can be described with a finite set of *meta rules* [41, 46]. On the other hand, we can try to use as equations those rules causing the divergence of the process. Let us assume that the two first axioms are used as equations and the two others as rules. The equational theory associated with the first two axioms has a finite and complete unification algorithm [45]. Checking the set of rules shows that it has the R,E-Church-Rosser property [46]. This is a good example of an ETRS that does not fall into the scope of our decidability results since the congruence class of x is infinite, whereas it can be worked out using the results of [37], where sufficient conditions are given for testing for the Church-Rosser property.

Before ending this section, let us point out that little is known about E-termination. A theorical study of the problem may be found in [39] and some effective, yet complex, methods in [14] and [68].
On the other hand, some progress has also been made recently for checking the Church-Rosser property using a weaker termination property than E-termination, namely the termination of the used reduction relation. Two different approaches can be used, one is reported in [64] and [40], the other in [39]. The properties used in place of coherence in these works are much

stronger and involve many restrictions in practice. The second approach, however, is w
suited to some equational theories like the associative commutative one.

4 The E-completion Procedure

Given a set of axioms, the E-completion procedure attempts to obtain a confluent and coherent set of rules that are as far *inter-reduced* as possible. Completely interreduced sets of rules can be expected to have a normal form property, i.e. they depend upon the equational theory $\sim^A$ rather than upon a particular presentation of it. Although this goal is easily achieved for the standard completion algorithm [59], it turns out to be one of the main difficulties in the design of the general E-completion procedure. As a consequence, we will discuss several variants of our algorithm, with only one of them having the property. On the other hand, we will see how to get a completely interreduced set of rules from a given Church-Rosser set of rules that does not have the property. An second expected property of a completion algorithm is its use for testing for the Church-Rosser property of a given set of rules. This can be achieved by using a very simple trick: to avoid the starting rules to be reduced, they can be protected during an initialization phase. Another way to achieve this goal is to apply the previous preprocessing on the starting set in order to start E-completion with a set of interreduced rules. Finally, our algorithm can be used as a semi-decision procedure for A-equality as usual.

4.1 A Very General Completion Procedure for ETRS

As a main feature of our algorithm, coherence is *dynamically* ensured, unlike in Peterson and Stickel's AC-completion algorithm, where coherence is ensured by systematically adding the so called AC-*extended rules* $f(f(x,l_1),l_2)\rightarrow f(x,r)$ with $x\notin\mathcal{V}(l_1)\cup\mathcal{V}(l_2)$, to those rules $f(l_1,l_2)\rightarrow r$ having an AC-top function symbol f [67].

The E-completion procedure works on a set P of pairs, a set R of rewrite rules, a constant set E of equations and a well founded **E-reduction ordering**, i.e. a compatible quasi ordering $\succeq$ such that its associated equivalence $\simeq$ contains $\sim^E$ and its associated strict ordering $\succ$ is well founded. See [10] for an introduction to termination proof methods and [39] for an introduction to E-termination proof methods.

There are two ways of using the E-completion procedure, that differ upon initialization:
•To built an R'-Church-Rosser set of rules from a given set of axioms. In that case, it starts with an empty set R of rules and a set P of pairs initialized with those axioms that are not in E.
•To check the R'-Church-Rosser property of an ETRS made up from a set E of equations and an E-terminating set R of rules. In that case, it starts with an empty set P of pairs and the set R itself, whose rules must be *protected*, else some of them could be removed if they are reducible by some other rules as explained next. If the Church-Rosser property is not satisfied, the set of rules may then be completed as before.

During the completion process, P is updated by picking a pair at a time in order to make a new rule and dropping in, on the one hand those rules whose left-hand side becomes reducible by a newly introduced rule, and on the other hand the newly computed critical pairs. Pairs in P are further compared using the E-reduction ordering $\succ$, producing rules for R.
The rewriting system R is divided into a set Rl of left-linear rules and a set Rnl of non-left-linear rules. Each rule $l\rightarrow r$ is:
•*labelled* by an integer n and denoted $n{:}l\rightarrow r$, $l_n\rightarrow r_n$, or simply n. The label tells us when the rule was created.

•*marked* as soon as all its critical pairs with the axioms of E and with the previously crea
rules have been computed.

In addition, two particular features have been added in order to ensure the coherence proper
protected rules and *extensions*. Let $S = \{\ll p=\sigma d, q=\sigma(g[v \leftarrow r]) \gg \mid \sigma \in CSU(g/v, l, E)\}$ be the se
E-critical pairs of a linear or nonlinear rule n:l→r on an equation g→d of E at occurrence
Remember that the left member, p, of any of these E-critical pairs in S has to be R'-reducible
ensure coherence.
•Assume first that p is R'-reducible, but only at the top by a non-left-linear rule $k:l_k \rightarrow r_k$ su
that p is an E-instance of l_k and not simply an instance. In that case, p is said to
top-reducible by the non linear rule k and k **protected** for coherence of n. Witho
protection, such a coherence critical pair ≪p,q≫ could satisfy the required property at so
step of the completion and not at a further step if the left hand side of the previous non-l
linear rule k has become reducible and the rule be dropped into P. As long as a rule
protected, its left hand side is not checked for reduction and the rule cannot be removed fr
the current rewriting system. Note that a rule can be protected for coherence of several rules.
•Assume now that the rule n is not in Rl and there exists in S an E-critical pair ≪p,q≫ wh
left member is R'-irreducible. Then an **extended rule** for n is added to Rnl, obtained from t
extended pair ≪g[v←l],g[v←r]≫ which must be directed from left to right, since l≻r and ≻
compatible. As Peterson and Stickel's associative commutative ones, extended rules reduce
top all left members of E-critical pairs in S, since p is an instance of g[v←l] modulo $\sim^{\mathcal{E}}$:
$p = \sigma d \sim^E \sigma g = \sigma(g[v \leftarrow g/v]) = \sigma g[v \leftarrow \sigma(g/v)] \sim^E \sigma g[v \leftarrow \sigma l] = \sigma(g[v \leftarrow l])$, using the homomorph
properties of substitutions.
As a consequence, no protection is needed for the other coherence pairs of l→r on g→d
occurrence v. Note in addition that an extended rule is a particular case of a rule in R
reducing p at the top. They are therefore implicitly protected.

A similar problem arises with coherence pairs coming from the superposition of a rule in Rl w
an equation. Under the same previous circumstances, non left linear rules will be protected
extensions added. However, the extensions are much simpler here: It is the rule p→q or q→
according to which set SCP(Rl,E) or SCP(E,Rl) the coherence pair ≪p,q≫ belongs to (the ri
side of the coherence pair must be reducible). These extensions may belong to Rl and/or to R
but do not need any protection, since they reduce p (or q) without using any E-equality steps
before.

Let **Ext(n)** be the set of those rules added by the procedure for coherence of n or of a rule
Ext(n). When the left hand side of rule n becomes reducible, rule n and all the non protect
rules in Ext(n) are removed from the current rewriting system, which may require a compl
data structure. More generally, when a new rule l→r is introduced by the process, the oth
rules are checked for simplification. A rule l'→r' is called **simplifiable** by l→r iff:
•l' is a true instance of l (and not an E-instance) or a strict subterm of l' is an instance of l
an E-instance if l→r ∈ Rnl). In other words, l' is reducible at the top or one of its str
subterms is E-reducible.
•It can be protected (an extended rule is a particular case of a protected rule), but only f
coherence of rules simplifiable by l→r.

Note the links between the definition of *simplifiability* of a rule and of *top-reducibility* of a coherence pair. Actually, the problem is the same: we want some reducibility property to remain after some rules have been deleted.

```
PROCEDURE E-COMPLETION (P, R, E, ≻, n)
IF P is not empty
THEN  choose a pair (p,q) in P ; p' = p↓; q' = q↓;[4]
      CASE  p' ~E q'   THEN  R = E-COMPLETION (P-{(p,q)}, R, E, ≻, n)  recursive call 1
            p' ≻ q'    THEN  l=p'; r=q'; (P,R) = SIMPLIFICATION (P-{(p,q)}, R, l→r);
                             R = E-COMPLETION (P, RU{n:l→r}[5], E,≻, n+1)  recursive call 2
            q' ≻ p'    THEN  l=q'; r=p'; (P,R) = SIMPLIFICATION (P-{(p,q)}, R, l→r);
                             R = E-COMPLETION (P, RU{n:l→r}, E,≻, n+1)  recursive call 2'
                       ELSE STOP with FAILURE[6]
      END CASE;
ELSE  IF all rules in R are marked
      THEN RETURN R; STOP with SUCCESS
      ELSE Choose an unmarked rule m:l→r w.r.t. fairness selection hypothesis;
      (n,P,R) = CRITICAL-PAIRS (m:l→r, R, E,n); Mark the rule m:l→r in R ;
      R = E-COMPLETION (P, R, E, ≻, n)  tail recursive call 3
      END IF
END IF
END E-COMPLETION
```

Note the elegant form of this tail recursive procedure, making easy its understanding, its proof and its transformation into an efficient iterative procedure.
The SIMPLIFICATION procedure transforms a set of rules into a quasi interreduced one. Rules where only the left hand side is *simplifiable* must become new pairs, since their orientation may change. On the other hand, rules whose right hand side only is reducible remain as rules, since their orientation does not change. Note that simplifiability is a particular case of R'-reducibility, but that some R'-reducible rules are not simplifiable: the reason is that some R'-reducible rules cannot be removed without loosing the coherence property. This will be clear from the proof. As a consequence, the resulting set of rules will not be fully interreduced.

```
PROCEDURE SIMPLIFICATION(P, R, l→r)
K = {k in R | k is simplifiable by l→r};
K' = {k in K | k is not an extension rule};
P = P ∪ {(l'k,rk) | k∈K', lk →[l→r] l'k};
R = {lk→r'k | lk→rk∈R, k∉K, rk →*R'U{l→r}' rk↓ = r'k}[7];
```

---

[4] The normal forms of p and q can be computed with any relation R' ranging between R' and R/E.

[5] As might be expected, l→r must be unmarked and unprotected.

[6] As indicated in [19], some cases can be solved by adding new function symbols.

[7] Marked and/or protected rules remain marked and/or protected after reduction.

RETURN (P,R)
END SIMPLIFICATION

An efficient implemention of this procedure is not straightforward: It requires a careful computation of both sets K and K'. Note that extended rules disappear when they become *simplifiable*.
The Procedure CRITICAL-PAIRS computes critical pairs, sets protections or adds extension rules if necessary and returns a new set P of pairs of terms.

PROCEDURE CRITICAL-PAIRS(m:l→r, R; E,n)
IF m:l→r ∈ Rl
THEN $P = \bigcup_{k\leq m,k\in R} SCP(m,k) \cup \bigcup_{k\leq m,k\in Rl} SCP(k,m)\} \cup \bigcup_{k\leq m,k\in Rnl} CSECP(k,m)$;
FOR any ≪p,q≫ ∈ SCP(l→r,E) (resp. SCP(E,l→r))
DO IF p (resp. q) is irreducible THEN l'=p; r'=q↓; (resp. l'=q; r'=p↓;)
(P,R) = SIMPLIFICATION (P,R,l'→r');
R = R ∪ {n:l'→r'}[8]; n=n+1
ELSE $p \rightarrow^{R'}_{[j]} p'$; P = P ∪ {p',q}; (resp. $q \rightarrow^{R'}_{[j]} q'$; P = P ∪ {p,q'};)
IF p (resp. q) is *top-reducible* by j THEN Protect j for coherence of m ENDIF
END IF
END DO
ELSE $P = \bigcup_{k\leq m,k\in R} CSECP(m,k) \cup \bigcup_{k\leq m,k\in Rl} SCP(k,m)\} \cup \bigcup_{k\leq m,k\in Rnl} CSECP(k,m)$;
FOR any g→d∈E, $v \in \mathcal{D}(g)$, $v \neq \epsilon$ such that $\mathcal{U}$=CSU(l,g/v,E)≠∅
DO IF ∃ $\sigma \in \mathcal{U}$ such that σd is irreducible
THEN l'=g[v←l]; r'=g[v←r]; (P,R) = SIMPLIFICATION(P,R,l'→r');
R = R ∪ {n:l'→r'}[9]; n=n+1
ELSE FOR every σ in $\mathcal{U}$
DO $p=\sigma d \rightarrow^{R'}_{[j]} p'$; P = P ∪ {p',$\sigma$(g[$v$←r])};
IF p is *top-reducible* by j THEN Protect j for coherence of m END IF
END DO
END IF
END DO
END IF;
RETURN (n,P,R)
END CRITICAL-PAIRS

The E-completion procedure can stop with failure, stop with success or loop forever. In the first case, all what may be said is that every pair or rewrite rule generated so far is an equational consequence of the axioms. It is possible that trying again with another ordering would bring a success. We are interested in the two remaining cases, when all pairs considered in P at every recursive call, can be compared by ≻, no matter whether the algorithm stops or loops forever, yielding then a semi-decision procedure for A-equality. These two cases are considered in the forthcoming subsection in a way similar to [30].

[8] The extension l'→r' must be unmarked and unprotected. It can be added to Rl or Rnl.

[9] The extension must be unmarked, added to Rnl and protected.

4.2 A Complete Proof of Correctness

Let us introduce the following notations, consistent with those of [31]:
•P_i and R_i denote the values of arguments P and R at the ith recursive call. Since the recursive calls of the completion procedure are tail recursive, proving properties of the procedure will be easy by induction: Assuming a property is true at the ith recursive call, we will prove it for the next recursive call by checking what happens along each path from the beginning to a tail recursive call. We will use the notation *case n* to indicate that we are interested in the path to the tail recursive call n. Since the two recursive calls 2 and 2' are almost the same, we will always make the proof for case 2 only.
•$\mathcal{R} = \bigcup_i R_i$ is the set of all rules generated during the process. $\mathcal{R}$ is split into left linear rules $\mathcal{R}$l and non left linear ones $\mathcal{R}$nl.
•$R_\infty = \{l\rightarrow r \text{ in } \mathcal{R} \mid \exists i\ \forall j>i,\ l\rightarrow r \text{ is in } R_j\}$. In other words, R_∞ is the set of those rules which are never reduced, neither on their left hand side nor on their right hand side by other rules. If the completion procedure stops, R_∞ is its result and is finite. If it does not stop, R_∞ is infinite and is the limit of R. R_∞ is also split into a set of left linear rules and a set of non left linear ones. Since these subsets are the limits of Rl and Rnl, they are respectively denoted by Rl_∞ and Rnl_∞.

According to these different sets, we consider two reduction relations: $\rightarrow^{\mathcal{R}'}$ is $\rightarrow^{\mathcal{R}l} \cup \rightarrow^{\mathcal{R}nl,E}$ and $\rightarrow^{R_\infty}$ is $\rightarrow^{Rl_\infty} \cup \rightarrow^{Rnl_\infty,E}$. Notice that $\rightarrow^{R_\infty}$ is included into $\rightarrow^{\mathcal{R}}$, $\rightarrow^{R_\infty/E}$ is included into $\rightarrow^{\mathcal{R}/E}$, $\rightarrow^{R_\infty}$ is included into $\rightarrow^{\mathcal{R}'}$. Since $\mathcal{R}$ is E-terminating, so is R_∞.

As in [31], the first part of the proof consists of stating that $\mathcal{R}$ and E generate the equational theory $\sim^A$.

Lemma 17: ∀i>0, (a) $\sim^{R_{i+1}} \subset \sim^{P_i \cup R_i \cup E}$ and (b) $\sim^{P_{i+1}} \subset \sim^{P_i \cup R_i \cup E}$.

Proof: By case analysis, according to the possible tail recursive calls in E-COMPLETION.
case 1: $P_{i+1} = P_i - \{(p,q)\}$ and $R_{i+1} = R_i$ and both (a) and (b) are obvious, since normalization is a particular case of equational deduction.
case 2 and 2': (P_{i+1},R_{i+1}) = SIMPLIFICATION$(P_i - \{(p,q)\},R_i,l\rightarrow r)$. Therefore,
$P_{i+1} = P_i \setminus \{(p,q)\} \cup \{(l'_k,r_k) \mid k \text{ in } K'\}$ and $R_{i+1} = \{l_k \rightarrow r'_k \mid k \notin K\} \cup \{l\rightarrow r\}$.
If $t \sim^{R_{i+1}} t'$, then $t \sim^{P_i \cup R_i \cup E} t'$, since $l \sim^{P_i \cup R_i \cup E} r$ and $l_k \sim^{R_i \cup l\rightarrow r \cup E} r'_k$ and thus $l_k \sim^{P_i \cup R_i \cup E} r_k$.
If $t \sim^{P_{i+1}} t'$, then $t \sim^{P_i \cup R_i \cup E} t'$, since $l'_k \sim^{R_i \cup l\rightarrow r} r_k$ and thus $l'_k \sim^{P_i \cup R_i \cup E} r_k$.
case 3: (P_{i+1},R_{i+1}) = CRITICAL-PAIRS(m,R_i,E).
Then $\sim^{P_{i+1}} \subset \sim^{P_i \cup R_i \cup E}$, since critical pairs are computed with respect to rules of equational deduction.
As new rules in R_{i+1} are extended pairs that are also computed with respect to rules of equational deduction, $\sim^{R_{i+1}} \subset \sim^{P_i \cup R_i \cup E}$. □

Corollary: $\sim^{\mathcal{R} \cup E} \subset \sim^A$.

Proof: Easy consequence of Lemma 17. □

Note that the converse requires a precise proof, since some rules may have been deleted duri
the completion process.
The next two lemmas are an important point of the proof. They assert that the $\mathcal{R}$/E-confluen
modulo E of any pair put in P is ensured.

Lemma 18: $\forall i \geq 0\ \forall (p,q) \in P_i$, (p,q) will be selected at some recursive call $j > i$.

Proof: This is equivalent to prove that there exists $k > i$ such that P_k is empty. To ea
recursive call i, let us associate the multiset of terms denoted $\{P_i\ R_i\}$ built from the le
and right hand sides of all pairs of terms in P_i and R_i. The numbers of axioms in P_i an
rules in R_i are denoted by $|P_i|$ and $|R_i|$ respectively. Let us now consider the followi
ordering on these multisets:
$\{P_i\ R_i\} \gg \{P_j\ R_j\}$ iff lexicographically:

a. $|P_i|+|R_i| > |P_j|+|R_j|$ b. $|P_i| > |P_j|$ c. $\{P_i\ R_i\} \succ_{mult} \{P_j\ R_j\}$

where $\succ_{mult}$ is the multiset extension of the E-reduction ordering $\succ$ used in the Procedu
E-COMPLETION.
Since the definition is lexicographic, it is sufficient to prove that each case is well founde
which is straightforward.
Let us now prove that $\{P_i,R_i\} \gg \{P_{i+1},R_{i+1}\}$ by case analysis:
case 1: $P_{i+1} = P_i - \{(p,q)\}$ and $R_{i+1} = R_i$. then $|P_i|+|R_i| > |P_{i+1}|+|R_{i+1}|$.
case 2 or 2': $P_{i+1} = P_i - \{(p,q)\} \cup \{(l'_k,r_k) \mid k \in K'\}$ and $R_{i+1} = \{l_k \to r'_k \mid k \notin K\} \cup \{l \to r$
If K' is a strict subset of K, $|P_i|+|R_i| > |P_{i+1}|+|R_{i+1}|$.
If $K=K'$, $|P_i|+|R_i|=|P_{i+1}|+|R_{i+1}|$. If K is empty, $|P_i| > |P_{i+1}|$, else K has at least on
element which is a $\mathcal{R}'$-reducible term of $\{P_i,R_i\}$ and in this case $\{P_i,R_i\} \succ_{mult} \{P_{i+1},R_{i+1}$
since each rule in $\mathcal{R}'$ has been checked for termination with $\succ$.

Lemma 19: $\forall i \geq 0$, $\forall (p,q) \in P_i$, $\exists p', q'$ such that $p \stackrel{+}{\to}_{\mathcal{R}/E} p'$, $q \stackrel{+}{\to}_{\mathcal{R}/E} q'$ and $p' \sim^E q'$.

Proof: From previous lemma, we know that any pair (p,q) in P_i is selected at som
recursive call j. It satisfies the claimed property since either $p{\downarrow}R^* \sim^{\mathcal{E}} q{\downarrow}R^*$, o
$p{\downarrow}R^* \to q{\downarrow}R^*$ (resp. $q{\downarrow}R^* \to p{\downarrow}R^*$) $\in R_{j+1}$.

We are now able to state the first part of the proof of the completion procedure:

Proposition 20: $\sim^A = \sim^{\mathcal{R}UE}$

Proof: From the previous lemma applied with i=0, therefore P_0=R, it follows that $\sim^{\mathcal{R}}$
therefore $\sim^A$, is included into $\sim^{\mathcal{R}UE}$. The result then follows from the last corollary.

The second part of the proof consists of proving the Church-Rosser property for R_∞. It wi
follow from the fact that the critical pairs of $\mathcal{R}$ satisfy the properties of Theorem 15.
To be sure that a rule cannot be ignored indefinitely in the selection process of the ELSE branc
of the THEN main branch of the procedure E-COMPLETION, a **fairness selection hypothesis**
required for the choice of an unmarked rule in R: For any rule labelled k, there exists
recursive call such that:
•either rule k is reduced by the newly introduced rule and removed from the current set o
rules,
• or rule k is selected and CRITICAL-PAIRS(k, R, E,n) is computed.

Combined with lemma 19, this will enable us to prove confluence of confluence and coherence
pairs.

Before stating and proving the Church-Rosser property of R_∞, we need to prove the well
foundedness of the reduction relation used in the proof. Let $\mapsto$ be the relation
$\to^{R/E} \cup \to^{\text{strict-subterm/E}} \cup \to^{\text{strict-subsumption/E}}$, where $\to^{\text{strict-subsumption/E}}$ is the strict ordering
associated with the following preordering: $s \to^{\text{subsumption/E}} t$ iff $\exists s', t'$ such that $s \sim^E s'$, $t \sim^E t'$
and $s' = \sigma t'$ for some substitution σ. First, $\to^{\text{strict-subsumption/E}}$ will be supposed well founded.
As already noted, this is usually the case as soon as the theory E has a finite unification
algorithm. The rest of the termination proof is based on additional commutation lemmas.

Lemma 21: $s \stackrel{+}{\to}^{\text{strict-subsumption/E}} t' \stackrel{+}{\to}^{R/E} t \Rightarrow$
$\exists s' \in \mathcal{T}$ such that $s \stackrel{+}{\to}^{R/E} s' \stackrel{+}{\to}^{\text{strict-subsumption}} t$.

Proof: The proof is similar to that of lemma 12 and is left to the reader. □

Lemma 22: $s \stackrel{+}{\to}^{\text{strict-subterm/E}} t \stackrel{+}{\to}^{\text{strict-subsumption/E}} t' \Rightarrow$
$\exists s'', s'' \in \mathcal{T}$ such that $s \stackrel{+}{\to}^{\text{strict-subsumption/E}} s' \sim^E s'' \stackrel{+}{\to}^{\text{strict-subterm}} t'$.

Proof: The proof is by induction on the length of the entire derivation. If s=t, the result
is obvious. If t=t', the result comes from lemma 13.
Else $s \stackrel{+}{\to}^{\text{strict-subterm/E}} s_1 \to^{\text{strict-subterm/E}} t \to^{\text{strict-subsumption/E}} t_2 \stackrel{+}{\to}^{\text{strict-subsumption/E}} t'$.
By definition, $\exists\ t_i \in \mathcal{T}$ and $\sigma \in \Sigma$ which is not a renaming modulo E such that
$t \sim^E \sigma t_1 \to^{\text{strict-subsumption}} t_1 \sim^E t_2$, therefore $t \sim^E \sigma t_2 \to^{\text{strict-subsumption}} t_2$. Since the
same process works with the strict-subterm relation as already noticed in lemma 13, we
get: $s_1 \sim^E s_2 \to^{\text{strict-subterm}} \sigma t_2 \to^{\text{strict-subsumption}} t_2$. We can now make strict-subterm
and strict-subsumption commute. By definition, $\exists \upsilon$ such that $s_2/\upsilon = \sigma t_2$. Assume
without lost of generality that $\mathcal{V}(t_2) \cap \mathcal{V}(s_2) = \emptyset$, therefore $\mathcal{D}(\sigma) \cap \mathcal{V}(s_2) = \emptyset$ (else, we
must perform a variable renaming on t_2, which eventually renames the variables of t' at
the end). Now, $s_2 = \sigma(s_2[\upsilon \leftarrow t_2])$ and thus $s_2 \to^{\text{strict-subsumption}} s_2[\upsilon \leftarrow t_2] \to^{\text{strict-subterm}} t_2$.
The result is finally obtained by pushing the strict-subterm reduction to the end using a
first application of the induction hypothesis from $s_2[\upsilon \leftarrow t_2]$ to t' and then by a second
application of the induction hypothesis to the whole derivation except the last strict-
subterm reduction. □

Corollary:
$\mapsto = \stackrel{+}{\to}^{R/E} \circ \stackrel{+}{\to}^{\text{strict-subsumption}} \circ \stackrel{+}{\to}^{\text{strict-subsumption/E}} \circ \stackrel{+}{\to}^{\text{strict-subterm}} \circ \stackrel{+}{\to}^{\text{strict-subterm/E}}$

Proof: follows from lemmas 12, 21 and 22. □

As a consequence of this corollary, the relation $\mapsto$ is well founded under almost the same
hypotheses as in the previous section, since the strict-subsumption relation is well founded and
the strict-subsumption/E relation is well founded by the added hypothesis.

Theorem 23: Let $R = R_I \cup R_{nI}$ be an E-terminating set of rules such that the rules in R_I are
left linear, a complete and finite unification algorithm exist for the theory E, E-congruence
classes are finite and the E-subsumption preordering is well founded.

Then R_∞ is R'_∞-Church-Rosser modulo E provided that the fairness selection hypothesis satisfied.

Proof: Let us denote by $t\downarrow$ the R'_∞-normal form of t, and write $s \Downarrow t$ iff $\exists s', t'$ such that $s \twoheadrightarrow^{R'_\infty} s'$, $t \twoheadrightarrow^{R'_\infty} t'$ and $s' \sim^E t'$, and $s \Downarrow^{\mathcal{R}/E} t$ iff $\exists s'$ such that $s \twoheadrightarrow^{\mathcal{R}/E} s'$ and $t \twoheadrightarrow^{\mathcal{R}/E} s'$. Since $\sim^A$ is the same as $\sim^{\mathcal{R}\cup E}$, we may actually prove that $t_0 \sim^{\mathcal{R}\cup E} t_{n+1} \Rightarrow t_0\downarrow \Downarrow t_{n+1}\downarrow$. Let $M=\{t_0,\ldots,t_n,t_{n+1}\}$ such that $\forall i\in[0..n]\ t_i \vdash\!\!\!\mapsto^{\mathcal{R}\cup E} t_{i+1}$. The proof is very similar to that of the Church-Rosser Theorem 15, and works by multiset induction on $\mapsto$. Three cases have to be distinguished as usual. Each one uses a particular local property defined next thanks to the following notation: $s_0 \Downarrow s_n$ iff $\vdash\!\!\!\mapsto^{\mathcal{R}\cup E} s_1 \vdash\!\!\!\mapsto^{\mathcal{R}\cup E} \ldots s_n \vdash\!\!\!\mapsto^{\mathcal{R}\cup E} s_{n+1}$ such that $\exists j\ \forall i\in[1..n]\ t_j \not\twoheadrightarrow s_i$. As a consequence, the whole multiset $\{s_0 \ldots s_{n+1}\}$ is strictly smaller than $\{t_j\}$, thus smaller than M. This will enable us to apply multiset induction easily on $\{s_0 \ldots s_{n+1}\}$. Note in addition that $\Downarrow$ is transitive for a given t_j and that $s \mathcal{R}' s'$, $s \Downarrow s'$ and $s \Downarrow^{\mathcal{R}/E} s'$ imply all $s \Downarrow s'$, provided s and s' are proper sons of t_j for $\mapsto$.

- $t_n \vdash\!\!\!\mapsto^E t_{n+1}$: If t_n is R_∞-irreducible, then so is t_{n+1} by the *local coherence* property that we redefine next using the previous notations and the result follows from induction hypothesis applied to the multiset $\{t_0 \ldots t_n\}$. Else, *local coherence* is applied first as shown on the left part of Figure 6 and then induction to the multiset $\{t_0 \ldots t_n\ t'_n \ldots t'_{n+1}\}$ which is strictly smaller than M, since the terms between t'_n and t'_{n+1} are all proper sons of t_{n+1} for $\mapsto$ by definition of $\Downarrow$. This is step 2 on the same figure.

- $t_{n+1} \to^{\mathcal{R}} t_n$: the result follows from *reducibility* of $\mathcal{R}$ by R_∞, which then allows us to apply the induction hypothesis to the multiset $\{t_0 \ldots t_n \ldots t'_{n+1}\}$ whose terms are all smaller than t_{n+1} for $\mapsto$. This is shown on the right part of Figure 6.

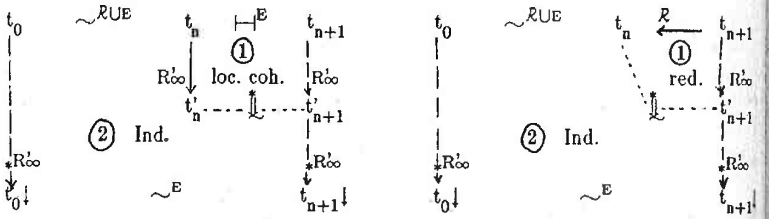


Figure 6: *Church-Rosser Proof using local coherence (left) or reducibility (right)*

- $t_n \to^{\mathcal{R}} t_{n+1}$: We apply first the induction hypothesis on the multiset $\{t_0 \ldots$ Since t_n is not in $\mathcal{R}$-normal form, it is not in R'_∞-normal form by the reducibility property, making clear step 1 in Figure 7. The proof then follows from *local confluence* (step 2) and induction on the multiset $\{t'_n \Downarrow t_{n+1}\}$ (step 3).

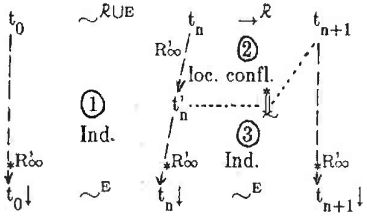


Figure 7: *Church-Rosser Proof using local confluence*

We are now left with proving the three properties of reducibility, local coherence and local confluence, assuming the Church-Rosser induction hypothesis on multisets strictly smaller than M.

Let us first prove reducibility, specifically:

$\forall t$ such that $t \sim^E t_{n+1}$ and $t \to^{\mathcal{R}}_{[\epsilon,\sigma,k:l\to r]} t'_1$ or $t \to^{\mathcal{R}'}_{[v\neq\epsilon,\sigma,k:l\to r]} t'_1$, $\exists\ t'_2$ such that $t \twoheadrightarrow^{R'_\infty} t'_2$ and $t'_1 \Downarrow t'_2$.

Note that standard matching is required here for reduction of t at the top, but not for reduction of a strict subterm of t.

- *case 1:* Assume $v\neq\epsilon$. Then t/v is a proper son of t for $\mapsto$ and the induction hypothesis may be applied to the multiset $\{t/v\ t'_1/v\}$ yielding $t/v\downarrow \sim^E t'_1/v\downarrow$. Since $\mathcal{R}$ is E-terminating, t must be different from $t\downarrow$. The property is then easily lifted to (t,t'_1) yielding the result.

- *case 2:* Assume $v=\epsilon$ and that σ is not a variable renaming. Then l is a proper son of t for $\mapsto$ and the same reasoning as in case 1 applies.
  For the remainder of the proof, v can be supposed to be ϵ and σ to be a variable renaming, hence $t=\sigma l$ and $t'_1=\sigma r$.

- *case 3:* Some rule labelled k belongs to R_∞, say $l\to r'$ with $r \twoheadrightarrow^{\mathcal{R}'} r'$, therefore $t \to^{R_\infty}_{[\epsilon,\sigma,k:l\to r']} t'_2 = \sigma r'$ and $t'_1 \twoheadrightarrow^{\mathcal{R}'} t'_2$, hence $t'_1 \Downarrow t'_2$.

- *case 4:* The left-hand side l of the rule k has been reduced by another rule $l''\to r''$, therefore $l/v \sim^E \tau l''$ for some occurrence $v\neq\epsilon$ and substitution τ or $l/\epsilon = \tau l''$, since left members of rules are not reduced at the top modulo E. Let $l' = l[v\leftarrow\tau r'']$. Since (l',r) has been dropped to P, by Lemma 19, the pair (l',r) has become convergent at some further recursive call, therefore $l' \Downarrow r$.
  Let $t \to^{\mathcal{R}'}_{[v,\sigma\tau,l''\to r'']} t''_1$. Since $t'_1 = \sigma r$ and $t''_1 = t[v\leftarrow\sigma\tau r''] = \sigma(l[v\leftarrow\tau r''])$, we have $t'_1 \Downarrow t''_1$. We are therefore left with proving the reducibility of the pair $(t\ t''_1)$.
  Let us suppose that τ is a variable renaming and v is ϵ. Then l'' is an instance of l and the rule $l''\to r''$ would never have been generated. Therefore $v\neq\epsilon$ or τ is not a variable renaming, we can apply the already proved cases 1 and 2 and we are done.

Note the crucial fact of l being an instance (and not an E-instance) of l'' when l'' reduces l

at the top. Else, it would have been necessary to use the (not yet proved) local coherence property to lift the reducibility of $r\mathrm{I}^{\mathrm{u}}$ to l in case 2, where the induction uses the E-subsumption ordering (and by transitivity in case 4).

Let us now prove local coherence, specifically:

$\forall t,t',t^{\mathrm{u}}$ such that $t_n \sim^{\mathcal{E}} t$, $t \vdash^{\mathrm{E}}_{[\upsilon,\sigma,g\to d]} t'$ and $t \to^{\mathrm{R}\dot{\infty}}_{[\nu,\sigma,l\to r]} t^{\mathrm{u}}$, $\exists t'_1$ such that $t' \to^{\mathrm{R}\dot{\infty}} t'_1$ and $t'_1 \downarrow t^{\mathrm{u}}$.

The cases to be now discussed according to the respective positions of υ and ν are the same as in Theorem 15. Only cases 5, 7 and 8 differ. Numbering and notation remain exactly the same:

- *case 5:* By fairness hypothesis, the critical pairs of $l\to r$ and $g\to d$ have been computed at some iteration j. By the critical pairs lemma, there exist a critical pair of $\ll p,q\gg$ in $\mathrm{SCP}(\mathrm{E},\mathrm{Rl}_j)$ and a substitution τ such that $t'=\tau p$, $t^{\mathrm{u}}=\tau q$. This is step 1 in Figure 8. From the procedure CRITICAL-PAIRS, we know that $p \to^{\mathcal{R}'}_{[\omega,\theta,k':l'\to r']}$ The rule $l'\to r'$ can be the added rule $p\to q\downarrow$ and in that case $q\downarrow=p'\downarrow$, or the pair (p',q) has been dropped into P. In this last case, we know from Lemma 19 that $p'\downarrow^{\mathcal{R}/\mathrm{E}}q$, therefore $p'\downarrow q$. As a consequence, in both cases, we have $\tau q \downarrow \tau p'$. This is step 2 in Figure 8. The rest of the proof is now split into three cases:

- *case 5.a:* $\omega\neq\epsilon$. In that case, the induction hypothesis is applied to the multiset $\{\tau p/\omega, \tau p'/\omega\}$. Since $\mathcal{R}$ is E-terminating, there must exist a term t'_1 such that $\tau p/\omega \to^{\mathrm{R}\dot{\infty}}$ and $\tau p'/\omega \downarrow t'_1$. As a consequence, the pair $(t',\tau p')$ satisfies the same property. This is step 3 on the left side of Figure 8.

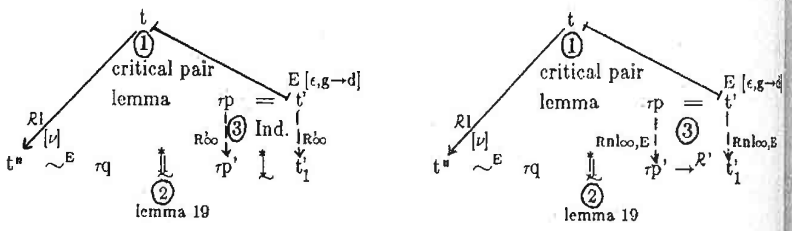


Figure 8: *Local Coherence Proof, cases 5.a (left) and 5.b (right)*

- *case 5.b:* $\omega=\epsilon$ and p is not an instance of l' but an E-instance. Then $l'\to r' \in \mathcal{R}\mathrm{nl}$ and has been protected (it may be an extended rule). Since $l\to r$ is in Rl_∞, a rule $k':l'\to$ must be in Rnl_∞ with $r' \to^{\mathcal{R}'} r^{\mathrm{u}}$, therefore $\tau p' = \tau\theta r' \to^{\mathcal{R}'} \tau\theta r^{\mathrm{u}}$. Now $t' = \tau p \sim^{\mathrm{E}} \tau\theta l' \to^{\mathrm{Rnl}_\infty}_{[\epsilon,\tau\theta,k']} \tau\theta r^{\mathrm{u}}$. This is step 3 on the right side of Figure 8, with $t'_1 = \tau\theta r^{\mathrm{u}}$.

- *case 5.c:* $\omega=\epsilon$ and $l'\to r' \in \mathcal{R}\mathrm{l}$ or $l'\to r' \in \mathcal{R}\mathrm{nl}$ and p is a true instance of l'. Then we conclude from the already proved reducibility property that $t' \to^{\mathrm{R}\dot{\infty}} t'_1$, $\tau p' \downarrow t'_1$ and we are done.

- *case 7:* similar reasoning as in case 5.

- *case 8:* By fairness hypothesis and the E-critical pairs lemma, there exists an iteration j, a pair $\ll p=\theta d, q=\theta(g[\nu\leftarrow\theta r])\gg$ in $\mathrm{CSECP}(\mathrm{Rnl}_j,\mathrm{E})$ and a substitution τ such that $\sigma \sim^{\mathcal{E}} \tau\theta$ $[\mathcal{V}(g)\cup\mathcal{V}(l)]$, therefore $t' \sim^{\mathcal{E}} \tau p$ with E-equality steps taking place outside of $\mathcal{G}(d)$ and $t^{\mathrm{u}} \sim^{\mathcal{E}} \tau q$. From the CRITICAL PAIRS procedure, we know that $p \to^{\mathcal{R}'}_{[\omega,\theta,k':l'\to r']} p'$ and, as in case 5, $\tau p' \downarrow \tau q$. This is step 1 in Figure 9.

The next step in the proof allows us to transform the $\mathcal{R}'$-rewriting from τp to $\tau p'$ into a $\mathrm{R}\dot{\infty}$-rewriting from τp to some t''_0 related to $\tau p'$. Two cases are to be distinguished: If $\omega=\epsilon$, $k':l'\to r' \in \mathcal{R}\mathrm{nl}$ and p' is not a true instance of l', then k' has been protected. Since $l\to r$ is in Rnl_∞, a rule $k':l'\to r^{\mathrm{u}}$ must be in Rnl_∞ with $r' \to^{\mathcal{R}'} r^{\mathrm{u}}$. Let now $t''_0 = \tau\theta r^{\mathrm{u}}$. Then $\tau p \to^{\mathrm{Rnl}_\infty}_{[\epsilon,\tau\theta,k']} t''_0$, and $\tau p' = \tau\theta r' \to^{\mathcal{R}'} t''_0$, therefore $\tau p' \downarrow t''_0$.

If k' applies at occurrence $\omega\neq\epsilon$ or p' is a true instance of l', then the already proved reducibility property applies and yields $\tau p \to^{\mathrm{R}\dot{\infty}} t''_0$ for some t''_0 and $\tau p' \downarrow t''_0$.

This is step 2 in Figure 9. The remainder of the proof is then very similar to the corresponding one in Theorem 15, applying the already proved local coherence property until t' is reached, since no equality step from τp to t' takes place at occurrence ϵ.

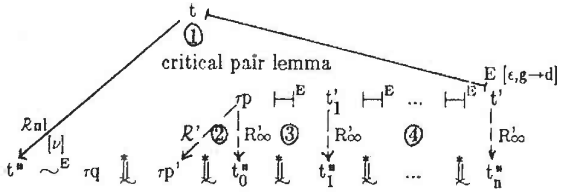


Figure 9: *Local Coherence Proof, case 8*

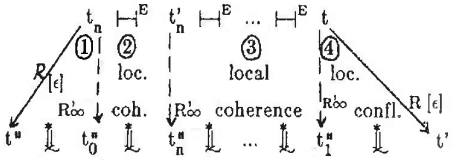


Figure 10: *Local Confluence Proof, case 4*

Let us now prove local confluence, specifically: For any t' and t^{u} such that $t_n \to^{\mathrm{R}\dot{\infty}}_{[\upsilon,\sigma,g\to d]} t'$ and $t_n \to^{\mathcal{R}}_{[\nu,\sigma,l\to r]} t^{\mathrm{u}}$, $t' \downarrow t^{\mathrm{u}}$.

We only point out what differs from the proof of Theorem 15. Cases 1 and 2 are the same. Case 3 is the same when there is no overlapping, else the fairness selection hypothesis and Lemma 19 are used as in the local coherence proof. Case 4 uses a first reducibility step to make appear a $\mathrm{R}\dot{\infty}$-reduction from t_n instead of a $\mathcal{R}$-reduction, the rest of the proof is entirely similar. □

As a corollary of Theorem 23 and Proposition 20, we obtain:

Theorem 24:
•R_∞ is R'_∞-convergent modulo E.
•$\mathcal{R}$ is $\mathcal{R}'$-convergent modulo E.
•The congruences $\sim^A$, $\sim^{\mathcal{R}\cup E}$ and $\sim^{R_\infty\cup E}$ are equal.

As in theorem 15, we conjecture that some hypotheses could be removed, precisely the h hypotheses about the finiteness of the E-congruence classes and the well foundedness of t E-subsumption preordering.

Let us finally prove that the E-completion algorithm can always be considered as a se decision procedure for $\sim^A$, assuming that a decision procedure for $\sim^{\mathcal{E}}$ is known. As in t standard completion algorithm, if the two terms to be checked for A-equality are not in t same congruence class, then the semi-decision procedure will loop for ever if the E-completi procedure loops forever.

Proposition 25: s $\sim^A$ t iff there exists a recursive call i such that s $\downarrow_{\mathcal{R}'_i}$ s', t $\downarrow_{\mathcal{R}'_i}$ t' a s' $\sim^E$ t' for some s' and t'.

Proof: It follows the lines of the proof of the corresponding result in [31] obtained for t case of an empty E.

This result can be extended in a straightforward way to semi-decide that two sets of axio (R,E) and $(\mathcal{R},\mathcal{E})$ generate the same equational theory, provided the hypotheses of theorem 23 satisfied: We can start the procedure E-COMPLETION to check that all axioms in $R\cup\mathcal{E}$ equational consequences of RUE. If this is the case, the process will stop after a finite numb of calls, even if the E-COMPLETION procedure loops forever. Then we can start the symmet check. However, this process of checking two sets of axioms for equivalence is time consum and we will see in section 4.4 an efficient and elegant way of achieving the same goal when E-COMPLETION procedure provides a R'-Church-Rosser sets of rules after a finite number steps.

4.3 Protection versus Inter-Irreducibility

Let us now discuss some issues regarding our Completion Procedure, especially the trade between protecting rules for coherence and reducing rules for the generated set of rules to ha a normal form property as in the standard case [59].
Compared to the other known completion algorithms [49, 50, 52, 67, 30, 36], the main n difficulty here arises from the coherence property of the rewriting relation $\rightarrow^{R,E}$. In the classi completion algorithm of Knuth and Bendix, proved in [31], E is empty and of course, coherence property is needed. The completion algorithm modulo E, designed in [30], u standard rewritings with left linear rules only. Our proof makes clear that we do not need protect any rule if Rnl is empty. On the other hand, Peterson and Stickel systematica introduce for each rule that has an associative commutative top function symbol its associat commutative extensions that are implicitly protected and thus enforce the coherence property.

Various strategies to deal with the problem of coherence can be imagined. Let us discuss some of them with their respective drawbacks and advantages:

- In the previous method, our aim was to avoid introducing extensions of rules if the coherence property was ensured by an already existing rule. The lack of power of the $\rightarrow^{R,E}$ relation compelled us to protect some non left linear rules. Once protected, a rule cannot be removed from the rewriting system, even if its left hand side becomes reducible, unless the rule it was protecting has disappeared. Of course, the resulting rewriting system may not be entirely inter-reduced.

- However, systematically adding extensions can be very interesting for some cases: assume that whenever a non left linear rule E-overlaps an equation at occurrence o, the adequate extension of this rule is added. We do not need computing a complete set of E-overlappings, since this extension automatically reduces all the corresponding E-critical pairs of this rule on the equation at occurrence o: we only need to know that E-unification is possible at this occurrence. In many cases this strategy will speed up the completion process. Notice also that in this case, we do not need to protect any rule, except of course the extensions, since we are no longer interested in finding an existing rule which reduces the adequate member of the coherence pairs. As a main drawback of this method, extensions can be many and redundant, and cannot be reduced.

- Another approach consists in modifying the E-COMPLETION procedure by protecting both left linear and non linear rules that reduce a coherence pair at the top under the same conditions as previously. This allows local coherence proofs without the use of reducibility, case 5.c of the local coherence proof becoming similar to case 5.b which makes no use of reducibility. As a consequence, the reducibility proof can now be performed after the local coherence proof and make use of it. As noticed at the end of case 4 of that proof, reducibility becomes now satisfied even if left members of rules are reduced modulo E in the SIMPLIFICATION procedure, provided the rule l→r is added to Rnl. This will yield a set R_∞ of rules that can be expected to be more interreduced than previously.

- A fourth method can even be designed that yields a set R_∞ of completely interreduced rules. This can be done by another modification of the SIMPLIFICATION Procedure: if the left-hand side of a rule k, protected for coherence of a rule k' with an equation g=d, becomes reducible, then k is removed from the rewriting system and the E-critical pairs of k with g=d are computed again. If the process terminates, we are sure that all the critical pairs have been computed, and thus the coherence of k' is ensured. But this strategy does not satisfy the fairness assumption hypothesis when the procedure loops and the infinite set R_∞ is no longer garanteed to be R'_∞-Church-Rosser modulo E. However, we could modify the fairness selection hypothesis to take into account the fact that some rule would have to be chosen infinitely many times and some others only finitely many times with respect to a generalized fairness selection hypothesis. Such algorithms already exist that were designed for the purpose of implementing concurrent procedures.

Finally, let us emphasize that all our results remain true, except the decidability results, if the theory E has a complete unification algorithm that may sometimes return an infinite set of unifiers, provided it returns a finite one at each call. For this case, we might want the

E-unification procedure to be called only when it is really needed, in order to minimize the possibility of starting an infinite computation because of the E-unification process. This suggests use of the variant that adds extensions in a systematic way. Note in addition that we might require that the E-unifiers are computed a finite number at a time if there are infinitely many, in order to respect the fairness assumption hypothesis and still have a semi-decision procedure.

4.4 Complete sets of rules in normal form

As seen in the last section, the completion procedure may give as a result a Church-Rosser set of rules that are not fully interreduced. As a consequence, we cannot expect such a set to have a normal form property as in the standard case [13, 59, 53]: the result of the completion procedure will depend upon a particular presentation of the theory as well as upon the implementation of the choice functions used in the procedure. Our goal here is twofold:
•First, to define precisely what kind of normal forms can be expected for a Church-Rosser set of rules.
•Second, to give an algorithm to transform any Church-Rosser set of rules into its normal form.

Let us say that two sets R and $\mathcal{R}$ of rules are **E-equivalent**, or simply equivalent whenever E is known from the context, iff they generate the same theory $\sim^A$ where A is the whole set of axioms of both E on one hand, and R or $\mathcal{R}$ on the other hand.

Assume we are now given a given E-reduction ordering $\succeq$ whose associated strict ordering $\succ$ is well-founded. This ordering can be thought of as the ordering used in the E-COMPLETION procedure: its role is to prove the E-termination property of the set R of rules. We will say that R is **compatible** with $\succeq$ if this is the case.

As a consequence, whenever the previous goal is achieved, it follows that the E-equivalence problem for two Church-Rosser sets of rules compatible with the same E-reduction ordering $\succeq$ is decidable. As a practical consequence, the fact that the resulting Church-Rosser set of rules produced by the E-COMPLETION procedure is not completely interreduced is not a severe drawback anymore: just apply the adequate normalization postprocessing.

We now define what it means for a Church-Rosser set of rules to be in normal form:

Definition 26: An **E-complete set of normalized reductions** R is a set of rules that has the following properties:
(1) R is E-terminating and R'-Church-Rosser for some R'.
(2) The left members of rules are R'-irreducible.
(3) The right members of rules are R'-irreducible.

We now show that such sets of reductions have a normal form property. To do so, we first introduce a useful technical lemma:

Lemma 27: Assume R is R'-Church-Rosser modulo E and compatible with $\succeq$. Then a term t is R'-irreducible iff it is minimal for $\succeq$ in its A-congruence class.

Proof: If t is R'-irreducible, then for any t' $\sim^A$ t, t' $\stackrel{*}{\rightarrow}^{R'}$ t'↓ $\sim^E$ t↓ = t. Hence, t' $\succeq$ t. Conversely, if t is R'-reducible to s, then t $\succ$ s. □

We are now ready for the main theorem of E-complete sets of normalized reductions:

Theorem 28: Let R and $\mathcal{R}$ be two E-complete sets of normalized reductions compatible with the same E-reduction ordering $\succeq$, and generating (with the help of the equations in E) the same equational theory $\sim^A$. Then R and $\mathcal{R}$ are identical up to E-equality.

Proof: We prove that for any rule l→r in $\mathcal{R}$, there exists a (unique) rule l'→r' in R such that l $\sim^E$ l' and r $\sim^E$ r'. Since R and $\mathcal{R}$ (with the help of E) present the same equational theory and R is R'-Church-Rosser and r is $\mathcal{R}$'-irreducible, thus R'-irreducible by lemma 27, l $\stackrel{*}{\rightarrow}^{R'} \circ \sim^E$ r. Since l is $\mathcal{R}$'-reducible thus R'-reducible by lemma 27 and strict subterms of l are $\mathcal{R}$'-irreducible thus R'-irreducible by the same lemma, l $\rightarrow_{[\epsilon,\sigma,l'\rightarrow r'\in R]} \circ \stackrel{*}{\rightarrow}^{R'}$ r.
Assume now that σ is not E-equivalent to a variable renaming. The same reasoning shows that l' is $\mathcal{R}$'-reducible at the top by a rule l"→r" of $\mathcal{R}$ which cannot be the rule l→r, since σ is not E-equivalent to a variable renaming. Now, since the second reduction applies at the top of l', l becomes reducible by l"→r", which is impossible by the hypothesis that the rules of $\mathcal{R}$ are interreduced. Therefore σ is a variable renaming and l $\sim^E$ l'. Since r' and r are supposed to be irreducible, we finally get r' $\sim^E$ r by the Church-Rosser property and we are done. The uniqueness of the rule l'→r' then follows from the inter-irreducibility of the rules in R. □

A similar result had already been obtained by [53], precisely for the case where R' = R/E. According to our thesis that an "abstract" rewriting relation R' must be used for sake of generality, our result is therefore more general and applies for sets of rules that are R/E-Church-Rosser. This will be used in the following.

The next three lemmas show how to transform an R'-Church-Rosser set of E-terminating rules into an E-complete set of normalized reductions. This will be done by three different kinds of transformations applied to the starting set, each one preserving a Church-Rosser property. The two first transformations preserve the R'-Church-Rosser property, while the last one only preserves the R/E-Church-Rosser property. As a consequence, the two first transformations can be usefully applied to the R'-Church-Rosser set of rules given by the procedure E-COMPLETION. Actually, only the second one is useful, since right members of rules are fully simplified in the procedure SIMPLIFICATION.

Lemma 29: Assume R is an E-terminating and R'-Church-Rosser set of rules.
Let $\mathcal{R} = \{l \rightarrow r\downarrow \mid l \rightarrow r \in R\}$. Then $\mathcal{R}$ is equivalent to R and $\mathcal{R}$'-Church-Rosser modulo E.

Proof: $\mathcal{R}$ is clearly E-terminating since R is so. In order to prove the result, we just prove that any term has the same normal forms for both rewriting relations. First, they define the same sets of normal forms, since left members of rules are the same and are used with the same matching algorithm (which is implicit in our definition of $\mathcal{R}$). Let now t $\stackrel{*}{\rightarrow}^{\mathcal{R}'}$ t↓. Then t $\stackrel{*}{\rightarrow}^{R'}$ t↓ since R is R'-Church-Rosser and the $\mathcal{R}$'-normal form of t is in R'-normal form. □

Lemma 30: Assume R is an E-terminating and R'-Church-Rosser set of rules. Let $\mathcal{R}$ be obtained from R by removing any rule from Rl whose left member is R'-reducible, or any rule from Rnl whose left member is R'-reducible on top by another rule of Rnl. Then $\mathcal{R}$ is equivalent to R and is $\mathcal{R}$'-Church-Rosser modulo E.

Proof: $\mathcal{R}$ is clearly E-terminating since R is. As previously, we now prove that any term

has the same normal forms for both rewriting relations. This is done here by noetherian induction on the relation $\mapsto$ defined in Section 4.2. Let $t \rightarrow^{R'}_{[v,\sigma,l\to r]} t' \not\rightarrow^{R'} t\downarrow$. If $v\neq\epsilon$, the result is easily obtained by induction on t/v.
If $l\to r$ is in $\mathcal{R}$, the result is easily obtained by induction on t'.
For the remaining cases, $v=\epsilon$, therefore $t=\sigma l$, and $l\to r$ has been removed, that is $l \rightarrow^{R'}_{[v',\sigma',l'\to r']} l^*$, with $l'\to r'$ in $\mathcal{R}$.
If $l\to r \in$ Rl, then $t=\sigma l \rightarrow^{\mathcal{R}} t^*=\sigma l[v'\leftarrow\sigma' r']$ and the result is obtained by the induction hypothesis applied to t^*, together with the R'-Church-Rosser property of R that forces t' and t^* to have E-equal R'-normal forms.
If $l\to r \in$ Rnl, then $t \sim^E \sigma l \rightarrow^{\mathcal{R}'} t^*=\sigma l[v'\leftarrow\sigma' r']$, since $v'=\epsilon$ in this case by definition of $\mathcal{R}$ and $l'\to r' \in$ Rnl, and the result is obtained by induction hypothesis applied to t^* and the R'-Church-Rosser property as before. □

Reducing the starting set of rules as long as possible using lemmas 29 and 30 clearly yields a set of rules $\mathcal{R}$ satisfying requirements (1) and (3) of definition 26. Moreover, it will also satisfy the part of requirement (2) concerning left members of rules in Rl. Now, either the whole set of rules satisfies requirement (2), i.e. left members of rules in Rnl are also $\mathcal{R}'$-irreducible and the resulting set of rules $\mathcal{R}$ is an E-complete set of $\mathcal{R}'$-normalized reductions equivalent to the starting one, or it does not and we need to apply as many times as necessary the third transformation studied in the next lemma. If this is the case, we will get an E-complete set of $\mathcal{R}$/E-normalized reductions.

Lemma 31: Assume R is an E-terminating and R'-Church-Rosser set of rules. Let $\mathcal{R}$ be obtained from R by removing any rule in Rnl whose left member is R'-reducible on a strict subterm or Rl-reducible on the top. Then $\mathcal{R}$ is equivalent to R and $\mathcal{R}$/E-Church-Rosser.

Proof: Note first that the new set of rules is still E-terminating. We prove now the two other properties by induction on $\mapsto$ as in the proof of Lemma 30. The two first cases are the same and the case where $v=\epsilon$ and $l\to r$ has been removed from R requires minor changes: Since the E-equality step between t and σl can not be absorbed anymore by the E-matching step used in the R' reduction relation it is absorbed here by the E-equality step used in the $\mathcal{R}$/E reduction relation. This explains why we only get a $\mathcal{R}$/E-Church-Rosser property. □

Applying this last lemma results in an E-complete set of $\mathcal{R}$/E-normalized reductions that is unique up to E-equality by Theorem 28, allowing us to check for equivalence two different sets of axioms expected to be presentations of a same theory. But the complete set of normalized reductions can be used for this purpose only, since it is not $\mathcal{R}'$-Church-Rosser anymore: For computing in the theory $\sim^A$, we will use the $\mathcal{R}'$-Church-Rosser set of rules obtained by applying the first two transformations to the set of rules given by the E-COMPLETION procedure, since the rewriting relation associated with it is more efficient.

5 Conclusion

Let us emphasize that splitting the set of rules into left linear rules and non left linear rules allows to compute less complete sets of E-unifiers, since E-unification is only needed for non left linear rules. On the other hand, by using a set Rl of linear rules one gets sometimes into a

divergent process, that would otherwise converge with all the rules considered to be in Rnl. Let us recall the example of Section 3 of a set of rules which is R,E-Church-Rosser modulo E but not Rl∪Rnl,E-Church-Rosser modulo E:
Let 0, 1, -1 be constant function symbols and + a binary infix symbol which is associative and commutative.
Let R be the set of linear rules: (1) $0+x \to x$, (2) $1 + -1 \to 0$, (3) $(x+1) + -1 \to x$.
Then R is R,E-Church-Rosser modulo associativity and commutativity.
However, using our algorithm with Rl = R, the procedure generates an infinite set of linear rules (all supposed to be in Rnl) among which are the following:
(4) $x+0 \to x$, by overlapping rule 1 and axiom of commutativity on top.
(5) $-1 + (x+1) \to x$, by overlapping rule 3 and axiom of commutativity on top.
(6) $(1+x) + -1 \to x$, by overlapping rule 5 and axiom of commutativity on top.
(7) $-1 + (1+x) \to x$, by overlapping axiom of commutativity into rule 5 at occurrence 2.
(8) $(x+(y+1)) + -1 \to x+y$, by overlapping the associativity axiom into rule 3 at occurrence 1.
....
This example shows that performing E-unification is costly but also powerful.

Let us now point out some interesting research directions for future work.

- First, the last open problem of infinite congruence classes should be addressed, since many interesting cases such as equipotency and identity fall in this category. Also the requirement that the E-subsumption preordering is well founded should be studied in detail: it may be possible that the property is true as soon as a minimal and complete unification algorithm exists for the theory E.

- Second, particular instances of our algorithm should be studied carefully as it was done for associativity and commutativity [67]. We believe that various kinds of permutative axioms have similar properties. This would allow an improvement over previous algorithms [50, 36]. Making various experiments should help a lot for studying such improvements. We are actually starting such experiments with the REVE system [55, 47].

- Third, we believe that our completion procedure can be improved: we conjecture that the transformation expressed in Lemma 30 could be incorporated into the algorithm (the one expressed in lemma 29 is already achieved in the algorithm). This is not straightforward, since the set R of rules becomes R'-Church-Rosser at the end, but does not have the property during the run of the algorithm. However, a rule can be considered to be *Church-Rosser* as soon as its critical pairs have been computed. As a consequence, dropping such rules into P when they become reducible in the sense of Lemma 30 could be sound, but this has to be proved.

- Last, we can imagine interesting rewriting relation other than those studied here: In OBJ [25], David Plaisted has implemented a very efficient reduction relation for the case of associative commutative operators (and even for associative commutative idempotent operators having an identity): Before applying an associative commutative matching algorithm to find a redex, the term to be reduced is sorted in a lexicographic way. This technique allows one to use a much more efficient associative commutative matching algorithm which makes the reductions much faster [69]. This technique should be studied carefully and, possibly extended to other cases.

6 Acknowledgements

The theory of *Equational Term Rewriting Systems* presented here lacks many examples. W apologize for this drawback and explain the reason: interesting examples are simply intractib by hand. Only computer experiments can provide such examples and we intend to present suc computer experiments in a future report, as soon as we are finished with the implementation these results in the REVE 3 system. REVE is a Term Rewriting Laboratory developed by th EURECA group at the CRIN and John Guttag's group at MIT. REVE 1 has bee implemented by Pierre Lescanne [55]. REVE 2 has been implemented by Randy Forgaard [2] The main core of REVE 3 has been implemented by Claude and Helene Kirchner [47]. Mar other people at CRIN and at MIT have also contributed to this last implementation which still going on. We are happy to thank them all here, especially Claude Kirchner.
We are also grateful to Jose Meseguer for carefully reading the manuscript and to Rand Forgaard for exhaustively pointing out the typos.

References

[1] Aho, A., Sethi, R. and Ullman, J.D., *Code Optimization and Finite Church-Rosser Theorems,* Design and Optimization of Compilers, Rustin, R., ed., Prentice Hall, , 1972, pp. 89-105.

[2] Ballantyne, A.M. and Lankford, D.S., *New Decision Algorithms for Finitely Presented Commutative Semigroups,* Computer & Mathematics with Applications, 7 (1981), pp. .

[3] Bergman, G.M., *The Diamond Lemma for Ring Theory,* Advances in Mathematics, 29 (1978), pp. 178-218.

[4] Book, R., *Confluent and Other Types of Thue Systems,* Journal of the ACM, 29 (1982), pp. 171-182.

[5] Buchberger, B. and Loos, R., *Algebraic Simplifications,* Computer Algebra, Symbolics and Algebraic Computations, Computing Supplementum 4, Buchberger, B., Collins, G.E. and Loos, R., ed., Springer Verlag, , 1982, pp. 11-43.

[6] Cohn, P.M., *Universal Algebra,* Harper and Row, , 1965.

[7] Corbin, J. and Bidoit, M., *A Rehabilitation of Robinson's Unification Algorithm,* Proceedings, 1983 IFIP Congress, , ed., North-Holland, , 1983, pp. .

[8] Deransart, P., *Dérivation de Programmes PROLOG a partir de Spécifications Algébriques,* Tech. Rep. , INRIA, France, , 1983.

[9] Dershowitz, N., *A note on Simplification Orderings,* Information Processing letters, 9 (1979), pp. 212-215.

[10] Dershowitz, N., *Ordering for Term-rewriting Systems,* Journal of Theoretical Computer Science, 17 (1982), pp. 279-301. Preliminary version in 20th Symposium on Foundations Of Computer Science, 1979.

[11] Dershowitz, N., *Computing with Term Rewriting Systems,* Tech. Rep. , University of Illinois, , 1983. to appear in Information and Control.

[12] Dershowitz, N. and Manna, Z., *Proving Termination with Multiset Orderings,* Communications of the ACM, 22 (1979), pp. 465-476. Also in Proceedings 6th International Conference on Algebra, Languages and Programming, 1979.

[13]
Dershowitz, N. and Marcus, L., *Existence and Construction of Rewrite Systems,* Tec Rep. , The Aerospace Corporation, , 1982.

[14]
Dershowitz, N., Hsiang,J., Josephson, N. and Plaisted,D., *Associate-Commutative Rewriting,* Proceedings 10th International Joint Conference on Artificial Intelligence, (1983), pp. .

[15]
Evans, T., *The word problem for Abstract Algebras,* J. London Math. Soc., 26 (1951), pp. 64-71.

[16]
Evans, T., , , (1963), pp. .

[17]
Fages, F., Formes Canoniques dans les Algèbres Boolèennes et Application a la Dèmonstration Automatique en Logique du Premier Ordre, . Thèse de troisieme cycle Universitè Paris 7.

[18]
Fages F., *Associative-Commutative Unification,* Proceedings 7th Conference on Automated Deduction, Napa Valley, (1984), pp. .

[19]
Fages, F. and Huet, G., *Unification and Matching in Equational Theories,* Proceedings 5th Conference on Automata, ALgebra and Programming, L'Aquila, , , 1983, pp. .

[20]
Fay, M., *First Order Unification in Equational Theories,* Proceedings, 4th Conference on Automated Deduction, , ed., Springer-Verlag, Lecture Notes in Computer Science, Volume 87, , 1979, pp. 161-167.

[21]
Forgaard, R., A Program for Generating and Analyzing Term Rewriting Systems, . Master's thesis, MIT-LCS.

[22]
Fribourg, L., *A Superposition Oriented Theorem Prover,* Proceedings 10th International Joint Conference on Artificial Intelligence, (1983), pp. .

[23]
Goguen, J. A., *How to Prove Algebraic Inductive Hypotheses without Induction: with Applications to the Correctness of Data Type Representations,* Proceedings, 5th Conference on Automated Deduction, W. Bibel and R. Kowalski, ed., Springer-Verlag, Lecture Notes in Computer Science, Volume 87, , 1980, pp. 356-373.

[24]
Goguen, J.A. and Meseguer, J., *Equality, Types, Modules and Generics for Logic Programming,* , (1984), pp. .

[25]
Goguen, J. A., Meseguer, J., and Plaisted, D., *Programming with Parameterized Abtract Objects in OBJ,* Theory and Practice of Software Technology, , ed., North-Holland, , 1982, pp. 163-193.

[26]
Guttag, J. V., Horowitz, E. and Musser, D. R., *Abstract Data Types and Software Validation,* Communications of the ACM, (1978), pp. .

[27]
Hsiang, J., *Refutational Theorem Proving using Term Rewriting Systems,* Artificial Intelligence Journal, (), pp. . To appear.

[28]
Hsiang, J., Dershowitz, N., *Rewrite Methods for Clausal and non Clausal Theorem Proving,* Proceedings 10th International Conference on Algebra, Languages and Programming, (1983), pp. .

[29]
Hsiang, J. and Plaisted, D., *Deductive Program Generation,* Tech. Rep. , University of Illinois, Computer Science Department, , 1982.

[30]
Huet, G., *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems,* Journal of the Association for Computing Machinery, 27 (1980), pp. 797-821. Preliminary version in 18th Symposium on Foundations Of Computer Science, IEEE, 1977.

[31]
Huet, G., *A Complete Proof of Correctness of the Knuth and Bendix Completion Algorithm,* Journal of Computer Systems and Sciences, 23 (1981), pp. 11-21.

[32]
Huet, G., Hullot, J.M., *Proofs by Induction in Equational Theories with Constructors,* Journal of the Association for Computing Machinery, 25 (1982), pp. 239-266. Preliminary version in Proceedings 21th Symposium on Foundations Of Computer Science, IEEE, 1980.

[33]
Huet, G. and Oppen, D., *Equations and Rewrite Rules: A Survey,* Formal Language Theory: Perspectives and Open Problems, Book, R., ed., AConference on Automated DEductionmic Press, , 1980, pp. .

[34]
Hullot, J.M., *Associative-Commutative Pattern Matching,* Proceedings 9th International Joint Conference on Artificial Intelligence, (1979), pp. .

[35]
Hullot, J.M., *Canonical Forms and Unification,* Proceedings, 5th Conference on Automated Deduction, W. Bibel and R. Kowalski, ed., Springer-Verlag, Lecture Notes in Computer Science, Volume 87, , 1980, pp. 318-334.

[36]
Jeanrond, J., *Deciding Unique Termination of Permutative Rewrite Systems: Choose your Term Algebra Carefully,* Proceedings, 5th Conference on Automated Deduction, W. Bibel and R. Kowalski, ed., Springer-Verlag, Lecture Notes in Computer Science, Volume 87, , 1980, pp. .

[37]
Jouannaud, J.P., *Church-Rosser Computations with Equational Term Rewriting Systems,* J. ACM, (submitted), pp. . Preliminary version in Proceedings 4th Conference on Automata, Algebra and Programming, L'Aquila, Springer Lecture Notes in Computer Science number 159, 1983.

[38]
Jouannaud, J.P., Lescanne, P., *On Multiset Ordering,* Information Processing letters, 15 (1982), pp. .

[39]
Jouannaud, J.P. and Munoz M., *Termination of a Set of Rules Modulo a Set of Equations,* Proceedings 7th Conference on Automated Deduction, Napa Valley, (1984), pp. .

[40]
Jouannaud, J.P., Kirchner, H. and Remy, J.L., *Church-Rosser Properties of Weakly Terminating Equational Term Rewriting Systems,* Proceedings 8th International Joint Conference on Artificial Intelligence, Karlsruhe, (1983), pp. .

[41]
Jouannaud, J.P., Kirchner, C. and Kirchner, H., *Algebraic Manipulations as a Unification and Matching Strategy for Equations in Signed Binary Trees,* Proceedings 7th International Joint Conference on Artificial Intelligence, Vancouver, (1981), pp. .

[42]
Jouannaud, J.-P., Kirchner, C., Kirchner, H., *Incremental Construction of Unification Algorithms in Equational Theories,* Automata, Languages and Programming, Barcelona 1983, , , 1983, pp. 361-373.

[43]
Jouannaud, J.-P., Lescanne, P., Reinig, F., *Recursive Decomposition Ordering,* IFIP Working Conference on Formal Description of Programming Concepts II, , ed., North-Holland, 1983, edited by D. Bjorner., , 1983, pp. .

[44]
Kamin, S. and Levy, J.J., Attempts for Generalizing the Recursive Path Ordering, Unpublished draft.

[45]
Kirchner, C., *A New Equational Unification Method: A Generalization of Martelli-Montanari Algorithm,* Proceedings 7th Conference on Automated Deduction, Napa Valley, (1984), pp. .

[46]
Kirchner, C. and Kirchner, H., Résolution d'Equations dans les Algèbres Libres et les Variétés Equationelles d'Algèbres, . Thèse de troisieme cycle, Université Nancy I.

[47]
Kirchner, C. and Kirchner, H., *Current Implementation of the general E-completion Algorithm,* Tech. Rep. , Centre de Recherches en Informatique de Nancy, , 1983. Internal Note of the EURECA Group.

[48]
Kirchner, H., *A General Inductive Completion Algorithm and Application to Abstract Data Types,* Proceedings 7th Conference on Automated Deduction, Napa Valley, (1984), pp. .

[49]
Knuth, D. and Bendix, P., *Simple Word Problems in Universal Algebra,* Computational Problems in Abstract Algebra, J. Leech, ed., Pergamon Press, , 1970, pp. .

[50]
Lankford, D. and Ballantyne, A., *Decision Procedures for Simple Equational Theories with Permutative Axioms: Complete Sets of Permutative Reductions,* Tech. Rep. , Univ. of Texas at Austin, Dept. of Mathematics and Computer Science, , 1977.

[51]
Lankford, D. and Ballantyne, A., *Decision Procedures for Simple Equational Theories with Commutative Axioms: Complete Sets of Commutative Reductions,* Tech. Rep. , Univ. of Texas at Austin, Dept. of Mathematics and Computer Science, , 1977.

[52]
Lankford, D. and Ballantyne, A., *Decision Procedures for Simple Equational Theories with Associative Commutative Axioms: Complete Sets of Associative Commutative Reductions,* Tech. Rep. , Univ. of Texas at Austin, Dept. of Mathematics and Computer Science, , 1977.

[53]
Lankford, D.S. and Butler, G.,, , Tech. Rep. , Mathematics Department, Louisiana Tech University, , 1983.

[54]
Lechenadec, P., *Canonical Forms in Finitely Presented Algebras,* Proceedings 7th Conference on Automated DEduction, Napa Valley, (1984), pp. .

[55]
Lescanne P., *Computer Experiments with the REVE Term Rewriting Systems Generator,* Proceedings, 10th ACM Symposium on Principles of Programming Languages, , ed., ACM, , 1983, pp. .

[56]
Lescanne, P., *Term Rewriting Systems and Algebra,* Proceedings 7th Conference on Automated Deduction, Napa Valley, (1984), pp. .

[57]
Lescanne, P., *How to Prove Termination? An Approach to the Implementation of a New Recursive Decomposition Ordering,* Proceedings 6th Conference on Automata, ALgebra and Programming, Bordeaux, (1984), pp. .

[58]
Livesey, M. and Siekman, J., *Unification of Bags and Sets,* Tech. Rep. , Institut fur Informatik I, Universitat Karlsruhe, , 1976.

[59]
Mètivier, B., *About the Rewriting Systems produced by the Knuth-Bendix Completion Algorithm,* Information Processing letters, 16 (1983), pp. .

[60]
Marchand, P., *Langages d'Arbres, Langages dans les Algèbres Libres,* Ph.D. Thesis, Universite de NANCY, , 1981.

[61]
Milner, R., *A Theory of Type Polymorphism in Programming,* Journal of Computer and System Sciences, 17 (1978), pp. 348-375.

[62]
Musser, D., *On Proving Inductive Properties of Abstract Data Types,* Proceedings, 7th ACM Symposium on Principles of Programming Languages, (1980), pp. .

[63]
Newman, M.H.A., *On Theories with a Combinatorial Definition of 'Equivalence',* An of Math., 43 (1942), pp. 223-243.

[64]
Padawitz, P., *Equational Data Type Specification and Recursive Program Scheme,* IFIP Working Conference on Formal Description of Programming Concepts II, , ed., North-Holland, 1983, edited by D. Bjorner., , 1983, pp. .

[65]
Paterson M.S. and Wegman M.N., *Linear Unification,* Journal of Computer Systems and Sciences, 16 (1978), pp. 158-167.

[66]
Peterson, G., *A Technique for Establishing Completeness Results in Theorem Proving with Equality,* SIAM Journal of Computing, 12 (1983), pp. 82-100.

[67]
Peterson, G. and Stickel, M., *Complete Sets of Reductions for Some Equational Theories,* Journal of the ACM, 28 (1981), pp. 233-264.

[68]
Plaisted, D., *An Associative Path Ordering,* Tech. Rep. , University of Illinois, Computer Science Department, , 1983.

[69]
Plaisted, D., , . Personal Communication.

[70]
Plotkin, G., *Building-in Equational Theories,* Machine Intelligence, 7 (1972), pp. 73-90.

[71]
Raoult, J.C., *Finiteness Results in Term Rewriting Systems,* RAIRO Informatique Theorique, 4 (1981), pp. 373-391.

[72]
Sethi, R., *Testing for the Church-Rosser Property,* Journal of the Association for Computing Machinery, 21 (1974), pp. 671-679.

[73]
Sethi, R., *Control Flow Aspects of Semantics-Directed Compiling,* ACM-Transactions On Programming Languages And Systems, 5 (1983), pp. 554-595.

[74]
Siekman, J. and Szabo, P., *A Noetherian and Confluent Rewrite System for Idempotent Semigroups,* Semigroup Forum, 25 (1982), pp. 83-110.

[75]
Siekman, J. and Szabo, P., *Universal Unification and Classification of Equational Theories,* Proceedings 6th Conference on Automated DEduction, Lecture Notes in Computer Science, 138 (1982), pp. .

[76]
Stickel, M.E., *A complete unification Algorithm for Associative-Commutative functions,* Journal of the ACM, 28 (1981), pp. 423-434. Preliminary version in Proceedings 4th International Joint Conference on Artificial Intelligence, Tbilissi, 1975.

List of Figures

IMPLEMENTATION OF A GENERAL COMPLETION PROCEDURE PARAMETERIZED BY BUILT-IN THEORIES AND STRATEGIES

Claude Kirchner and Helene Kirchner(*)

Centre de Recherche en Informatique de Nancy
BP 239
54506 Vandoeuvre Les Nancy Cedex
France

ABSTRACT

This paper describes a software which presents an unified point of view of several known completion procedures. It allows working with built-in theories and strategies and is aimed to perform proofs and experiments in term rewriting systems.

1. INTRODUCTION

The Knuth and Bendix's completion procedure [Knuth1970] originally computes a term rewriting system R which generates the same equivalence on terms as a given set of equations A. Moreover R is proved to have the so-called Church-Rosser property, which allows deciding equality of two terms by using rewritings only, provided R satisfies uniform termination. A proof method for A-equality is derived which consists of computing irreducible forms of the two members of an equational theorem and checking for their identity.

This method was extended to handle the case of an equational term rewriting system, that is a pair composed of a set of rules R and a set of axioms E. A first approach due to Lankford and Ballantyne [Lankford1977] handles the case of permutative axioms that generate finite E-congruence classes. The case of infinite E-congruence classes is studied in [Huet1980,Peterson1981,Jouannaud1983]. Huet's approach is restricted to sets R of left linear rules, while Peterson and Stickel's one is restricted to theories

(*) This research was partly supported by the GRECO of programmation and by ADI under Grant 82/767.

E defined by left and right linear axioms and for which a finite and complete unification algorithm is known. These results were unified by Jouannaud who described the underlying computations used in both approaches. Moreover his results allowed dropping all the previous linearity conditions. Based on these ideas, a very general completion procedure was described in [Jouannaud1984] that subsumes all known completion algorithms. The purpose of this paper is to describe an implementation of all these algorithms, perform experiments and compare them. Our experimental version is hereafter called REVEUR-3 since it has been designed as an extension of the REVE software, written in CLU [Liskov1981]. REVE is a rewrite rule laboratory, that is a generator of term rewriting systems based on the Knuth and Bendix's completion procedure. Among many interesting features, it provides automatic or semi-automatic proofs of termination of the generated rewriting system. Two previous versions of REVE are REVE-1 [Lescannel983] and REVE-2 [Forgaard1984] which is currently the distributed version of REVE.

After a review of the theoretical framework in section 2, we emphasize in section 3 the originality of REVEUR-3 and describe some of our experiments in the last section.

2. THEORETICAL FRAMEWORK

This paper is aimed at readers who are familiar with the basic notions of term rewriting systems and completion procedure, including terms, occurrences, substitutions, equational equality generated by a set of axioms denoted ~E, rewriting rule, rewriting system, normal form of a term t denoted hereafter t↓, critical pair. Definitions of these concepts can be gleaned from [Huet1980a].

All the theoretical bases of this section can be found in [Jouannaud1984]. We only remind here the main concepts.

2.1. Equational rewriting

The main originality of REVEUR-3 is to allow equational rewriting that is to use mixed sets of rules and equations. This need comes from the fact that some permutative equations like commutativity cannot be oriented into rewrite rules without compromising the finite termination of the rewriting process. In order to take into account such equations, the first idea is to work on equivalence classes of terms. Thus if E is the set of (non directed) axioms, the set of terms is quotiented by the equivalence relation ~E. A set of rewrite rules R induces then a reduction relation on equivalence classes:

T -> T' iff there exists t in T and t' in T' such that t ->R t'

where ->R is the standard rewriting relation on terms defined by :

t ->R t' iff there exist a subterm t1 of t at occurrence u
a substitution σ
and a rule g -> d in R
such that t1 = σ(g) and t' = t[u \ σ(d)] (which denotes the term t where

t1 has been replaced by σ(d)).

The reduction relation on E-equivalence classes of terms cannot be efficiently implemented except perhaps in some particular cases. Moreover it can be undecidable when E-equivalence classes are infinite. Thus different attempts have been made in order to define a rewriting relation on terms which simulates the reduction relation ->.

Peterson and Stickel proposed a second type of term rewriting, called rewriting modulo E, which needs an E-matching algorithm:

t ->R,E t' iff there exist a subterm t1 of t at occurrence u,
a substitution σ
and a rule g-> d in R
such that t1 ~E σ(g) and t' = t[u \ σ(d)]

Obviously the standard rewriting relation is a particular case of the second one but it is conceptually simpler than the other and computationally more efficient.

Another term rewriting relation has been imagined by Jouannaud, which combines these two ones. Splitting the set of rules into two parts, left linear rules Rl and non left linear ones Rnl, he defines ->(RlURnl,E) as either a rewriting using a rule of Rl or a rewriting modulo E using a rule of Rnl. This idea allowed him to generalize previous results of Huet and Peterson and Stickel.

For any binary relation ->, let us denote by (->)-1 the symmetric relation, -+-> its transitive closure, -*-> its reflexive transitive closure and <-*-> the equivalence relation generated by ->.

We are now ready to define several Church-Rosser properties.

The reduction relation -> on E-equivalence classes of terms is said Church-Rosser iff
T <-*-> T' implies there exists T" such that T -*-> T" <-*- T'.

In Huet's, Peterson and Stickel's and Jouannaud's approaches, other Church-Rosser properties are used. They are defined on terms and no more on E-equivalence classes of terms. All of them imply the Church-Rosser property on E-equivalence classes of terms.

2.2. Correspondence between rewriting relation and Church-Rosser property.

Let us denote by ~(RUE) the equivalence relation which is the transitive closure of (->R U (->R)-1 U ~E).

The following table shows, for each previously mentionned approach, the correspondence between the rewriting relation and the Church-Rosser property.

Knuth and Bendix's method:
E empty, R all rules t ~(RUE) t'

->R	iff there exists t" s.t. t -*->R t" R<-*

Huet's method:

E non empty,	t ~(RUE) t'
R left linear rules	iff there exists t"1, t"2
->R	s.t. t -*->R t"1 ~E t"2 R<-*- t'.

Peterson and Stickel's method:

E linear axioms,	t ~(RUE) t'
R rules	iff there exists t"1, t"2
->R,E	s.t. t -*->R,E t"1 ~E t"2 R,E<-*- t'

Jouannaud's method:

any E non empty,	t ~(RUE) t'
Rl left linear rules	iff there exists t"1, t"2 s.t.
Rnl non left linear rules	t -*->(RlURnl,E) t"1 ~E t"2 (RlURnl,E)<-*- t'
->(RlURnl,E)	

Let us emphasize that these Church-Rosser properties are actually similar up to the rewriting relation used. If ->R' is this rewriting relation, the corresponding Church-Rosser property is:

t ~(RUE) t' iff there exists t"1, t"2
such that t -*->R' t"1 ~E t"2 R'<-*- t'.

In the following, we denote this property as the "R'-Church-Rosser property". As soon as it is satisfied and ->R' terminates, we get a decision procedure for (RUE)-equality by computing the R'-normal forms of t and t' and testing their E-equality. Thus to each choice of a ->R' rewriting relation, corresponds a theorem proving method. But notice the increasing power of the rewriting relations:

->Rl included into ->(Rl U Rnl,E) included into ->(Rl U Rnl),E.

Thus we get corresponding sorted Church-Rosser properties and a classification of the different equational proof methods when there are axioms. For example, we can say that in general Huet's rewriting method is less powerful than Jouannaud's in the following sense: any equational theorem that can be proved with ->Rl as rewriting method can also be proved with ->(RlURnl,E). This last rewriting method is itself less powerful than Peterson and Stickel's one. Nevertheless let us already mention that the implementation of a rewrite rule laboratory providing the different possibilities allowed us to compare these methods with respect to other criteria, as described in the last section.

2.3. A general completion procedure.

The aim of a completion procedure is to compute from a set of equations P and a set of axioms E, a set of rewrite rules R such that ~(RUE) is equal to ~(PUE) and such that a Church-Rosser property is satisfied.

We give here a recursive form of the general completion procedure implemented in REVEUR-3. It works with a set of rules R, a set of equations P and a possibly empty set E of axioms. Axioms of E can be seen as defining a theory or

as defining properties of operators. For example, E can define an associative commutative theory, that is more precisely one or more operators op1, op2, ..., opi,... satisfying the following axioms:

(x opi y) = (y opi x)
((x opi y) opi z) = (x opi (y opi z))

Contrary to the fixed set E, P is a set of equations which evolves during the completion process. It is the set of equations which have to be directed into rules. In addition a well-founded reduction ordering > is provided for this purpose, which must be compatible with the E-equivalence classes (i.e. t ~E t' implies t ≧ t' and t' ≧ t).

This procedure is said general because any known completion procedure can be expressed as a particular instance of it, using parameterization at different levels. First the procedure can be parameterized by the set of axioms E. Second, the four main operations in a completion procedure are:

- normalization of terms,
- orientation of equations into rules,
- simplification of other rules and equations using the new added rules,
- computation of critical pairs between rules and between rules and axioms.

For each of them, changing some parameters leads to a different behaviour of the completion process. For example, the choice of the rewriting relation used in

---

```
PROCEDURE E-COMPLETION(P, R, E, >)
IF P is not empty
THEN choose a pair (p, q) in P
     p' = p↓ ; q' = q↓
        CASE p' ~E q' THEN E-COMPLETION(P-{(p, q)}, R, E, >)
             p' > q'  THEN (P, R) = SIMPLIFICATION(P-{(p, q)}, R, p'->q')
                           E-COMPLETION(P, RU{p'->q'}, E, >)
             q' > p'  THEN (P, R) = SIMPLIFICATION(P-{(p, q)}, R, q'->p')
                           E-COMPLETION(P, RU{q'->p'}, E, >)
                      ELSE STOP with FAILURE
        END CASE
ELSE IF all rules in R are marked
     THEN STOP with SUCCESS
     ELSE Choose an unmarked rule l->r
          (P, R) = CRITICAL-PAIRS(l->r, R, E)
          Mark the rule l->r in R
          E-COMPLETION(P, R, E, >)
     END IF
END IF
END E-COMPLETION
```

---

the normalization implies a specific Church-Rosser property and thus a new proof method for deciding equational equality. Thus parameterization can also be introduced at the level of these basic operations.

Before developing this idea which appears as the originality of REVEUR-3, let us first point out some theoretical problems which are introduced by the fact to work with axioms.

2.4. Theoretical problems

From the theoretical study of the problem, it is made clear that two properties, namely confluence and coherence are necessary and sufficient conditions for Church-Rosser properties, assuming the termination of the reduction relation on E-equivalence classes -> and provided these classes are finite.
In addition to confluence, coherence is required to enable computations in E-equivalence classes; more precisely coherence is the necessary and sufficient condition for -> and ->R' to have the same normal forms.

Coherence and confluence can be checked on an adapted notion of critical pairs. Thus when working modulo a set of axioms E, critical pairs must be computed both between rules (confluence critical pairs) and between rules and axioms (coherence critical pairs). The first ones must satisfy the confluence property, the second ones the coherence property depicted below:

For any confluence critical pair (p,q) :
p -*-> p' ~E q' <-*- q (confluence property)

For any coherence critical pair (p,q) :
p -*-> p' ~E q' <-*- q1 <- q (coherence property)

Notice that coherence needs to reduce at least once the right hand side of a coherence critical pair obtained for instance by overlapping a rule l->r into an axiom g=d at occurrence u. This term q is an instance of d, denoted $\sigma(d)$. The difficulty comes from the fact that the rule which is used to perform the reduction of q at some step of the completion process, can be later removed and replaced by a new one, during the SIMPLIFICATION procedure. Thus to ensure coherence, it can be necessary to protect an existing rule which reduces a right hand side of a coherence critical pair. When no such rule exists, it is necessary to introduce a new rule q -> $p\downarrow$ or an extension. The extension introduced for a non left linear rule l->r is defined as $g[u\backslash l]\to g[u\backslash r]\downarrow$. Notice that this definition generalizes Peterson and Stickel's extensions and that such a rule reduces the right hand side of the corresponding coherence critical pair because $\sigma(d) \sim_E \sigma(g) \sim_E \sigma(g[u\backslash l])$. Except when the rule l->r is deleted, neither extensions nor protected rules are allowed to be deleted from the rewriting system, since this would compromise the Church-Rosser property.

Up to now two methods have been proposed in order to ensure the coherence:

- the first one consists of testing reducibility of the right hand side of coherence critical pairs with already existing rules. This method avoids introducing useless extensions but needs to protect some rules.
- the second method consists of automatically adding extensions for any

rule in the system. The form and the number of extensions can be deduced from the axioms and the rule itself. In the associative commutative theory case, it is proved that it is not necessary to introduce extensions of extensions. But in general, this method possibly leads to add infinitely many extensions.

The second theoretical problem arises when trying to work with theories E with non finite equivalence classes. Putting in E an axiom like idempotency (x+x = x), or involution -(-x) = x, leads to this situation. The tools developed in this case are yet more complex and up to now, we only got a sufficient condition to ensure the Church-Rosser property, which seems a little too strong.

As a third problem, let us point out that orientation of equations into rules is also more difficult in the equational case. More precisely, what we want to get is the termination of the reduction relation -> on E-equivalence classes, called E-termination property. Little is known about this property. A theoretical study of the problem can be found in [Jouannaud1984a] and effective, yet complex methods for associative commutative theories in [Dershowitz1983]. More recent results in this last case are given in [Bachmair1985].

3. ORIGINALITY OF REVEUR-3

Our main objective was to implement a general completion procedure which allows both built-in equational theories E and experimentation of the different completion processes mentionned before. This aim implies the modularity of the system in order to allow easy changes and enrichments.

Thus from the user external point of view the software provides different functionalities and seems to have different behaviours. In this way it is a rewrite rule laboratory in the same vein as its predecessors REVE-1 and REVE-2. On the contrary, from the designer internal point of view, it appears as a general and unified procedure. Let us detail and specify more precisely these ideas.

3.1. Built-in equational theories.

The parameterization of the general completion procedure by the set of axioms E involves generalizations of three basic operations: equality decision, matching and unification. All of them have to take into account the existence of equational properties on some function symbols. For example, a symbol + may be associative and commutative, another distributive on +, a third one may have no property. To each property corresponds a set of axioms which is explicitely used in the completion procedure to compute coherence critical pairs. On the other hand, equality decision, matching and unification use the properties of operators and work on terms which are built from all these symbols. In [Kirchner1984], it is shown that such a unification procedure can be designed in a very general way. Briefly, it is based on three operations: decomposition of equations (which determines their common part and their sets of disagreements), merging all the conditions on the same variable, and normalization (which replaces a no longer decomposable equation, say (t=t'), by a system of equations

whose solutions are also solutions of (t=t')). Normalization works on equations (t=t') where the top symbols of t and t' have the same equational property and thus is based on specific processes in the built-in equational theories.

Thus working with built-in theories together means to attach equational properties to function symbols, to introduce explicit axioms and to design specific processes used for equality decision, matching and unification. In REVEUR-3 a built-in equational theory has been implemented as a module composed of axioms and of special procedures: equality, matching, unification. The name of the theory is attached to the operators satisfying the axioms. Up to now, the empty theory (E is the empty set of axioms) and the associative-commutative theory are implemented.

3.2. Strategies.

A strategy is a set of parameters which determine a particular behaviour of the completion process. We have grouped together under this concept several more or less original ideas.

- From our theoretical study of E-completion, the idea arises to design a system which works with different rewritings and use different methods to test the coherence property.
- The state of art about automatic orientation compelled us to introduce at least an interactive way to orient them. The introduction of a possible choice between a manual and an automatic orientation has been conforted by some other experiments with REVE.
- In the same vein, we have been convinced that it can be useful to choose a particular superposition strategy between rules either for efficiency reasons or in order to perform experiments.

In REVEUR-3, different parameters of a completion process may be set according to the user's choices and according to them, the system has a different behaviour. We review in this section three kinds of choices which are effectively implemented. Of course a lot of other ones could be added.

3.2.1. Choice of the rewriting relation.

Let us explain more precisely how we have implemented the choice of the rewriting relations. The completion procedure always works with two sets of rules named R1 and R2. In general, standard rewriting is performed using rules of R1, while rewriting modulo the axioms E is performed using rules of R2. Now, according to the different Church-Rosser properties the user may choose, the sets of rules are built as follows.

Knuth and Bendix's method:

E empty, R all rules	all rules are put in R1

Huet's method:

E non empty,	only left linear rules are allowed and
R left linear rules	put in R1

Peterson and Stickel's method:

E linear axioms,	all rules are put in R2
R rules	

Jouannaud's method:

any E non empty,	left linear rules are put in R1
R1 left linear rules	non left linear rules are put in R2
Rnl non left linear rules	

The choice of the rewriting relation is thus entirely implemented only by the way to introduce rules in R1 or R2.

3.2.2. Choice of coherence check.

According to the rewriting relation chosen by the user, a strategy for ensuring the coherence may be proposed. The choice arises in the way to add extensions. Of course, with an empty set of axioms E or Huet's method, there is no need to add extensions. But with other methods the user can choose between checking the coherence with already existing rules, or systematically adding extensions for the rules of R2. As mentionned before, this last method, actually chosen by Peterson and Stickel, is valid with associative commutative theories. Thus the user who wants to make experiments in associative commutative theories may choose between the two possibilities and the completion process will use the corresponding procedure for coherence critical pairs.

3.2.3. Choice of superposition strategy.

Two superposition strategies are provided in order to overlap rules: each rule with all the previously introduced (or older) ones, or each rule with smaller ones. Since each rule is superposed with all rules which are before it in the list of rules, changing the superposition strategy is equivalent to change the way to classify rules when they are introduced in the list. If the list is sorted in decreasing order according to the age of the rule, the superpositions will be made with all the previously introduced rules. Whereas if the list is sorted by increasing order with respect of the size of the rules, each rule will be superposed with smaller ones.

3.3. A general completion procedure.

Beyond the design of general procedures for equality decision, matching and unification working with built-in theories, some other generalizations are required for the general completion procedure we propose.

Problems are due to the complexity of the E-completion process: some

procedures, especially normalization and simplification, need standard rewritings associated with a first subset of rules and rewritings modulo E associated with an other subset. Thus the reduction process itself is parameterized by the type of matching, which can be either usual matching, or E-matching. The same feature appears at the critical pairs level, where unification must be performed sometimes in the empty theory, sometimes in the E theory.

On the other hand, let us mention that a complex implementation of a rewriting system was needed because checking the coherence property needs to add extensions and to protect some rules. Accordingly the simplification process of the rewriting system by the new introduced rules becomes much more complex and needs access to extensions and to protected rules.

4. EXPERIMENTS.

We present in this section some examples which allow comparisons between different strategies. From experiments the following idea arises: according to the strategies used in REVEUR-3, the same starting set of equations can be completed using different methods which are more or less powerful with respect to some criteria. A first criterion is the termination of the completion process: a strategy which allows finding a finite rewriting system R such that (RUE) is equivalent to the starting equational theory can be considered as more interesting than a strategy with which the completion process fails with a non orientable equation or generates an infinite set of rewrite rules. A second criterion illustrated in our examples is the time consumed by the completion process. This time is strongly related to the efficiency of the rewriting relation which can be considered as a third criterion: it is clear that rewriting modulo E is very expensive and less it is used, more efficient is the rewriting. Thus according to the efficiency criterion, we can say that ->(R1 U Rn1,E) is more efficient than ->(R1 U Rn1),E.

We now study on three examples the effect of changing the rewriting strategy on the generated set of rules.

5. Abelian groups.

For abelian groups, Lankford and Ballantyne together with Peterson and Stickel have found a term rewriting system satisfying the R,E-Church-Rosser property. The same experiment was performed with REVEUR-3. Our completion process was initialized with the rewriting relation ->R,E and the following sets of axioms and equations:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
```

User equations:

1. (x + 0) == x

2. (x + i(x)) == 0

No critical pair equations.

No rewrite rule in R1.

No rewrite rule in R2.

REVEUR-3 terminates with the message:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
```

No rewrite rule in R1.

Rewrite rules in R2:

1. i(0) -> 0
2. (x + 0) -> x
   Which has for extensions:
3. (z + (x + 0)) -> (z + x)

4. i(i(x)) -> x
5. (x + i(x)) -> 0
   Which has for extensions:
6. (z + (x + i(x))) -> z

7. i((x + z)) -> (i(z) + i(x))

Your system is complete!

Using now the rewriting relation ->(R1 U Rn1,E), REVEUR-3 terminates with the message:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
```

Rewrite rules in R1:

1. i(0) -> 0
2. (x + 0) -> x
3. (0 + x) -> x
4. i(i(z)) -> z
5. i((z + x)) -> (i(z) + i(x))

Rewrite rules in R2:

6. (x + i(x)) -> 0
   Which has for extensions:
7. ((x + i(x)) + z) -> z

Your system is complete!

Notice that now the rule 0 + x -> x is needed to insure equivalence between -> and ->(R1 U Rn1,E)-reducibility. The second completion is twice faster than the first one. It is due to the fact that less associative-commutative unification and matching are used in the second case. This result is new in the following sense: it provides a rewriting relation (here ->(R1 U R2,E)) which allows deciding equality in abelian groups and which is more efficient than the previous one proposed by Peterson and Stickel.

5.1. Commutative monoid with two generators and identity

Let us now consider the set of equations:

```
0 + x == x
a + b == 0
(x + a) + b == x
```

and assume the associativity and commutativity of the + symbol. Using ->R,E as rewriting relation, REVEUR-3 terminates with the message:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
```

No rewrite rule in R1.

Rewrite rules in R2:

1. (0 + x) -> x
   Which has for extensions:
2. (z + (0 + x)) -> (z + x)

3. (a + b) -> 0
   Which has for extensions:
4. (z + (a + b)) -> z

5. ((x + a) + b) -> x
   Which has for extensions:
6. (z + ((x + a) + b)) -> (z + x)

Your system is complete!

But using ->R1 or ->(R1 U Rn1,E) as rewriting relation, we get an infinite set of rules whose first ones are described in the following message:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
```

Rewrite rules in R1:

```
 1. (0 + x) -> x
 2. (x + 0) -> x
 3. (a + b) -> 0
 4. (b + a) -> 0
 5. ((x + a) + b) -> x
 6. (a + (b + z)) -> z
 7. (b + (a + z)) -> z
 8. ((x + b) + a) -> x
 9. (b + (x + a)) -> x
10. ((a + x) + b) -> x
11. ((b + z) + a) -> z
12. (a + (y + b)) -> y
13. ((x + a) + (b + z)) -> (x + z)
14. ((x + (y + a)) + b) -> (x + y)
15. (a + ((b + z) + z1)) -> (z + z1)

...

37. ((x + a) + ((b + z) + z1)) -> ((x + z) + z1)
38. ((x + (y + a)) + (b + z)) -> ((x + y) + z)
47. ((x1 + b) + ((x + a) + z)) -> (x1 + (x + z))

...
```

No rewrite rule in R2.

This last completion method is thus less interesting than the previous one, since it does not terminate. This example illustrates the difference of power between the two completion methods.

5.2. Arithmetic theory.

This third example is again a case where ->(R1 U Rn1,E) rewriting can be usefully chosen.

Let + and * be addition and multiplication declared as associative and commutative, s be the successor function and ** the exponentiation function. Stickel proposed a rewriting system which has the Church-Rosser property:

```
(0 + x) -> x
(0 * x) -> 0
(s(x) + y) -> s((x + y))
(s(x) * y) -> ((x * y) + y)
(x * (y + z)) -> ((x * y) + (x * z))
(x ** 0) -> s(0)
(s(0) ** x) -> s(0)
(x ** s(y)) -> (x * (x ** y))
(x ** (y + z)) -> ((x ** y) * (x ** z))
((x ** y) * (z ** y)) -> ((x * z) ** y)
```

REVEUR-3 with the rewriting strategy ->(R1URn1,E) terminates with the message:

You are currently working modulo the following axioms:

```
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
((x * y) * z) == (x * (y * z))
(x * y) == (y * x)
```

Rewrite rules in R1:

```
 1.  (0 + x) -> x
 2.  (0 * x) -> 0
 3.  (x + 0) -> x
 4.  (x * 0) -> 0
 5.  (x ** 0) -> s(0)
 6.  (s(0) ** x) -> s(0)
 7.  (s(x) + y) -> s((x + y))
 8.  (y + s(x)) -> s((x + y))
 9.  (s(x) * y) -> ((x * y) + y)
10.  (y * s(x)) -> ((x * y) + y)
11.  (x ** s(y)) -> (x * (x ** y))
12.  (x * (y + z)) -> ((x * y) + (x * z))
13.  ((y + z) * x) -> ((x * y) + (x * z))
14.  (x ** (y + z)) -> ((x ** y) * (x ** z))
```

Rewrite rules in R2:

```
15.  ((x ** y) * (z ** y)) -> ((x * z) ** y)
          Which has for extensions:
16.       (((x ** y) * (z1 ** y)) * z) -> (((x * z1) ** y) * z)
```

Your system is complete!

6. CONCLUSION.

The first main idea that can be extracted from this work is that there is not an unique method to complete a set of equations as soon as there is a built-in theory. The originality of the REVEUR-3 system is based upon this idea. As a consequence, REVEUR-3 needs interaction with the user who chooses his method. But let us also emphasize that the underlying theoretical approach both unifies different known completion processes and increases their power, in the sense that the general completion algorithm implemented in REVEUR-3 is able to deal with a larger class of equational theories: all the linearity conditions introduced by Huet or by Peterson and Stickel have been dropped in our approach. The power of the REVEUR-3 system is due to this fact.

Up to now our experiments have been performed with associative commutative built-in theories and thus do not provide unknown results about decidability of equational theories. Nevertheless our contribution was to provide a more efficient rewriting method for example for abelian groups and arithmetic theories. In this way our results can be seen as improving previous ones. In addition, REVEUR-3 has been designed in order to support any built-in theory for which algorithms for equality decision, matching and unification are known. The next developments of REVE will include such implementations. Among them let us mention for instance the minus theory [Kirchner1984] defined by the following axioms:

```
-(-x) = x
-(f(x,y)) = f(-y, -x)
```

and the permutative theory [Jeanrond1980,Kirchner1984a] defined by:

```
f(f(x, y) ,z) = f(f(x, z), y).
```

ACKNOWLEDGEMENTS

We sincerely thank Jean-Pierre Jouannaud, Pierre Lescanne and Jean-Luc Remy for their comments about a preliminary version of this paper.

REFERENCES

Bachmair1985.
L. Bachmair and D. Plaisted, "Associative Path Orderings," 1st Conference on Rewriting Techniques and Applications, (1985).

Dershowitz1983.
N. Dershowitz, J. Hsiang, N.A. Josephson, and D.A. Plaisted, "Associative-Commutative rewriting," Proceedings of the 8th IJCAI, Karlsruhe (West Germany), pp. 940-944 (1983).

Forgaard1984.
R. Forgaard and J.V. Guttag, "REVE: A Term Rewriting System Generator with Failure-Resistant Knuth-Bendix," MIT-LCS (1984).

Huet1980.
G. Huet, "Confluent reductions: abstract properties and applications to term rewriting systems," J. of ACM Vol. 27(4) pp. 797-821 (Oct. 1980).

Huet1980a.
G. Huet and D. Oppen, "Equations and Rewrite Rules: A Survey," in Formal Languages: Perspectives And Open Problems, ed. Book R.,Academic Press (1980).

Jeanrond1980.
H. J. Jeanrond, "Deciding Unique Termination of Permutative Rewriting Systems: Choose Your Term Algebra Carefully," Proceedings 5th international Conference on Automated Deduction, Springer Verlag, (1980).

Jouannaud1983.
J.P. Jouannaud, "Confluent and coherent equational term rewriting systems. application to proofs in abstract data types," Proceeding of the 8th colloquium on trees an algebra and programming, (1983).

Jouannaud1984.
J.P. Jouannaud and H. Kirchner, "Completion of a set of rules modulo a set of equations," Proceedings 11th ACM Conference of Principles of Programming Languages, (1984).

Jouannaud1984a.
J.P. Jouannaud and M. Munoz, "Termination of a set of rules modulo a set of equations," Proceedings 7th Conference on Automated Deduction Vol. 170(1984).

Kirchner1984.
C. Kirchner, "A new equational unification method: A generalisation of Martelli-Montanari's Algorithm," Proceedings 7th international Conference on Automated Deduction, pp. 224-247 (1984).

Kirchner1984a.
C. Kirchner, "Standardization of equational unification: Abstract and examples," CRIN Internal report 84-R-074 (1984).

Knuth1970.
D. Knuth and P. Bendix, "Simple Word Problems in Universal Algebras," Computational Problems in Abstract Algebra Ed. Leech J., Pergamon Press, pp. 263-297 (1970).

Lankford1977.
D.S. Lankford and A.M. Ballantyne, "Decision Procedures for Simple Equational Theories With Permutative Axioms: Complete Sets of Permutative Reductions," Report Atp-37, Dept. of Mathematics And Computer Science U.Texas, Austin (April 1977).

Lescanne1983.
P. Lescanne, "Computer Experiments with the REVE Term Rewriting System Generator," pp. 99-108 in 10th ACM Conf. on Principles of Programming

4. Généralisation à des unions de réécritures

Toute notion de réécriture sur les termes est basée sur une notion de filtrage adaptée. Habituellement un filtre est défini comme une substitution appliquant un terme sur un autre. Un point de vue différent, élaboré en collaboration avec Claude Kirchner, est adopté ici: on définit le filtrage comme une application qui à deux termes fait correspondre un ensemble, éventuellement vide, de substitutions. L'intérêt de ce point de vue est de pouvoir ensuite faire varier la méthode de filtrage en fonction des termes considérés. Cela permet de traiter de façon uniforme toute réécriture équationnelle définie comme une union de réécritures utilisant des méthodes de filtrage différentes. Une première approche de ce type de réécriture est étudiée dans la première partie de ce chapitre où l'on mélange la réécriture standard par des règles linéaires à gauche et la réécriture modulo E. Une présentation analogue de l'unification en tant qu'application qui à deux termes fait correspondre un ensemble de substitutions, permet de décider de la propriété de Church-Rosser pour ce type très général de réécriture équationnelle.

4.1. Filtrage dans une théorie équationnelle

On désigne par Subst l'ensemble des substitutions de $T(F,X)$ et par $P(Subst)$ l'ensemble des parties de Subst.

Définition 36 : Soit E un ensemble d'axiomes. Une opération de filtrage dans la théorie équationnelle $\sim^E$ est une application $Filtre_E$ de $T(F,X)\times T(F,X)$ à valeur dans $P(Subst)$ telle que

* pour toute substitution σ appartenant à $Filtre_E(t,t')$, $\sigma(t) \sim^E t'$
* $Filtre_\emptyset(t,t')$ est inclus dans $Filtre_E$.

Toute substitution σ appartenant à $Filtre_E(t,t')$ est un **filtre modulo E** ou

E-filtre de t vers t'. O

Exemple 7 :

1) On retrouve la notion habituelle de filtrage lorsque $Filtre_{\emptyset}$ est l'application qui à deux termes t et t' associe soit le filtre de t vers t' dans la théorie vide, soit l'ensemble vide lorsque t ne filtre pas vers t'. On notera $F_{\emptyset}$ cette application.

2) De même la notion habituelle de E-filtrage est obtenue en prenant pour $Filtre_E(t,t')$ l'application qui à deux termes t et t' associe l'ensemble des filtres de t vers t' dans la théorie E. On notera F_E cette application.

3) De façon plus générale, on peut par exemple définir $Filtre_E(t,t')$ comme étant $F_{\emptyset}(t,t')$ si t est linéaire et $F_E(t,t')$ si ce n'est pas le cas.

4) A la suite de Pedersen (loc.cit.), on peut prendre pour valeur de $Filtre_E(t,t')$ l'ensemble des substitutions σ telle que $\sigma(t) \sim^E t'$, les axiomes de E n'étant appliqués, dans la E-égalité précédente, qu'en dessous d'une occurrence de variable de t. Avec les notations de Pedersen, t appartient à $\{\sigma_E(t)\}$, où $\{\sigma_E(t)\}$ désigne l'ensemble des termes obtenus à partir de $\sigma(t)$ en effectuant des étapes de E-égalités en dessous des variables de t.

5) On peut aussi définir dans ce formalisme la notion de filtrage contraint, dans lequel on exige que, pour toute variable x du domaine de σ, $\sigma(x)$ appartienne à un ensemble de termes donné.

6) On peut également faire varier les axiomes avec lesquels on filtre suivant un critère sur les termes, ce qui généralise le troisième exemple.

∇

Ce formalisme permet donc d'inclure dans l'opération de filtrage toutes les conditions restrictives que l'on souhaite imposer à la notion habituelle de filtrage dans une théorie équationnelle. L'application $Filtre_E$ peut être vue comme un algorithme qui prend deux termes en entrée et retourne un ensemble de filtres. Cet algorithme peut retourner son résultat en tenant compte de certaines propriétés du terme qui filtre vers l'autre: dans la pratique, pour discriminer le résultat, on utilisera

- soit un critère de linéarité du terme t,
- soit un critère d'appartenance de t à un sous-ensemble de termes.

Cette notion de filtrage permet de définir un préordre sur les termes.

Définition 37 : On appelle **préordre de filtrage** associé à $Filtre_E$, et on note $\leq$, le préordre défini sur $T(F,X)\times T(F,X)$ par

$t \leq t'$ si et seulement si $Filtre_E(t,t')$ est non vide.

O

Notation : $t \leq t'$ si et seulement si il existe une substitution σ appartenant à $Filtre_E(t,t')$. Lorsqu'il sera utile de préciser la substitution σ utilisée pour comparer t et t', on notera $t \leq^{\sigma} t'$.

Ce préordre est compris entre le préordre de filtrage dans la théorie vide noté $\leq_{\emptyset}$ et défini par

$t \leq_{\emptyset} t'$ si et seulement si il existe σ telle que $\sigma(t) = t'$

et le préordre de E-filtrage défini par

$t \leq_E t'$ si et seulement si il existe σ telle que $\sigma(t) \sim^E t'$.

Ce préordre s'étend aux substitutions de façon classique:

Définition 38 : Soit V un ensemble de variables inclus dans X. On définit sur SubstxSubst le préordre $\leq [V]$ par

$$\alpha \leq \beta \ [V]$$

si et seulement si il existe une substitution γ telle que pour tout terme t tel que V(t) soit inclus dans V, γ appartient à $\text{Filtre}_E(\alpha(t),\beta(t))$. ○

Définition 39 : Le préordre $\leq$ est **conservé sur les sous-termes** si et seulement si $t \leq^{\sigma} t'$ implique:

pour toute occurrence u appartenant à dom(t), $t_{|u} \leq^{\sigma} t'_{|u}$. ○

Exemple 8 :

- Le préordre de filtrage dans la théorie vide $\leq_{\emptyset}$ est conservé sur les sous-termes, alors que le préordre de E-filtrage $\leq_E$ ne l'est pas.
- Le préordre défini par

$t \leq t'$ si et seulement si t' appartient à $\{\sigma_E(t)\}$

est conservé sur les sous-termes. ∇

4.2. Réécriture équationnelle

Cette notion de filtrage permet d'introduire de façon abstraite la notion de réécriture. Cela nous permettra en particulier de traiter de manière unifiée les différentes **réécritures sur les termes** connues à ce jour.

Définition 40 : Un **système de réécriture équationnelle** est un couple (R,E) constitué d'un ensemble de règles R et d'un ensemble d'équations E tels que

- pour toute équation g=d de E, V(g)=V(d). Une telle équation est dite **non effaçante**
- toute règle de réécriture notée g->d vérifie V(d) inclus dans V(g).

La **RE-réécriture** notée ->RE est la réécriture associée à R et à Filtre_E et définie par :

$$t \ \text{->RE}[u,\sigma,g\text{->}d] \ t'$$

si et seulement si - g->d appartient à R

- σ appartient à $\text{Filtre}_E(g,t_{|u})$
- et $t' = t[u \leftarrow \sigma(d)]$. ○

Notation : Si on suppose que les règles de R sont numérotées, et si la règle g->d porte le numéro k alors $t \text{->RE}[\sigma,u,g\text{->}d] \ t'$ est aussi noté $t \text{->RE}[u,\sigma,k] \ t'$.

Définition 41 : On appelle **congruence engendrée** par ->RE la fermeture réflexive symétrique et transitive de cette relation, notée <-*->RE. La fermeture symétrique transitive de ->RE est notée <-+->RE. ○

On dira qu'un ensemble d'axiomes A est équivalent à RUE, lorsque l'égalité engendrée par A, notée $\sim^A$, coincide avec <-*->RE.

Définition 42 : On appelle **RE-dérivation** toute suite de RE-réécritures. On dit que t est **RE-irréductible** ou qu'il est en **RE-forme normale** ou encore qu'il est **RE-normalisé** s'il n'existe pas de terme t' tel que t se RE-réécrive en t'.

De même on dit qu'une substitution σ est **RE-normalisée** si pour tout élément x de son domaine, $\sigma(x)$ est RE-normalisé. ○

Notation : On notera $t\downarrow_{RE}$ ou simplement $t\downarrow$ une forme normale de t pour la relation ->RE.

Exemple 9 : En reprenant dans l'ordre les cas de l'exemple précédent, on retrouve

1) la réécriture habituelle notée ->R associée à $F_{\emptyset}$.

2) la réécriture de Peterson et Stickel, qui est notée ->R,E et qui est associée à F_E.

3) la réécriture ->($R_1 \cup R_{nl}$,E) introduite dans [Jouannaud1983] qui consiste à utiliser un filtre de $F_{\emptyset}$ pour réécrire par une règle linéaire à gauche, et utiliser un E-filtre de F_E pour réécrire par une règle non linéaire à gauche.

4) la réécriture de Pedersen définie par

$$t \to_{RE[u,\sigma,g\to d]} t'$$

si et seulement si $t_{|u}$ appartient à $\{\sigma_E(g)\}$ et $t' = t[u \leftarrow \sigma(d)]$.

5) la réécriture contrainte avec un ensemble de méta-règles, définie dans le chapitre 3.

6) la réécriture avec un ensemble R union disjointe d'ensembles R_i, à chaque R_i étant associé un sous-ensemble E_i d'axiomes de E et une méthode de filtrage. ∇

Certaines approches de la réécriture équationnelle, telle celle décrite par Lankford et Ballantyne (loc.cit), utilisent la réécriture notée ->R/E et définie par $\sim^E .\to_R .\sim^E$. Cette relation simule la réécriture induite par R dans les classes d'équivalence modulo $\sim^E$ et n'est pas décrite par le formalisme précédent. Ce n'est pas un hasard, puisqu'on cherche à

formaliser ici les diverses réécritures sur les termes et non sur les classes d'équivalence de termes. On a bien sûr, pour toute théorie A = (RUE), l'inclusion des relations:

$$\to_R \subseteq \to_{RE} \subseteq \to_{R,E} \subseteq \to_{R/E}$$

Définition 43 : La relation ->RE est **E-noethérienne** ou **noethérienne modulo E** si et seulement si ->R/E est noethérienne. ○

On dira aussi par abus de langage, R est E-noethérien ou noethérien modulo E.

Notation : Pour une paire de termes (t_1,t_2), on note

- $t_1 \downarrow_{R/E} t_2$, si et seulement si il existe un terme t tel que

$$t_1 \to^*_{R/E} t \text{ et } t_2 \to^*_{R/E} t.$$

- $t_1 \downarrow t_2$ si et seulement si il existe des termes t'_1 et t'_2 tels que

$$t_1 \to^*_{RE} t'_1,\ t_2 \to^*_{RE} t'_2 \text{ et } t'_1 \sim^E t'_2.$$

La notion de réécriture est étroitement liée à celle du préordre induit par le filtrage utilisé pour réécrire. En effet, si t est réductible par la règle g->d et le filtre σ à l'occurrence u, $g \leq^{\sigma} t_{|u}$. Pour généraliser les résultats obtenus dans [Jouannaud1984] et présentés dans la première partie de ce chapitre, il est nécessaire d'introduire deux propriétés de la réécriture par rapport au préordre ≤.

Définition 44 : La RE-réécriture est **compatible** avec le préordre ≤ sur le couple de termes (t,t') si et seulement si

$t \leqq^{\sigma} t'$ et t RE-réductible en t_1

implique

t' RE-réductible en t'_1 et $t'_1 \downarrow_{R/E} \sigma(t_1)$.

○

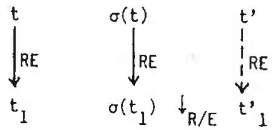


Fig 1. Compatibilité de ->RE et $\leqq$

Définition 45 : La RE-réécriture est **compatible au sommet** avec le préordre $\leqq$ sur le couple de termes (t,t') si et seulement si

$t \leqq^{\sigma} t'$ et t RE-réductible en t_1 à l'occurrence ε

implique

t' RE-réductible en t'_1 et $t'_1 \downarrow_{R/E} \sigma(t_1)$.

○

Exemple 10 :

1) ->R est compatible avec le préordre de filtrage standard $\leqq_{\emptyset}$ pour tout couple de termes.

2) ->R,E n'est pas compatible avec $\leqq_E$ en général, mais il est compatible au sommet avec $\leqq_E$ pour tout couple de termes.

3) $->(R_1 \cup R_{nl}, E)$ est compatible avec le préordre associé sur tous les couples de termes (t,t') tels que $Filtre_E(t,t') = F_{\emptyset}(t,t')$. $->(R_1 \cup R_{nl}, E)$ est compatible au sommet avec le préordre associé sur tous les couples de termes (t,t') tels que $Filtre_E(t,t') = F_E(t,t')$.

4) La réécriture de Pedersen est compatible avec le préordre associé à sa méthode de filtrage. En effet, si $t \leqq^{\mu} t'$, t' appartient à $\{\mu_E(t)\}$. Ce qui signifie que $t' = \mu'(t)$ avec $\mu'(x) \sim^E \mu(x)$ pour toute variable x de t. Si t est réductible par g, il existe une occurrence u et une substitution σ telles que $t_{|u}$ appartient à $\{\sigma_E(g)\}$. De la même façon, $t_{|u} = \sigma'(g)$ avec $\sigma'(x) \sim^E \sigma(x)$ pour toute variable x de g. On en déduit que $\mu'(t)_{|u} = \mu'(t_{|u}) = \mu'.\sigma'(g)$ et $\mu'.\sigma'(x) \sim^E \mu'.\sigma(x)$ pour toute variable x de g. Donc t' est réductible par g->d en $t'_1 = t'[u \leftarrow \mu'.\sigma(d)] = \mu'(t[u \leftarrow \sigma(d)] = \mu'(t_1)$. D'autre part $\mu'(t_1)$ appartient à $\mu_E(t_1)$ puisque $V(t_1)$ est inclus dans V(t).

5) Toutes les relations de réécriture équationnelle précédentes sont compatibles avec $\leqq_{\emptyset}$ pour tout couple de termes. ∇

La propriété de compatibilité est à rapprocher de celle de cohérence définie dans la première partie de ce chapitre. En effet si la RE-réécriture est compatible avec $\leqq$ sur le couple de termes (t,t'), la relation ->R/E vérifie la propriété de cohérence sur le couple $(\sigma(t),t')$.

Remarque : Il est facile de prouver que si $\leqq$ est conservé sur les sous-termes et vérifie :

$$t \leqq^{\sigma} t_1 \leqq^{\alpha} t' \text{ implique } t \leqq^{\alpha.\sigma} t'$$

alors ->RE est compatible avec $\leqq$ sur tout couple de termes (t,t'). C'est le cas pour ->R et la réécriture de Pedersen.

4.3. Propriété de Church-Rosser modulo E

Reprenons brièvement les définitions de confluence et cohérence données dans la première partie du chapitre, en les adaptant à la relation

->RE.

Définition 46 : Soient (R,E) un système de réécriture équationnelle, A = (R∪E) et ->RE la relation de réécriture équationnelle.

->RE est **Church-Rosser modulo E** si et seulement si pour tous termes t_1, t_2,

$$t_1 \sim^A t_2 \text{ implique } t_1 \downarrow t_2$$

->RE est **confluente modulo E** si et seulement si pour tous termes t, t_1, t_2,

$$t \text{ -*->RE } t_1 \text{ et } t \text{ -*->RE } t_2 \text{ implique } t_1 \downarrow t_2$$

->RE est **cohérente modulo E** si et seulement si pour tous termes t, t_1, t_2,

$$t \text{ -+->RE } t_1 \text{ et } t \sim^E t_2 \text{ implique}$$

il existe un terme t'_2 tel que

$$t_2 \text{ ->RE } t'_2 \text{ et } t_1 \downarrow t_2 \quad \circ$$

La relation entre ces propriétés s'énonce de la façon suivante:

Théorème 6 : (Théorème de Church-Rosser)

Si ->RE est E-noethérienne, les propriétés suivantes sont équivalentes:

(1) ->RE est Church-Rosser modulo E

(2) ->RE est confluente modulo E et cohérente modulo E

(3) pour tous termes t,t', $t \sim^A t'$ si et seulement si $t\downarrow_{RE} \sim^E t'\downarrow_{RE}$.

Preuve: Elle est donnée dans la première partie de ce chapitre pour une relation de réécriture notée R'. □

Définition 47 : Le système de réécriture équationnelle (R,E) est **convergent** si et seulement si il existe une relation de réécriture ->RE qui est E-noethérienne et possède la propriété de Church-Rosser modulo E. ○

Remarque : Si (R,E) est convergent, alors la relation ->R/E est E-noethérienne et vérifie elle aussi la propriété de Church-Rosser suivante: pour tous termes t_1, t_2, $t_1 \sim^A t_2$ implique $t_1 \downarrow_{R/E} t_2$.

Afin maintenant de pouvoir décider de la propriété de Church-Rosser modulo E à partir de l'ensemble des règles R et des équations E, il faut introduire la notion de paire critique et auparavant celle d'unification.

4.4. Unification

Définition 48 : Soit E un ensemble d'axiomes et $UNIF_E$ une application de $T(F,X)\times T(F,X)$ à valeur dans P(Subst) telle que pour tous termes t et t',

* pour toute σ dans $UNIF_E(t,t')$, $\sigma(t) \sim^E \sigma(t')$

* pour tous termes t et t' tels que V(t) et V(t') soient disjoints, $Filtre_E(t,t')$ est inclus dans $UNIF_E(t,t')$.

* pour toute substitution ρ, pour tous termes g, t et t' tels que ρ appartienne à $Filtre_E(t,t')$ et tels que D(ρ) et V(g) soient disjoints, si γ appartient à $UNIF_E(g,t')$, alors γ.ρ appartient à $UNIF_E(g,t)$. ○

Remarque : La deuxième condition assure la cohérence entre le filtre utilisé et l'unification. La troisième condition permet de s'assurer que l'ensemble des unificateurs est suffisamment stable pour que l'on puisse retrouver les propriétés classiques.

Exemple 11 : Nous suivons toujours l'ordre des exemples précédents.

1) On retrouve la notion d'unification dans la théorie vide lorsque $UNIF_\emptyset$ est l'application qui à deux termes t et t' associe l'ensemble des unificateurs de t et t' dans la théorie vide. On notera $U_\emptyset$ cette application.

2) De même la notion habituelle de E-unification est obtenue en prenant

pour $UNIF_E(t,t')$ l'application qui à deux termes t et t' associe l'ensemble des unificateurs de t et t' dans la théorie E. On notera U_E cette application.

3) De façon plus générale, on peut par exemple définir $UNIF_E(t,t')$ comme étant $U_\emptyset(t,t')$ si t est linéaire et $U_E(t,t')$ si ce n'est pas le cas.

4) On peut également restreindre $UNIF_E(t,t')$ à l'ensemble des E-unificateurs σ de t et t' tels que $\sigma(t) \sim^E \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap (V(t) \cup V(t'))$.

5) On peut aussi définir dans ce formalisme la notion d'unification contrainte, dans laquelle on exige que, pour toute variable x du domaine de σ, $\sigma(x)$ appartienne à un ensemble de termes donné.

6) On peut également faire varier les axiomes avec lesquels on unifie suivant un critère sur les termes. ∇

A cette notion générale d'unification, on associe la notion adaptée d'ensemble générateur, c'est à dire d'ensemble complet relatif à $UNIF_E$.

Définition 49 : Soit E un ensemble d'axiomes et $Unif_E$ une application de $T(F,X) \times T(F,X)$ à valeur dans P(Subst) telle que pour tous termes t et t',

(1) pour toute σ dans $Unif_E(t,t')$, $D(\sigma)$ est inclus dans $V(t) \cup V(t')$ et $I(\sigma)$ est choisi d'intersection vide avec W, où W est un ensemble de variables qui contient $V(t) \cup V(t')$.

(2) $Unif_E(t,t')$ est inclus dans $UNIF_E(t,t')$.

(3) pour toute σ' dans $UNIF_E(t,t')$, il existe σ dans $Unif_E(t,t')$ tel que

$$\sigma \leq \sigma' \ [V(t) \cup V(t')].$$

$Unif_E(t,t')$ est appelé **ensemble complet d'unificateurs** relativement à $Unif_E$ et à $Filtre_E$.

Il est souvent intéressant pour des raisons d'efficacité d'imposer la condition de minimalité suivante:

(4) pour toutes σ et ρ appartenant à $Unif_E$,

$$\sigma \leq \rho \ [V] \text{ implique } \sigma = \rho.$$

L'ensemble $Unif_E(t,t')$ est alors dit **ensemble complet minimal** d'unificateurs. $\circ$

Il est important de noter que le préordre utilisé pour comparer les substitutions de $UNIF_E(t,t')$ est le préordre induit par $Filtre_E(t,t')$. L'application $Unif_E$ dépend donc des termes t et t' uniquement et est étroitement reliée à la valeur de $Filtre_E$ en (t,t'). L'intérêt de cette définition est là encore de pouvoir faire varier éventuellement la méthode d'unification suivant un critère sur les termes. L'application $Unif_E$ peut être vue comme un algorithme qui prend deux termes t et t' en entrée et retourne un ensemble d'unificateurs complet par rapport à $UNIF_E(t,t')$ et à $Filtre_E$. Cet algorithme peut retourner son résultat en tenant compte de certaines propriétés des deux termes à unifier, comme pour le filtrage.

Notation :

1) On notera $UP_\emptyset$ l'application $Unif_E$ qui associe à tous termes t et t' soit leur unificateur principal (ou minimal) dans la théorie vide, soit l'ensemble vide lorsque t ne s'unifie pas avec t'.

2) On notera ECU_E l'application $Unif_E$ qui à deux termes t et t' associe soit un ensemble complet d'unificateurs dans la théorie E, soit l'ensemble vide lorsque t ne s'unifie pas avec t' dans la théorie E.

4.5. Paires critiques

A cette notion d'unification correspond une notion adaptée de paires

critiques.

Définition 50 : Un terme t non réduit à une variable se $Unif_E$-superpose dans un terme t' à une occurrence u de G(t') avec un ensemble complet Θ de superpositions si et seulement si $\Theta = Unif_E(t,t'|_u)$.

Etant données deux paires de termes (g,d) et (l,r) telles que V(g) et V(l) soient disjoints, et que l se superpose dans g à l'occurrence u avec l'ensemble complet de superpositions Θ, alors l'ensemble

$$\{(p,q) \mid p=\theta(d),\ q=\theta(g[u\leftarrow r]) \text{ pour tout } \theta \text{ dans } \Theta\}$$

est un **ensemble complet de paires critiques** de la paire (l,r) dans la paire (g,d) à l'occurrence u. A chaque superposition θ de Θ est associé le terme superposé θ(g). ○

Exemple 12 :

1) On retrouve la notion de paire critique définie par Knuth et Bendix en prenant $Unif_E = UP_\emptyset$ sur tous les termes.

2) On retrouve la notion de E-paire critique définie par Peterson et Stickel en prenant $Unif_E = ECU_E$ sur tous les termes.

3) Mais on peut également choisir pour $Unif_E(t,t')$

- si t est un membre gauche de règle:
  - * $ECU_E(t,t')$ si t n'est pas linéaire
  - * $UP_\emptyset(t,t')$ quand t est linéaire.
- si t est un membre gauche ou droit d'équation et t' un membre gauche de règle linéaire:
  - * $UP_\emptyset(t,t')$.

On obtient alors la notion de paire critique adaptée à la réécriture $\to(R_1 UR_{n1},E)$.

4) On peut également restreindre $Unif_E(t,t')$ à

- si t et t' sont des membres gauches de règle:

l'ensemble des E-unificateurs σ de t et t' tels que $\sigma(t) \overset{E}{\sim} \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap (V(t) \cup V(t'))$.

- si t (ou symétriquement t') est un membre gauche d'équation:

l'ensemble des E-unificateurs σ de t et t' tels que $\sigma(t) \overset{E}{\sim} \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap V(t')$ (symétriquement V(t)).

On définit ainsi une notion de paire critique adaptée à la réécriture de Pedersen. ∇

Définition 51 : Une **paire critique de confluence** est obtenue par superposition de deux règles entre elles. Une **paire critique de cohérence** est obtenue par superposition d'une règle avec une équation de E. ○

Le fait que l'on puisse ramener le problème de confluence à l'étude de la convergence de paires critiques repose sur le lemme des paires critiques qui s'énonce de la façon suivante:

Lemme 2 : Supposons que

- soit $t \to R[\varepsilon,\sigma,l\to r]\ t_1$ et $t \to RE[u,\sigma,g\to d]\ t_2$
- soit $t \mid\!-\!\mid E[\varepsilon,\sigma,l\to r]\ t_1$ et $t \to RE[u,\sigma,g\to d]\ t_2$
- soit $t \to RE[\varepsilon,\sigma,l\to r]\ t_1$ et $t \mid\!-\!\mid E[u,\sigma,g\to d]\ t_2$

où u est une occurrence de non variable dans dom(l), différente de ε dans le troisième cas. Supposons de plus que, dans la dernière éventualité, le préordre associé à →RE est conservé sur les sous-termes.

Alors il existe une paire critique (p,q) dans un ensemble complet de paires critiques de g->d dans l->r à l'occurrence u, telle que $p \leqq t_1$ et $q \leqq t_2$, où $\leqq$ est le préordre associé à la méthode de filtrage utilisée dans l'application de ->RE.

Preuve: Dans les deux premiers cas, σ appartient à $Filtre_E(g,t_{|u})$ et $t = \sigma(l)$. Donc, par les deux dernières conditions de la définition de $UNIF_E$, σ appartient à $UNIF_E(g,l_{|u})$. Il existe alors α appartenant à $Unif_E(g,l_{|u})$ tel que $\alpha \leqq \sigma$ [V] où $V = V(l) \cup V(g)$. Il existe une substitution γ telle que pour tout terme s tel que V(s) soit inclus dans V, γ appartient à $Filtre_E(\alpha(s),\sigma(s))$. $(p,q)=(\alpha(r),\alpha(l[u \leftarrow d]))$ est une paire critique de g->d dans l->r à l'occurrence u. De plus par définition de γ, γ appartient à $Filtre_E(p,t_1)$ et à $Filtre_E(q,t_2)$.

Dans le dernier cas, σ appartient à $Filtre_E(l,t)$ et σ appartient à $Filtre_E(g,t_{|u})$. Puisque le préordre est conservé sur les sous-termes, σ appartient à $Filtre_E(l_{|u},t_{|u})$. Donc σ appartient à $UNIF_E(g,l_{|u})$ et on conclut comme dans le cas précédent. □

Exemple 13 :

1) Si on choisit pour ->RE la relation ->R, (p,q) satisfait la condition suivante: il existe une substitution σ telle que $\sigma(p)=t_1$ et $\sigma(q)=t_2$.

2) Si on choisit pour ->RE la relation ->R,E, (p,q) satisfait la condition suivante: il existe une substitution σ telle que $\sigma(p) \sim^E t_1$ et $\sigma(q) \sim^E t_2$.

3) Si on choisit pour ->RE la réécriture de Pedersen, (p,q) satisfait la condition suivante: il existe une substitution σ telle que t_1 est dans $\{\sigma_E(p)\}$ et t_2 est dans $\{\sigma_E(q)\}$. ∇

4.6. Théorème de décidabilité de la propriété de Church-Rosser

Dans ce formalisme, le théorème de décidabilité de la propriété de Church-Rosser pour les systèmes de réécriture équationnelle s'énonce ainsi:

Théorème 7 : Soit R un ensemble de règles E-noethérien, et E une théorie telle que $Unif_E$ existe et que les classes d'équivalence modulo $\sim^E$ soient finies. Supposons que ->RE soit une réunion d'un nombre fini n de relations de réécriture équationnelle $->R_iE$ telles que

* pour tout i = 1,...k, $->R_iE$ est compatible sur $T(F,X)xT(F,X)$ avec le préordre associé $\leqq^i$ qui est conservé sur les sous-termes,

* pour tout i = k+1,...n, $->R_iE$ compatible au sommet avec le préordre associé $\leqq^i$ sur $T(F,X)xT(F,X)$.

Alors ->RE est Church-Rosser modulo E si et seulement si

- toutes les paires critiques de confluence (p,q) vérifient $p \downarrow_{R/E} q$.
- toutes les paires critiques de cohérence (p,q) obtenues par superposition d'une règle dans une équation satisfont

  il existe p' tel que p ->RE p' et $p' \downarrow_{R/E} q$.
- toutes les paires critiques de cohérence (p,q) obtenues par superposition d'une équation dans une règle de $\bigcup_{i=1}^{k} R_i$ satisfont

  il existe q' tel que q ->RE q' et $p \downarrow_{R/E} q'$.

Preuve: La preuve s'obtient facilement à partir de la preuve du théorème 15 de la première partie du chapitre en prenant $\bigcup_{i=1}^{k} \rightarrow R_iE$ à la place de $->R_1$ et $\bigcup_{i=k+1}^{n} \rightarrow R_iE$ à la place de $->R_{n1}$,E. Les cas 4 et 5 de la preuve se généralisent en utilisant la propriété de compatibilité de $->R_iE$ avec $\leqq^i$ pour i=1,...k. □

Exemple 14 : Il est donc possible de mélanger la réécriture standard, la réécriture de Pedersen et celle de Peterson et Stickel en définissant $\bigcup_{i=1}^{k} \rightarrow_{R_i E}$ comme la réunion de la réécriture standard avec un ensemble de règles R_1 et de la réécriture à la Pedersen avec un ensemble différent de règles R_2. En effet ces deux types de réécriture ont les mêmes propriétés de compatibilité avec leur préordre associé qui est conservé sur les sous-termes. $\bigcup_{i=k+1}^{n} \rightarrow_{R_i E}$ est réduit à la réécriture de Peterson et Stickel avec un ensemble de règles R_3. ∇

4.7. Procédure de complétion équationnelle

La procédure de complétion équationnelle a pour arguments un ensemble P de paires de termes, un ensemble R de règles de réécriture, un ensemble constant d'équations E et un ordre de E-réduction noethérien (c'est-à-dire un préordre compatible avec la structure de F-algèbre tel que l'équivalence associée contienne $\sim^E$ et tel que l'ordre strict associé soit noethérien).

La procédure de complétion est décrite de façon détaillée dans la première partie du chapitre dans le cas de la réécriture $\rightarrow_{(R_1 \cup R_{n1}, E)}$. Les notions de **règles protégées** et d'**extensions** introduites dans cette partie se généralisent en remplaçant la distinction entre R_1 et R_{n1} par une discrimination sur $\bigcup_{i=1}^{k} \rightarrow_{R_i E}$ et $\bigcup_{i=k+1}^{n} \rightarrow_{R_i E}$. La procédure PAIRES-CRITIQUES calcule les paires critiques de confluence et cohérence, et ajoute des protections ou des extensions si c'est nécessaire. Rappelons brièvement qu'une règle est marquée lorsque ses paires critiques de cohérence et de confluence avec les règles précédemment introduites ont été calculées.

Une **hypothèse d'équité** est nécessaire pour assurer qu'aucune règle ne sera indéfiniment ignorée dans le processus de sélection pour le calcul des paires critiques:

pour toute règle engendrée par la procédure, il existe un appel récursif tel que

- soit la règle est simplifiée par une nouvelle règle introduite
- soit la règle est choisie et ses paires critiques sont calculées.

Soit R+ l'ensemble de toutes les règles engendrées par la procédure. La notion de simplifiabilité, utilisée par la procédure SIMPLIFICATION pour supprimer des règles de R+, se définit maintenant de la façon suivante:

Définition 52 : Posons $\leqq = \bigcup_{i=1}^{n} \leqq^i$. $l' \rightarrow r'$ est **simplifiable** par $l \rightarrow r$ si et seulement si les deux conditions suivantes sont satisfaites:

1. - ou bien il existe une occurrence u appartenant à dom(l'), différente de ε, et $l \leqq l'|_u$
   - ou bien $l \leqq^i l'$ et $\rightarrow_{R+E}$ est compatible avec $\leqq^i$ sur le couple (l,l').

2. $l' \rightarrow r'$ n'est protégée que par des règles elle-mêmes simplifiables. ○

En d'autres termes, ou bien l' est R+E-réductible par $l \rightarrow r$ à une occurrence différente de ε, ou bien l' est R+E-réductible au sommet par $l \rightarrow r$ avec une opération de filtrage induisant un préordre compatible avec la réécriture équationnelle. Une façon de réaliser cette condition est de ne réduire les règles au sommet qu'avec la réécriture $\rightarrow_R$.

```
PROCEDURE E-COMPLETION(P,R,E,>)

 SI P n'est pas vide
 ALORS choisir une paire (p,q) dans P; p'=p↓; q'=q↓;

      CAS p' ~E q' ALORS R = E-COMPLETION(P-{(p,q)},R,E,>)
          p' > q'  ALORS l = p', r = q';
                   (P,R) = SIMPLIFICATION(P-{(p,q)},R,E,l->r)
                   R = E-COMPLETION(P,RU{l->r},E,>)
          q' > p'  ALORS l = q', r = p';
                   (P,R) = SIMPLIFICATION(P-{(p,q)},R,E,l->r)
                   R = E-COMPLETION(P,RU{l->r},E,>)
          SINON ARRET en ECHEC
      FIN CAS

 SINON SI toutes les règles de R sont marquées
      ALORS RETOURNER R; ARRET en SUCCES
      SINON choisir une règle non marquée (l->r) en respectant
            l'hypothèse d'équité
            (P,R)=PAIRES-CRITIQUES(l->r,R,E);
            marquer la règle l->r dans R
            R=E-COMPLETION(P,R,E,>)
      FIN SI
 FIN SI

FIN E-COMPLETION
```

Les preuves données dans la première partie du chapitre, dans le cas de la réécriture $\to(R_1 \cup R_{n1}, E)$ se généralisent aisément en remplaçant la distinction entre R_1 et R_{n1} par une discrimination sur $\bigcup_{i=1}^{k} \to_{R_i E}$ et $\bigcup_{i=k+1}^{n} \to_{R_i E}$.

Considérons alors le système de règles R_∞ engendré par la procédure.

Théorème 8 : Soit (R,E) un système de réécriture équationnelle tel que $Unif_E$ existe, que les classes de congruence modulo E soient finies et que le préordre de E-filtrage $\leq_E$ soit noethérien. Alors $\to R_\infty E$ est Church-Rosser modulo E.

Preuve: Elle se calque aisément sur la preuve du Théorème 23 de la première partie du chapitre. La réécriture $\to R+E$ est la réunion d'un nombre

fini n de relations de réécriture équationelle $\to R+_i E$ telles que

* pour tout i = 1,...k, $\to R+_i E$ est compatible sur $T(F,X) \times T(F,X)$ avec le préordre associé $\leq^i$ qui est conservé sur les sous-termes,

* pour tout i = k+1,...n, $\to R+_i E$ est compatible au sommet avec le préordre associé $\leq^i$ sur $T(F,X) \times T(F,X)$.

La réductibilité de R+ par R_∞ se définit de la façon suivante:

pour tous termes t, t'_1 et t_{n+1} tels que

- $t \sim^E t_{n+1}$ et $t \to R+E[v \neq \varepsilon, \sigma, l \to r]\ t'_1$ ou
- $t \sim^E t_{n+1}$ et $t \to R+E[\varepsilon, \sigma, l \to r]\ t'_1$

il existe un terme t'_2 tel que $t \to R_\infty E\ t'_2$ et t'_1 équivalent à t'_2 par l'équivalence engendrée par R+E.

Le cas 5.b de la preuve de cohérence locale utilise le fait que pour i compris entre k+1 et n, $\to R+_i E$ est compatible au sommet avec le préordre associé $\leq^i$. □

Il est important de noter que pour un même ensemble initial d'équations, des partitions différentes des ensembles de règles engendrées conduisent à des stratégies de complétion différentes et plus ou moins puissantes. Cette remarque a été développée en détail dans la deuxième partie du chapitre, dans le cas de la réécriture $\to R_1 \cup R_{n1}, E$.

Remarque : Lorsque les classes d'équivalence modulo E ne sont pas finies, il est possible de donner une condition suffisante à tester sur les paires critiques de cohérence pour assurer la propriété de Church-Rosser (voir la première partie du chapitre).

Il est intéressant de noter également le résultat d'unicité suivant:

étant donné un ordre de E-réduction et une théorie équationnelle A, il existe un unique système de réécriture équationnelle tel que

- RUE est équivalent à A
- R est E-noethérien
- R est R/E-Church-Rosser (i.e. $t \overset{A}{\sim} t'$ si et seulement si $t \downarrow_{R/E} t'$)
- les règles de R sont interréduites.

Un tel système est obtenu en normalisant par la relation ->R/E le système de réécriture retourné par la procédure de complétion équationnelle.

4.8. Travaux reliés

Un certain nombre d'autres travaux ont été menés autour de la procédure de complétion de Knuth et Bendix, c'est-à-dire dans le cas où E est vide. Notons d'abord une idée de Kapur permettant de réduire le nombre de paires à réduire et orienter: si θ est une superposition et qu'il existe une variable x de D(θ) telle que θ(x) soit réductible, la paire critique correspondant à θ est convergente. La preuve de cette propriété est facile en utilisant une induction noethérienne et s'incorpore simplement dans nos preuves.

Citons d'autre part les travaux de Winkler et Buchberger [Winkler1983], prolongés par ceux de Kuchlin [Kuchlin1985]. Dans le premier article, un critère permettant de prédire la convergence de certaines paires critiques est donné. Ce critère permet ainsi de diminuer le nombre de paires critiques à considérer et donc d'éliminer certaines réductions superflues. Il permet également de diminuer la taille de l'ensemble des paires critiques en attente, ainsi que le nombre de paires de termes à orienter. Le deuxième article améliore ces résultats en donnant un critère plus facilement implantable et montre que l'efficacité de la procédure de

complétion en est considérablement améliorée. Il serait intéressant d'étudier l'extension de leurs critères à la procédure de complétion équationnelle générale.

CHAPITRE 3:

PROBLEMES DE DIVERGENCE

ET META-REGLES

CHAPITRE 3: PROBLEMES DE DIVERGENCE ET META-REGLES

1. Introduction

L'intérêt des preuves par complétion est souvent limité en pratique par le fait que le processus de complétion engendre une infinité de paires critiques toutes orientables en règles de réécriture. De plus l'unicité du système de réécriture engendré par la procédure de complétion, un ordre étant donné, permet d'affirmer que s'il y a divergence pour cet ordre, il n'existe pas d'autre complétion permettant de n'engendrer qu'un nombre fini de règles. Il peut se produire deux types de divergence, en profondeur et en largeur.

- La divergence en profondeur est par exemple provoquée par l'existence d'une règle g->d qui se superpose successivement sur une règle g_1->d_1 puis sur la règle g_2->d_2 obtenue à partir de cette superposition et ainsi de suite. Ce phénomène apparait lorsqu'il existe un sous-terme commun à d et à g_1 qui s'unifie avec g. Si σ est l'unificateur de ce sous-terme avec g, la règle engendrée par superposition aura comme membre gauche la forme normale de $\sigma(g_1)$. Si dans celle-ci apparait à nouveau le sous-terme de d, le processus se réitère et les membres gauches des règles engendrées sont les formes normales de $\sigma(g_1)$, $\sigma(\sigma(g_1))$, $\sigma(\ldots(\sigma(g_1)\ldots)\ldots$ L'exemple typique est celui de la règle d'associativité d'un symbole f qui peut produire une divergence dès qu'il existe une autre règle contenant un sous-terme du type f(x,t).

- La divergence en largeur n'apparait que dans la complétion équationnelle et résulte de l'existence, pour certaines équations, d'un ensemble complet infini d'unificateurs dans la théorie E.

L'apparition de l'un de ces deux types de divergence soulève des problèmes à la fois d'ordre pratique et théorique. En effet, la preuve de la procédure de complétion reste valide lorsque le système de règles, engendré à partir d'un nombre fini d'axiomes d'une théorie équationnelle, possède une infinité de règles. Ce système infini permet encore, théoriquement, de semi-décider l'égalité de deux termes finis dans la théorie équationnelle initiale. Si les deux termes sont égaux, il est en effet possible de trouver une itération de la procédure, donc un système de règles fini, telle que les deux termes se réduisent en une même forme normale (ou en des formes normales E-égales s'il s'agit de réécriture équationnelle). Néanmois la procédure de complétion ne permet pas de prouver que deux termes ne sont pas égaux et, en ce sens, ne fournit alors qu'une procédure de semi-décision. De même lorsque, en superposant modulo E deux membres gauches de règles, l'algorithme d'unification modulo E retourne un ensemble complet infini de E-unificateurs, la procédure de complétion doit théoriquement assurer la convergence d'une infinité de paires critiques associées à ces deux règles en ajoutant en général une infinité de règles.

Dans la pratique, il est exclu de considérer des ensembles infinis de règles. Une première approche a été proposée par J.Y.Cras dans [Cras1983] pour traiter ce problème: une extension de REVE qui détecte un certain type de divergence de la procédure de complétion, a été implantée. Les règles engendrées par le processus de complétion sont réparties de façon dynamique en un certain nombre de suites "régulières". Deux critères de régularité sont envisagés: le premier est l'existence d'une relation de récurrence explicite, mais cela pose des problèmes de détection qui ne sont pas

résolus dans cette approche. Le deuxième est l'existence d'une relation de filiation, c'est-à-dire que les règles sont classées d'après leur origine. Plus précisément une règle appartient à une famille donnée si elle est obtenue par superposition d'une même règle dans une règle précédente de la famille. Quand le système détecte une suite de termes régulière et que celle-ci atteint une certaine longueur, en pratique très faible, il considère celle-ci comme infinie et cherche à définir des lemmes qui la réduiront. Dans la plupart des cas, le lemme doit être fourni par l'utilisateur et bien sûr prouvé par lui.

Le problème de la détection des suites régulières rejoint celui de l'inférence d'une relation de récurrence sur une famille de termes, utilisé dans la synthèse de programme à partir d'exemples [Jouannaud1981]. L'outil des arbres signés est également proposé dans [Kirchner1982] pour aider à la découverte de relations de récurrence. Cependant ce problème ne sera pas abordé dans le cadre de cette thèse. Nous avons plutôt cherché dans ce chapitre à débroussailler un problème difficile et les résultats que nous donnons ne prétendent pas constituer un traitement exhaustif du problème. Nous nous sommes fixés les trois objectifs suivants:

1) Le premier est de proposer une notion de schématisation et à mettre en évidence les conditions qu'il faut vérifier pour que cette schématisation soit possible. Le but poursuivi dans la schématisation d'un ensemble infini de règles de réécriture est de trouver un système fini de méta-règles qui permettront de décider de l'égalité dans la théorie équationnelle correspondante. Soulignons que, contrairement à d'autres approches utilisant l'introduction de lemmes valides pour faire une preuve, telle celle de Boyer et Moore [Boyer1977], nous nous plaçons ici dans l'algèbre libre

et non dans l'algèbre initiale de la théorie équationnelle. De façon informelle, une méta-règle est une règle possédant des méta-variables, c'est-à-dire des variables qui ne peuvent prendre leurs valeurs que dans un domaine donné. Pour construire une méta-règle, il faut d'abord déterminer une famille de règles dont les membres gauches ont un squelette commun. Ce squelette est défini comme le plus grand terme G tel qu'il existe un filtre modulo une relation d'équivalence décidable, notée ~, de ce terme vers chaque terme de la même famille. Dans les cas de divergence en profondeur que nous considérons, cette équivalence est engendrée par un sous-ensemble R_0 de l'ensemble infini de règles. Dans le cas d'une divergence en largeur, cette équivalence est engendrée par les équations de E. Des méta-variables s'introduisent là où l'on applique le filtre. Les structures sont les images des méta-variables par le filtre et engendrent leur domaine. On infère alors une méta-règle G->D dont le membre gauche est le squelette de la famille infinie. Pour réduire un terme avec une méta-règle, on définit une réécriture contrainte notée -->>. La schématisation est correcte si l'ensemble des méta-règles ainsi déterminées possède la propriété que la congruence engendrée par la relation -->>> (définie comme -->> suivie de ~) coincide avec celle de la théorie équationnelle de départ. Plusieurs conditions permettant d'assurer la correction d'une schématisation sont proposées. Une fois trouvée une schématisation correcte, il est valide d'utiliser les méta-règles pour prouver un théorème équationnel, ou pour résoudre des équations, comme nous le verrons dans un chapitre ultérieur.

2) Le deuxième problème est de proposer une méthode permettant de prouver que la relation de réécriture -->>> est convergente, afin à la fois de prouver la correction et d'obtenir une procédure de décision de la théorie équationnelle de départ. Au lieu de travailler directement sur la notion de

réécriture contrainte, on élargit le problème en l'étendant à l'ensemble des méta-termes. Pour cela, une opération de méta-réduction est définie sur cet ensemble. La méta-réduction est une extension conservative de l'opération de réduction contrainte sur les termes. Il est alors possible de donner des conditions nécessaires et suffisantes pour que la méta-réduction possède la propriété de convergence dans l'ensemble des méta-termes. Celle-ci implique la convergence de la réécriture contrainte dans l'ensemble des termes.

3) Enfin le dernier problème est de faire le lien avec la complétion équationnelle. Dans certains cas de divergence en profondeur que nous caractérisons, compléter un ensemble de méta-règles se ramène à une complétion modulo l'équivalence ~.

2. Position du problème

Afin de cerner mieux le problème, il s'agit d'abord de répondre à la question: quels types de systèmes de réécriture infinis veut-on représenter par des méta-règles? Donnons quelques exemples simples mais significatifs des problèmes qui se posent.

Exemple 15 : Les arbres binaires signés:

Le système de réécriture considéré est

```
-(-(x)) -> x
-(f(x,y)) -> f(-(y),-(x))
f(-x,f(x,y)) -> y
f(f(y,x),-x) -> y
```

A partir de ces règles, la procédure de complétion diverge en engendrant deux familles infinies de règles:

$f(-x,f(x,y))\to y$,

$f(f(-x_2,-x_1),f(f(x_1,x_2),y)\to y$

$f(f(-x_2,x_1),f(f(-x_1,x_2),y)\to y$

$f(f(x_2,-x_1),f(f(x_1,-x_2),y)\to y$

$f(f(f(-x_3,-x_2),-x_1),f(f(x_1,f(x_2,x_3)),y)\to y$

....

et

$f(f(y,x),-x) \to y$

$f(f(y,f(x_1,x_2)),f(-x_2,-x_1))\to y$

$f(f(y,f(-x_1,x_2)),f(-x_2,x_1))\to y$

$f(f(y,f(x_1,-x_2)),f(x_2,-x_1))\to y$

$f(f(y,f(x_1,f(x_2,x_3))),f(f(-x_3,-x_2),-x_1))\to y$

....

∇

Exemple 16 : Associativité et idempotence: les équations

$f(f(x,y),z) = f(x,f(y,z))$
$f(x,x) = x$

orientées de gauche à droite, engendrent les deux familles de règles:

$f(x_1,f(x_2,\ldots,f(x_k,f(x_1,\ldots,f(x_k,z)\ldots.)) \to f(x_1,f(x_2,\ldots,f(x_k,z)\ldots)$

et

$f(x_1,f(x_2,\ldots,f(x_k,f(x_1,\ldots,f(x_{k-1},x_k)\ldots.)) \to f(x_1,f(x_2,\ldots,f(x_{k-1},x_k)\ldots)$

∇

Exemple 17 : Un deuxième exemple assez proche du précédent est celui des

axiomes de distributivité et associativité, lorsque la distributivité est orientée de manière à factoriser les expressions. Les axiomes

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

orientés de gauche à droite, engendrent la famille de règles:

$(x_n*(\ldots*(x_1*z)\ldots)+(y_k*(\ldots*(y_1*z)\ldots) \to ((x_n*(\ldots*x_1)\ldots)+(y_k*(\ldots*y_1)\ldots))*z$

∇

Exemple 18 : Un autre exemple dû à F.Bellegarde montre que l'on peut avoir des cas plus complexes:

les équations

$(x+y)+z = x+(y+z)$
$(x+y)*z = (x*z) + (y*z)$
$(x+a).z = (x*z) + a$

sont supposées orientées de gauche à droite et engendrent alors

$(x_1+(x_2+\ldots+(x_k+a)\ldots)).z \to ((x_1*z)+((x_2*z)+\ldots+((x_k*z) + a)\ldots)$

∇

Exemple 19 : Une seule règle réussit parfois à engendrer une famille infinie. Prenons l'exemple (dû à M.Ardis [Ardis1980] et cité par N.Dershowitz dans [Dershowitz1982]) de la règle

$f(g(f(x)))\to g(f(x))$

Elle engendre la famille de règles:

$f(g^i(f(x))) \to g^i(f(x))$

∇

Exemple 20 : Cet exemple décrit un type de représentation des entiers. Il est emprunté à J.Y.Cras et dû à M.C. Gaudel. La fonction rn associe à l'entier i sa représentation rn(i). Les fonctions pred et succ portent sur les entiers et decr sur les représentations.

Les règles

```
test(rn(x),rn(succ(x))) -> lt
rn(pred(x)) -> decr(rn(x))
```

engendrent

```
test(rn(x),rn(succ(x))) -> lt
test(decr(rn(x)),rn(succ(pred(x)))) -> lt
test(decr(decr(rn(x))),rn(succ(pred(pred(x))))) -> lt
     ....
```

∇

Exemple 21 : Montrons aussi un exemple de divergence en largeur: il est emprunté à la résolution par complétion de l'équation

```
f(x,f(x,f(a,y)))=x
```

dans la théorie des arbres signés avec un symbole binaire f et un symbole unaire h. La théorie est décrite par le système de réécriture équationnelle

```
E : -(-(x)) = x
    -(f(x,y)) = f(-(y),-(x))
    -h(x) = h(-x)

R : f(-x,f(x,y)) -> y
    f(f(y,x),-x) -> y
```

Par superposition modulo E des règles:


```
RES(f(x,f(x,f(a,y))),x) -> SOL(x,y)
et f(-x,f(x,y)) -> y
```

on obtient la famille infinie de règles:

$RES(f(a,y),f(x_1,-x_1)) \rightarrow SOL(f(x_1,-x_1),y)$

$RES(f(a,y),h(f(x_1,-x_1))) \rightarrow SOL(h(f(x_1,-x_1)),y)$

$RES(f(a,y),h(h(f(x_1,-x_1)))) \rightarrow SOL(h(h(f(x_1,-x_1))),y)$

$RES(f(a,y),h(h(h(f(x_1,-x_1))))) \rightarrow SOL(h(h(h(f(x_1,-x_1)))),y)$

...

dûe au fait que l'unification modulo E de f(x,f(x,f(a,y))) et f(-x,f(x,y)) retourne un ensemble complet infini d'unificateurs modulo E

$\{(x \leftarrow f(x_1,-x_1)),$

$(x \leftarrow h(f(x_1,-x_1))),$

$(x \leftarrow h(h(f(x_1,-x_1)))),$

$(x \leftarrow h(h(h(f(x_1,-x_1))))),\ldots\}$

solutions de l'équation x=-x. ∇

Dans toute la suite, R_0 est un système de réécriture confluent et noethérien et E un ensemble d'équations pour lequel il existe un algorithme complet de E-unification.

Pour avoir un formalisme unifié permettant de traiter à la fois les cas de divergence en profondeur et en largeur, nous introduisons une relation d'équivalence ~ que nous supposerons décidable. Dans le cas d'une divergence en profondeur de la procédure standard de Knuth-Bendix, ~ doit être interprétée comme la congruence engendrée par un système de réécriture convergent R_0, dont le rôle et la détermination seront précisés dans le

paragraphe suivant. Dans le cas de divergence en profondeur de la procédure de complétion équationnelle, ~ est la congruence engendrée par un système de réécriture équationnelle (R_0,E) supposé aussi convergent. Dans le cas d'une divergence en largeur de la procédure de complétion équationnelle, ~ doit être interprétée comme l'équivalence engendrée par E.

D'autre part, $<-*->R_\infty$ désigne dans toute la suite la congruence engendrée sur les termes par le système de réécriture (éventuellement équationnelle) R_∞ (ou (R_∞,E)).

3. Schématisation

Le but de ce paragraphe est de proposer une notion de schématisation et de caractériser les systèmes de réécriture infinis schématisables en ce sens. Intuitivement il y a deux idées dans le concept de schématisation proposé ici: la première est de remplacer un ensemble infini de règles par un nombre fini de méta-règles. Ainsi la condition pour qu'un système de réécriture soit schématisable est que l'on puisse trouver des sous-ensembles de règles dont les membres gauches présentent une certaine régularité que nous formaliserons. A chacun de ces sous-ensembles est alors associée une méta-règle. La deuxième idée, contenue dans le concept de schématisation correcte, est que l'ensemble des méta-règles ainsi obtenu doit permettre de décider de la théorie équationnelle.

Dans tout ce paragraphe, les techniques d'inférence jouent un rôle primordial dans la découverte des méta-règles. Nous avons tenté de les localiser le plus précisément possible, en sachant qu'elles sont encore peu satisfaisantes.

3.1. Suite généralisable de termes

Dans un premier temps, il s'agit de caractériser les suites infinies de termes étudiées. Intuitivement une suite de termes est généralisable s'il existe un squelette commun à tous les termes de la suite et des branches, c'est-à-dire des ensembles de termes pour lesquels il existe un processus de formation descriptible, par exemple une relation de récurrence.

Définition 53 : Une suite de termes (g_k) admet un généralisé modulo ~ si et seulement si il existe un terme g tel que pour tout g_k, il existe une substitution σ_k telle que $\sigma_k(g) \sim g_k$. ○

Cette notion étend la notion de généralisé dans l'ensemble des termes présentée par exemple dans [Huet1976].

Exemple 22 : Cas d'une divergence en profondeur:

Les équations

$$f(f(x,y),z) = f(x,f(y,z))$$
$$f(x,x) = x$$

orientées de gauche à droite, engendrent une première famille de règles:

$$f(x_1,f(x_1,z)) \to f(x_1,z)$$
$$f(x_1,f(x_2,f(x_1,f(x_2,z)))) \to f(x_1,f(x_2,z))$$
$$\dots$$
$$f(x_1,f(x_2,\dots,f(x_k,f(x_1,\dots,f(x_k,z)\dots.)\dots) \to f(x_1,f(x_2,\dots,f(x_k,z)\dots)$$

Considérons la suite (g_k) de leur membres gauches et le système de réécriture:

$$R_0 : f(f(x,y),z) \to f(x,f(y,z))$$

La suite de termes (g_k) admet un généralisé modulo R_0 puisqu'il existe un

terme $g = f(x,f(x,z))$ tel que pour tout g_k, il existe une substitution σ_k telle que $\sigma_k(g) \text{ -*->}_{R_0} g_k$.

Les σ_k sont:

$(x \leftarrow x_1)$,

$(x \leftarrow f(x_1,x_2))$,

$(x \leftarrow f(x_1,f(x_2,x_3)))$,

$(x \leftarrow f(x_1,f(x_2,f(x_3,x_4))))...$

∇

Exemple 23 : Cas d'une divergence en largeur:

Dans la théorie es arbres signés où

E : -(-(x)) = x
-(f(x,y)) = f(-(y),-(x))
-h(x) = h(-x)

R : f(-x,f(x,y)) -> y
f(f(y,x),-x) -> y
RES(f(x,f(x,f(a,y))),x) -> SOL(x,y)

considérons la suite (g_k) des membres gauches des règles suivantes:

RES(f(a,y),f(x_1,-x_1)) -> SOL(f(x_1,-x_1),y)

RES(f(a,y),h(f(x_1,-x_1))) -> SOL(h(f(x_1,-x_1)),y)

RES(f(a,y),h(h(f(x_1,-x_1)))) -> SOL(h(h(f(x_1,-x_1))),y)

RES(f(a,y),h(h(h(f(x_1,-x_1))))) -> SOL(h(h(h(f(x_1,-x_1)))),y)
...

La suite de termes (g_k) admet un généralisé modulo E puisqu'il il existe un terme g = RES(f(a,y),x) tel que pour tout g_k, il existe une substitution σ_k telle que $\sigma_k(g) \overset{E}{\sim} g_k$.

Les σ_k sont:

$(x \leftarrow f(x_1,-x_1))$,

$(x \leftarrow h(f(x_1,-x_1)))$,

$(x \leftarrow h(h(f(x_1,-x_1))))$,

$(x \leftarrow h(h(h(f(x_1,-x_1)))))$,...

∇

Considérons l'ensemble des généralisés modulo ~ d'une suite (g_k). Cet ensemble de termes est non vide puisqu'il contient les variables. Il est partiellement ordonné par le préordre de filtrage.

Définition 54 : Un **plus grand généralisé** de la suite de termes (g_k) quand il existe, est un élément maximal (pour le préordre de filtrage) non réduit à une variable, dans l'ensemble des généralisés. ○

Exemple 24 : Dans les deux exemples précédents, g est un plus grand généralisé. ∇

Lorsque ~ est simplement l'égalité syntaxique sur les termes, le plus grand généralisé existe toujours, car c'est la borne inférieure de l'ensemble des termes de la suite dans l'inf-demi-treillis bien fondé $(T(F,X)/Ren,\leqq)$, où Ren est la relation de renommage des variables et $\leqq$ l'ordre induit par le préordre de filtrage $\leqq_\emptyset$.

Le rôle du plus grand généralisé est de représenter la plus grande partie commune à tous les termes d'une même famille. Le choisir maximal permet de mieux localiser les sous-termes qui grossissent et de trouver le processus de formation de ces sous-termes. C'est ce que nous allons

formaliser maintenant avec les notions de branche et de domaine engendré par une branche.

Définition 55 : Etant donnée une suite de termes (g_k) admettant un plus grand généralisé noté g, pour toute variable x de g, on définit Bx = $\{\sigma_k(x)\}$. On appellera **branche de la suite** (g_k) tout ensemble Bx non réduit à une variable. ○

Nous allons imposer de plus que chaque branche soit un ensemble générateur de termes que l'on peut reconnaitre. Cela nécessite d'introduire quelques définitions supplémentaires.

Définition 56 : Le **domaine** engendré par la branche Bx=$\{\sigma_k(x)\}$ est l'ensemble des termes

$\{t \mid t \sim \gamma(\sigma_k(x))$ pour toute substitution $\gamma\}$. ○

Exemple 25 : Reprenons l'exemple 8. Le domaine engendré par Bx = $\{x_1$, $f(x_1,x_2)$, $f(x_1,f(x_2,x_3))$, $f(x_1,f(x_2,f(x_3,x_4)))...\}$ est l'ensemble des termes tout entier. Ce sera toujours le cas pour les exemples de divergence en profondeur dans lesquels Bx contient une variable. ∇

Exemple 26 : Considérons la suite de règles:

$$f(g^i(f(x))) \rightarrow g^i(f(x))$$

R_0 est ici l'ensemble ø et le généralisé des membres gauches est le terme f(g(x)). La branche

$$Bx = \{f(x_1),\ g(f(x_1)),\ g(g(f(x_1))),...\}$$

engendre l'ensemble des termes dont le symbole de tête est f ou g. ∇

Exemple 27 : Reprenons l'exemple 9 de divergence en largeur: le domaine engendré par Bx = $\{f(x_1,-x_1)$, $g(x_2,f(x_1,-x_1),-x_2)$, $h(g(x_2,f(x_1,-x_1),-x_2))$, $g(x_3,g(x_2,f(x_1,-x_1),-x_2),-x_3),...\}$ est l'ensemble des solutions modulo E de l'équation (x=-x). Notons que Bx est un ensemble complet de E-solutions. ∇

Remarquons qu'un même terme du domaine peut en général être obtenu par substitution à partir de plusieurs termes de la branche. Cependant il est possible d'en trouver un de taille maximale dans les exemples précédents.

Nous aurons besoin dans la suite de reconnaitre les termes du domaine. C'est pourquoi nous allons imposer qu'un domaine soit un ensemble récursif. Rappelons tout d'abord cette notion [Shoenfield1967].

Définition 57 : On dit qu'un ensemble de termes $\{t_1,t_2,...,t_n,...\}$ est **récursif** si il existe un algorithme permettant de décider si un terme appartient à cet ensemble. ○

La propriété pour un ensemble d'être récursif est indécidable en général. Pourtant il est possible de donner des conditions suffisantes assurant cette propriété. Ce sera le cas en particulier s'il existe une grammaire engendrant ce langage ou un automate le reconnaissant. Un cas particulier, important pour l'étude des divergences en largeur, est celui où DOM est l'ensemble des termes satisfaisant modulo E une équation donnée avec une seule variable x. Pour décider de l'appartenance d'un terme t à l'ensemble, il suffit dans ce cas de substituer t à la variable x dans e et de tester la E-égalité des termes obtenus.

Exemple 28 : Dans les trois exemples précédents, DOM est récursif. ∇

Nous sommes maintenant en mesure de définir une suite généralisable:

Définition 58 : Une suite de termes (g_k) est **généralisable** si et seulement si elle admet un plus grand généralisé et des branches qui engendrent des ensembles récursifs. ○

Remarquons que, pour une suite, la propriété d'être généralisable est indécidable, du fait que, pour un ensemble, la propriété d'être récursif l'est.

3.2. Méta-terme et instanciations associées à une suite généralisable

A la suite généralisable (g_k) est associé un terme contenant une nouvelle sorte de variables, avec des contraintes sur les valeurs qui peuvent leur être affectées.

Définition 59 : On appelle **ensemble de variables contraintes** un couple (MX,DOM) composé d'un ensemble MX de nouvelles variables et de leur domaine DOM qui est un ensemble récursif.
Les variables de MX sont appelées **méta-variables** et notées X,Y,Z.... ○

A des branches distinctes sont en général associés des domaines différents. Il faudra alors introduire autant d'ensembles de méta-variables que de domaines. Pour ne pas alourdir la formalisation, nous supposerons que les branches engendrent toutes le même domaine DOM et nous n'introduirons qu'un seul ensemble de méta-variables. C'est d'ailleurs le cas dans tous les exemples que nous traitons ici.

Définition 60 : On appellera **méta-terme** tout terme de la F-algèbre libre engendrée par XUMX et notée T(F,XUMX).
Si t est un méta-terme, MV(T) désigne l'ensemble de ses variables et méta-

variables.
Une **méta-règle** est un couple de méta-termes noté G->D tel que MV(G) contienne MV(D). ○

Notation : Les méta-termes seront notés par des lettres capitales T,U,G,D...

Dans tout ce qui suit, nous supposerons la congruence ~ étendue aux méta-termes de la façon suivante: ~ est la plus petite congruence sur T(F,XUMX) contenant toutes les paires $(\sigma(g),\sigma(d))$, pour toute équation g=d définissant ~, et pour toute substitution σ de T(F,XUMX).

Définition 61 :

- On appelle **squelette de la suite généralisable** (g_k) le plus grand généralisé dans lequel toute variable z telle que Bz n'est pas réduit à une variable est remplacée par une méta-variable Z.
- L'**ensemble des structures** S associé à une méta-variable Z est l'ensemble Bz. Le **domaine** d'une méta-variable Z de MX à laquelle est associé l'ensemble de structures S est l'ensemble de termes DOM engendré par S.
- On dira alors que (g_k) est **schématisable** par ce méta-terme G et le domaine DOM. ○

Précisons tout de suite un cas important où une suite de règles est schématisable: supposons qu'il existe une relation de récurrence modulo un système de réécriture confluent et noethérien entre les $(g_k->d_k)$, c'est-à-dire que pour tout k>1, g_k et d_k puissent s'exprimer en fonction de g_{k-1} et d_{k-1} respectivement, à l'aide d'un même filtre σ_{k-1}. En d'autres termes, $g_k \sim (\sigma_{k-1}(g_{k-1}))$ et $d_k \sim (\sigma_{k-1}(d_{k-1}))$.
Alors pour tout k>1, $g_k->d_k$ peut s'exprimer en fonction de $g_1->d_1$:

$g_k \rightarrow d_k \sim (a_k(g_1)) \rightarrow (a_k(d_1))$.

où $a_k = \sigma_{k-1}.\sigma_{k-2}\ldots\sigma_1$. La méta-règle est alors la règle $g_1 \rightarrow d_1$ dans laquelle les variables x appartenant au domaine de la substitution a_k sont remplacées par des méta-variables. Les branches s'écrivent sous la forme $\{a_k(x)\}$. De telles relations de récurrence peuvent être détectées automatiquement par une technique de filtrage en tête appliquée récursivement d'abord sur les termes de la suite, puis sur les filtres obtenus jusqu'à obtenir un filtre constant. Pour plus de détails sur cette technique utilisée en synthèse automatique de programmes à partir d'exemples, nous renvoyons le lecteur à [Jouannaud1981].

Définition 62 : Une **instanciation des méta-variables** est une application ϕ de l'ensemble des méta-variables MX dans leur domaine DOM. Une **instanciation principale** des méta-variables est une application ψ de l'ensemble des méta-variables MX dans l'ensemble des structures. ○

Les hypothèses faites sur les structures permettent de décomposer toute instanciation des méta-variables en une substitution de T(F,X) et une instanciation principale, modulo l'équivalence ~.

Lemme 3 : Pour tout méta-terme T et pour toute instanciation ϕ des méta-variables de T, il existe une substitution γ de T(F,X) et une instanciation principale ψ des méta-variables de T telles que $\phi \sim \gamma.\psi$ [MV(T)].

Preuve: Soit ϕ une instanciation des méta-variables de T. Pour toute méta-variable Z, il existe une structure s et une substitution γ telles que $\gamma(s) \sim \phi(Z)$. En posant $\psi(Z) = s$, on obtient $\phi \sim \gamma.\psi$ [MV(T)]. □

Il faut noter que cette décomposition nécessite de renommer les variables des structures associées quand il y a plusieurs méta-variables, ce que nous ferons implicitement.

Exemple 29 : Prenons le cas où ~ est l'égalité syntaxique, où l'ensemble des structures contient f(x) et engendre l'ensemble des termes commençant par f ou g. Soit T le méta-terme g(f(X),f(Y)), et $\phi=(X \leftarrow f(t_1))(Y \leftarrow f(t_2))$ où t_1 et t_2 sont des termes quelconques. On choisira alors: $\psi=(X \leftarrow f(x_1))(Y \leftarrow f(x_2))$ et $\gamma=(x_1 \leftarrow t_1)(x_2 \leftarrow t_2)$. ∇

Cette décomposition se généralise à toute application de XUMX dans T(F,X):

Corollaire 1 : Pour tout méta-terme T et toute application σ de XUMX dans T(F,X), il existe une substitution δ de T(F,X) et une instanciation principale ψ des méta-variables de T telles que $\sigma \sim \delta.\psi$ [MV(T)].

Preuve: Soit σ une application de XUMX dans M(F,XUMX). σ se décompose en α définie sur les variables de T et β définie sur les méta-variables de T. D'après le lemme 1, il existe alors γ et ψ telles que $\gamma.\psi \sim \beta$ [MV(T)]. On en déduit que:

$\alpha.\gamma.\psi \sim \sigma$ [MV(T)], d'où le résultat en posant $\delta=\alpha.\gamma$. □

Exemple 30 : Dans l'exemple précédent, considérons T = g(g(X,Y),x) et

$\sigma = (X \leftarrow f(t_1))(Y \leftarrow f(t_2))(x \leftarrow t_3)$.

On choisit alors:

$\psi=(X \leftarrow f(x_1))(Y \leftarrow f(x_2))$ et $\delta=(x_1 \leftarrow t_1)(x_2 \leftarrow t_2)(x \leftarrow t_3)$. ∇

Définissons maintenant un système de règles schématisable par un ensemble de méta-règles.

Définition 63 : Un système de réécriture infini R_∞ est **schématisable par un ensemble de méta-règles** MR et un ensemble de structures S si et seulement si pour toute méta-règle G->D de MR et toute instanciation principale ψ de ses méta-variables, $\psi(G)$ est ~-équivalent au membre gauche d'une règle de R_∞. ○

Remarquons qu'une règle (sans méta-variable) est aussi une méta-règle. Dans la suite, quand nous parlerons de règles, nous ferons référence au système de réécriture R_∞ obtenu par complétion, et nous parlerons de méta-règles pour le système MR obtenu par schématisation, qui peut bien sûr contenir des méta-règles sans méta-variables.

Dire que le membre gauche d'une méta-règle G est après instanciation principale ~-équivalent au membre gauche d'une règle de R_∞, traduit le fait que G est le plus grand généralisé modulo ~ d'une suite (g_k); tous les g_k sont alors réductibles par -->> au sommet par la méta-règle de membre gauche G. Si l'on n'impose pas cette restriction, on pourrait obtenir la situation suivante:

Exemple 31 : Les règles

```
f(x+e) -> f(x)
(x+y)+z -> x+(y+z)
```

engendrent

```
f(x+e) -> f(x)
f(x+(y+e)) -> f(x+y)
f(x+(y+(z+e)) -> f(x+(y+z))
     ...
```


Dans la suite des membres gauches de ces règles, les sous-termes à l'occurrence 1 forment une suite

x+e, x+(y+e), x+(y+(z+e)) ...

généralisable par le méta-terme X+e et le domaine DOM engendré par

S = {x, x+y, x+(y+z) ...}.

De même, dans la suite des membres droits, les sous-termes à l'occurrence 1 se généralisent par X. Cependant l'introduction de la règle (X+e) -> X modifierait complètement la théorie équationnelle. Par contre la définition que nous donnons conduit à la méta-règle f(X+e)->f(X). ▽

Pour schématiser un ensemble infini de règles, la méthode proposée ici est la suivante: à partir de l'ensemble R_i des règles à une itération i de la procédure de complétion, on détermine une congruence ~ incluse dans $<-*->R_\infty$. On infère des méta-règles dont les membres gauches généralisent, modulo ~, les premières règles engendrées, ainsi que le domaine DOM et son ensemble de structures S. On vérifie alors l'hypothèse que toute la suite de termes que généralise le membre gauche de la méta-règle est bien engendrée par complétion. Ceci peut se faire dans tous les exemples que nous considérons, par un raisonnement par récurrence du type suivant:

- $\psi_0(G)$ est ~-équivalent au membre gauche g_0 d'une règle de R_∞
- si $\psi_i(G)$ est ~-équivalent au membre gauche g_i d'une règle de R_∞, alors $\psi_{i+1}(G)$ est ~-équivalent au membre gauche g_{i+1} d'une règle de R_∞, en prouvant que g_i se superpose avec une autre règle pour donner une nouvelle règle dont le membre gauche est ~-équivalent à $\psi_{i+1}(G)$.

Nous avons vu apparaitre dans l'exemple 31, la nécessité d'assurer la correction de la schématisation. Schématiser correctement un système de

réécriture R_∞ consiste à trouver un nombre fini de méta-règles permettant de décider de $<-*->R_\infty$. Pour cela il faut auparavant préciser ce que l'on entend par réécrire avec un tel système.

3.3. Définition de la réécriture contrainte sur les termes

En général, la notion habituelle de réduction, même modulo l'équivalence ~, ne convient pas.

Exemple 32 : Reprenons l'exemple de la règle

$$f(g(f(x)))->g(f(x)).$$

Elle engendre la suite de règles:

$$f(g^i(f(x))) -> g^i(f(x))$$

qui admet pour squelette le méta-terme $f(g(X)) -> g(X)$ et il parait judicieux de choisir cette méta-règle pour représenter la famille de règles considérée. La branche

$$Bx = \{f(x_1),\ g(f(x_1)),\ g(g(f(x_1))),\ldots\}$$

engendre l'ensemble des termes dont le symbole de tête est f ou g. Les termes $f(g(a))$ et $g(a)$ satisfont $f(g(a)) -> g(a)$. Pourtant $f(g(a))$ et $g(a)$ ne sont pas équivalents dans la théorie initiale et ne sont donc jamais R_∞-équivalents. ∇

La raison qui empêche d'avoir la réciproque apparait clairement sur cet exemple: on a "oublié" la contrainte sur les méta-variables en filtrant $f(g(X))$ vers $f(g(a))$. En effet la valeur $X->a$ n'est pas une valeur autorisée pour la méta-variable X car elle n'appartient pas à son domaine. On va donc définir une autre relation de réécriture, que nous appelons

réécriture contrainte, qui tient compte du domaine des méta-variables.

Notation : Soit MR un ensemble de méta-règles notées G->D.

Définition 64 : Soit Filtre-contraint l'application de $T(F,X\cup MX)\times T(F,X\cup MX)$ dans l'ensemble des parties de l'ensemble des substitutions de $T(F,X\cup MX)$ qui a un couple (T,T') associe

- l'ensemble des substitutions $\sigma^\sim$ satisfaisant:
  - -- $\sigma^\sim$ est un filtre modulo ~ de T vers T'
  - -- pour toute méta-variable Z de MV(T), $\sigma^\sim(Z)$ appartient à DOM,

  si T est un méta-terme et T' un terme,
- l'ensemble vide sinon. ○

Remarque : L'application Filtre-contraint est une opération de filtrage dans la théorie équationnelle ~, dans le sens où nous l'avons définie au chapitre 2, lorsqu'on suppose que la congruence ~ est étendue à $T(F,X\cup MX)$. Il faut noter qu'en fait Filtre-contraint n'a besoin d'être définie que sur $T(F,X\cup MX)\times T(F,X)$. Sa valeur sur $T(F,X\cup MX)\times T(F,X\cup MX)$ tout entier ne nous intéresse pas dans la mesure où l'on ne réduit que des termes.

Reprenons les trois exemples précédents:

Exemple 33 : Associativité et idempotence:

A l'ensemble infini de règles engendré par les équations

$$f(f(x,y),z) = f(x,f(y,z))$$
$$f(x,x) = x$$

on associe

$$R_0 : f(f(x,y),z) -> f(x,f(y,z)).$$

L'ensemble des structures

$S = \{x_1,\ f(x_1,x_2),\ f(x_1,f(x_2,x_3)),\ f(x_1,f(x_2,f(x_3,x_4)))...\}$ engendre l'ensemble des termes tout entier. Filtre-contraint est dans ce cas l'application qui au couple (T,T'), tel que T' soit dans T(F,X), associe l'ensemble des filtres modulo R_0 de T vers T'. ∇

Cet exemple se généralise car dans la pratique, il est fréquent que l'ensemble des structures contienne une variable.

Définition 65 : On dira qu'une méta-règle est **sans contraintes** si et seulement si l'ensemble des structures associé à l'ensemble des méta-variables contient une variable.
On dira qu'un ensemble de méta-règles est sans contraintes si et seulement si chacune de ses méta-règles est sans contraintes. ○

Dans ce cas, nous avons déjà remarqué que DOM est alors l'ensemble de tous les termes. De plus, les méta-règles sont, à un renommage près des méta-variables par des variables, des règles de R_∞.

Corollaire 2 : Si MR est un ensemble de méta-règles sans contraintes, Filtre-contraint(T,T'), lorsque T' appartient à T(F,X), est l'ensemble des filtres modulo ~ de T vers T'.

Exemple 34 : Considérons la suite de règles:

$$f(g^i(f(x))) \rightarrow g^i(f(x))$$

R_0 est ici l'ensemble vide. L'ensemble des structures

$$S = \{f(x_1),\ g(f(x_1)),\ g(g(f(x_1))),...\}$$

engendre l'ensemble des termes dont le symbole de tête est f ou g.

Filtre-contraint est dans ce cas l'application qui au couple (T,T'), tel que T' soit un terme de T(F,X), associe le filtre σ^- de T vers T' si pour tout Z, $\sigma^-(Z)$ commence par le symbole f, ou bien l'ensemble vide sinon. ∇

Exemple 35 : Cas d'une divergence en largeur:

Dans la théorie des arbres signés où

E : -(-(x)) = x
-(f(x,y)) = f(-(y),-(x))
-h(x) = h(-x)

R : f(-x,f(x,y)) -> y
f(f(y,x),-x) -> y
RES(f(x,f(x,f(a,y))),x) -> SOL(x,y)

pour la suite de règles

$RES(f(a,y),f(x_1,-x_1)) \rightarrow SOL(f(x_1,-x_1),y)$

$RES(f(a,y),h(f(x_1,-x_1))) \rightarrow SOL(h(f(x_1,-x_1)),y)$

$RES(f(a,y),h(h(f(x_1,-x_1)))) \rightarrow SOL(h(h(f(x_1,-x_1))),y)$ ...

on choisit l'ensemble de structures

$$S = \{f(x_1,-x_1),\ g(x_2,f(x_1,-x_1),-x_2),\ h(g(x_2,f(x_1,-x_1),-x_2)),$$
$$g(x_3,g(x_2,f(x_1,-x_1),-x_2),-x_3),...\}$$

qui engendre l'ensemble des solutions modulo E de l'équation (x=-x). Filtre-contraint est dans ce cas l'application qui au couple (T,T'), tel que T' soit dans T(F,X), associe un filtre modulo E σ^-, de T vers T' si pour tout Z, $\sigma^-(Z)$ est solution de (x=-x), ou bien l'ensemble vide s'il n'existe aucun filtre modulo E vérifiant cette condition. ∇

Ce dernier exemple révèle une difficulté non résolue du filtrage contraint. En effet il ne suffit pas de disposer d'un algorithme de filtrage modulo ~ calculant un filtre modulo ~ (ou même un ensemble complet de filtres modulo ~), pour savoir faire du filtrage contraint. Il se peut en effet que le ou les filtres retournés ne satisfassent pas les conditions du domaine, alors qu'il en existe d'autres les satisfaisant.

A la notion de filtrage contraint et à l'ensemble de méta-règles MR correspond la notion de réécriture contrainte.

Définition 66 : La relation de **réécriture contrainte** -->> avec l'ensemble de méta-règles MR est définie **sur les termes** par:

$$t \dashrightarrow\!\!> [u,\sigma^{\sim},G\to D]\ t'$$

si et seulement si

- G->D appartient à MR
- $\sigma^{\sim}$ appartient à Filtre-contraint$(G,t_{|u})$
- $t' = t[u \leftarrow \sigma^{\sim}(D)]$. ○

Définissons également la relation -->>> par:

t -->>> t' si et seulement si t -->> t'_1 et $t' \sim t'_1$.

En d'autres termes -->>> = -->>.~

Nous sommes maintenant en mesure de définir la notion de schématisation correcte.

3.4. Correction de la schématisation

Une schématisation est correcte si l'ensemble des méta-règles obtenues

permet de décider de l'égalité dans la théorie A initiale.

Définition 67 : La schématisation de R_∞ par MR est **correcte** si et seulement si <-*->R_∞ coïncide avec <<<-*->>>. ○

Remarque : Puisque la preuve de la procédure de complétion permet d'affirmer que $\sim^A$ coincide avec <-*->R_∞, il est équivalent de dire $\sim^A$ coincide avec <<<-*->>> dans la définition précédente. Notons d'autre part que <<<-*->>> sera facilement décidable si -->>> est convergente, c'est-à-dire si -->>> est noethérienne et possède la propriété de Church-Rosser:

t <<<-*->>> t' si et seulement si il existe t'' tel que

t -*->>> t'' et t' -*->>> t''.

Notre but est maintenant de caractériser une schématisation correcte.

Proposition 1 : La schématisation de R_∞ est correcte si et seulement si

(a) pour toute équation g=d de A, g <<<-*->>> d.

(b) pour toute méta-règle G->D et toute instanciation principale ψ, $\psi(G)$ <-*->R_∞ $\psi(D)$

(c) ~ est incluse dans <-*->R_∞.

Preuve: ● Montrons d'abord que les deux propositions suivantes sont équivalentes:

(1) Si t <-*->R_∞ t', alors t <<<-*->>> t'.

(a) pour toute équation g=d de A, g <<<-*->>> d.

(1) => (a) est évident.

(a) => (1) : On a dans ce cas $t \sim^A t'$ et donc t <<<-*->>> t'.

● Montrons ensuite que

(2) Si t <<<-*->>> t' alors t <-*->R_∞ t'

est équivalente à la conjonction de:

(b) pour toute méta-règle G->D et toute instanciation principale ψ, $\psi(G)$ <-*->R_∞ $\psi(D)$

(c) ~ est incluse dans <-*->R_∞.

(2) => (b) et (c) est évident.

(b) et (c) => (2): il suffit de vérifier que t -->>> t' implique t <-*->R_∞ t'. Si t n'est pas équivalent modulo ~ à t', t est réductible par une règle de MR avec -->>.

Posons t -->>[u,σ^-,G->D] t_1 ~ t'. On a $\sigma^-(G)$ ~ $t_{|u}$. D'après le corollaire 1, il existe alors δ et ψ telles que $\delta.\psi \sim \sigma^-$ [MV(G)]. On en déduit que:

$\delta.\psi(G) \sim \sigma^-(G) \sim t_{|u}$ et $\delta.\psi(D) \sim \sigma^-(D) \sim t_{1|u}$.

Or $\psi(G)$ <-*->R_∞ $\psi(D)$ implique $\delta(\psi(G))$ <-*->R_∞ $\delta(\psi(D))$. Par conséquent, $t_{|u}$ <-*->R_∞ $t_{1|u}$ et t <-*->R_∞ t_1 <-*->R_∞ t'. □

Vérifier la correction d'une schématisation est un problème difficile dans la mesure où il faut tester la propriété (b) sur toutes les instanciations principales. Nous proposons dans le paragraphe suivant des conditions suffisantes permettant de prouver la correction.

4. Conditions suffisantes de correction

Examinons tout d'abord les conditions (a), (b) et (c) équivalentes à la correction.

La condition (a) est facile à tester si -->>> est convergente. Nous reviendrons sur ce problème par la suite.

La condition (c) est automatiquement satisfaite dans le cas d'une divergence en largeur, car dans ce cas <-*->R_∞ est l'équivalence engendrée par (R_∞ ∪ E) qui contient bien l'équivalence engendrée par E. Dans le cas d'une divergence en profondeur, la condition (c) est remplie de façon évidente si R_0 est inclus dans R_∞.

Des conditions suffisantes pour assurer la condition (b) peuvent être données:

- si l'ensemble des structures contient une variable, (b) équivaut à:

(b_1) pour toute méta-règle G->D, G <-*->R_∞ D.

- si toute instanciation principale ψ s'écrit ψ_0^i, où ψ_0 est une instanciation principale, (b) est équivalent à:

(b_2) pour toute méta-règle G->D, $\psi_0(G)$ <-*->R_∞ $\psi_0(D)$.

Cela résulte de la stabilité de la congruence <-*->R_∞ par substitutions.

- de manière plus générale encore, si toute instanciation principale ψ s'écrit $F^i(\psi_0)$, où F est un contexte et ψ_0 une instanciation principale, (b) est équivalent à:

(b_3) pour toute méta-règle G->D, $\psi_0(G)$ <-*->R_∞ $\psi_0(D)$.

Cela résulte de la stabilité de la congruence <-*->R_∞ par contexte.

Remarquons qu'en tout état de cause, cette condition (b) ne fait intervenir que des instanciations principales et non quelconques: cela permet de tenir compte de la forme des structures pour la prouver.

Voyons un premier cas simple où l'on peut prouver la correction. La première idée qui vient à l'esprit est de considérer -> comme un symbole de fonction binaire et de supposer que la suite infinie de règles engendrées (g_k->d_k) est une suite généralisable de termes de symbole de tête ->. La méta-règle associée à une telle suite est son squelette.

Proposition 2 : Supposons que ~ est incluse dans <-*->R_∞ et que la suite de

règles engendrée par la procédure de complétion est schématisable par le méta-terme G->D et le domaine DOM. Alors la schématisation de R_∞ par {G->D} est correcte.

Preuve: - Pour toute règle g_k->d_k de R_∞, il existe ψ telle que $\psi(G) \sim g_k$ et $\psi(D) \sim d_k$. Donc <-*->R_∞ (qui coincide avec $\sim^A$) est incluse dans <<<-*->>> et la condition (a) de la proposition 1 est satisfaite.

- Pour toute instanciation ψ, $\psi(G) \sim g_k$ et $\psi(D) \sim d_k$ où g_k->d_k est une règle de R_∞. La condition (b) est bien satisfaite.

- La condition (c) est remplie par hypothèse. □

Ce formalisme permet de traiter les exemples simples que nous avons déjà considérés. Examinons quelles méta-règles leur correspondent.

Exemple 36 : les arbres binaires signés

A partir des équations

```
-(-(x)) = x
-(f(x,y)) = f(-(y),-(x))
f(-x,f(x,y)) = y
f(f(y,x),-x) = y
```

orientées de gauche à droite, la procédure de complétion diverge en engendrant deux familles infinies de règles:

$f(-x,f(x,y))\to y$,

$f(f(-x_2,-x_1),f(f(x_1,x_2),y)\to y$

$f(f(-x_2,x_1),f(f(-x_1,x_2),y)\to y$

$f(f(x_2,-x_1),f(f(x_1,-x_2),y)\to y$

$f(f(f(-x_3,-x_2),-x_1),f(f(x_1,f(x_2,x_3)),y)\to y$

....

La congruence ~ est celle engendrée par le système de réécriture

```
R0 : -(-(x)) -> x
     -(f(x,y)) -> f(-(y),-(x))
```

Les méta-règles sont:

```
f(-X,f(X,y)) -> y
f(f(y,X),-X) -> y
```

La famille des structures est l'ensemble des arbres en R_C-forme normale dont toutes les feuilles sont des variables distinctes, par exemple:

```
x, -x, f(x,y), f(-x,y), f(x,-y), f(-x,-y), f(x,f(y,z)), f(-x,f(y,z))...
f(f(x,y),f(x',y'))...
```

∇

Exemple 37 : Associativité et distributivité

Les axiomes

```
((x*y)*z) = (x*(y*z))
(x*z)+(y*z) = (x+y)*z
```

orientés de gauche à droite, engendrent la famille de règles:

$(x_n*(...*(x_1*z)...)+(y_k*(...*(y_1*z)...) \to ((x_n*(...*x_1)...)+(y_k*(...*y_1)...))*z$

L'ensemble des structures est l'ensemble des peignes construits avec * et à variables toutes distinctes:

x_1, (x_1*x_2), $(x_1*(x_2*x_3))$, $(x_1*(x_2*(x_3*x_4)))$...

et engendre l'ensemble des termes. L'équivalence ~ est l'égalité syntaxique. La méta-règle est alors:

$(Y*z)+(X*z) \to (Y+X)*z$

∇

Exemple 38 : Les équations

$$(x+y)+z = x+(y+z)$$
$$(x+y)*z = (x*z) + (y*z)$$
$$(x+a).z = (x*z) + a$$

orientées de gauche à droite, engendrent les règles:

$$(x_1+(x_2+\ldots+(x_k+a)\ldots)).z \rightarrow (x_1*z+(x_2*z+\ldots+(x_k*z + a))).$$

L'ensemble des structures est

$$x_1,\ x_1+x_2,\ x_1+(x_2+x_3)\ \ldots$$

Si l'on choisit pour R_0 le système de réécriture:

$$(x+y)+z \rightarrow x+(y+z)$$
$$(x+y)*h \rightarrow (x+h)*(y+h)$$

La suite de règles admet pour généralisé la méta-règle:

$$(X+a).h \rightarrow X*h + a$$

∇

Exemple 39 : Considérons la suite de règles:

$$f(g^i(f(x))) \rightarrow g^i(f(x))$$

La congruence ~ est l'égalité syntaxique. L'ensemble des structures

$$S = \{f(x_1),\ g(f(x_1)),\ g(g(f(x_1))),\ldots\}$$

engendre l'ensemble des termes dont le symbole de tête est f ou g.

On a alors une méta-règle :

$$f(g(X)) \rightarrow g(X).$$

∇

Exemple 40 : type abstrait

Les règles

```
test(rn(x),rn(succ(x))) -> lt
rn(pred(x)) -> decr(rn(x))
```

engendrent

```
test(rn(x),rn(succ(x))) -> lt
test(decr(rn(x)),rn(succ(pred(x)))) -> lt
test(decr(decr(rn(x))),rn(succ(pred(pred(x))))) -> lt
    ....
```

L'ensemble des structures

$$S = \{x,\ pred(x),\ pred(pred(x)),\ \ldots\}$$

engendre tous les termes de T(F,X). La congruence ~ est celle engendrée par

$$R_0 : rn(pred(x)) \rightarrow decr(rn(x))$$

La méta-règle est alors

```
test(rn(X),rn(succ(X))) -> lt.
```

∇

Exemple 41 : divergence en largeur

Dans la théorie es arbres signés où

E : $-(-(x)) = x$
$-(f(x,y)) = f(-(y),-(x))$
$-h(x) = h(-x)$

R : $f(-x,f(x,y)) \to y$
$f(f(y,x),-x) \to y$
$RES(f(x,f(x,f(a,y))),x) \to SOL(x,y)$

la suite de règles

$RES(f(a,y),f(x_1,-x_1)) \to SOL(f(x_1,-x_1),y)$

$RES(f(a,y),h(f(x_1,-x_1))) \to SOL(h(f(x_1,-x_1)),y)$

$RES(f(a,y),h(h(f(x_1,-x_1)))) \to SOL(h(h(f(x_1,-x_1))),y)$

$RES(f(a,y),h(h(h(f(x_1,-x_1))))) \to SOL(h(h(h(f(x_1,-x_1)))),y)$
...

L'ensemble des structures est l'ensemble des solutions minimales de l'équation x=-x modulo la théorie E des arbres signés.

La méta-règle est alors

$RES(f(a,y),X) \to SOL(X,y)$

∇

La proposition 2 se généralise au cas où R_∞ est réunion d'un nombre fini de suites de règles schématisables. Illustrons ce cas par l'exemple des axiomes d'associativité et idempotence.

Exemple 42 : Associativité et idempotence

Les équations

$f(f(x,y),z) = f(x,f(y,z))$
$f(x,x) = x$

orientées de gauche à droite, engendrent deux familles de règles:

$f(x_1,f(x_1,z)) \to f(x_1,z)$

$f(x_1,f(x_2,f(x_1,f(x_2,z)))) \to f(x_1,f(x_2,z))$
...
$f(x_1,f(x_2,\ldots,f(x_k,f(x_1,\ldots,f(x_k,z)\ldots.)$
$\to f(x_1,f(x_2,\ldots,f(x_k,z)\ldots)$

$f(x_1,x_1) \to x_1$

$f(x_1,f(x_2,f(x_1,x_2))) \to f(x_1,x_2)$
...
$f(x_1,f(x_2,\ldots,f(x_k,f(x_1,\ldots,f(x_{k-1},x_k)\ldots.)$
$\to f(x_1,f(x_2,\ldots,f(x_{k-1},x_k)\ldots)$

Les deux familles sont schématisées respectivement par les méta-règles

$f(X,f(X,z)) \to f(X,z)$
$f(X,X) \to X$

en posant

$R_0 : f(f(x,y),z) \to f(x,f(y,z))$

L'ensemble des structures est l'ensemble:

$S = \{x_1, f(x_1,x_2), f(x_1,f(x_2,x_3)), f(x_1,f(x_2,f(x_3,x_4)))\ldots\}$

Le domaine engendré par S est l'ensemble des termes tout entier.

Si l'on considère uniquement la méta-règle

$f(X,X) \to X$

qui schématise la deuxième famille, on s'aperçoit que celle-ci réduit, à l'occurrence 1, par réécriture contrainte toutes les règles de la première famille, puisque:

$f(x_1,f(x_2,\ldots,f(x_k,f(x_1,\ldots,f(x_k,z)\ldots.) \sim$

$f(f(f(x_1,f(x_2,\ldots x_k)\ldots),f(x_1,f(x_2,\ldots x_k)\ldots)),z)$

Cet exemple met en évidence qu'il n'est pas nécessaire de trouver une

méta-règle pour chaque famille de règles. Nous verrons plus loin que la complétion de méta-règles permet de retrouver que la première méta-règle est redondante. ∇

Cependant les hypothèses faites dans la proposition 2 sont peu satisfaisantes pour plusieurs raisons:

- on fait une hypothèse d'inférence sur la "forme" de R_∞ tout entier, alors que dans la pratique, on arrêtera le processus de complétion qui boucle en cours de route. Il n'est jamais possible en pratique de s'assurer que la procédure a amorcé toutes les suites infinies possibles au moment où on l'arrête.

- la schématisation en tête d'une suite de règles est en général trop restrictive. Reprenons pour le voir l'exemple 18.

Exemple 43 : Les équations

```
(x+y)+z = x+(y+z)
(x+y)*z = (x*z) + (y*z)
(x+a).z = (x*z) + a
```

orientées de gauche à droite, engendrent les règles:

$$(x_1+(x_2+\ldots+(x_k+a)\ldots)).z \rightarrow (x_1*z+(x_2*z+\ldots+(x_k*z + a))).$$

L'ensemble des structures est

$$x_1,\ x_1+x_2,\ x_1+(x_2+x_3)\ \ldots$$

Supposons que l'on choisisse pour R_0 la seule règle

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

Il est facile de voir que si l'on schématise séparément les membres gauches et les membres droits des règles, on obtient des méta-termes (X+a).h et

(x*z + Y) qui ne peuvent constituer une méta-règle.

Outre la condition syntaxique que les variables et méta-variables du membre droit doivent être aussi dans le membre gauche, le choix de ce dernier est aussi limité par le fait que l'on veut pouvoir utiliser les méta-règles pour pouvoir décider de la théorie équationnelle.

Ces considérations conduisent à choisir pour système de règles et méta-règles:

```
(X+a).h -> X*h + a
(x+y)*h -> (x+h)*(y+h)
```

Elles vérifient

pour tout $g_k \sim \psi(G)$, $g_k \text{ -*->>> } d_k$.

∇

C'est pour donner cette plus grande latitude de choix que, dans la définition d'un système R_∞ schématisable, on a imposé seulement que la suite des membres gauches soit schématisable. Cependant, comme le met en évidence l'exemple précédent, un inconvénient majeur est alors que le choix du membre droit relève essentiellement de l'intuition, même s'il est limité par l'exigence de correction.

Notons encore que des exemples plus complexes que les nôtres nécessitent plusieurs méta-règles pour réduire les membres gauches et éventuellement aussi les membres droits des règles de R_∞. De tels exemples sont présentés dans [Bellegardel985].

Pour remédier à ces problèmes, nous allons faire maintenant des hypothèses un peu plus fortes sur les relations mises en jeu. Nous allons

exiger, d'une part la convergence de -->>>, d'autre part l'existence d'une relation d'ordre noethérienne contenant à la fois ->R∞ et -->>>. Ces hypothèses vont permettre

- de ne pas avoir à tester toutes les règles engendrées comme dans le paragraphe précédent: on se limitera au test de convergence suivant sur les équations de départ:

pour toute équation g=d de A, il existe un terme t tel que

g -*->>> t et d -*->>> t

- de tenir compte de simplifications possibles des membres droits des règles par des méta-règles,

- de tenir compte de réductions des membres gauches de règles par des méta-règles à des occurrences quelconques,

- de ne pas faire d'hypothèse d'inférence sur la forme de R∞.

Définition 68 : Soit ≦ une relation d'ordre, ≃ l'équivalence associée définie par

t ≃ t' si et seulement si t ≦ t' et t' ≦ t

et < l'ordre strict associé. < est un **ordre de réduction contrainte** associé à R∞, MR et ~ si et seulement si

* > est stable par les opérations de l'algèbre et les substitutions i.e. vérifie pour tous termes t_1 et t_2:

si $t_1 < t_2$, alors $t[u \leftarrow t_1] < t[u \leftarrow t_2]$ pour tout terme t et toute occurrence u dans dom(t)

si $t_1 < t_2$, alors $\sigma(t_1) < \sigma(t_2)$ pour toute substitution σ.

* pour tous termes t et t' non ~-équivalents tels que t ->R∞ t', t > t'.

* -->>> est incluse dans >

* ~ est incluse dans ≃. ○

Exemple 44 : Dans l'ensemble des arbres binaires signés, considérons la relation définie par:

t ≦ t' si et seulement si $|t|_f \leqq |t'|_f$

où $|t|_f$ désigne le nombre d'occurrences du symbole f dans t.

C'est bien évidemment un ordre sur les termes, stable par substitution et par contexte. Soit < l'ordre strict associé. L'équivalence associée ≃ est définie par:

t ≃ t' si et seulement si $|t|_f = |t'|_f$

Toute règle g_k -> d_k de R∞-R_0 vérifie $g_k > d_k$. Toute règle de R_0 vérifie g ≃ d. Les deux méta-règles vérifient G > D. On en déduit que > est un ordre de réduction contraint. ○

Remarque :

- Dans le cas d'une divergence en largeur, l'ordre utilisé pour orienter R∞ est un ordre de E-réduction (voir chapitre 2). Comme dans ce cas ~ est l'équivalence $\sim^E$, la seule condition qui reste à assurer est que -->>> est incluse dans >.

- Dans le cas d'une divergence en profondeur, notons R_0 le sous-ensemble des règles de R∞ qui engendre l'équivalence ~. Soit > la relation d'ordre utilisée par la procédure de complétion pour orienter les règles de R∞. Elle vérifie

si t -+->R∞ t' alors t > t'.

Cette relation est stable par les opérations de l'algèbre et par substitution. Comme R_0 est inclus dans R∞, toute règle g->d de R_0 vérifie g ≧ d. Pour que > soit un ordre de réduction contrainte, il reste à vérifier que:

* toute règle g->d de R_0 vérifie g ≦ d,

* pour toute méta-règle G->D et toute instanciation principale ψ, $\psi(G) > \psi(D)$. $>$ vérifie alors: si t $-*->>>$ t' alors t > t'.

Proposition 3 : Soit $\rightarrow$ = ($>$ $\cup$ $\rightarrow$strict-sous-terme). Si l'ordre de réduction contraint $>$ est noethérien, $\rightarrow$ est noethérienne.

Preuve: Cela résulte du fait que les deux relations sont noethériennes et que t $\rightarrow$strict-sous-terme t' > t" implique:
il existe t_1 tel que t > t_1 $\rightarrow$strict-sous-terme t". □

Proposition 4 : Soient A une théorie équationnelle, R_∞ un système de réécriture infini équivalent à A et schématisable par un ensemble de méta-règles MR. Si

* pour toute équation g=d de A, g $<<<-*->>>$ d
* $-->>>$ est convergente
* l'ordre de réduction contrainte associé est noethérien
* $\sim$ est incluse dans $<-*->R_\infty$,

alors la schématisation de R_∞ par l'ensemble MR de méta-règles est correcte.

Preuve: ● La condition (a) de la proposition 1, satisfaite par hypothèse, implique pour tous termes t et t',
si t $<-*->R_\infty$ t' alors t $<<<-*->>>$ t'.

● Montrons ensuite que
(2) pour tous termes t et t', si t $<<<-*->>>$ t' alors t $<-*->R_\infty$ t'.
Par hypothèse de convergence de $-->>>$, (2) est équivalente à
(2') Si t $-*->>>$ t_0 et t' $-*->>>$ t_0, alors t $<-*->R_\infty$ t',
ce qui se prouve par induction multi-ensemble avec la relation

noethérienne $\rightarrow$ = ($>$ $\cup$ $\rightarrow$strict-sous-terme). Considérons le multi-ensemble $\{\{t...t_0...t'\}\}$. Si t $\sim$ t' alors c'est terminé. Sinon, supposons par exemple que t $-->>>$ t", l'autre cas se traitant de façon symétrique. Par induction multi-ensemble, t" $<-*->R_\infty$ t' et il suffit de prouver que t $-->>>$ t" implique t $<-*->R_\infty$ t". Si t n'est pas équivalent modulo $\sim$ à t", t est réductible par une règle de MR avec $-->>$.

Posons t $-->>[u,\sigma^-,G->D]$ t_1 $\sim$ t". On a $\sigma^-(G) \sim t_{|u}$ et par le corollaire 1, $t_{|u} \sim \delta.\psi(G)$. Par hypothèse $\psi(G)$ est $\sim$-équivalent à un membre gauche d'une règle g->d de R_∞. Donc $t_{|u} \sim \delta(g)$ et $\delta(g)$ $-->>$ $\sigma^-(D)$.

● Cas 1: Si $u \neq \varepsilon$, on peut appliquer l'hypothèse d'induction au multi-ensemble $\{\{t_{|u}, t_{1|u}\}\}$, puisque t $\rightarrow$ $t_{|u}$ et t $\rightarrow$ $t_{1|u}$. Donc $t_{|u}$ $<-*->R_\infty$ $t_{1|u}$, ce qui implique par stabilité de cette congruence par substitution, inclusion de $\sim$ dans $<-*->R_\infty$, et stabilité de $<-*->R_\infty$ par contexte, t $<-*->R_\infty$ t".

● Cas 2: Il reste à prouver le cas où $u=\varepsilon$.

Comme t $\sim$ $\delta(g)$ $->R_\infty$ $\delta(d)$, t $<-*->R_\infty$ $\delta(d)$ et donc t $\sim^A$ $\delta(d)$, puis par hypothèse, t $<<<-*->>>$ $\delta(d)$.

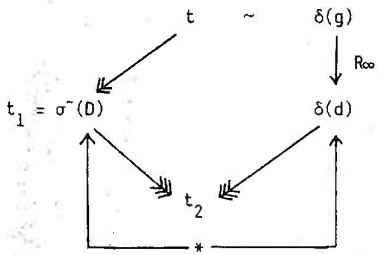
Comme d'autre part t $-->>$ $\sigma^-(D) = t_1$, t_1 $<<<-*->>>$ $\delta(d)$. Par convergence de $-->>>$, il existe t_2 tel que

$$t_1 \; -*->>> \; t_2 \text{ et } \delta(d) \; -*->>> \; t_2.$$

On peut alors appliquer l'hypothèse d'induction noethérienne au multi-ensemble des termes $\{\{t_1,\ldots,t_2,\ldots\delta(d)\}\}$ dont tous les éléments sont des fils stricts de t pour la relation $\rightarrow$. On obtient alors t_1 $<-*->R_\infty$ $\delta(d)$, puis

$t \sim \delta(g) \rightarrow_{R\infty} \delta(d) <-*->_{R\infty} t_1$.

La preuve est illustrée par la figure ci-dessous. □



Preuve de $t \dashrightarrow t_1 \Rightarrow t <-*->_{R\infty} t_1$ (cas 2)

Remarque : Avant de terminer ce paragraphe, notons une voie de recherche que nous n'avons pas explorée. On pourrait chercher à vérifier pour toute règle $(g_k \rightarrow d_k)$ de R_∞:

- pour tout $k>1$, $g_k \dashrightarrow^{\{G\rightarrow D\}} \psi_k(D) \rightarrow^{\{g_{k-1}\rightarrow d_{k-1}\}} d_k$
- $g_1 \dashrightarrow^{\{G\rightarrow D\}} \psi_1(D) \sim d_1$

Ce qui donne par récurrence

pour tout k, $g_k \dashrightarrow^{*}\dashrightarrow^{\{G\rightarrow D\}} \dashrightarrow^{*}\dashrightarrow d_k$.

et donc implique la condition $g_k \lll\dashrightarrow^{*}\dashrightarrow\gg d_k$ donnée précédemment.

Le problème qui se pose maintenant consiste à vérifier que $\dashrightarrow\gg$ est convergente, afin de prouver la correction de la schématisation. De plus, la propriété de convergence permettra d'utiliser dans la pratique la relation $\dashrightarrow\gg$ pour faire des preuves dans la théorie équationnelle $\sim^A$. Nous devons pour cela avoir un moyen de tester la convergence de $\dashrightarrow\gg$ sur l'ensemble de méta-règles, et éventuellement de compléter celui-ci pour

obtenir la convergence. Une telle procédure de complétion doit bien sûr travailler dans l'ensemble des méta-termes. Nous allons donc définir sur cet ensemble la relation de méta-réécriture et la méta-unification. Cela va permettre de définir un ensemble de méta-règles convergent et de donner des conditions nécessaires et suffisantes de convergence. Dans le cas où l'ensemble de méta-règles MR est sans contraintes, nous proposerons une procédure de complétion assurant la convergence de la relation de méta-réécriture sur $T(F,X\cup MX)$. Nous verrons ensuite que cette convergence implique celle de $\dashrightarrow\gg$ sur $T(F,X)$.

5. Méta-réécriture de méta-termes

On va maintenant définir directement des opérations de réduction et de superposition sur les méta-termes. Cela exige de définir au préalable égalité, filtrage et unification de méta-termes.

5.1. Méta-égalité

La méta-égalité est la congruence définie sur $T(F,X\cup MX)$ de la façon suivante:

Définition 69 : Deux méta-termes T_1 et T_2 sont **méta-égaux** et l'on notera $T_1 \equiv T_2$ si et seulement si pour toute instanciation ϕ de leur méta-variables, $\phi(T_1) \sim \phi(T_2)$. ○

Il est facile de voir que restreinte aux termes sans méta-variables, la méta-égalité coincide avec l'égalité engendrée par ~.

Lemme 4 : Deux méta-termes T_1 et T_2 sont méta-égaux, si et seulement si pour toute instanciation principale ψ de leur méta-variables, $\psi(T_1) \sim \psi(T_2)$.

Preuve: - Il est clair que si $T_1 \equiv T_2$, pour toute instanciation principale ψ, $\psi(T_1) \sim \psi(T_2)$, puisque ψ est une instanciation particulière.

- La réciproque est due au fait que toute instanciation ϕ se décompose en $\gamma.\psi$ modulo $\sim$, d'après le lemme 3. □

Remarquons qu'il est possible d'établir un rapport entre la méta-égalité et l'égalité dans une algèbre terminale qui est ici l'algèbre des méta-termes. Soit T(FUX) l'algèbre initiale construite sur F et X, les variables étant considérées comme des constantes. On enrichit cette algèbre en ajoutant une nouvelle sorte méta-terme, de nouvelles constantes de la nouvelle sorte, les méta-variables, et des opérateurs définis de la façon suivante: on définit d'abord une application de l'ensemble des méta-variables dans les structures (i.e. un sous-ensemble de T(FUX)) que l'on étend en une application de l'ensemble des méta-termes dans T(FUX). Soit ψ une telle fonction. Puisque son domaine est la nouvelle sorte et que ses valeurs sont dans la sorte primitive, c'est ce qu'on appelle un observateur dans la terminologie des types abstraits. Considérons alors l'algèbre terminale associée à la spécification de sortes {terme , méta-terme}, d'opérateurs FUXUMXU{ψ} et d'équations R_0 ou E. L'égalité dans l'algèbre terminale est caractérisée de la façon suivante:

T =term T' si et seulement si pour tout observateur ψ, $\psi(T)$ et $\psi(T')$ sont égaux dans l'algèbre initiale T(F,X). Ici $\psi(T) \sim \psi(T')$, puisque l'égalité sur T(FUX) est l'équivalence engendrée par R_0 ou E. L'égalité de deux méta-termes est en fait l'égalité dans cette algèbre terminale. Cette remarque rejoint les idées développées par L.Puel dans[Puel-Cormierl983].

Dans les exemples étudiés, les deux résultats suivants permettent de décider la méta-égalité.

Proposition 5 : Quand l'ensemble des structures contient une variable,

$T_1 \equiv T_2$ si et seulement si $T_1 \sim T_2$.

Preuve: - si $T_1 \equiv T_2$, pour toute instanciation principale ψ, $\psi(T_1) \sim \psi(T_2)$. Puisqu'il existe une interprétation ψ_0 qui envoie chaque méta-variable de T_1 et T_2 sur une variable de X, $\psi_0(T_1) \sim \psi_0(T_2)$ implique $T_1 \sim T_2$.

- Réciproquement, supposons que $T_1 \sim T_2$. Pour toute instanciation principale ψ, $\psi(T_1) \sim \psi(T_2)$. □

Exemple 45 : Cette proposition s'applique aux exemples 36,37,38,40,42 et 43. ∇

Proposition 6 : Quand toute instanciation principale ψ s'écrit $F^i(\psi_0)$ où F est un contexte et ψ_0 une instanciation principale, $T_1 \equiv T_2$ si et seulement si $\psi_0(T_1) \sim \psi_0(T_2)$.

Preuve: Cela résulte du fait que $\sim$ est stable par contexte. □

Exemple 46 : Cette proposition s'applique aux exemples 39 et 41. ∇

5.2. Méta-filtrage

La notion de méta-filtrage que nous proposons, étend la notion de filtrage dans une théorie équationnelle proposée dans le chapitre 2. Ici la congruence équationnelle est remplacée par la méta-égalité.

Définition 70 : Soit Méta-Filtre l'application de T(F,XUMX)xT(F,XUMX) dans l'ensemble des parties de l'ensemble des substitutions de T(F,XUMX) qui à un couple (T,T') associe l'ensemble des substitutions $\sigma^{\sim}$ satisfaisant:

- $\sigma^{\sim}(T) \equiv T'$ et
- pour toute méta-variable Z du domaine de $\sigma^{\sim}$, pour toute instanciation ψ, $\psi(\sigma^{\sim}(Z))$ appartient à DOM.

Un élément de Méta-Filtre(T,T') est appelé un **méta-filtre** de T vers T'. ○

Il faut noter qu'un filtre de T vers T' n'est pas toujours un méta-filtre.

Exemple 47 : Reprenons l'exemple où l'on schématise la suite de règles:

$$f(g^i(f(x))) \rightarrow g^i(f(x))$$

La congruence ~ est l'égalité syntaxique. L'ensemble des structures

$$S = \{f(x_1),\ g(f(x_1)),\ g(g(f(x_1))),\ldots\}$$

engendre l'ensemble des termes dont le symbole de tête est f.

On a alors une méta-règle :

$$f(g(X)) \rightarrow g(X).$$

Il n'existe pas de méta-filtre du méta-terme Z vers le méta-terme a à cause de la définition du domaine des méta-variables. ∇

Cependant si l'ensemble des méta-règles est sans contraintes, tout filtre modulo R_0 est un méta-filtre. Il peut donc exister un méta-filtre de T vers T' sans qu'il existe de filtre et il n'y a pas en général unicité du méta-filtre.

Dans le cas de méta-règles avec contraintes, la conception d'un algorithme calculant le résultat de Méta-Filtre soulève la même difficulté que pour le filtrage contraint, et il ne suffit pas de disposer d'un algorithme de filtrage modulo ~, pour savoir faire du méta-filtrage.

On peut établir le lemme suivant:

Lemme 5 : Soient T et T' deux méta-termes et $\sigma^{\sim}$ un méta-filtre de T vers T'. Pour toute instanciation principale ψ' de T', il existe une instanciation principale ψ de T et une substitution σ telles que $\psi'.\sigma^{\sim} \sim \sigma.\psi$ [MV(T)].

Preuve: Par définition d'un méta-filtre, pour toute instanciation principale ψ', $\psi'(\sigma^{\sim}(T)) \sim \psi'(T')$. Notons ϕ la substitution $\psi'.\sigma^{\sim}$. ϕ est une application de XUMX dans T(F,X), et par le corollaire 1, il existe une substitution σ de T(F,X) et une instanciation principale ψ des méta-variables de T telles que $\phi \sim \sigma.\psi$ [MV(T)]. On peut alors écrire $\psi'.\sigma^{\sim}(Z)) \sim \sigma.\psi(Z)$ pour tout Z de MV(T). □

Le résultat suivant met en évidence le lien existant entre Méta-Filtre et Filtre-contraint: à tout méta-filtre correspond une infinité de filtres contraints.

Proposition 7 : Soit t' un terme et T un méta-terme. Soit $\sigma^{\sim}$ appartenant à

Méta-Filtre(T,t). Pour toute instanciation principale ψ' de $\sigma^{\sim}(T)$, $\psi'.\sigma^{\sim}$ appartient à Filtre-contraint(T,t).

Preuve: Puisque $\sigma^{\sim}(T) \equiv t'$, pour toute instanciation principale ψ' de $\sigma^{\sim}(T)$, $\psi'.\sigma^{\sim}(T) \sim t'$. De plus d'après le lemme 5, $\psi'.\sigma^{\sim}(Z) \sim \sigma.\psi(Z)$ et donc $\psi'.\sigma^{\sim}(Z) \sim \sigma.\psi(Z)$ appartient à DOM. On en déduit que $\psi'.\sigma^{\sim}$ appartient à Filtre-contraint(T,t'). □

5.3. Méta-réduction

La relation de méta-réduction est la relation de réécriture associée à l'application Méta-Filtre.

Définition 71 : La relation de **méta-réduction** ==> avec un ensemble de méta-règles MR, est définie sur T(F,X∪MX) par:

$$T \Longrightarrow_{[u,\sigma^{\sim},G\to D]} T'$$

si et seulement si

- G->D appartient à MR,
- $\sigma^{\sim}$ appartient à Méta-Filtre(G,$T_{|u}$)
- $T' = T[u \leftarrow \sigma^{\sim}(D)]$. ○

A toute méta-réduction, il est possible d'associer une infinité de réductions de la façon suivante:

Lemme 6 : Si un méta-terme T se méta-réduit par une méta-règle G->D avec le méta-filtre $\sigma^{\sim}$ en T', alors pour toute instanciation principale ψ' de T, $\psi'(T)$ -->>> $\psi'(T')$. Plus précisément $\psi'(T)$ -->> $t' \sim \psi'(T')$.

Preuve: Posons $T_{|u} \equiv \sigma^{\sim}(G)$. Par le lemme 5, pour toute instanciation ψ' de T, il existe une instanciation ψ de $\sigma^{\sim}(G)$ et une substitution σ telle que $\psi'.\sigma^{\sim} \sim \sigma.\psi$ [MV(G)].

On en déduit que $\sigma.\psi(G) \sim \psi'.\sigma^{\sim}(G) \sim \psi'(T_{|u}) = \psi'(T)_{|u}$.

Donc $\sigma.\psi$ est un filtre modulo ~ de G vers $\psi'(T)_{|u}$ tel que pour toute méta-variable Z de G, $\sigma.\psi(Z) \sim \psi'.\sigma^{\sim}(Z)$ appartient à DOM.

$\psi'(T)$ est réductible pour -->> à l'occurrence u par la méta-règle G->D en $t' = \psi'(T)[u \leftarrow \sigma.\psi(D)]$. De plus

$T' = T[u \leftarrow \sigma^{\sim}(D)]$ et $\psi'(T') = \psi'(T)[u \leftarrow \psi'.\sigma^{\sim}(D)] \sim t'$.

□

Notation : Notons ==>> la relation ==>.≡ définie sur les méta-termes par

$$T \Longrightarrow\!\!> T'$$

si et seulement si il existe T_1 tel que $T \Longrightarrow T_1 \equiv T'$.

Proposition 8 : Si t et t' appartient à T(F,X), $t \Longrightarrow\!\!> t'$ si et seulement si t -->>> t'.

Preuve:

* $t \Longrightarrow\!\!> t'$ implique t -->>> t':

Cela résulte directement de la proposition précédente et du fait que la méta-égalité restreinte aux méta-termes coincide avec ~.

* t -->>> t' implique $t \Longrightarrow\!\!> t'$:

t -->> $t_0 \sim t'$. Remarquons d'abord que $t_0 \equiv t'$. Si t -->> t_0, il existe une méta-règle de MR, disons G->D, et un filtre contraint $\sigma^{\sim}$ tel que $\sigma^{\sim}(G) \sim t_{|u}$. Pour toute ψ', $\psi'(\sigma^{\sim}(G)) \sim \psi'(t_{|u})$ et donc

$\sigma^{\sim}(G) \equiv t_{|u}$. $\sigma^{\sim}$ est un méta-filtre de G vers $t_{|u}$.

Donc $t \Longrightarrow t[u \leftarrow \sigma(D)] = t_0$. □

Corollaire 3 : L'équivalence engendrée par la relation ==>> sur les méta-termes, notée <<=*=>>, étend l'équivalence <<<-*->>> sur les termes.

En conséquence, pour tous termes t et t', t <<<-*->>> t' équivaut à t <<=*=>> t'. S'il existe un méta-terme T" tel que t =*=>> T" et t' =*=>> T", T" est dans ce cas un terme t" tel que t -*->>> t" et t' -*->>> t". Pour prouver la convergence de -->>> sur les termes, on va d'abord établir un théorème de Church-Rosser pour la relation ==>> sur les méta-termes.

5.4. Théorème de Church-Rosser pour la méta-réécriture

Les opérations sur les méta-termes qui viennent d'être définies permettent de définir un ensemble de méta-règles convergent et de donner des conditions nécessaires et suffisantes pour que la méta-réécriture possède la propriété de Church-Rosser.

Définition 72 : Un **ensemble de méta-règles** MR est **convergent** si et seulement si la relation ==>> est noethérienne et possède la propriété de Church-Rosser:

pour tous méta-termes T et T' tels que T <<=*=>> T', il existe un méta-terme T" tel que T =*=>> T" et T' =*=>> T'. ○

Dans le cas où l'ensemble des méta-règles est sans contraintes, la relation ==> coincide avec la réécriture équationnelle modulo ~.

Proposition 9 : Désignons par E l'ensemble d'équations engendrant l'équivalence ~. Si le système de réécriture équationnelle (MR,E) est convergent, l'ensemble de méta-règles sans contraintes MR est convergent.

Preuve: La convergence de (MR,E) implique la propriété de Church-Rosser: pour tous méta-termes T et T' tels que T <-*->MR,E T', il existe des méta-termes T_1 et T'_1 tels que T -*->MR,E T_1, T' -*->MR,E T'_1 et $T_1 \overset{E}{\sim} T'_1$. Comme <-*->MR,E coincide avec <<=*=>> (car elles coincident toutes les deux avec la congruence engendrée par MRUE sur les méta-termes), cela implique la convergence de l'ensemble de méta-règles MR. □

Nous pouvons alors appliquer les résultats obtenus dans le chapitre 2.

Corollaire 4 : Pour les ensembles de méta-règles sans contraintes, si la congruence ~ est engendrée par un ensemble d'équations non effaçantes E, telles que les classes d'équivalence modulo E soient finies, que le préordre de E-filtrage soit noethérien et telles qu'il existe un algorithme complet et fini de E-unification, la procédure de complétion équationnelle retourne un ensemble de méta-règles convergent.

Dans le cas général de méta-règles avec contraintes, on peut pousser un peu plus loin l'étude de la convergence, en proposant des propriétés locales qui lui sont équivalentes.

Notation : Notons --> la relation de méta-réécriture sans méta-égalité définie par T -->[u,$\sigma^{\sim}$,G->D] T' si et seulement si

- pour toute méta-variable Z de $D(\sigma^{\sim})$ et toute instanciation principale ψ, $\psi(\sigma^{\sim}(Z))$ appartient à DOM.

- $T_{|u} = \sigma^{\sim}(G)$
- $T' = T[u \leftarrow \sigma^{\sim}(D)]$.

Définition 73 :

==> est **localement confluente** si et seulement si pour tous méta-termes T, T_1, T_2 tels que T --> T_1 et T ==> T_2, il existe T' tel que T_1 =*=>> T' et T_2 =*=>> T'.

==> est **cohérente** avec ≡ si et seulement si pour tous méta-termes T, T_1, T_2 tels que $T \equiv T_1$ et T ==> T_2, il existe T' tel que T_1 =+=>> T' et T_2 =*=>> T'. ○

La propriété de Church-Rosser se caractérise par ces deux propriétés.

Théorème 9 : Supposons que ==>> soit noethérienne. ==>> est Church-Rosser si et seulement si

1) ==> est localement confluente

2) ==> est cohérente avec ≡.

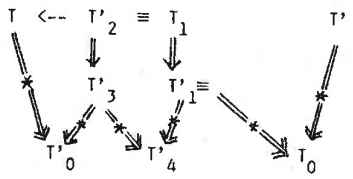
Preuve: La preuve se fait par induction noethérienne sur les multi-ensembles avec ==>>. Soient T <<=*=>> T' et $\{\{T, T_1, T_2, \ldots T'\}\}$ le multi-ensemble des méta-termes apparaissant dans cette preuve. Comme le multi-ensemble $\{\{T_1, T_2 \ldots T'\}\}$ est plus petit que $\{\{T, T_1, T_2, \ldots T'\}\}$, on peut lui appliquer l'hypothèse d'induction:

$$T_1 \text{ =*=>> } T_0 \text{ et } T' \text{ =*=>> } T_0.$$

- Si T ==>> T_1, c'est terminé.
- Si T <<== T_1,
  - -- soit T' =*=>> T_1 et la preuve est terminée,
  - -- soit T_1 ==> T'_1 ≡.=*=>> T_0 et T' =*=>> T_0.

Supposons que T_1 ==>[u,σ~,G->D] T. Alors $T_{1|u} \equiv \sigma^{\sim}(G)$. Posons

$T'_2 = T_1[u \leftarrow \sigma^{\sim}(G)]$. $T_1 \equiv T'_2$ et puisque T_1 est réductible par ==>, par cohérence de ==> avec ≡, T'_2 ==> T'_3 =*=>> T'_4 et T'_1 =*=>> T'_4. Supposons que T'_2 ==>[v,σ~,L->R] T'_3. Comme d'autre part, T'_2 ->[u,σ~,G->D] T, la confluence locale de ==> permet de dire qu'il existe T'_0 tel que que T =*=>> T'_0 et T'_3 =*=>> T'_0. Il reste à appliquer l'hypothèse d'induction au multi-ensemble $\{\{T'_0, \ldots, T'_3, \ldots, T'_4, \ldots, T'_1, \ldots, T_0\}\}$. La preuve est représentée sur la figure suivante:



□

5.5. Méta-unification

La notion de méta-unification est, comme le méta-filtrage, une extension de la notion d'unification dans une théorie équationnelle.

Définition 74 : Soit Méta-UNIF l'application de T(F,XUMX)xT(F,XUMX) dans l'ensemble des parties de l'ensemble des substitutions de T(F,XUMX) qui a un couple (T,T') associe l'ensemble des substitutions σ~ satisfaisant:

- $\sigma^{\sim}(T) \equiv \sigma^{\sim}(T')$ et
- pour toute méta-variable Z du domaine de σ~, pour toute instanciation ψ, $\psi(\sigma^{\sim}(Z))$ appartient à DOM. Toute substitution appartenant à Méta-UNIF(T,T') est appelée un **méta-unificateur** de T et T'. ○

L'application Méta-Unif associe à tout couple de méta-termes T et T' un ensemble générateur de Méta-UNIF(T,T'). Plus précisément

Définition 75 : Soit Méta-Unif une application définie sur T(F,X⊎MX)xT(F,X⊎MX), qui à deux méta-termes T et T' associe l'ensemble Méta-Unif(T,T') des substitutions de T(F,X⊎MX) telles que

(1) pour toute $\sigma^{\sim}$ dans Méta-Unif(T,T'), D($\sigma^{\sim}$) est inclus dans MV(T)⊎MV(T') et I($\sigma^{\sim}$) est choisi d'intersection vide avec W, où W est un sous-ensemble de X⊎MX qui contient MV(T)⊎MV(T').

(2) Méta-Unif(T,T') est inclus dans Méta-UNIF(T,T').

(3) pour toute $\alpha^{\sim}$ dans Méta-UNIF(T,T'), il existe $\sigma^{\sim}$ dans Méta-Unif(T,T') tel que $\sigma^{\sim} \leq \alpha^{\sim}$ [MV(T)⊎MV(T')] où $\leq$ est le préordre associé à Méta-Filtre comme dans le chapitre 2.

Méta-Unif(T,T') est appelé **ensemble complet de méta-unificateurs** de T et T'. ○

Lorsque l'ensemble des structures contient une variable, un méta-unificateur est un unificateur modulo R_0. Il n'y a donc pas en général unicité de l'unificateur minimal. Dans ce cas particulier, un algorithme de méta-unification est connu dès qu'un algorithme de R_0-unification est connu. Ce n'est pourtant pas le cas en général. La conception d'un algorithme calculant le résultat de Méta-Unif soulève des difficultés. L'une d'entre elles apparait lorsqu'il y a plusieurs ensembles de méta-variables avec des domaines distincts. L'unification de deux méta-variables dans des ensembles différents doit prendre en compte les domaines.

La notion de méta-unification permet de définir une notion de paire critique entre deux méta-règles.

Définition 76 : Soient deux méta-règles G->D et L->R telles que MV(G)∩MV(L) = ∅, et que Θ = Méta-Unif(L,$G_{|u}$) soit non vide. L'ensemble

$$\{(P,Q) \mid P=\theta^{\sim}(D),\ Q=\theta^{\sim}(G[u\leftarrow R]) \text{ pour tout } \theta^{\sim} \text{ dans } \Theta\}$$

est un **ensemble complet de paires critiques** de la méta-règle L->R dans la méta-règle G->D à l'occurrence u.

Une paire critique (P,Q) est **confluente** si et seulement si

P =*=>> R et Q =*=>> R. ○

Lemme 7 :

Si $\gamma^{\sim}(G)$ ==>[u,$\gamma^{\sim}$,L->R] T où u est une occurrence dans dom(G) différente de ε, alors il existe une paire critique (P,Q) dans un ensemble complet de paires critiques de L->R dans G->D à l'occurrence u et une méta-substitution $\alpha^{\sim}$ telle que $\alpha^{\sim}(P) \equiv \gamma^{\sim}(D)$ et $\alpha^{\sim}(Q) \equiv T$.

Preuve: Puisque $\gamma^{\sim}(G)_{|u} \equiv \gamma^{\sim}(L)$, L et $G_{|u}$ sont méta-unifiables et il existe alors un méta-unificateur $\sigma^{\sim}$ appartenant à Méta-Unif(L,$G_{|u}$) et une méta-substitution $\alpha^{\sim}$ tels que $P = \sigma^{\sim}(D)$, $Q = \sigma^{\sim}(G[u\leftarrow R])$, $\alpha^{\sim}(P) \equiv \gamma^{\sim}(D)$ et $\alpha^{\sim}(Q) \equiv T$. □

A partir de ces résultats, il apparait clairement que pour pouvoir écrire une procédure de complétion de méta-règles, il faut qu'il soit possible de tester la cohérence de ==> avec ≡ sur une notion adaptée de paires critiques. Nous avons vu que dans le cas, en pratique fréquent, de méta-règles sans contraintes, la procédure de complétion équationnelle permet de conclure. Terminons en traitons deux exemples.

Exemple 48 : Associativité et idempotence

Nous avons vu qu'une schématisation possible de l'ensemble de règles engendré par complétion des deux axiomes d'associativité et idempotence de f, conduit à deux méta-règles

```
f(X,f(X,z)) -> f(X,z)
f(X,X) -> X
```

D'autre part, puisque ces méta-règles sont sans contraintes, la complétion des méta-règles est ici une complétion équationnelle modulo l'équation d'associativité de f. Or la première méta-règle est ici simplifiable par la deuxième modulo l'associativité de f. Le système de méta-règles permettant de décider de l'égalité dans la théorie équationnelle est donc réduit à la seule méta-règle f(X,X) -> X. Notons que dans ce cas, nous avons pu conclure sans utiliser l'unification associative (pour laquelle il existe des ensembles d'unificateurs infinis). ∇

Exemple 49 : Dans les arbres binaires signés, la congruence ~ est celle engendrée par le système de réécriture

```
R0 : -(-(x)) -> x
     -(f(x,y)) -> f(-(y),-(x))
```

Les méta-règles sont:

```
f(-X,f(X,y)) -> y
f(f(y,X),-X) -> y
```

Les méta-règles sont sans contraintes et la complétion des méta-règles est une procédure de complétion modulo R_0. Bien que les classes d'équivalence modulo R_0 ne soient pas finies, il a été possible de tester la convergence de ces méta-règles [Kirchner1982]. ∇

6. Conclusion

Les résultats de ce chapitre sont loin de résoudre tous les problèmes posés par la divergence de la procédure de complétion. Ils permettent seulement d'ouvrir une brèche dans ce problème difficile. Nous avons proposé une notion de schématisation adaptée aux preuves dans l'algèbre libre d'une théorie équationnelle. Nous avons donné un certain nombre de cas où l'utilisation de méta-règles pour décider de l'égalité dans cette théorie équationnelle est licite. Nous avons également montré qu'il est possible de compléter un ensemble de méta-règles dans le cas où elles sont sans contraintes et où l'égalité ~ est définie par un ensemble d'équations non effaçantes pour lesquelles il existe un algorithme complet d'unification.

De nombreux problèmes restent à résoudre:

- la détermination du plus grand généralisé modulo ~. Ce problème est relié au choix des équations engendrant ~, choix qui n'est pas évident dans le cas d'une divergence en profondeur.
- la conception d'algorithmes de filtrage contraint. Cela soulève des problèmes que nous avons déjà évoqués.
- la décidabilité des opérations de méta-égalité, méta-filtrage et méta-unification, qui n'est complètement résolue que dans le cadre des méta-règles sans contraintes.
- l'écriture d'une procédure de complétion de méta-règles. C'est un problème ouvert important, qui n'est lui aussi résolu que dans le cas particulier des méta-règles sans contraintes et d'une congruence ~ engendrée par des équations non effaçantes. On peut également se poser le problème de comparer complétion équationnelle et complétion de méta-règles dans le cas de méta-règles sans contraintes.

Mais les idées développées ici ont probablement d'autres applications. En particulier, la notion de schématisation proposée peut être utilisée pour aborder d'autres problèmes que celui de la complétion. Nous en donnons un autre exemple dans cette thèse, qui est le problème de la résolution d'équations par surréduction. D'autres utilisations sont proposées par F.Bellegarde (loc.cit.).

CHAPITRE 4:

COMPLETION INDUCTIVE

CHAPITRE 4: COMPLETION INDUCTIVE

1. Introduction

Depuis une dizaine d'années sont apparus des langages de programmation basés sur des structures de données abstraites, tels SIMULA, ALGOL68, HOPE, CLEAR, CLU, OBJ et ι. Une structure de données abstraite est construite en groupant dans un module toutes les procédures qui définissent des fonctions de base manipulant une sorte de données. Les opérations qui implantent une structure sont clairement séparées des opérations qui l'utilisent sans avoir besoin de connaitre sa représentation, qui est en ce sens cachée. Il a été largement reconnu depuis que cette approche structurée facilite la spécification, l'écriture, la lecture, les modifications et les raisonnements sur les programmes. Dans le cadre de tels langages, se posent des problèmes de correction ou d'équivalence de représentations, de preuve de propriétés du type de données. Pour formaliser et résoudre ce genre de problème, les notions de types de données abstrait et concret ont été introduites. Un type de données concret est une algèbre hétérogène [Goguen1977]. Pour définir une algèbre associée à un type de données concret, on commence par déterminer les différentes sortes de données, puis on définit les opérations qui permettent de construire des éléments, d'en sélectionner certains ou de tester certaines de leurs propriétés. On obtient ainsi la signature de l'algèbre associée au type de données concret. Les relations entre les divers éléments sont décrites par un ensemble d'équations A et on s'intéresse alors aux algèbres qui sont des modèles de A. Le type de données abstrait standard introduit par le groupe ADJ est l'algèbre initiale des modèles de A. Dans cette approche, l'ensemble d'équations A seul définit les données sans faire référence à une

quelconque implantation. Une spécification algébrique du type abstrait est alors constituée de sa signature et des équations A.

Ce chapitre traite un certain nombre de problèmes de logique équationnelle qui apparaissent dans l'étude des spécifications algébriques des types abstraits. Plus précisément, le but est de montrer comment utiliser systèmes de réécriture et procédures de complétion pour répondre à deux types de questions:

- comment prouver des théorèmes dans l'algèbre initiale d'une théorie par des méthodes équationnelles et plus précisement par complétion?
- comment enrichir une spécification de base avec de nouveaux opérateurs sans altérer celle-ci?

L'idée d'utiliser la procédure de complétion pour prouver des théorèmes dans l'algèbre initiale est connue depuis longtemps. Elle a été proposée par Musser en 1980 et a été largement étudiée depuis. Rappelons en brièvement le principe. Pour prouver par induction qu'une équation e est valide dans l'algèbre initiale d'une spécification définie par un ensemble d'équations A, il s'agit intuitivement de prouver que l'adjonction de e ne modifie pas la congruence engendrée par A sur les termes sans variables. Ce type de preuve par complétion entre dans le cadre mathématique des preuves par consistance dans les systèmes de preuve formels [Kapur1984].

Les résultats de ce chapitre prolongent ceux de Musser [Musser1980], Goguen [Goguen1980], Huet et Hullot [Huet1982], Remy [Remy1982] et Paul [Paul1984]. Tous imposent des restrictions importantes. Musser et Goguen supposent une spécification complète de l'égalité et font jouer un rôle essentiel au type booléen. Huet et Hullot supposent que les constructeurs

sont donnés et qu'il n'y a aucune relation entre eux. Remy et Paul acceptent des relations entre les constructeurs à condition qu'elles soient décrites par un système de règles convergent et que le raisonnement équationnel soit complet dans l'algèbre initiale définie par les constructeurs. Ces différentes approches sont brièvement passées en revue avant d'en détailler une plus récente, due à Kounalis. Elle est basée sur l'idée que, lorsque les équations d'une spécification constituent un système de réécriture convergent R, l'équation e est valide si elle peut être orientée en une règle l->r telle que l possède la propriété de réductibilité inductive (i.e. toutes les instances closes de l sont réductibles) et telle que R∪{l->r} soit encore convergent. Une procédure de preuve de validité s'en déduit. Elle consiste à tester la réductibilité inductive des membres gauches de règles au cours de la complétion. Elle est décrite ici dans le cadre général d'une spécification possédant un système de réécriture équationnelle convergent.

Notre première contribution au problème de preuve de théorèmes dans l'algèbre initiale consiste à montrer comment la construction structurée d'une spécification s'exploite de façon intéressante au niveau des preuves. L'idée est simple: si pour prouver un théorème dans une spécification qui est un enrichissement d'une spécification de base, des lemmes dans cette base sont utiles, ils sont prouvés en ne considérant que le sous-ensemble des équations qui sont nécessaires à la preuve. De plus, la spécification de base possédant un système de réécriture équationnelle convergent, la même technique de preuve par complétion peut être employée. Cette idée, détaillée d'abord dans le cas d'un seul niveau d'enrichissement est ensuite exploitée dans le cadre de la construction de spécifications hiérarchiques.

A une telle construction est associé un graphe de dépendance et un opérateur mis à un noeud ne fait intervenir dans sa définition que les opérateurs définis à des noeuds descendants. Aux feuilles, on suppose qu'il est possible de décider de l'égalité inductive, c'est-à-dire de prouver qu'une équation est valide dans l'algèbre initiale de la théorie correspondante. Habituellement une feuille contient les constructeurs et les relations entre eux, s'il y en a. Si tout est basé sur le type booléen, comme dans l'approche de Musser et Goguen, l'égalité inductive revient à décider que vrai est différent de faux. S'il n'y a pas de relations entre les constructeurs, l'égalité inductive coincide avec l'égalité entre termes construits uniquement sur les constructeurs, qui est syntaxiquement décidable. C'est l'approche de Huet et Hullot. Dans notre cadre général, il suffira de supposer que toutes les spécifications correspondant aux feuilles possèdent un système de réécriture équationnelle convergent, car il sera alors possible d'utiliser la technique de preuve inductive proposée par E.Kounalis.

Une procédure de complétion adaptée aux preuves par induction dans une spécification hiérarchique est présentée et prouvée ici. Cet algorithme travaille de façon hiérarchisée. Supposons que l'on veuille prouver par induction une propriété qui contient des symboles définis au noeud n. Si en calculant des superpositions, la procédure engendre une équation qui contient uniquement des symboles définis en dessous du noeud i, i descendant de n, cette équation doit être prouvée valide dans la spécification correspondant au sous-graphe de sommet i. Si i est une feuille, on utilise la procédure de décision à cette feuille, sinon l'algorithme est récursivement appelé au noeud i. La procédure complète ainsi les spécifications par des conséquences inductives qui apparaissent comme des

lemmes permettant de prouver le théorème donné.

Cet algorithme est aussi utilisé pour valider l'ajout dans une spécification hiérarchique d'un ou plusieurs nouveaux opérateurs et des axiomes les définissant. La spécification à enrichir, dite de base est supposée définie par un système de réécriture équationnelle convergent sur les termes sans variables. Dans un enrichissement de la spécification de base, on exige que l'ajout des nouveaux opérateurs et des nouveaux axiomes ne détruise pas ce qui est déjà construit. Cela est assuré par la propriété de consistance de la nouvelle spécification par rapport à la spécification de base. Si les nouveaux symboles sont définis par des règles de réécriture et des équations formant un système de réécriture équationnelle noethérien, la complétion de l'ensemble de toutes les règles en testant des conditions syntaxiques, fournit une spécification consistante par rapport à la spécification de base. Il peut être intéressant par ailleurs de s'assurer que, lors d'un enrichissement, les nouveaux termes créés s'expriment en fonction des anciens. C'est ce qui est assuré par la propriété de complétude suffisante par rapport à la spécification de base. Si les nouveaux symboles sont définis de facon complète par des règles de réécriture, la complétion de l'ensemble de toutes les règles fournit une spécification complète et consistante par rapport à la spécification de base. Le système de réécriture total possède aussi la propriété de Church-Rosser sur les termes sans variables et on peut alors ajouter un noeud au graphe de la hiérarchie.

L'intérêt de cette approche est qu'elle exploite au maximum la construction hiérarchique des spécifications, ce qui est très important en pratique pour la validation de spécifications volumineuses. Cette idée de

structuration des preuves en utilisant la structuration de la construction apparait par exemple dans le langage ι [Honda1979].

Il faut souligner que dans tout ce qui vient d'être décrit, l'utilisation d'axiomes non orientés est autorisée. En effet les procédures de preuve inductive décrites apparaissent comme des adaptations de la procédure de complétion équationnelle implantée dans REVEUR-3. Tout ensemble d'équations E non effaçantes pour lesquelles un algorithme de E-unification est connu est autorisé. Cela généralise par là les travaux de Huet et Hullot qui proposaient d'admettre des opérateurs définis associatifs et commutatifs. C'est là le deuxième apport que nous faisons au problème de preuve dans l'algèbre initiale.

2. Notions fondamentales

Ce paragraphe présente les concepts et résultats fondamentaux utilisés dans la suite de ce chapitre.

2.1. Types de données concret et abstrait

Définition 77 : Un **type de données concret** de signature Σ est une Σ-algèbre hétérogène.
Une classe d'isomorphisme de Σ-algèbres est une classe d'équivalence de Σ-algèbres pour la relation Isom suivante: M Isom M' si et seulement si il existe un isomorphisme f de M dans M' tel que f(M) = M'.
Un **type de données abstrait** de signature Σ est une classe d'isomorphisme de Σ-algèbres. ○

On définit alors, à un isomorphisme près, un type abstrait comme l'algèbre initiale d'une classe d'algèbres.

Définition 78 : Une classe K de Σ-algèbres est fermée par isomorphisme si et seulement si pour toute Σ-algèbre M de K, si M' est une Σ-algèbre isomorphe à M alors M' est élément de K. Si K est une classe de Σ-algèbres fermée par isomorphisme et possédant une algèbre initiale, alors on appelle **type abstrait** associé à K la classe des algèbres initiales dans K (toutes isomorphes entre elles). ○

Notation : Rappelons les notations adoptées:
- si K est la classe de toutes les Σ-algèbres, nous noterons $I(\Sigma)$ son algèbre initiale et $T(\Sigma)$ son domaine.
- si K est la classe de toutes les Σ-algèbres satisfaisant un ensemble d'axiomes A, l'algèbre initiale sera désignée par $I(\Sigma,A)$. Son domaine est l'ensemble quotient $T(\Sigma)/\sim^A$.

Définition 79 : Une **spécification** est la donnée d'une signature Σ sur un ensemble de sortes S et d'un ensemble d'équations A entre les termes définis par cette signature. ○

Une spécification définit une algèbre initiale caractérisée par des domaines et des applications interprétant les opérations. Le type abstrait associé à une spécification donnée (Σ,A) est l'algèbre initiale de la classe des modèles de A.

2.2. Egalité inductive

Puisque dans l'approche type abstrait que nous suivons ici, nous nous intéressons à l'algèbre initiale de la classe des modèles d'un ensemble d'équations A, nous nous intéressons par là même aux congruences sur les termes clos. Nous en définissons deux: la première est simplement la restriction aux termes clos de la congruence équationnelle engendrée par les

équations de A. Mais une (Σ,A)-algèbre initiale satisfait en général bien plus d'équations que celles qui sont déduites de A uniquement par pur raisonnement équationnel. La deuxième congruence qui nous intéresse est donc la congruence inductive que nous définissons de la façon suivante:

Définition 80 : L'**égalité inductive** de deux termes t et t', notée t ~ind(A) t' est définie par:
t ~ind(A) t' si et seulement si l'équation t=t' est valide dans l'algèbre initiale I(Σ,A) ou encore si et seulement si pour toute substitution close σ, σ(t) $\sim^{A}$ σ(t').
On dit aussi que (t = t') est un **théorème inductif** de A. ○

Il est clair que si t et t' sont deux termes non clos tels que t $\sim^{A}$ t', alors t ~ind(A) t', la réciproque étant fausse, puisqu'il suffit d'exhiber un théorème dont la preuve nécessite une induction sur la structure des termes clos. Un exemple classique est celui de l'associativité de l'addition sur les entiers relatifs.

Exemple 50 :

```
S = {int}

Σ:  0 : → int
    succ  : int → int
    pred : int → int
    plus : int, int → int


A:  succ(pred(x)) = x
    pred(succ(x)) = x
    plus(0,y) = y
    plus(succ(x),y) = succ(plus(x,y))
    plus(pred(x),y) = pred(plus(x,y))
```

Les équations


```
plus(x,y) = plus(y,x)
plus(plus(x,y),z) = plus(x,plus(y,z))
```

sont valides mais ne peuvent être prouvées par raisonnement équationnel à partir des axiomes précédents. ∇

Cependant pour comparer les égalités inductives engendrées par deux ensembles d'équations distincts, il suffit de comparer les restrictions aux termes clos des congruences équationnelles, et réciproquement. C'est ce qu'exprime le lemme suivant:

Lemme 8 : ~ind(A) coincide avec ~ind(A_0) si et seulement si $\sim^{A}$ coincide avec $\sim^{A_0}$ sur les termes clos.

Preuve:

- => Si t et t' sont deux termes clos tels que t $\sim^{A}$ t', t ~ind(A) t', donc t ~ind(A_0) t', d'où t $\sim^{A_0}$ t'.
- <= Si t et t' sont deux termes quelconques tels que t ~ind(A) t', pour toute sustitution close σ, σ(t) $\sim^{A}$ σ(t'), donc σ(t) $\sim^{A_0}$ σ(t') et t ~ind(A_0) t'. □

2.3. Enrichissement

Un concept fondamental en programmation structurée est celui d'enrichissement. Enrichir une spécification consiste à introduire une nouvelle sorte, des opérations dont le profil contient à la fois les anciennes sortes et la nouvelle, ainsi que des équations définissant ces opérations. Un cas particulier d'enrichissement est l'ajout sans nouvelle

sorte d'opérateurs et d'équations. Il est indispensable de s'assurer que ces nouvelles équations n'ont aucun effet sur les sortes initiales, c'est-à-dire qu'elles ne changent ni les domaines ni les applications préalablement définies. Pour cela, on exige que la spécification enrichie, restreinte aux anciennes sortes et opérations, soit l'image par un homomorphisme injectif de la spécification initiale. En général, on exigera de plus que cet homomorphisme soit surjectif.

Pour formaliser cela, nous définissons d'abord la Σ_0-réduite d'une Σ-algèbre.

Définition 81 : Soient deux signatures Σ et Σ_0, telles que Σ_0 est incluse dans Σ. La **Σ_0-réduite** d'une Σ-algèbre M est la Σ_0-algèbre M' telle que

- pour toute sorte s de Σ_0, les domaines de sorte s coincident, i.e. $(D_{M'})_s = (D_M)_s$
- pour toute opération f de Σ_0, $f_{M'} = f_M$. ○

Exemple 51 : Prenons la sorte entier naturel avec $\Sigma_0 = \{0, succ\}$. $I(\Sigma_0,\emptyset)$ admet pour domaine l'ensemble des termes $\{succ^n(0) \mid n \geq 0\}$.
Enrichissons la spécification en ajoutant une nouvelle constante a: $\Sigma = \Sigma_0 \cup \{a\}$. $I(\Sigma,\emptyset)$ admet maintenant comme domaine $\{succ^n(0) \mid n \geq 0\} \cup \{succ^n(a) \mid n \geq 0\}$. La Σ_0-réduite de $I(\Sigma,\emptyset)$ est l'algèbre admettant pour domaine $\{succ^n(o) \mid n \geq 0\} \cup \{succ^n(a) \mid n \geq 0\}$ et pour opérations 0 et succ définies comme

$0 : \rightarrow 0$

$succ : t \rightarrow succ(t)$ pour t de la forme $succ^n(0)$ ou $succ^n(a)$.

Il existe un homomorphisme de $I(\Sigma_0,\emptyset)$ dans la Σ_0-réduite de $I(\Sigma,\emptyset)$. Mais ici il n'est pas surjectif car les termes $succ^n(a)$ ne sont les images

d'aucun terme de $T(\Sigma_0)$. On dit parfois qu'une telle algèbre n'est pas finiment engendrée. ∇

Définition 82 : Etant donné un enrichissement (Σ,A) de (Σ_0,A_0), soit h l'unique Σ_0-homomorphisme de $I(\Sigma_0,A_0)$ dans la Σ_0-réduite de $I(\Sigma,A)$. L'enrichissement est dit **protégé** si et seulement si h est injectif. Il est dit **complètement protégé** si et seulement si h est un isomorphisme. ○

Cela se traduit par les notions de complétude suffisante et consistance dues à Guttag [Guttag1978].

Définition 83 :

- (Σ,A) est **suffisamment complète** par rapport à (Σ_0,A_0) si et seulement si pour tout terme t de $T(\Sigma)$ de sorte initiale, il existe un terme t' de $T(\Sigma_0)$ tel que $t \sim^A t'$.
- (Σ,A) est **consistante** par rapport à (Σ_0,A_0) si et seulement si pour tous termes t_1, t_2 de $T(\Sigma_0)$ de sorte initiale, $t_1 \sim^A t_2$ implique $t_1 \sim^{A_0} t_2$. ○

Soient (Σ,A) et (Σ,A_1) deux enrichissements de (Σ_0,A_0) tels que $\sim^A$ est incluse dans $\sim^{A_1}$. Alors (Σ,A) suffisamment complète par rapport à (Σ_0,A_0) implique (Σ,A_1) suffisamment complète par rapport à (Σ_0,A_0), (Σ,A_1) consistante par rapport à (Σ_0,A_0) implique (Σ,A) consistante par rapport à (Σ_0,A_0).

Proposition 10 : (Σ,A) est un enrichissement protégé de (Σ_0,A_0) si et seulement si (Σ,A) est consistante par rapport à (Σ_0,A_0).
(Σ,A) est un enrichissement complètement protégé de (Σ_0,A_0) si et seulement si (Σ,A) est consistante et suffisamment complète par rapport à (Σ_0,A_0).

Preuve: Considérons une sorte s initiale et notons simplement $T(\Sigma_0)$ et $T(\Sigma)$ les termes de sortes s dans les deux spécifications. Puisque $\sim^{A_0}$ est incluse dans $\sim^{A}$, la classe d'équivalence modulo $\sim^{A_0}$ d'un terme de $T(\Sigma_0)$ est incluse dans la classe d'équivalence modulo $\sim^{A}$ du même terme.

Soit donc $h : T(\Sigma_0)/\sim^{A_0} \rightarrow T(\Sigma)/\sim^{A}$

$$[t]\sim^{A_0} \rightarrow [t]\sim^{A}$$

- h est surjectif si et seulement si (Σ,A) est suffisamment complète par rapport à (Σ_0,A_0)

- h est injectif si et seulement si (Σ,A) est consistante par rapport à (Σ_0,A_0). □

2.4. Validité et consistance

La validité dans l'algèbre initiale de théorèmes qui nécessitent une preuve par induction est étroitement reliée à la propriété de consistance. Nous donnons dans cette section deux lemmes de validité qui sont à la base des méthodes de preuve développées dans la suite.

Le premier lemme de validité exprime simplement que des équations sont valides dans l'algèbre initiale des modèles de A si et seulement si leur adjonction à A ne modifie pas la congruence engendrée sur les termes.

Lemme 9 : (1er lemme de validité)

Soit A_p un ensemble de Σ-équations. Les équations de A_p sont valides dans $I(\Sigma,A)$ si et seulement si $(\Sigma,A\cup A_p)$ est consistante par rapport à (Σ,A).

Preuve: Si les équations de A_p sont valides, pour tous termes clos t_1 et t_2, t_1 et t_2 équivalents modulo $(A\cup A_p)$ implique $t_1 \sim^{A} t_2$. Réciproquement soit t=t' une équation de A_p. Pour toute substitution σ, $\sigma(t)$ et $\sigma(t')$ sont égaux modulo A_p et par consistance $\sigma(t) \sim^{A} \sigma(t')$. □

Pour pouvoir prendre en compte les spécifications obtenues par enrichissement complètement protégé, on raffine un peu ce lemme. Comme on s'intéresse dans ce cas à la consistance par rapport à la spécification de base, on fait intervenir la propriété de complétude suffisante pour s'y ramener. Le deuxième lemme de validité exprime que deux enrichissements emboités suffisamment complets d'une même spécification de base sont consistants en même temps. Pour prouver la validité d'une équation dans l'algèbre initiale d'une spécification suffisamment complète, il suffit de prouver la consistance de la spécification enrichie par l'équation par rapport à la spécification de base.

Lemme 10 : (2eme lemme de validité)

Soit (Σ,A) un enrichissement suffisamment complet par rapport à une spécification de base (Σ_b,A_b) et A' un ensemble de Σ-équations telles que $\sim^{A}$ est incluse dans $\sim^{A'}$ pour les termes de sortes initiales. Alors les trois propositions suivantes sont équivalentes:

(1) (Σ,A') est consistante par rapport à (Σ_b,A_b)

(2) (Σ,A) est consistante par rapport à (Σ_b,A_b) et toute équation de A' de sorte initiale est valide dans $I(\Sigma,A)$

(3) (Σ,A) est consistante par rapport à (Σ_b,A_b) et $\sim^{A}$ coincide avec $\sim^{A'}$ sur l'ensemble des termes clos de sorte s initiale.

Preuve: ● (1) => (2)

Il est clair que (Σ,A) est consistante par rapport à (Σ_b,A_b) puisque $\sim^A$ est incluse dans $\sim^{A'}$ sur les termes de sorte initiale. D'autre part, supposons qu'il existe une équation $t=t'$ de A' de sorte initiale qui n'est pas valide dans $I(\Sigma,A)$. On peut donc trouver une substitution close σ telle que $\sigma(t)$ et $\sigma(t')$ ne soient pas A-équivalents. Comme $\sigma(t)$ et $\sigma(t')$ sont deux termes de $T(\Sigma)$ de sorte initiale et que (Σ,A) est suffisamment complète par rapport à (Σ_b,A_b), il existe des termes de sorte initiale t_0 et t'_0 dans $T(\Sigma_b)$ tels que $\sigma(t) \sim^A t_0$ et $\sigma(t') \sim^A t'_0$. Comme t_0 et t'_0 ne peuvent pas être égaux modulo A, ils ne le sont pas modulo A_b. Pourtant puisque $\sim^A$ est incluse dans $\sim^{A'}$ sur les termes de sorte initiale, $t_0 \sim^{A'} \sigma(t) \sim^{A'} \sigma(t') \sim^{A'} t'_0$. Les deux termes t_0 et t'_0 dans $T(\Sigma_b)$ satisfont $t_0 \sim^{A'} t'_0$ et t_0 n'est pas A_b-équivalent à t'_0, ce qui contredit la consistance de (Σ,A') par rapport à (Σ_b,A_b).

● (2) => (3)

Soient deux termes t et t' de $T(\Sigma)$ tels que $t \sim^{A'} t'$. Puisque toute équation de A' est valide dans $I(\Sigma,A)$, toute étape de A'-égalité dans la preuve de $t \sim^{A'} t'$ peut être remplacée par une suite de A-égalités.

● (3) => (1) est évident. □

Le premier lemme de validité apparait comme un cas particulier du deuxième puisque si (Σ,AUA_p) est un enrichissement de (Σ,A) par des équations seulement, alors les équations de A_p sont valides dans $I(\Sigma,A)$ si

et seulement si l'enrichissement est consistant. Cela se déduit de (2)<=>(1) dans le lemme précédent avec $(\Sigma_b,A_b)=(\Sigma,A)$ et $A'=AUA_p$.

2.5. Principe d'induction sans induction

Le principe d'induction sans induction est basé sur l'idée d'utiliser des techniques purement équationnelles et plus précisément la réécriture pour prouver la propriété de consistance. Tester la consistance exige de savoir décider de l'égalité engendrée par les équations, au moins sur les termes clos. Ce sera le cas si l'on dispose d'une système de réécriture (éventuellement équationnelle) équivalent aux axiomes sur les termes clos. Les deux lemmes de validité se traduisent alors par des propriétés sur les formes normales des termes clos.

Définition 84 : Une spécification (Σ,A) est dite **convergente** si et seulement si il existe un système de réécriture équationnelle (R,E) convergent qui engendre sur les termes clos une relation d'équivalence coincidant avec $\sim^A$. On notera aussi dans ce cas la spécification (Σ,RUE). ○

Si (Σ,RUE) et (Σ_0,R_0UE_0) sont deux spécifications convergentes telles que (Σ,RUE) est un enrichissement de (Σ_0,R_0UE_0), nous supposerons toujours implicitement que Σ_0, R_0 et E_0 sont respectivement inclus dans Σ, R et E, et que la relation de réécriture équationnelle $->R_0E_0$, utilisée dans (Σ,R_0UE_0), est incluse dans la relation ->RE utilisée dans (Σ,RUE). Cette restriction signifie simplement qu'on ne change pas la méthode de filtrage associée aux règles de R_0 lorsqu'on les considère comme des règles de R.

Considérons tout d'abord le cas d'un enrichissement par des équations seulement. La consistance de (Σ,RUE) par rapport à (Σ,R_0UE_0) se traduit par une propriété sur les formes normales des termes de $T(\Sigma)$.

Lemme 11 : Soit (Σ,R_0UE_0) une spécification convergente et (Σ,RUE) un enrichissement de (Σ,R_0UE_0) par des équations seulement. Supposons de plus que (Σ,RUE) est convergente. Alors (Σ,RUE) est consistante par rapport à (Σ,R_0UE_0), si et seulement si

- tout terme clos de $T(\Sigma)$ a les mêmes ensembles de formes normales pour les deux relations $->RE$ et $->R_0E_0$
- les classes d'équivalence de $T(\Sigma)$ modulo E et E_0 coincident.

Preuve: - Supposons d'abord (Σ,RUE) consistante. Soit t un terme de $T(\Sigma)$, t_1 et t_0 des formes normales de t pour respectivement $->RE$ et $->R_0E_0$. Comme t_1 et t_0 sont égaux modulo (RUE), par consistance, t_1 et t_2 sont égaux modulo (R_0UE_0). Par convergence de (Σ,R_0UE_0) et puisque t_1 et t_0 sont deux termes R_0E_0-irréductibles, $t_1 \sim^{E_0} t_0$. t_1 étant RE-irréductible, t_0 l'est aussi.

- Réciproquement, soient deux termes t_1 et t_2 de $T(\Sigma)$ égaux modulo (RUE). Cela implique qu'ils ont des formes normales E-égales pour $->RE$. Par hypothèse, ce sont aussi des formes normales E_0-égales pour $->R_0E_0$. D'où t_1 et t_2 sont égaux modulo (R_0UE_0). □

Remarque : Notons que l'hypothèse de convergence de (R,E) n'intervient que dans la preuve de la réciproque.

La procédure de complétion permet de construire des systèmes de réécriture équationnelle (R,E) et (R_0,E_0) convergents. La consistance de la spécification convergente (Σ,RUE) par rapport à la spécification initiale (Σ,R_0UE_0) est satisfaite si les systèmes obtenus définissent sur les termes initiaux les mêmes formes normales. Pour faire une preuve de validité, on ajoute les assertions à prouver et on complète l'ensemble. Si le système résultant définit les mêmes formes normales que le système initial, les équations sont valides.

Supposons maintenant que (Σ,RUE) inclus dans $(\Sigma,R'UE')$ sont deux enrichissements suffisamment complets par rapport à (Σ_b,R_bUE_b), que (Σ_b,R_bUE_b) et $(\Sigma,R'UE')$ sont convergentes et que (Σ,RUE) est consistante par rapport à (Σ_b,R_bUE_b). La consistance de $(\Sigma,R'UE')$ par rapport à (Σ,RUE) se traduit cette fois par une propriété sur les formes normales des termes de $T(\Sigma_b)$ uniquement. La complétude suffisante de (Σ,RUE) par rapport à (Σ_b,R_bUE_b) permet ici de se passer de la convergence de (Σ,RUE).

Lemme 12 : Soient (Σ_b,R_bUE_b), (Σ,RUE) et $(\Sigma,R'UE')$ des spécifications telles que:

- (Σ_b,R_bUE_b) est une spécification convergente,
- (Σ,RUE) est consistante par rapport à (Σ_b,R_bUE_b) et tout terme de $T(\Sigma)$ contenant un symbole de $\Sigma - \Sigma_b$ est RE-réductible,
- les classes d'équivalence de $T(\Sigma_b)$ modulo E et E' coincident,
- $(\Sigma,R'UE')$ est convergente.

Alors $(\Sigma,R'UE')$ est consistante par rapport à (Σ,RUE), si et seulement si tout terme clos t de $T(\Sigma_b)$ a les mêmes ensembles de formes normales pour les deux relations $->RE$ et $->R'E'$.

Preuve: - Supposons $(\Sigma,R'UE')$ consistante par rapport à (Σ,RUE). Soient t un terme de $T(\Sigma_b)$, t_1 et t_0 des formes normales de t pour respectivement $->RE$ et $->R'E'$. t_1 et t_0 étant RE-irréductibles ne peuvent contenir de symbole de $\Sigma - \Sigma_b$. Ce sont donc deux termes de $T(\Sigma_b)$ égaux modulo (R'UE'). Par consistance de $(\Sigma,R'UE')$ et de (Σ,RUE),

ils sont égaux modulo $(R_b \cup E_b)$ et donc t_1 et t_0 sont égaux modulo E_b puisqu'ils sont R_bE_b-irréductibles.

- Réciproquement, soient deux termes t_1 et t_2 de $T(\Sigma)$ tels que t_1 et t_2 soient égaux modulo $(R'\cup E')$. Par complétude suffisante de $(\Sigma,R\cup E)$ par rapport à $(\Sigma_b,R_b\cup E_b)$, il existe t'_1 et t'_2 dans $T(\Sigma_b)$ tels que t_1 soit égal modulo $(R\cup E)$ à t'_1 et t_2 à t'_2. D'où t'_1 et t'_2 sont égaux modulo $(R'\cup E')$ et ont des $R'E'$-formes normales égales modulo E'. Puisque ce sont aussi des RE-formes normales égales modulo E par hypothèse, t_1 et t_2 sont égaux modulo $(R\cup E)$. □

C'est cette dernière méthode, utilisant la complétude, qui a été employée dans les premières approches du problème de preuve par induction que nous allons décrire maintenant.

3. Différentes approches du problème

Historiquement deux approches ont été proposées par Musser et Goguen d'une part, et par Huet et Hullot d'autre part. Nous allons brièvement les rappeler en montrant qu'elles sont en fait des instances du processus que nous venons de décrire. Contrairement à ces deux approches utilisant la propriété de complétude suffisante, celle proposée par E.Kounalis [Jouannaud1985] est basée directement sur l'hypothèse que les spécifications sont convergentes.

3.1. Approche de Musser et Goguen

L'approche de Musser, clarifiée et améliorée ensuite par Goguen, consiste à enrichir la spécification par une axiomatisation de l'égalité, en

introduisant

- une nouvelle sorte newbool avec les constantes vrai et faux,
- un prédicat d'égalité, c'est-à-dire une opération $=_s$ de profil (s.s;newbool) pour chaque sorte s de la spécification initiale,
- de nouvelles équations axiomatisant ces opérations et telles que étant donné deux termes clos t et t', on puisse prouver $(t=_s t')$=vrai si et seulement si t et t' peuvent être prouvés égaux dans la spécification initiale par les règles de déduction équationnelle. On suppose de plus qu'on ne peut pas prouver vrai=faux.

Exemple 52 : Les équations

$(x =_s x) = vrai$
$(0 =_s succ(x)) = faux$
$(succ(x) =_s 0) = faux$
$(succ(x) =_s succ(y)) = (x =_s y)$

donnent une axiomatisation de l'égalité pour les entiers naturels. ∇

Cet enrichissement doit bien sûr être complètement protégé et il fournit alors ce qu'on appelle un enrichissement égalitaire de la spécification initiale.

Définition 85 : Soit BOOL la signature de sorte newbool et d'opérateurs constants vrai et faux. Soit $\Sigma^=$ la réunion de BOOL et d'une signature Σ enrichie, pour chaque sorte s différente de newbool, d'une opération $=_s$ de profil (s.s;newbool). Soit $A^=$ un ensemble de $\Sigma^=$-équations. Alors $(\Sigma^=,A^=)$ est un **enrichissement égalitaire** de (Σ,A) si et seulement si il satisfait les conditions suivantes:

(1) Egalité équationnelle:

pour chaque sorte s différente de newbool dans $\Sigma^=$ et pour tous termes clos t et t' de sorte s,

a. l'équation $(t =_s t') = vrai$ est déductible de $A^=$ si et seulement si $(t = t')$ est déductible de A.

b. l'équation $(t =_s t') = faux$ est déductible de $A^=$ si et seulement si $(t = t')$ n'est pas déductible de A.

(2) Consistance:

(vrai = faux) n'est pas déductible de $A^=$ pour la sorte newbool. ○

En d'autres termes, le prédicat d'égalité dans l'enrichissement est caractérisé par la déduction équationnelle dans la spécification d'origine.

Pour ce type d'enrichissement, il est facile de vérifier que l'enrichissement est complètement protégé.

Lemme 13 : Soit (Σ,A) un enrichissement de (Σ_0,A_0) tel que:

- il existe une seule sorte s_0 dans $\Sigma - \Sigma_0$
- les opérations et les constantes de $\Sigma - \Sigma_0$ ont toutes pour codomaine s_0
- les équations de $A - A_0$ ont toutes pour sorte s_0.

Alors l'enrichissement est complètement protégé.

Preuve: Voir [Meseguer1983]. □

Les conditions imposées dans la définition d'un enrichissement égalitaire impliquent à la fois les propriétés de complétude suffisante et de consistance de $(\Sigma^=,A^=)$ par rapport à (Σ,A) et par rapport à l'algèbre de sorte newbool, ceci dès que $\sim^A$ est décidable.

Lemme 14 : Si $(\Sigma^=,A^=)$ est un enrichissement égalitaire de (Σ,A) et si $\sim^A$ est décidable, alors $(\Sigma^=,A^=)$ est consistante et complète par rapport à (Σ,A) et par rapport à (BOOL,ø)

Preuve: $(\Sigma^=,A^=)$ est consistante par rapport à (BOOL,ø) par définition d'un enrichissement égalitaire. D'autre part, tout terme de $T(\Sigma^=)$ de sorte newbool peut être soit l'une des constantes vrai ou faux ou un terme $(t =_s t')$. Dans ce dernier cas, puisque $\sim^A$ est décidable, soit $t \sim^A t'$ et donc $(t =_s t') = vrai$, soit t n'est pas A-équivalent à t' et donc $(t =_s t') = faux$. Dans tous les cas, tout terme de sorte newbool est équivalent modulo $A^=$ à un terme de T(BOOL).

La consistance et la complétude par rapport à (Σ,A) résultent du lemme 13. □

Le problème de la validité d'un ensemble d'équations dans une algèbre initiale se réduit à celui de la consistance de l'enrichissement égalitaire augmenté de ces équations.

Théorème 10 : Soit $(\Sigma^=,A^=)$ un enrichissement égalitaire de (Σ,A). Un ensemble A_p de Σ-équations est valide dans $I(\Sigma,A)$ si et seulement si (vrai=faux) ne se déduit pas de $A^= \cup A_p$ par les règles de déduction équationnelle.

Preuve: Par application du lemme 10 (2eme lemme de validité), en prenant $(\Sigma_b,A_b)=(BOOL,ø)$, $(\Sigma,A)=(\Sigma^=,A^=)$ et $(\Sigma',A')=(\Sigma^=,A^= \cup A_p)$, on obtient: l'ensemble A_p de Σ-équations est valide dans $I(\Sigma^=,A^=)$ si et seulement si (vrai=faux) ne se déduit pas de $A^= \cup A_p$ par les règles de déduction équationnelle. Par consistance de $(\Sigma^=,A^=)$ par rapport à (Σ,A), l'ensemble A_p de Σ-équations est valide dans $I(\Sigma^=,A^=)$ si et

seulement si l'ensemble A_p de Σ-équations est valide dans $I(\Sigma,A)$.

□

En termes de réécriture, ceci signifie simplement que les formes normales des termes de base qui sont ici vrai et faux uniquement, sont les mêmes pour les systèmes de réécriture associés à $A^=$ et à $A^= \cup A_p$. La propriété de complétude suffisante de $(\Sigma^=,A^=)$ par rapport à BOOL est ici implicitement utilisée pour se ramener à l'étude des formes normales des termes de sorte newbool.

On en déduit une méthode de preuve par induction d'un ensemble d'assertions A_p dans $I(\Sigma,A)$:

(1) Calculer un enrichissement égalitaire $(\Sigma^=,A^=)$ de (Σ,A) tel que l'égalité engendrée par $A^=$ soit équivalente à celle engendrée par un système de réécriture équationnelle convergent, en utilisant une procédure de complétion équationnelle.

(2) Compléter de même $A^= \cup A_p$.

(3) Si la complétion termine ou engendre une infinité de règles, et si avec le système résultant vrai et faux ont des formes normales distinctes, les équations de A_p sont valides dans $I(\Sigma,A)$

(4) Si au cours de la complétion, l'équation vrai=faux est engendrée, au moins une des équations de A_p n'est pas valide dans $I(\Sigma,A)$.

3.2. Approche de Huet et Hullot

Contrairement à la précédente, cette méthode ne requiert pas l'introduction de prédicats d'égalité. Par contre elle utilise l'existence de constructeurs, c'est-à-dire pour chaque sorte s, l'existence d'une

sous-signature Σ_0 et le fait que (Σ,A) soit un enrichissement consistant et suffisamment complet de $(\Sigma_0,\emptyset)$. Ici encore la complétude suffisante permet de se ramener à la comparaison des formes normales sur les termes construits uniquement sur les constructeurs. Comme il n'y a à l'origine aucune relation entre constructeurs, il suffit que l'adjonction des équations à prouver n'en engendre pas.

On en déduit une méthode de preuve par induction d'un ensemble d'assertions A_p dans $I(\Sigma,A)$:

(1) Vérifier la complétude suffisante de (Σ,A) par rapport à $(\Sigma_0,\emptyset)$.

(2) Appliquer la procédure de complétion à $(A \cup A_p)$.

(3) Si la complétion termine ou engendre une infinité de règles, et si les termes clos de $T(\Sigma_0)$ sont irréductibles, les équations de A_p sont valides dans $I(\Sigma,A)$

(4) Si au cours de la complétion, une équation dont les deux membres sont des termes de $T(\Sigma_0)$ est engendrée, au moins une des équations de A_p n'est pas valide dans $I(\Sigma,A)$.

Le fait qu'il n'y ait pas d'équation sur les constructeurs permet d'introduire des simplifications sur les équations engendrées. Ainsi, lorsqu'une équation $c(t_1,\ldots,t_n) = c'(t'_1,\ldots,t'_n)$ est engendrée avec c et c' appartenant à Σ_0, soit $c \neq c'$ et l'algorithme s'arrête en réfutation, soit $c=c'$ et l'équation est remplacée par les n équations $(t_i=t'_i)$. Ces simplifications permettent d'améliorer considérablement l'efficacité du processus et parfois même de le faire terminer, alors que la complétion sans ces simplifications diverge. Remarquons que sur les entiers, avec les constructeurs 0 et succ, les simplifications permettent de remplacer toute équation succ(t) = succ(t') par l'équation t=t', tout comme le fait dans la

démarche de Goguen, l'équation (succ(x) $=_s$ succ(y)) = (x $=_s$ y). Dans cette approche en effet, l'axiomatisation de l'égalité est implicite grâce au fait qu'il n'y ait pas d'équations entre les constructeurs.

Les limitations de cette méthode d'induction sans induction sont discutées dans [Lankford1981].

3.3. Approche de Paul

E.Paul propose une généralisation de la méthode de Huet et Hullot au cas où il existe des relations entre constructeurs, sous réserve que les deux conditions suivantes soient vérifiées:

- on peut construire à partir des relations entre constructeurs un système de réécriture convergent éventuellement modulo l'associativité et la commutativité de certains constructeurs.
- l'ensemble de ces relations possède la propriété dite de complétude inductive qui traduit le fait que, pour la théorie équationnelle qu'elles définissent, l'égalité équationnelle coincide avec l'égalité inductive, ou encore que le raisonnement équationnel est complet. C'est le cas en pratique pour de nombreuses spécifications courantes, en particulier sur les entiers relatifs sur lesquels on a les relations suivantes entre les constructeurs succ et pred: succ(pred(x)) = x et pred(succ(x)) = x.

Un autre apport est d'introduire un procédé de simplification d'équations généralisant celui de Huet et Hullot. Les simplifications sont codées sous forme de relation non équationnelles qui ont la forme suivante:

$$t_1 = t'_1 \Rightarrow t = t'$$

avec

- soit t et t' sont des termes construits uniquement sur les constructeurs

- soit (t=t') est l'équation (vrai=faux).

Ces relations doivent être fournies au départ et être valides dans l'algèbre initiale $I(\Sigma,A)$. Ainsi sur les entiers naturels on donne:

succ(x) = succ(y) => x = y
succ(x) = 0 => vrai = faux.

le calcul des paires critiques dans l'algorithme de complétion doit alors comporter aussi le calcul de paires critiques entre une règle l->r et une relation de ce type. Une telle paire critique existe si l=r est unifiable avec $t_1=t'_1$ et elle est définie comme la paire $(\sigma(t)=\sigma(t'))$. L'unification doit se faire modulo la commutativité du symbole =. Il est à noter que la paire critique ainsi calculée n'est autre que le résolvant de la clause (l=r :-) avec (t=t' :- $t_1=t'_1$). Au niveau de l'examen des équations, l'algorithme s'arrête en réfutation dès la découverte d'une équation (vrai=faux). Par rapport aux algorithmes précédents, celui-ci a la propriété de fabriquer des équations qui ne sont pas des conséquences équationnelles des équations de départ. Elles ne sont donc pas valides dans tous les modèles des équations de départ, mais elles sont valides dans l'algèbre initiale puisque les règles d'inférence utilisées pour les déduire le sont.

3.4. Approche de Kounalis et Jouannaud

Cette approche toute récente donne un nouvel éclairage au problème. Elle montre comment se passer de la complétude suffisante dans le cas de spécifications convergentes, en s'appuyant directement sur le lemme 11. Le concept fondamental mis en évidence est celui de réductibilité inductive pour un système de réécriture équationnelle (R,E). Nous présentons maintenant cette approche, détaillée et prouvée dans la thèse de E.Kounalis

[Kounalis1985], sous une forme légèrement plus générale, grâce à la notion de RE-réécriture.

4. Preuve de validité dans une spécification convergente

Le concept de réductibilité inductive (ou quasi-réductibilité dans des travaux antérieurs de E.Kounalis et JP.Jouannaud) est apparu à l'origine dans l'étude de la propriété de complétude suffisante. En cherchant des conditions permettant d'assurer cette propriété, on utilise le fait qu'un symbole f est défini de façon complète si tout terme clos commençant par f est réductible. Lorsqu'on essaie de faire une telle preuve en examinant les termes non clos, on s'aperçoit qu'il existe des termes non clos irréductibles dont toutes les instances closes sont réductibles.

4.1. Réductibilité inductive

Définition 86 : Soit (R,E) un système de réécriture équationnelle noethérien modulo E. Un terme t possède la propriété de **RE-réductibilité inductive** si et seulement si pour toute substitution close σ, $\sigma(t)$ est RE-réductible. ○

On peut également définir une notion de R/E-réductibilité inductive, pour la relation de réécriture dans les classes d'équivalence de termes modulo E. Cependant, nous nous intéressons uniquement ici aux relations de réécriture définies sur les termes. Dans toute la suite, lorsqu'on parle de RE-réductibilité inductive dans une spécification convergente $(\Sigma,R\cup E)$, on suppose implicitement que ->RE est la relation de réécriture associée au système de réécriture équationnelle (R,E) décrivant la théorie.

Exemple 53 : Le terme plus(x,y) dans la spécification des entiers relatifs

décrites ci dessous a la propriété de RE-réductibilité inductive, en choisissant par exemple pour ->RE la réécriture ->R,E.

S = {int}

Σ: 0 : → int
succ : int → int
pred : int → int
plus : int, int → int

R: succ(pred(x)) = x
pred(succ(x)) = x
plus(0,y) = y
plus(succ(x),y) = succ(plus(x,y))
plus(pred(x),y) = pred(plus(x,y))

E: plus(x,y) = plus(y,x)
plus(plus(x,y),z) = plus(x,plus(y,z))

∇

La décidabilité de la RE-réductibilité inductive est étudiée dans [Jouannaud1985]. Lorsque E est l'ensemble vide et R un ensemble de règles linéaires à gauche, la R-réductibilité inductive est décidable. Il en est de même pour certaines théories E, parmi lesquelles les théories permutatives. Le cas général est un problème encore ouvert.

Cette propriété est directement liée à une propriété concernant les formes normales d'un terme, qui s'exprime de la façon suivante:

Lemme 15 : Soient (R,E) un système de réécriture équationnelle noethérien modulo E, R_p un ensemble de règles g->d telles que g a la propriété de RE-réductibilité inductive, E_p un ensemble d'équations g=d telles que g et d aient la propriété de RE-réductibilité inductive. Alors un terme clos t est en forme normale pour ->RE si et seulement si il est en forme normale pour ->R'E' où $R'=R\cup R_p$ et $E'=E\cup E_p$.

On en déduit le théorème suivant:

Théorème 11 : Soit (Σ,RUE) une spécification convergente. Soient R_p un ensemble de règles g->d telles que g ait la propriété de RE-réductibilité inductive, E_p un ensemble d'équations g=d telles que g et d aient la propriété de RE-réductibilité inductive. Si (RUR_p,EUE_p) est convergent, alors les équations de $R_p \cup E_p$ sont valides dans $I(\Sigma,RUE)$.

La notion de réductibilité inductive permet de donner un théorème de validité qui est une adaptation du lemme 9 (1er lemme de validité) aux systèmes de réécriture équationnelle.

Théorème 12 : Soit (Σ,RUE) une spécification convergente. Soient R_p un ensemble de règles g->d telles que g ait la propriété de RE-réductibilité inductive, E_p un ensemble d'équations g=d telles que g et d aient la propriété de RE-réductibilité inductive. Posons $R'=RUR_p$ et $E'=EUE_p$. S'il existe un système de réécriture équationnelle (R_∞,E') convergent, tel que pour tout terme clos t, les formes normales de t pour $->R_\infty E'$ et pour $->R'E'$ coincident, alors les équations de $R_p \cup E_p$ sont valides dans $I(\Sigma,RUE)=I(\Sigma,R_\infty UE)$.

Preuve: Supposons qu'il existe une règle g->d de R_p qui n'est pas valide dans $I(\Sigma,RUE)$. Il existe alors une substitution σ telle que $\sigma(g)$ et $\sigma(d)$ ne sont pas RUE-équivalents. Ce qui implique que $\sigma(g)\downarrow_{RE}$ et $\sigma(d)\downarrow_{RE}$ ne sont pas E-équivalents. Par le lemme 15, $\sigma(g)\downarrow_{R'E'}$ et $\sigma(d)\downarrow_{R'E'}$ ne sont pas non plus E'-équivalents. Donc $\sigma(g)\downarrow_{R_\infty E'}$ n'est pas E'-équivalent à $\sigma(d)\downarrow_{R_\infty E'}$. On contredit ainsi la confluence de (R_∞,E'). De la même manière, s'il existe une équation g=d de E_p qui n'est pas valide dans $I(\Sigma,RUE)$, on contredit la cohérence

de (R_∞,E') modulo E'. L'identité des algèbres initiales résulte du fait qu'elles sont toutes deux identiques à $I(\Sigma,R'UE')$, la première à cause de la validité des équations de $R_p \cup E_p$ dans $I(\Sigma,RUE)$, la deuxième à cause de l'identité des congruences engendrées par $(R_\infty UE')$ et $(R'UE')$ sur les termes clos de $T(\Sigma)$. □

Afin de prouver la validité d'une équation dans une spécification convergente, on l'ajoutera donc soit à l'ensemble de paires à orienter en règles, soit à l'ensemble d'équations non orientées, et on complètera l'ensemble en vérifiant que chaque règle ajoutée a un membre gauche ayant la propriété de RE-réductibilité inductive pour le système de réécriture équationnelle en cours.

4.2. Preuve inductive dans une spécification convergente

Comme dans la procédure de complétion équationnelle, la procédure de preuve inductive a pour arguments un ensemble de paires de termes P, un ensemble de règles R, un ensemble constant d'équations E et un ordre de E-réduction noethérien >.

Chaque règle possède une étiquette qui permet de retrouver l'ordre dans lequel elles ont été introduites. Cela permet d'implanter facilement l'hypothèse d'équité nécessaire au moment du choix d'une règle pour le calcul des paires critiques. Pour ne pas alourdir les notations, les étiquettes ne sont pas mentionnées dans l'algorithme. Chaque règle est également marquée si et seulement si ses paires critiques avec toutes les règles d'étiquette inférieure et toutes les équations de E ont été calculées.

La procédure PREUVE-INDUCTIVE a pour but de prouver ou réfuter un ensemble d'équations $R_p \cup E_p$ dans une spécification convergente définie par le système de réécriture équationnelle (R_0, E_0).

```
Initialisation:
  P = R_p, R = R_0, E = E_0 ∪ E_p

PREUVE-INDUCTIVE(P,R,E,>)
 SI P n'est pas vide
 ALORS choisir une paire (p,q) dans P; p'=p↓; q'=q↓;

       SI p' ~E q' ALORS R = PREUVE-INDUCTIVE(P-{(p,q)},R,E,>)
       SINON  l->r = ORIENTATION(p',q',>)
              SI NON(IND-REDUCTIBLE(l,R_0,E_0))
              ALORS ARRET en REFUTATION
              FIN SI
              (P,R) = SIMPLIFICATION(P-{(p,q)},R,l->r)
              R = PREUVE-INDUCTIVE(P,R∪{(l->r)},E,>)
       FIN SI
 SINON SI toutes les règles de R sont marquées
       ALORS RETOURNER R; ARRET en SUCCES
       SINON choisir une règle non marquée l->r
             en respectant l'hypothèse d'équité .
             (P,R) = PAIRES-CRITIQUES(l->r,R,E);
             marquer la règle l->r dans R
             R = PREUVE-INDUCTIVE(P,R,E,>)
       FIN SI
 FIN SI
FIN PREUVE-INDUCTIVE
```

La procédure ORIENTATION compare les deux termes p' et q' donnés avec l'ordre donné et s'arrête en échec si c'est impossible.

```
ORIENTATION(p,q,>)
 CAS p > q ALORS RETOURNER p->q
     q > p ALORS RETOURNER q->p
     SINON ARRET en ECHEC
 FIN CAS
FIN ORIENTATION
```

La procédure IND-REDUCTIBLE(t,R_0,E_0) teste si le terme t a la propriété de R_0E_0-réductibilité inductive, Les procédures SIMPLIFICATION et PAIRES-CRITIQUES sont les mêmes que celles utilisées dans la procédure de complétion équationnelle.

La correction de la procédure de preuve inductive se traduit par le théorème suivant:

Théorème 13 : Soit $(\Sigma, R_0 \cup E_0)$ une spécification convergente, R_p un ensemble de règles et E_p un ensemble d'équations (g=d) non effaçantes tel qu'un algorithme complet de $E_0 \cup E_p$-unification est connu et que g et d aient la propriété de R_0E_0-réductibilité inductive. Supposons que la procédure PREUVE-INDUCTIVE est initialisée avec les arguments R_p, R_0, $E_0 \cup E_p$ et <. Chaque équation de $R_p \cup E_p$ est valide dans $I(\Sigma, R_0 \cup E_0)$ si et seulement si PREUVE-INDUCTIVE ne s'arrête pas en échec ou réfutation.

Précisons un peu ce que signifie ce théorème:

a) Si la procédure ne s'arrête pas en échec ou réfutation, soit R_∞ le système de règles retourné. Alors

- R_∞ est Church-Rosser modulo $(E_0 \cup E_p)$ sur les termes clos de $T(\Sigma)$
- les égalités inductives engendrées par $(R_\infty \cup E_0 \cup E_p)$ et $(R_0 \cup R_p \cup E_0 \cup E_p)$ coincident
- chaque équation de $R_p \cup E_p$ est valide dans $I(\Sigma, R_0 \cup E_0)$
- $I(\Sigma, R_0 \cup E_0) = I(\Sigma, R_0 \cup R_p \cup E_0 \cup E_p) = I(\Sigma, R_\infty \cup E_0 \cup E_p)$.

b) Si la procédure s'arrête en réfutation, il existe une équation de $R_p \cup E_p$ qui n'est pas valide dans $I(\Sigma, R_0 \cup E_0)$

c) Réciproquement si une équation de $R_p \cup E_p$ n'est pas valide dans $I(\Sigma, R_0 \cup E_0)$, alors la procédure s'arrête en réfutation ou en échec.

La preuve de ce théorème s'appuie sur le lemme suivant:

Lemme 16 : Un terme clos est en forme normale pour $->R_0E_0$ si et seulement si il est en forme normale pour les systèmes de réécriture équationnelle ultérieurement engendrés par la procédure au cours de la complétion.

Le problème de vérifier la consistance d'un enrichissement par rapport à une spécification de base se différencie de la preuve de la validité d'un ensemble d'équations lorsqu'on ajoute de nouveaux symboles. Reprenons un exemple donné par JP.Jouannaud et E.Kounalis (loc.cit).

Exemple 54 : Soit Σ_b = {0,succ}, R_0 = {succ(succ(0)) -> 0} définissant la spécification de base. On l'enrichit avec un nouveau symbole pred et une équation pred(x) = succ(x). Cet enrichissement est bien consistant et l'ensemble des deux règles

succ(succ(0)) -> 0
pred(x) -> succ(x)

est convergent. Pourtant pred(x) n'a pas la propriété de R_0-réductibilité inductive, et la procédure précédemment décrite termine en réfutation si l'on choisit un ordre sur les termes tel que pred(x) > succ(x). ∇

Pour tester la consistance d'un enrichissement, nous proposons dans ce qui suit un autre point de vue utilisant, dans une situation semblable à celle présentée dans cet exemple, la structuration de l'enrichissement.

5. Spécifications structurées

Nous allons maintenant montrer comment exploiter la notion d'enrichissement d'une spécification dans la procédure de complétion. L'idée est que quand on enrichit une spécification, on introduit au moins un nouveau symbole et des équations qui contiennent ce symbole. Lorsque, pour prouver la consistance de cet enrichissement, on utilise la procédure de complétion, il apparait en général des paires critiques qui ne font intervenir que des symboles de la spécification initiale et il s'agit alors

de prouver que ces équations sont valides. Comme nous supposerons toujours que cette spécification est convergente, il est possible d'utiliser la méthode de preuve inductive que l'on vient de décrire pour cela.

5.1. Spécifications structurées complètes ou convergentes

Le concept de spécification structurée apparait par exemple dans [Wirsing1982]. Dans le cadre des systèmes de réécriture, il est défini par J.L.Remy dans sa thèse (loc.cit).

Définition 87 : Soit $\Sigma_b \cup \Sigma_d$ une partition de la signature Σ. On appelle

- Σ_b-**équation** toute équation g=d dont les deux membres sont des termes de $T(\Sigma_b,X)$.
- Σ_b-**règle** toute Σ_b-équation orientée g->d telle que V(g) contienne V(d).
- Σ_d-**équation** toute équation g=d dont l'un des membres est un terme de $T(\Sigma,X)-T(\Sigma_b,X)$.
- Σ_d-**règle** toute Σ_d-équation orientée g->d dont le membre gauche g est un terme de $T(\Sigma,X)-T(\Sigma_b,X)$ tel que V(g) contienne V(d).

Une Σ_b-équation ou une Σ_d-équation g=d est **non effaçante** si et seulement si V(g)=V(d). ○

Il faut noter qu'avec cette définition, une règle x->t avec t dans $T(\Sigma,X)$ ne peut pas être une Σ_d-règle.

Définition 88 : Une spécification (Σ,A) est une **spécification structurée** de base (Σ_b,A_b) si et seulement si $\Sigma = \Sigma_b \cup \Sigma_d$, $A = A_b \cup A_d$

(1) A_b est un ensemble de Σ_b-équations divisé en un ensemble d'équations non effaçantes E_b et un ensemble de règles R_b.

(2) L'équivalence engendrée par A_b est décidable à l'aide du système de réécriture équationnelle convergent (R_b,E_b).

(3) A_d est un ensemble de Σ_d-équations divisé en un ensemble d'équations non effaçantes E_d et un ensemble de règles R_d.

(4) Soient $R=R_b \cup R_d$ et $E=E_b \cup E_d$. ->RE est E-noethérienne.

De plus

(5) Si tout terme t de $T(\Sigma)$ possède une forme normale $t\downarrow$ dans $T(\Sigma_b)$ pour la relation ->RE, la spécification est dite **structurée complète** de base (Σ_b,A_b).

(5') Si (R,E) est convergent, la spécification est dite **structurée convergente** de base (Σ_b,A_b). ○

Faisons quelques remarques sur cette définition:

- Le seul lien exigé entre la relation de réécriture utilisée dans (Σ_b,A_b) et celle de (Σ,A) est que $->R_bE_b$ est inclus dans ->RE, ce qui est immédiat dès que R_b est inclus dans R et E_b dans E et que la méthode de filtrage utilisée pour chaque règle de R_b est conservée.
- L'hypothèse de terminaison de ->RE implique celle de $->R_bE_b$.
- La condition (5) implique la propriété de complétude suffisante de (Σ,A) par rapport à (Σ_b,A_b). Si un terme a plusieurs formes normales distinctes, il est facile de vérifier qu'elles sont toutes dans $T(\Sigma_b)$ dès que l'une d'entre elles y est: sinon en effet cette forme normale serait elle-même un terme de $T(\Sigma)$ et à cause de la condition (5) serait réductible.

La propriété de complétude suffisante étant indécidable, des conditions suffisantes et les tests correspondants ont été proposés par [Huet1982,Bidoit1981,Thiel1984,Kounalis1985a]. Beaucoup d'entre eux sont bâtis sur la vérification de la condition suivante équivalente à la condition (5): tout terme $t=f(t_1,\ldots,t_n)$ tel que f appartient à Σ_d et $t_1,\ldots,t_n$ appartiennent à $T(\Sigma_b)$ est RE-réductible.

Il est important de noter que si la spécification est complète, tout terme t de $T(\Sigma,X)-T(\Sigma_b,X)$ a la propriété de RE-réductibilité inductive. Sinon il existerait une substitution close σ et un terme clos $\sigma(t)$ contenant un symbole de Σ_d qui est RE-irréductible. Ce qui contredit la complétude suffisante.

La notion de spécification structurée est illustrée par l'exemple de la spécification suivante des entiers relatifs:

Exemple 55 : Les entiers relatifs

$S = \{int\}$

Σ_b:
```
0 : → int
succ : int → int
pred : int → int
```

Σ_d:
```
opp  : int, int → int
plus : int, int → int
```

R_b:
```
succ(pred(x)) -> x
pred(succ(x)) -> x
```

$E_b = \emptyset$

R_d:
```
opp(0) -> 0
opp(succ(x)) -> pred(opp(x))
opp(pred(x)) -> succ(opp(x))
plus(0,y) -> y
plus(succ(x),y) -> succ(plus(x,y))
plus(pred(x),y) -> pred(plus(x,y))
```

E_d:
```
plus(x,y) = plus(y,x)
plus(plus(x,y),z) = plus(x,plus(y,z))
```

∇

Il est intéressant de noter tout de suite quelques propriétés simples qui découlent des conditions syntaxiques imposées aux règles et équations d'une spécification structurée:

Lemme 17 : Soit (Σ,A) une spécification satisfaisant les conditions (1) et (3) de la définition précédente. Alors pour tous termes t et t' tels que t appartient à $T(\Sigma_b,X)$:

a) t et t' E_b-équivalents implique t' appartient à $T(\Sigma_b,X)$

b) $t \text{ -*->}_{R_b E_b} t'$ implique t' appartient à $T(\Sigma_b,X)$

c) $t \sim^{E} t'$ implique t et t' E_b-équivalents

d) $t \text{ -*->}_{RE} t'$ implique $t \text{ -*->}_{R_b E_b} t'$.

Preuve: a) se prouve par induction sur la longueur de la plus petite preuve de E_b-équivalence de t et t'.

b) se prouve par induction sur la longueur de la dérivation.

c) et d) se déduisent du fait que si t est dans $T(\Sigma_b,X)$ ni les règles de R_d ni les équations de E_d ne peuvent s'appliquer. □

La relation qui existe entre une spécification structurée de base BSPEC et un enrichissement consistant par rapport à BSPEC repose sur la propriété de convergence. C'est ce qu'exprime le théorème suivant:

Théorème 14 : Soit SPEC=$(\Sigma,R\cup E)$ une spécification structurée de base BSPEC=$(\Sigma_b,R_b\cup E_b)$ telle que (R,E) soit un système de réécriture équationnelle convergent sur les termes clos de $T(\Sigma)$. Alors SPEC est consistante par rapport à BSPEC.

Preuve: Puisque (R,E) est convergent sur les termes clos, pour tous termes

t et t' $(R\cup E)$-équivalents, il existe des termes t_1 et t'_1 tels que $t \text{ -*->}_{RE} t_1$, $t' \text{ -*->}_{RE} t'_1$ et $t_1 \sim^{E} t'_1$.

Si t et t' de plus sont dans $T(\Sigma_b)$, d'après le lemme précédent:

$t \text{ -*->}_{R_b E_b} t_1$, $t' \text{ -*->}_{R_b E_b} t'_1$ et t_1 et t'_1 sont E_b-équivalents.

Donc t et t' sont $(R_b\cup E_b)$-équivalents, ce qui prouve la consistance. □

Remarquons que la propriété de consistance d'une spécification structurée $(\Sigma,R\cup E)$ par rapport à sa base ne nécessite ni la complétude suffisante ni la convergence de $(\Sigma,R\cup E)$.

Dans ce qui suit, nous prendrons toujours en compte à la fois des spécifications complètes et des spécifications convergentes, afin de mettre en évidence que les approches décrites dans le paragraphe 3 sont des cas particuliers de la nôtre.

5.2. Preuve de validité et consistance dans une spécification structurée

Nous allons maintenant spécialiser les lemmes 9 et 10 de validité aux spécifications structurées.

Théorème 15 : (Théorème de validité) Soient $(\Sigma,R\cup E)$ une spécification structurée de base $(\Sigma_b,R_b\cup E_b)$, et $(\Sigma,R\cup R_p\cup E\cup E_p)$ un enrichissement sans nouvelle sorte de $(\Sigma,R\cup E)$ par un ensemble R_p de Σ_d-règles et un ensemble E_p de Σ_d-équations non effaçantes. Supposons de plus que si $R_p\cup E_p$ est non vide, l'une des conditions suivantes est remplie:

(cas 1) $(\Sigma,R\cup E)$ est complète par rapport à $(\Sigma_b,R_b\cup E_b)$

(cas 2) $(\Sigma,R\cup E)$ est convergente, R_p est un ensemble de règles g->d telles que g ait la propriété de (R,E)-réductibilité inductive, E_p est un ensemble

d'équations g=d telles que g et d aient la propriété de (R,E)-réductibilité inductive.

Si il existe un système de réécriture R_∞ tel que

- R_∞ est Church-Rosser modulo EUE_p sur les termes clos de $T(\Sigma)$
- Les égalités engendrées par $(R_\infty UEUE_p)$ et (RUR_pUEUE_p) coincident sur les termes clos de $T(\Sigma)$
- $(\Sigma,R_\infty UEUE_p)$ est une spécification structurée de base (Σ_b,R_bUE_b).

Alors

a) toute équation de R_pUE_p est valide dans $I(\Sigma,RUE)$

b) (Σ,RUE) est consistante par rapport à (Σ_b,R_bUE_b)

c) Si (Σ,RUE) est complète par rapport à (Σ_b,R_bUE_b), alors $(\Sigma,R_\infty UEUE_p)$ est complète par rapport à (Σ_b,R_bUE_b).

Preuve: (cas 1)

Puisque R_∞ est Church-Rosser modulo (EUE_p), $(\Sigma,R_\infty UEUE_p)$ est consistante par rapport à (Σ_b,R_bUE_b) par le théorème 14, et donc (Σ,RUR_pUEUE_p) l'est aussi. La spécification structurée (Σ,RUE) de base (Σ_b,R_bUE_b) est suffisamment complète par rapport à (Σ_b,R_bUE_b) et le lemme 10 (2eme lemme de validité) s'applique à (Σ_b,R_bUE_b), (Σ,RUE) et (Σ,RUR_pUEUE_p).

(cas 2)

a) Par application du théorème 12 à (Σ,RUE), R_p et E_p.

b) se déduit du théorème 14. □

Le système R_∞ est construit par complétion en utilisant la procédure de complétion équationnelle. La contrainte supplémentaire à prendre en compte est de type syntaxique: puisque $(\Sigma,R_\infty UEUE_p)$ doit être une

spécification structurée de base (Σ_b,R_bUE_b), les règles ajoutées dans (Σ,RUE) doivent être des Σ_d-règles. La procédure d'orientation d'une paire critique doit donc prendre en compte cette contrainte.

Le problème suivant est de traiter les équations dont les deux membres appartiennent à $T(\Sigma_b,X)$. Si une telle équation $(t=t')$ est valide dans $I(\Sigma_b,R_bUE_b)$, l'égalité engendrée par $R_bUE_bU\{t=t'\}$ sur $T(\Sigma_b)$ coincide avec celle engendrée par R_bUE_b. La congruence sur les termes clos de la spécification de base n'étant pas modifiée, on peut considérer à la place de BSPEC une spécification de base $BSPEC_\infty$ dans laquelle on a ajouté des lemmes, c'est-à-dire des équations valides dans $I(\Sigma_b,R_bUE_b)$, nécessaires pour prouver la validité de R_pUE_p. Il faut donc supposer que la validité d'une équation est décidable dans la spécification de base. Jusqu'à récemment, on considérait que c'était une hypothèse très restrictive, bien que la plupart des techniques proposées pour faire des preuves de validité aient supposé des conditions impliquant cette propriété:

- dans l'approche de Musser et Goguen, la définition d'un enrichissement égalitaire permet de se ramener au cas où la spécification de base est toujours le type booléen.
- dans l'approche de Huet et Hullot, la spécification de base est celle des constructeurs et puisqu'il n'y a aucune relation entre eux, l'égalité inductive se ramène à un test d'égalité syntaxique.
- dans l'approche de Paul, l'égalité inductive est supposée coincider exactement avec l'égalité équationnelle et donc se teste par des techniques de réécriture.

Dans le cas général, puisque la spécification de base est convergente, la méthode décrite précédemment et testant la réductibilité inductive peut

être utilisée pour prouver des lemmes dans la spécification de base.

5.3. Complétion inductive dans une spécification structurée

Nous proposons maintenant une procédure de complétion qui, à partir d'une spécification structurée, complète ou convergente, $(\Sigma, R \cup E)$ de base $(\Sigma_b, R_b \cup E_b)$, et d'assertions à prouver $R_p \cup E_p$, construit une spécification structurée convergente.

Nous supposons donc que la signature Σ est partitionnée en deux signatures Σ_b et Σ_d. L'ensemble d'équations non orientées E est partitionné en deux ensembles E_b et E_d. Nous supposons aussi que pour tout ensemble de règles R, nous disposons de fonctions B et D telles que B(R) est l'ensemble des Σ_b-règles de R, D(R) est l'ensemble des Σ_d-règles de R. Il est nécessaire de définir B et D comme des fonctions puisqu'au cours d'une complétion, l'ensemble des règles peut varier.

Comme dans la procédure de complétion équationnelle, la procédure de complétion inductive a pour arguments un ensemble de paires de termes P, un ensemble de règles R, un ensemble constant d'équations E et un ordre de E-réduction noethérien >. Nous avons ajouté un nouvel argument qui est un booléen ir qui indique si l'on doit tester la réductibilité inductive des membres gauches des règles introduites. Ce booléen est initialisé à vrai si l'on fait une preuve de validité et à faux si l'on fait une preuve de consistance.

Chaque règle possède une étiquette qui permet de retrouver l'ordre dans lequel elles ont été introduites. Cela permet d'implanter facilement l'hypothèse d'équité nécessaire au moment du choix d'une règle pour le calcul des paires critiques. Il faut supposer ici que B(R) et D(R) ont leur

règles étiquetées par des ensembles distincts isomorphes à l'ensemble des entiers naturels. Pour ne pas alourdir les notations, nous ne mentionnerons pas les étiquettes dans l'algorithme. Chaque règle est également marquée si et seulement si ses paires critiques avec toutes les règles d'étiquette inférieure et toutes les équations de E ont été calculées.

```
Initialisation:
P = Rp, R = Rb ∪ Rd, E = Eb ∪ Ed ∪ Ep.
ir = vrai si Rp ∪ Ep ≠ ø faux sinon

E-COMPLETION INDUCTIVE(P,R,E,>,ir)
 SI P n'est pas vide
 ALORS choisir une paire (p,q) dans P; p'=p↓; q'=q↓;
       CAS p' ~E d q' ALORS R=E-COMPLETION INDUCTIVE(P-{(p,q)},R,E,>,ir)
           p' et q' sont dans T(Σb,X) ALORS
                 RES=PREUVE-INDUCTIVE({(p',q')},B(R),Eb,>)
                 SI p'=q' n'est pas valide dans I(Σb,B(R)∪Eb)
                 ALORS ARRET en REFUTATION
                 FIN SI
                 R=E-COMPLETION INDUCTIVE(P-{(p,q)},R-B(R)∪RES,E,>,ir)
           SINON l->r = ORIENTATION(p',q',>,ir)
                 (P,R)=SIMPLIFICATION(P-{(p,q)},R,E,l->r)
                 R=E-COMPLETION INDUCTIVE(P,R∪{(l->r)},E,>,ir)
       FIN CAS
 SINON SI toutes les règles de D(R) sont marquées
       ALORS RETOURNER R; ARRET en SUCCES
       SINON choisir une règle non marquée l->r en respectant
             l'hypothèse d'équité
             (P,R)=PAIRES-CRITIQUES(l->r,R,E);
             marquer la règle l->r dans R
             R=E-COMPLETION INDUCTIVE(P,R,E,>,ir)
       FIN SI
 FIN SI
FIN E-COMPLETION INDUCTIVE
```

La procédure ORIENTATION tient compte en plus de l'ordre donné, des conditions syntaxiques imposées pour construire une spécification structurée. De plus elle teste, quand on fait une preuve de validité, si le membre gauche de la règle à ajouter a la propriété de réductibilité inductive dans le système de réécriture équationnelle de départ. Ce test rend

toujours la valeur vrai si la spécification d'origine $(\Sigma_b \cup \Sigma_d, R_b \cup R_d \cup E_b \cup E_d)$ est suffisamment complète par rapport à sa base. Il est donc inutile si l'on teste la complétude avant de lancer la complétion. Par contre, si l'on sait que $(\Sigma_b \cup \Sigma_d, R_b \cup R_d \cup E_b \cup E_d)$ est convergente mais non suffisamment complète, le test doit être fait pour s'assurer que la règle introduite est valide.

```
ORIENTATION(p,q,>,ir)
 CAS (p∈T(Σ,X)-T(Σb,X) AND q∈T(Σb,X) AND p>q)
       ALORS l := p, r := q
     (q∈T(Σ,X)-T(Σb,X) AND p∈T(Σb,X) AND q>p)
       ALORS l := q, r := p
     (p,q∈T(Σ,X)-T(Σb,X) AND p>q)
       ALORS l := p, r := q
     (p,q∈T(Σ,X)-T(Σb,X) AND q>p)
       ALORS l := q, r := p
     SINON ARRET en ECHEC
 FIN CAS
 SI ir ALORS
    SI NON(IND-REDUCTIBLE(l,RbURd,EbUEd))
    ALORS ARRET en REFUTATION
    FIN SI
 FIN SI
 RETOURNER l->r
 FIN SI
FIN ORIENTATION
```

L'appel de la procédure PREUVE-INDUCTIVE($\{(p',q')\}$,B(R),E_b,>)

- soit retourne un système de réécriture équationnelle (RES,E_b) convergent et tel que les égalités inductives engendrées par (RES$\cup E_b$) et ($R_b \cup E_b$) coincident,
- soit termine en réfutation si l'équation p'=q' n'est pas valide dans $I(\Sigma_b, R_b \cup E_b)$,
- soit termine en échec si la procédure de preuve utilisée termine de cette façon.

Cette procédure de preuve inductive d'une équation peut être

- soit un simple test d'identité syntaxique, comme dans les méthodes de

Huet et Hullot ou de Goguen,

- soit une preuve d'égalité équationnelle, c'est-à-dire un test d'égalité des formes normales, comme dans la méthode de Paul où l'on suppose que dans la spécification de base, l'égalité inductive coincide avec l'égalité équationnelle,
- soit la procédure de preuve inductive que nous avons décrite précédemmment, puisque la spécification de base est convergente. Dans ce cas, la similitude des deux procédures conduit à remplacer simplement l'appel RES = PREUVE-INDUCTIVE($\{(p',q')\}$,B(R),E_b,>) par RES = E-COMPLETION INDUCTIVE($\{(p',q')\}$,B(R),E_b,>,vrai). Cette remarque sera plus largement exploitée dans le paragraphe suivant.

Il est à noter que l'arrêt en échec de la procédure E-COMPLETION INDUCTIVE peut provenir soit de l'arrêt en échec de la procédure d'orientation, soit de l'arrêt en échec de la procédure PREUVE-INDUCTIVE qui sont propagés dans la procédure appelante. La non terminaison de la procédure E-COMPLETION INDUCTIVE peut aussi provenir de celle de PREUVE-INDUCTIVE sur les équations de la spécification de base.

Les procédures SIMPLIFICATION et PAIRES-CRITIQUES sont les mêmes que celles utilisées dans la procédure de complétion équationnelle. Elles peuvent cependant être implantées plus efficacement en tenant compte des faits suivants:

- une Σ_b-règle ne peut pas être simplifiable par une Σ_d-règle, donc seules les règles de D(R) sont à simplifier lorsqu'on introduit une Σ_d-règle.
- une Σ_d-règle ne peut pas être superposée dans une Σ_b-règle, ni dans une Σ_b-équation de E. La procédure PAIRES-CRITIQUES calculera donc des superpositions des règles de B(R) et des équations de E_b dans la règle considérée,

ainsi que des superpositions des règles de D(R) et des équations de E_d dans la règle considérée et réciproquement.

La procédure E-COMPLETION INDUCTIVE peut être utilisée avec deux objectifs différents:

● Pour tester la consistance d'une spécification structurée construite en enrichissant une spécification (Σ_b,R_b,E_b) par un ensemble de Σ_d-règles R_d et un ensemble de Σ_d-équations non effaçantes E_d, la procédure E-COMPLETION INDUCTIVE est initialisée avec R_d comme ensemble de règles non marquées, R_b comme ensemble de règles marquées, et $E=E_b \cup E_d$ comme ensemble d'équations. Si elle ne s'arrête pas en échec ou réfutation, $(\Sigma,R_\infty \cup E)$ est une spécification structurée convergente de base $(\Sigma_b,B(R_\infty)\cup E_b)$. De plus les équivalences engendrées par $(B(R_\infty)\cup E_b)$ et $(R_b \cup E_b)$ sur $T(\Sigma_b)$ coincident et les équivalences engendrées par $(R_\infty \cup E)$ et $(R \cup E)$ sur $T(\Sigma)$ coincident également. On en déduit donc la consistance de $(\Sigma,R\cup E)$ par rapport à $(\Sigma_b,R_b\cup E_b)$ par le théorème 14.

● Pour faire des preuves de validité d'un ensemble $R_p \cup E_p$ d'équations effaçantes ou non, dans l'algèbre initiale d'une spécification $(\Sigma_b\cup\Sigma_d,R_b\cup R_d\cup E_b\cup E_d)$ structurée de base $(\Sigma_b,R_b\cup E_b)$, on commence par tester soit la complétude suffisante de $(\Sigma_b\cup\Sigma_d,R_b\cup R_d\cup E_b\cup E_d)$ par rapport à $(\Sigma_b,R_b\cup E_b)$, soit la convergence de $(\Sigma_b\cup\Sigma_d,R_b\cup R_d\cup E_b\cup E_d)$ et la $(R_b\cup R_d,E_b\cup E_d)$-réductibilité inductive des deux membres de toute équation de E_p. Puis la procédure E-COMPLETION INDUCTIVE est initialisée avec R_p comme ensemble de paires, $R_b\cup R_d$ comme ensemble de règles marquées et $E_b\cup E_d\cup E_p$ comme ensemble d'équations. Si la procédure ne s'arrête pas en échec ou réfutation, les équations de $R_p\cup E_p$ sont valides dans $I(\Sigma,R_b\cup R_d\cup E_b\cup E_d)$. Il est à noter qu'il n'est pas nécessaire dans ce deuxième cas de tester la

consistance de $(\Sigma,R_b\cup R_d\cup E_b\cup E_d)$ par rapport à $(\Sigma_b,R_b\cup E_b)$, mais cette propriété est obtenue comme conséquence de la méthode. Remarquons que la procédure permet de prouver des assertions qui ne sont pas orientables en règles de réécriture, par exemple la commutativité d'un opérateur ou tout axiome permutatif. Il suffit alors d'initialiser E_p avec cette équation, à condition toutefois de disposer d'un algorithme d'unification pour $E_b\cup E_d\cup E_p$.

Exemple 56 : Appliquée à la spécification des entiers relatifs donnée dans l'exemple précédent, la procédure termine, établissant ainsi la consistance de la spécification. On obtient de plus que $(\Sigma_b\cup\Sigma_d,R_b\cup R_d\cup E_b\cup E_d)$ est une spécification structurée convergente de base $(\Sigma_b,R_b\cup E_b)$.

Si on veut prouver dans cette spécification les assertions:

```
opp(opp(x)) = x
opp(plus(x,y)) = plus(opp(x),opp(y)),
```

on initialise R_p avec ces deux équations. La procédure termine, prouvant ainsi leur validité.

Supposons que l'on veuille enrichir cette spécification en introduisant la multiplication, décrite par l'opérateur mult. Posons maintenant

Σ_b = {0,succ,pred,opp,plus}

Σ_d = {mult} avec mult : int, int → int

R_d:
```
mult(0,x) -> 0
mult(pred(x),y) -> plus(opp(y),mult(x,y))
mult(succ(x),y) -> plus(y,mult(x,y))
mult(plus(x,y),z) -> plus(mult(x,z),mult(y,z))
mult(opp(x),y) -> opp(mult(x,y))
```

E_d:
```
mult(mult(x,y),z) = mult(x,mult(y,z))
mult(x,y) = mult(y,x)
```

La procédure de complétion inductive engendre et prouve :

```
opp(opp(x)) -> x
opp(plus(x,y)) -> plus(opp(x),opp(y))
plus(x,opp(x)) -> 0
```

et termine ensuite en succès.

Nous avons ici défini mult de façon complète sur Σ_b. Cependant, on obtient une spécification structurée convergente si l'on se contente de donner comme règles et équations:

R_d: mult(0,x) -> 0
mult(pred(x),y) -> plus(opp(y),mult(x,y))
mult(succ(x),y) -> plus(y,mult(x,y))

E_d: mult(mult(x,y),z) = mult(x,mult(y,z))
mult(x,y) = mult(y,x)

Il est alors possible de prouver la validité des équations suivantes orientées en règles:

```
mult(plus(x,y),z) -> plus(mult(x,z),mult(y,z))
mult(opp(x),y) -> opp(mult(x,y))
```

Les mêmes lemmes sont engendrés.

Donnons un exemple de preuve de non validité: considérons l'équation orientée en règle: mult(x,x) -> x. Sa superposition avec

```
mult(succ(x),y) -> plus(y,mult(x,y))
```

produit la règle

```
succ(plus(x,plus(x,x))) -> succ(x).
```

C'est une règle de la spécification de base qui se superpose alors avec

```
plus(succ(x),y)->succ(plus(x,y))
```

et produit

```
succ(succ(succ(succ(plus(x,x))))) -> succ(succ(x)).
```


Cette règle à son tour se superpose avec plus(0,x) -> x pour donner

```
succ(succ(succ(succ(0)))) -> succ(succ(0)).
```

Cette dernière règle n'a pas la propriété de réductibilité inductive, et la procédure termine donc en réfutation. ∇

5.4. Preuve de la procédure

La correction de la procédure est exprimée par le théorème de suivant:

Théorème 16 : Soit $\Sigma = \Sigma_b \cup \Sigma_d$ une signature, $SPEC=(\Sigma,R_b \cup R_d \cup E_b \cup E_d)$ une spécification structurée de base $BSPEC=(\Sigma_b,R_b \cup E_b)$, R_p un ensemble de Σ_d-règles et E_p un ensemble de Σ_d-équations non effaçantes telle qu'un algorithme complet de $(E_b \cup E_d \cup E_p)$-unification est connu. Supposons de plus que si $R_p \cup E_p$ est non vide, l'une des conditions suivantes est remplie:

(cas 1) $(\Sigma,R_b \cup R_d \cup E_b \cup E_d)$ est complète par rapport à $(\Sigma_b,R_b \cup E_b)$

(cas 2) $(\Sigma,R_b \cup R_d \cup E_b \cup E_d)$ est convergente, R_p est un ensemble de règles g->d telles que g ait la propriété de $(R_b \cup R_d,E_b \cup E_d)$-réductibilité inductive, E_p est un ensemble d'équations g=d telles que g et d aient la propriété de $(R_b \cup R_d,E_b \cup E_d)$-réductibilité inductive.

Supposons enfin que la procédure E-COMPLETION INDUCTIVE est initialisée avec les arguments R_p, $R_b \cup R_d$, $E_b \cup E_d \cup E_p$ et <.

Chaque équation de $R_p \cup E_p$ est valide dans $I(\Sigma,R_0 \cup E_0)$ et SPEC est consistante par rapport à BSPEC si et seulement si COMPLETION-INDUCTIVE ne s'arrête pas en échec ou réfutation.

Il faut noter qu'on cherche à assurer la propriété de Church-Rosser sur les termes sans variables uniquement, ce qui autorise à ajouter des équations qui ne sont pas des conséquences équationnelles mais des

théorèmes inductifs. De même la coincidence des congruences engendrées par le système initial et le système final est assurée sur les termes clos.

Reprenons les mêmes notations que pour la preuve de la procédure de complétion équationnelle:

- P_i et R_i sont les valeurs des arguments P et R au ieme appel récursif,
- R+ est la réunion des ensembles R_i, c'est-à-dire l'ensemble de toutes les règles engendrées au cours de la procédure.

Un lemme préliminaire établit un résultat utile sur les arguments de la procédure au cours des différents appels récursifs. Il signifie que la congruence engendrée sur les termes clos par l'ensemble des arguments n'est pas modifiée.

Lemme 18 : Sur les termes clos de $T(\Sigma)$, pour tout i, l'équivalence engendrée par $(P_{i+1} \cup R_{i+1} \cup E)$ est incluse dans l'équivalence engendrée par $(P_i \cup R_i \cup E)$.

Sur les termes clos de $T(\Sigma_b)$, pour tout i, les équivalences engendrées par $(B(R_{i+1}) \cup E_b)$ et $(B(R_i) \cup E_b)$ coincident.

Quand on teste la réductibilité inductive, pour tout terme clos t de $T(\Sigma)$, pour tout i, t est en $(R_b \cup R_d)E$-forme normale si et seulement si il est en $R_i E$-forme normale.

Preuve: En examinant successivement chaque appel récursif de la procédure:

- (1) P_{i+1} est inclus dans P_i et $R_{i+1}=R_i$. Le résultat est évident.
- (2) $P_{i+1} = P_i - \{(p,q)\}$ et $R_{i+1} = RES \cup (R_i - B(R_i))$. Les équivalences engendrées par $(RES \cup E_b)$, $(B(R_i) \cup \{p,q\} \cup E_b)$ et $(B(R_i) \cup E_b)$ coincident puisque p=q est un théorème inductif de $(B(R_i) \cup E_b)$ et par hypothèse sur la procédure PREUVE-INDUCTIVE.

Donc l'équivalence engendrée par $(R_{i+1} \cup E)$ coincide avec celle engendrée par $(R_i \cup E)$.

D'autre part, $B(R_{i+1})=RES$ et les équivalences engendrées par $(RES \cup E_b)$ et $(B(R_i) \cup E_b)$ coincident.

- (3) (P_{i+1},R_{i+1})=SIMPLIFICATION$(P_i-\{(p,q)\},R_i,l->r)$ avec $l->r$ = ORIENTATION(p',q',>). La preuve que l'égalité engendrée par $(P_{i+1} \cup R_{i+1} \cup E)$ est incluse dans celle engendrée par $(P_i \cup R_i \cup E)$ est la même que dans la preuve de la procédure de complétion équationnelle. Comme d'autre part la règle $l->r$ comporte dans ce cas au moins un symbole de Σ_d, elle est ajoutée à $D(R_i)$ et $B(R_{i+1}) = B(R_i)$.
- (4) (P_{i+1},R_{i+1})=PAIRES-CRITIQUES$(l->r,R_i,E)$. On prouve comme pour la procédure de complétion équationnelle que l'équivalence engendrée par $(P_{i+1} \cup R_{i+1} \cup E)$ est incluse dans celle engendrée par $(P_i \cup R_i \cup E)$.

D'autre part, les règles ajoutées à cette étape pour assurer éventuellement la cohérence sont calculées à partir de paires critiques entre la règle $l->r$ de $D(R_i)$ et d'autres règles ou équations. Par construction, les extensions ajoutées sont des Σ_d-règles. Donc $B(R_{i+1}) = B(R_i)$.

Enfin le résultat sur les formes normales résulte du lemme 16. □

On en déduit que l'équivalence engendrée sur $T(\Sigma)$ par $(P_i \cup R_i \cup E)$ est incluse dans celle engendrée par $(P_0 \cup R_0 \cup E) = (R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ et que sur les termes clos de $T(\Sigma_b)$ pour tout i, les équivalences engendrées par $(B(R_i) \cup E_b)$ et $(R_b \cup E_b)$ coincident.

Corollaire 5 : Sur $T(\Sigma)$, l'équivalence engendrée par $(R \cup E)$ est incluse dans celle engendrée par $(R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ et sur $T(\Sigma_b)$, les équivalences engendrées par $(B(R+) \cup E_b)$ et $(R_b \cup E_b)$ coincident.
Si t est un terme clos de $T(\Sigma)$, t est en forme normale pour $\to(R_b \cup R_d \cup R_p)(E_b \cup E_d \cup E_p)$ si et seulement si t est en forme normale pour $\to R+(E_b \cup E_d \cup E_p)$.

Les lemmes suivants établissent les différents points du théorème de correction.

Lemme 19 : R_∞ est Church-Rosser modulo $(E_b \cup E_d \cup E_p)$ sur les termes clos de $T(\Sigma)$ et les égalités inductives engendrées par $(R_\infty \cup E_b \cup E_d \cup E_p)$, $(R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ et $(R+ \cup E_b \cup E_d \cup E_p)$ coincident.

Preuve: On utilise les mêmes techniques que dans la preuve de la procédure de complétion équationnelle. Les identités des égalités inductives se déduisent de l'identité sur les termes clos des congruences engendrées par $(R_\infty \cup E_b \cup E_d \cup E_p)$, $(R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ et $(R+ \cup E_b \cup E_d \cup E_p)$. □

Corollaire 6 : Sur $T(\Sigma_b)$, les égalités engendrées par $(B(R_\infty) \cup E_b)$ et $(R_b \cup E_b)$ coincident.

Lemme 20 : $(\Sigma, R_\infty \cup E)$ est une spécification structurée convergente de base $(\Sigma_b, R_b \cup E_b)$. $(\Sigma, R_\infty \cup E)$ est suffisamment complète par rapport à $(\Sigma_b, R_b \cup E_b)$ si $(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$ est suffisamment complète par rapport à $(\Sigma_b, R_b \cup E_b)$.

Preuve: - Par construction, $(\Sigma, R_\infty \cup E)$ satisfait les conditions (1) et (3) de la définition d'une spécification structurée.

- $B(R_\infty)$ est Church-Rosser modulo E_b sur les termes de $T(\Sigma_b)$: en effet si t et t' sont deux termes de $T(\Sigma_b)$ tels que t et t' sont égaux modulo $(B(R_\infty) \cup E_b)$, alors t et t' sont égaux modulo $(R_\infty \cup E)$ et il existe t_1 et t'_1 E-égaux tels que

$t \to^*_{R_\infty E} t_1$ et $t' \to^*_{R_\infty E} t'_1$.

Par le lemme 17,

$t \to^*_{B(R_\infty) E_b} t_1$ et $t' \to^*_{B(R_\infty) E_b} t'_1$ et t_1 et t'_1 sont égaux modulo E_b.

- $\to_{R_\infty E}$ est E-noethérienne

- Supposons de plus que $(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$ est suffisamment complète par rapport à $(\Sigma_b, R_b \cup E_b)$. Soient t un terme de $T(\Sigma)$, t_1 une forme normale de t pour $\to(R_b \cup R_d)(E_b \cup E_d)$ et t_2 une forme normale de t pour $\to_{R_\infty E}$. Puisque SPEC est une spécification structurée, t_1 appartient à $T(\Sigma_b)$. Comme t_1 est égal à t modulo $(R_b \cup R_d \cup E_b \cup E_d)$, que t est égal à t_2 modulo $(R_\infty \cup E)$ et que l'équivalence engendrée par $(R_b \cup R_d \cup E_b \cup E_d)$ est incluse dans celle engendrée par $(R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ ou par $(R_\infty \cup E)$, t_1 et t_2 sont égaux modulo $(R_\infty \cup E)$. Par propriété de Church-Rosser, $t_1 \to^*_{R_\infty E} t'_2 \overset{E}{\leftarrow} t_2$. Par le lemme 10, $t_1 \to^*_{B(R_\infty) E_b} t'_2$, t'_2 est E_b-équivalent à t_2 et t_2 est dans $T(\Sigma_b)$.

- On déduit du fait que les équivalences engendrées par $(R_b \cup E_b)$ et $(B(R_\infty) \cup E_b)$ coincident sur $T(\Sigma_b)$, que $(\Sigma, R_\infty \cup E)$ est une spécification structurée convergente (éventuellement suffisamment complète) de base $(\Sigma_b, R_b \cup E_b)$. □

Lemme 21 : Si la procédure ne s'arrête pas en échec ou réfutation, alors

- SPEC=$(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$ est consistante par rapport à BSPEC=$(\Sigma_b, R_b \cup E_b)$
- toute équation de $R_p \cup E_p$ est valide dans $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$
- $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d) = I(\Sigma, R_b \cup R_d \cup E_b \cup E_d \cup E_p) = I(\Sigma, R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$.

Preuve: On déduit du lemme précédent et du théorème 15 que SPEC est consistante par rapport à BSPEC et que toute équation de $R_p \cup E_p$ est valide dans $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$. L'identité des algèbres initiales est facilement obtenue à partir du lemme 9 (1er lemme de validité). □

Considérons maintenant le cas d'arrêt en réfutation de la procédure.

Lemme 22 : Si la procédure s'arrête en réfutation,

- soit, dans le cas où l'on fait une preuve de validité, il existe une équation de $R_p \cup E_p$ qui n'est pas valide dans $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$,
- soit, si l'on fait une preuve de consistance, SPEC=$(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$ n'est pas consistante par rapport à BSPEC=$(\Sigma_b, R_b \cup E_b)$.

Preuve: Posons SPEC'=$(\Sigma, R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$. L'arrêt en réfutation de la procédure peut provenir de deux raisons:

- cas 1: arrêt en réfutation dans la spécification de base. Il existe alors une paire (p,q) de P_i telle p et q sont tous deux dans $T(\Sigma_b, X)$ mais ne sont pas E-égaux. De plus, p=q n'est pas valide dans $I(\Sigma_b, R_b \cup E_b)$. Alors il existe une substitution σ à valeurs dans $T(\Sigma_b)$, telle que $\sigma(p)$ n'est pas $(B(R_i) \cup E_b)$-équivalent à $\sigma(q)$, donc $\sigma(p)$ n'est pas $(R_b \cup E_b)$-équivalent à $\sigma(q)$. Les deux termes clos de $T(\Sigma_b)$, $\sigma(p)$ et $\sigma(q)$, sont donc égaux modulo (P_i) et non égaux modulo $(R_b \cup E_b)$. Ce qui prouve que SPEC$_i$=$(\Sigma, P_i \cup R_i \cup E)$ n'est pas consistante par rapport à BSPEC.

Mais d'autre part, sur $T(\Sigma_b)$, l'équivalence engendrée par $(P_i \cup R_i \cup E)$ est incluse dans celle engendrée par $(P_0 \cup R_0 \cup E) = (R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ d'après le lemme 18. Donc si SPEC$_i$ n'est pas consistante par rapport à BSPEC, il est clair que SPEC' ne l'est pas non plus.

--- Si l'on fait une preuve de consistance, $R_p \cup E_p$ est vide, SPEC'=SPEC et SPEC n'est donc pas consistante par rapport à BSPEC.

--- Dans le cas où l'on fait une preuve de validité, soit SPEC est suffisamment complète par rapport à BSPEC et il suffit alors d'appliquer le lemme 10 (2eme lemme de validité) à BSPEC, SPEC et SPEC', soit SPEC est convergente. Dans ce cas, $\sigma(p)$ et $\sigma(q)$ ne peuvent être équivalents modulo (R∪E) sinon ils le seraient modulo $(R_b \cup E_b)$.

- cas 2: dans le cas où l'on fait une preuve de validité, l'arrêt en réfutation peut aussi provenir de l'existence d'une règle l->r telle que l n'a pas la propriété de $(R_b \cup R_d, E_b \cup E_d)$-réductibilité inductive. On déduit du théorème 13 que $(\Sigma, R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ n'est pas consistante par rapport à $(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$. □

Il reste à prouver une réciproque.

Lemme 23 : Si, lorsqu'on fait une preuve de validité, une équation de $R_p \cup E_p$ n'est pas valide dans $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$ ou si, lorsqu'on fait une preuve de consistance, SPEC n'est pas consistante par rapport à BSPEC, alors la procédure s'arrête en réfutation ou en échec.

Preuve: Posons SPEC'=$(\Sigma, R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$.

- Supposons que l'on fasse une preuve de validité et qu'il existe

une équation de $R_p \cup E_p$ non valide dans $I(\Sigma, R_b \cup R_d \cup E_b \cup E_d)$. Alors par application du lemme 9, on obtient que SPEC' n'est pas consistante par rapport à SPEC.

-- Quand SPEC est convergente, par le théorème 13, on obtient que la procédure s'arrête en échec ou réfutation.

-- Quand SPEC est suffisamment complète par rapport à BSPEC, en appliquant le lemme 10 (2eme lemme de validité) à BSPEC, SPEC et SPEC', on obtient que SPEC' n'est pas consistante par rapport à BSPEC.

- Si l'on fait une preuve de consistance et que SPEC n'est pas consistante par rapport à BSPEC, alors SPEC'=SPEC et on est encore dans la même situation.

Il existe donc deux termes t et t' tels que t et t' soient égaux modulo $(R_b \cup R_d \cup R_p \cup E_b \cup E_d \cup E_p)$ et non égaux modulo $(R_b \cup E_b)$. Si la procédure termine en succès ou engendre une infinité de règles, il existe une itération i et des termes t_0 et t'_0 tels que

$t \rightarrow^*_{R_i E} t_0$, $t' \rightarrow^*_{R_i E} t'_0$ et $t_0 \sim^E t'_0$.

Mais comme t et t' sont dans $T(\Sigma_b)$,

$t \rightarrow^*_{B(R_i)E_b} t_0$, $t' \rightarrow^*_{B(R_i)E_b} t'_0$, t_0 et t'_0 sont égaux modulo E_b, donc t et t' sont égaux modulo $(B(R_i) \cup E_b)$ et par le corollaire 5, t et t' sont égaux modulo $(R_b \cup E_b)$, ce qui conduit à une contradiction. Donc la procédure termine en échec ou réfutation. □

Il est à noter que si la procédure s'arrête en échec, aucune conclusion ne peut en être tirée. Il est possible que l'une des équations de $R_p \cup E_p$ ne soit pas valide, ou que la propriété de consistance ne soit pas satisfaite, ou encore que l'ordre choisi ne soit pas assez puissant pour orienter l'une des équations.

6. Spécifications hiérarchiques structurées

Le concept de spécification structurée a été introduit à l'origine pour écrire des spécifications complexes à partir de plus simples et ceci de façon hiérarchique. Les langages de spécifications algébriques comme OBJ [Futatsugi1985] ou son prédécesseur CLEAR [Burstall1977] introduisent des mécanismes de construction de spécifications. Illustrons sur un exemple la construction hiérarchisée d'une spécification des paires d'entiers naturels et booléens.

Exemple 57 : Les spécifications de base sont

Nat_0

```
sorte entier
Σ : 0, succ
R = ø
E = ø
```

Nat_1 : on introduit l'opérateur plus

```
BSPEC : Nat0
Σ : plus
R : plus(0,x) -> x
    plus(succ(x),y) -> succ(plus(x,y))
E : plus(x,y) = plus(y,x)
    plus(plus(x,y),z) = plus(x,plus(y,z))
```

$Bool_0$

```
sorte bool
Σ : vrai, faux
R = ø
E = ø
```

$Bool_1$: on enrichit $Bool_0$ avec deux opérateurs non et et

BSPEC : $Bool_0$
Σ : non, et

R: non(vrai) -> faux
non(faux) -> vrai
faux et b -> faux
vrai et b -> b
b et b -> b

E: associativité et commutativité de et

$Paire_0$

sorte paire
BSPEC : $Nat_1 \cup Bool_1$
Σ : <;>, l, r
<;> : entier, bool -> paire
l : paire -> entier
r : paire -> bool

R: l(<i;b>) -> i
r(<i;b>) -> b

On peut prouver à ce stade que <l(p);r(p)> -> p.

$Paire_1$: on définit +

BSPEC : $Paire_0$
Σ : +
+ : paire,paire -> paire

R: <i;b> + <i;b'> -> <plus(i,i');b et b'>

La preuve de l'associativité de +, nécessite celle de plus et de et. ∇

Les trois opérations de construction que nous avons utilisées sont

- l'union qui à partir de deux spécifications en produit une nouvelle dont les sortes, les opérations, les règles et les équations sont les unions des sortes, des opérations, des règles et des équations des deux spécifications initiales. C'est cette opération qui permet de considérer

comme base de $Paire_0$ l'union des spécifications Nat_1 et $Bool_1$. Soulignons que nous ne considérons pas cette opération comme une opération d'enrichissement.

- l'enrichissement sans nouvelle sorte. La spécification enrichie hérite de toutes les sortes, opérations, règles et équations de la spécification initiale. C'est l'opération qui permet de construire Nat_1 sur Nat_0, $Bool_1$ sur $Bool_0$, $Paire_1$ sur $Paire_0$.

- l'enrichissement avec une nouvelle sorte. Là encore, la spécification enrichie hérite de toutes les sortes, opérations, règles et équations de la spécification initiale. Les nouveaux opérateurs introduits ont tous la nouvelle sorte dans leur profil. C'est l'opération utilisée pour construire $Paire_0$ sur $Bool_1 \cup Nat_1$. Dans tous les cas, la spécification de base est une union finie de spécifications structurées.

Arrivé à un certain stade de la construction, il faut faire des preuves de validité d'une équation dans l'algèbre initiale de la spécification ainsi construite. Nous allons décrire une procédure de complétion inductive adaptée à ce type de construction, en ce sens qu'elle exploite au maximum la structuration de la construction pour être plus efficace.

Formalisons tout d'abord le cadre dans lequel la procédure va travailler, en définissant de façon récursive les spécifications hiérarchiques structurées.

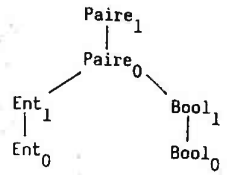
Définition 89 : On appelle **spécification hiérarchique structurée** une spécification structurée SPEC de base BSPEC telle que BSPEC soit une union finie de spécifications qui sont chacune

- soit une spécification hiérarchique structurée convergente
- soit une spécification convergente dite de base. ○

A une telle construction est associé un graphe acyclique de dépendance entre les spécifications, qui est celui de la relation d'ordre stricte entre les spécifications engendrée par:
SPEC > SPEC' si et seulement si SPEC' appartient à la base de SPEC.

Il est à noter qu'à chaque noeud peut être associé un ou plusieurs opérateurs définis à ce neud. Un opérateur mis à un noeud ne fait intervenir dans sa définition que les opérateurs définis à des noeuds de la descendance.

Exemple 58 : Le graphe associé à la spécification $Paire_1$ est le suivant:



∇

Nous avons vu précédemment que la preuve de validité d'une équation dans l'algèbre initiale d'une spécification structurée complète ou convergente supposait une procédure de preuve inductive dans la spécification de base. L'idée est de se dire qu'on peut à nouveau utiliser la complétion inductive pour prouver la validité des équations engendrées dans celle-ci.

Nous supposons donc qu'à chaque noeud du graphe on peut associer une signature Σ, un ensemble de règles R et un ensemble d'équations E. Σ est partitionnée en deux signatures Σ_b et Σ_d. L'ensemble d'équations non

orientées E est partitionné en deux ensembles E_b et E_d. Nous supposons aussi que pour tout ensemble de règles R, nous disposons de fonctions B et D telles que

- B(R) est l'ensemble des Σ_b-règles de R,
- D(R) est l'ensemble des Σ_d-règles de R.

$(\Sigma_b, B(R) \cup E_b)$ est une union finie de spécifications. $B_k(\Sigma)$, $B_k(R)$, $B_k(E)$ désignent respectivement les signatures, les règles et les équations de la kieme spécification de $(\Sigma_b, B(R) \cup E_b)$. Si $(\Sigma, R \cup E)$ est une feuille du graphe de la hiérarchie, on pose $B(R)=E_b=\emptyset$, $D(R)=R$, $E_d=E$.

La procédure de complétion inductive a maintenant un paramètre de plus qui est la signature de la spécification sur laquelle elle travaille.

```
Initialisation:
 P = R_p, R = R_b ∪ R_d, E = E_b ∪ E_d ∪ E_p,
 ir = vrai si R_p ∪ E_p ≠ ø faux sinon

E-COMPLETION INDUCTIVE(Σ,P,R,E,>,ir)
 SI P n'est pas vide
 ALORS choisir une paire (p,q) dans P; p'=p↓; q'=q↓;

      CAS - p' ~E_d q' ALORS
            R=E-COMPLETION INDUCTIVE(Σ,P-{(p,q)},R,E,>,ir)

          - (Σ,R∪E) n'est une feuille de la hiérarchie
            ET p' et q' sont dans T(B_k(Σ),X)) ALORS

            RES=E-COMPLETION INDUCTIVE(B_k(Σ),{(p',q')},B_k(R),B_k(E),>,vrai)

            R=E-COMPLETION INDUCTIVE(Σ,P-{(p,q)},R-B_k(R)∪RES,E,>,ir)

          - SINON l->r = ORIENTATION(p',q',>,ir)
                (P,R)=SIMPLIFICATION(P-{(p,q)},R,E,l->r)
                R=E-COMPLETION INDUCTIVE(Σ,P,R∪{(l->r)},E,>,ir)
      FIN CAS
 SINON SI toutes les règles de D(R) sont marquées
      ALORS RETOURNER R; ARRET en SUCCES
      SINON choisir une règle non marquée l->r en respectant
            l'hypothèse d'équité
            (P,R)=PAIRES-CRITIQUES(l->r,R,E);
            marquer la règle l->r dans R
            R=E-COMPLETION INDUCTIVE(Σ,P,R,E,>,ir)
      FIN SI
 FIN SI
FIN E-COMPLETION INDUCTIVE
```

On peut comprendre la procédure de deux manières différentes:

- la première est de considérer que la procédure travaille sur la hiérarchie de spécifications structurées et ses arguments sont tous les attributs associés à un noeud de la hiérarchie. Le processus consiste à parcourir le graphe depuis son sommet pour découvrir le noeud exact où doit se prouver un lemme. Une fois trouvé, on applique le principe d'induction sans induction, puisqu'on ajoute le lemme à ce noeud. Si cet ajout ne perturbe pas le sous-graphe issu de ce noeud, le lemme est valide et la procédure qui l'a engendré peut continuer.

- la deuxième façon de voir est de considérer qu'à chaque noeud de la hiérarchie est associée une procédure de complétion inductive indexée par la signature. L'appel

E-COMPLETION INDUCTIVE($B_k(\Sigma)$,{(p',q')},$B_k(R)$,$B_k(E)$,>,vrai) est en fait l'appel d'une procédure de preuve pour la validité de p'=q' dans la spécification ($B_k(\Sigma)$, $B_k(R) \cup B_k(E)$). A condition que cette procédure ne modifie pas l'algèbre initiale $I(B_k(\Sigma), B_k(R) \cup B_k(E))$, elle peut être autre chose qu'une complétion. Cette remarque est intéressante pour le cas où l'on peut disposer de procédures de preuve plus efficaces dans certaines spécifications.

La correction de la procédure résulte du même théorème que dans le cas d'une spécification structurée. La preuve en est cependant un peu plus complexe puisqu'elle nécessite en plus un raisonnement par récurrence sur le graphe acyclique associé à la spécification hiérarchique.

Nous allons donc supposer que le théorème de correction est vérifié pour toute spécification SPEC' de la base de SPEC et en déduire qu'il est vrai pour SPEC.

Cette fois-ci P_i et R_i désignent les arguments de la procédure au ieme appel récursif terminal. Seules les preuves des lemmes 18 et 22 sont à modifier.

- Dans le lemme 18, on utilise l'hypothèse de récurrence pour affirmer que, si RES est le résultat de l'appel E-COMPLETION INDUCTIVE($B_k(\Sigma)$,{(p,q)},$B_k(R)$,$B_k(E)$,>,vrai), les équivalences engendrées par RES, ($B(R_i) \cup \{p,q\}$) et ($B(R_i)$) coincident et p=q est un théorème inductif de ($B(R_i) \cup E_b$).

- Dans le lemme 22, on utilise l'hypothèse de récurrence pour dire que, si E-COMPLETION INDUCTIVE($B_k(\Sigma)$,{(p,q)},$B_k(R)$,$B_k(E)$,>) termine en réfutation, p=q n'est pas valide dans $I(B_k(\Sigma),B_k(R) \cup B_k(E))$.

Cette procédure de complétion hiérarchique et équationnelle n'est pas encore implantée. Cependant il existe déjà dans le logiciel REVE une version de la procédure PREUVE-INDUCTIVE dans le cadre de la réécriture non équationnelle.

7. Autres approches et perspectives

Pour conclure ce chapitre, citons quelques autres travaux importants sur des sujets très voisins.

L.Fribourg [Fribourg1984] étend le principe d'induction sans induction à la preuve de clauses équationnelles (i.e. de disjonctions d'équations et d'inéquations) dans l'algèbre initiale d'une théorie équationnelle. Sa méthode consiste à transformer le problème de validité en un problème de satisfiabilité pour lequel une procédure de réfutation complète, basée sur la superréduction, est utilisée.

L.Puel [Puel-Cormier1983] propose une méthode pour faire des preuves dans l'algèbre terminale. Cette approche est en effet parfois mieux adaptée au problème à traiter. En effet, dans l'approche algèbre initiale que nous avons proposée, deux termes sont équivalents si et seulement si on peut déduire leur égalité des axiomes de A. Or souvent on doit considérer comme équivalents des termes qui vérifient certaines propriétés, caractérisées par des fonctions à valeur dans un type de référence. Par exemple, si l'on considère le type Ensemble d'entiers, on veut que deux

ensembles soient égaux s'ils contiennent les mêmes éléments. Pour cela, on ajoute une fonction Appartient à valeur dans le type booléen (qui est dans ce cas le type de référence), ainsi que des équations définissant l'appartenance d'un élément à un ensemble. Deux ensembles e_1 et e_2 sont égaux si pour tout entier n, Appartient(e_1,n)=Appartient(e_2,n). Ce sont alors les notions d'algèbre terminale et d'équivalence terminale qui sont adaptées au problème. Or pour faire des preuves dans l'algèbre terminale, la méthode utilisée est encore une preuve par complétion. Plus précisément, sous les hypothèses de complétude suffisante et de consistance de la spécification SPEC par rapport à une spécification BSPEC du type de référence, et sous la condition que le type abstrait associé à BSPEC = (Σ_b,A_b) est l'algèbre initiale des modèles de A_b, l'algèbre terminale associée à SPEC existe et l'algorithme de complétion que nous donnons dans les spécifications structurées permet de faire des preuves de validité dans cette algèbre terminale.

Padawitz développe des techniques de preuve pour les spécifications paramétrées [Padawitz1985]. Le problème est en effet beaucoup plus complexe dans ce cas. Une propriété de persistance remplace alors la consistance. Une spécification paramétrée PAR est un couple <PSPEC,SPEC> constitué d'une spécification paramètre PSPEC et d'une spécification but SPEC. Un type abstrait paramétré est une classe d'algèbres-but dont chacune est librement engendrée par une algèbre d'une classe K d'algèbres paramètres. Le critère de persistance exprime que chaque algèbre du type abstrait paramétré préserve l'algèbre paramètre sur laquelle elle est librement engendrée. Pour tester la persistance de PAR, Padawitz propose la méthode suivante : décomposer PAR en BPAR et DPAR telles que BPAR contient les opérations et

les équations de PAR nécessaire à la construction du type. Puis montrer que BPAR est persistant et que PAR possède les propriétés de complétude suffisante et de consistance par rapport à BPAR. Il est clair que les résultats de ce chapitre peuvent trouver là une extension intéressante.

CHAPITRE 5:

LA PROCEDURE DE SUPERREDUCTION: UNE RESTRICTION DE LA COMPLETION

CHAPITRE 5: LA PROCEDURE DE SUPERREDUCTION, UNE RESTRICTION DE LA COMPLETION

1. Introduction

On étudie dans ce chapitre un premier lien entre complétion et programmation logique. Comme en programmation logique, on cherche à résoudre des requêtes exprimées à l'aide d'un prédicat, dans un domaine (ou univers) décrit par des équations. L'équivalence sur les termes du domaine est décrite par un système de réécriture équationnelle convergent. On ne considère que le seul prédicat d'égalité noté == et une requête est un terme de la forme (t == t').

L'interprète de ce language de programmation logique restreint est un programme qui, partant du terme d'entrée (t == t'), produit un ensemble de valeurs à attribuer aux variables de t et de t' pour que ces deux termes soient égaux dans la théorie équationnelle décrite par le système de réécriture convergent du domaine. Formellement, l'interprète résoud le problème de l'unification dans une théorie équationnelle. Cet interprète est la procédure de complétion, ou plus exactement une restriction de cette procédure, complète pour ce type de problème, c'est-à-dire trouvant un ensemble complet de solutions s'il existe. Nous décrivons ici cette restriction de la procédure de complétion, appelée NARROWER, qui implante une généralisation du processus de superréduction ou "narrowing" proposé par[Fay1979] et [Lankford1979a] permettant de traiter des théories équationnelles décrites par un système de réécriture équationnelle convergent.

La stratégie d'un langage de programmation comme PROLOG est la résolution linéaire qui présente certaines similitudes avec le processus de

surréduction introduit dans [Hullot1980a]. La surréduction est un calcul de paires critiques entre une règle du système de réécriture et l'un des termes de l'équation à résoudre. La principale différence entre la stratégie de PROLOG et celle de NARROWER est que PROLOG fait uniquement de la surréduction en tête, tandis que NARROWER fait de la surréduction sur tous les sous-termes suivie d'une étape de normalisation. Une réduction est une surréduction particulière, mais, alors que la surréduction est coûteuse car basée sur l'unification, la réduction basée sur le filtrage, l'est en général beaucoup moins. NARROWER réalise donc une amélioration importante en réduisant les termes sur lesquels il travaille. La puissance de NARROWER repose sur le fait qu'il combine les caractéristiques de la programmation logique pour inférer de nouveaux faits à partir d'anciens et les caractéristiques de la programmation fonctionnelle pour simplifier les conséquences et les conserver ainsi sous une forme plus efficace. Cette combinaison des deux stratégies fait d'un language de programmation basé sur la superréduction un bon candidat pour succéder à PROLOG.

Dans une première partie, les notions de surréduction et superréduction sont définies, dans le cadre général des systèmes de réécriture équationnelle. On y rappelle comment la superréduction permet d'obtenir une ensemble complet d'unificateurs d'une équation donnée. Malheureusement le processus a le gros inconvénient de terminer rarement, c'est pourquoi une deuxième partie est consacrée aux améliorations possibles. Toutes visent à réduire l'arbre de recherche des solutions, tout en gardant la complétude de l'ensemble des unificateurs retourné. L'implantation de NARROWER est ensuite décrite, prouvée et illustrée par un exemple dans la troisième partie. Enfin une dernière partie est consacrée à

la méta-superréduction, qui applique les idées proposées pour la complétion de méta-règles à la résolution d'équations.

2. Surréduction et Superréduction

Dans tout le chapitre, A est un ensemble d'axiomes réunion d'un ensemble d'équations E non effaçantes et d'un ensemble de règles R. Subst désigne l'ensemble des substitutions de T(F,X) et P(Subst) l'ensemble des parties de Subst. Rappelons qu'à une théorie E, nous avons associé au chapitre 2, une application $Filtre_E$ qui à deux termes t et t' de T(F,X) associe un ensemble de filtres modulo E de t et t'.

Pour un ensemble de règles R, la **RE-réécriture** notée ->RE est la réécriture associée à R et à $Filtre_E$ et définie par :

$$t \to_{RE[u,\sigma,g\to d]} t'$$

si et seulement si $\sigma \in Filtre_E(g, t_{|u})$ et $t' = t([u \leftarrow \sigma(d)])$.
On dit qu'une substitution σ est **RE-normalisée** si pour tout élément x de son domaine $\sigma(x)$ est RE-normalisé.

Parallèlement, on définit l'application $UNIF_E$ qui à deux termes t et t' de T(F,X) associe un ensemble d'unificateurs modulo E de t et t', contenant $Filtre_E(t,t')$ et vérifiant de plus la condition technique suivante, indispensable dans les preuves de ce chapitre:

* pour toute substitution ρ, pour tous termes g, t et t' tels que ρ appartient à $Filtre_E(t,t')$ et tels que $D(\rho)$ et $V(g)$ soient disjoints, si γ appartient à $UNIF_E(g,t')$, alors $\gamma.\rho$ appartient à $UNIF_E(g,t)$.

Soit U_E l'application qui à (t,t') associe l'ensemble de leurs unificateurs dans la théorie E.

A chaque application UNIF correspond une notion d'ensemble générateur de UNIF(t,t'). Soit $Unif_E$ une application qui à deux termes t et t' de T(F,X) associe un ensemble complet de E-unificateurs relatif à $UNIF_E$ et à $Filtre_E$. On notera ECU_E l'application $Unif_E$ associée à U_E. A tout couple de termes (t,t'), ECU_E associe un ensemble générateur de l'ensemble de tous les unificateurs de t et t' dans la théorie E.

Il est alors possible d'introduire les notions de surréduction et superréduction.

2.1. Surréduction

La surréduction est une opération similaire à la réduction mais qui utilise l'unification au lieu du filtrage.

Définition 90 : Soit (R,E) un système de réécriture équationnelle. Les règles de réécriture de R sont notées g->d. On note -^->RE la **surréduction** associée à R et à $Unif_E$ définie par :

$$t \text{ -}\hat{}\text{->RE}[u,\sigma,g\text{->}d]\ t'$$

si et seulement si σ appartient à $Unif_E(t_{|u}, g)$ et t' = σ(t[u←d]).
σ s'appelle **substitution surréductrice** de t vers t'. On note Surred(t,R,$Unif_E$) l'ensemble des substitutions surréductrices issues de t pour la méthode d'unification $Unif_E$ et l'ensemble de règles R. R et $Unif_E$ peuvent être implicites. ○

Notation : On note -^*-> la fermeture réflexive transitive de la relation -^->, et -^+-> sa fermeture transitive.

Notons que si t -^->RE[u,σ,g->d] t' alors σ(t) ->RE[u,σ,g->d] t'.

2.2. Superréduction

La RE-surréduction est une relation contenant la relation de RE-réduction. Il est donc en théorie inutile de combiner les deux. Mais bien que les étapes de réduction soient des cas particuliers des étapes de surréduction, la surréduction basée sur l'unification est plus coûteuse que la réduction basée sur le filtrage. En pratique, il est beaucoup plus simple et efficace de calculer un E-filtre (ou plus généralement un élément de $Filtre_E$) que de calculer un ensemble complet de E-unificateurs (respectivement $Unif_E$). C'est pourquoi nous introduisons, à la suite de Fay [Fay1978] la RE-superréduction qui combine une étape de RE-surréduction et la mise en RE-forme normale du terme surréduit. Formellement,

Définition 91 : Soit (R,E) un système de réécriture équationnelle. On note -^^->RE la **RE-superréduction** associée à R et à $Unif_E$ définie par :

$$t \text{ -}\hat{}\hat{}\text{->RE}[u,\sigma,g\text{->}d]\ t'$$

si et seulement si σ appartient à $Unif_E(t_{|u}, g)$ et t' est la RE-forme normale de σ(t[u←d]).
σ s'appelle **substitution superréductrice** de t vers t'. On note Supred(t,R,$Unif_E$) l'ensemble des substitutions superréductrices issues de t pour la méthode d'unification $Unif_E$ et l'ensemble de règles R. R et $Unif_E$ pourront être implicites lorsqu'il n'y a pas d'ambiguité. ○

Notation : On note -^*^-> la fermeture réflexive transitive de la relation -^^->, et -^+^-> sa fermeture transitive.

2.3. Surréduction et unification

Nous allons voir dans ce paragraphe en quel sens la surréduction con-

serve l'ensemble des A-solutions d'une équation. Les résultats sont énoncés sans preuve. Toutes peuvent être trouvées dans [Kirchner1985]

Le symbole == utilisé pour décrire les équations à résoudre est considéré dans tout ce chapitre comme un symbole de l'algèbre des termes considérée. On suppose qu'aucun axiome ne fait intervenir ce symbole. Cela afin de considérer les équations comme des termes irréductibles en tête.

Définition 92 :

- S et S' étant des ensembles de substitutions et α une substitution, l'ensemble S.α est défini par S.α = {$\sigma.\alpha$ | σ appartient à S}.
- Soit A un ensemble d'équations. On dit que S est **A-inclus** dans S' si et seulement si pour tout σ dans S, il existe σ' dans S' telle que $\sigma \sim^A \sigma'$ (i.e. pour toute variable x de X, $\sigma(x) \sim^A \sigma'(x)$).
- S et S' sont **A-égaux** si et seulement si ils sont A-inclus l'un dans l'autre. ○

Exemple 59 : Si on prend pour A la théorie décrite par les équations

$$-(-(x)) = x$$
$$-(f(x,y)) = f(-(y),-(x))$$

on a $\{(y \leftarrow x)\} \sim^A \{(y \leftarrow -(-x)),(y \leftarrow -(-(-(-x))))\}$ ∇

Notation : Une méthode de calcul des E-unificateurs $Unif_E$ étant déterminée, soient

$$SURSU_A(t,t') = \bigcup_{\sigma \in Surred(t==t',R,Unif_E)} U_A(t_1,t'_1).\sigma$$

où t_1 et t'_1 sont tels que

$$(t==t') \text{ -}\hat{}\text{->}RE[\sigma]\ (t_1==t'_1)$$

$$SUPERSU_A(t,t') = \bigcup_{\sigma \in Supred(t==t',R,Unif_E)} U_A(t_1,t'_1).\sigma$$

où t_1 et t'_1 sont tels que

$$(t==t') \text{ -}\hat{}\text{->}RE[\sigma]\ (t_1==t'_1)$$

Exemple 60 : Si on prend E = ø, R = {x+x -> x} et un algorithme d'unification standard sur les termes alors

$SURSU_A(y+(x+a),a) = U_A(x+a,a).(y \leftarrow x+a) \cup U_A(y+a,a).(x \leftarrow a)$ ∇

Le résultat suivant permet de s'assurer que la surréduction ne fait pas sortir de l'ensemble des A-solutions.

Lemme 24 : $SURSU_A(t,t')$ est inclus dans $U_A(t,t')$

La preuve de la réciproque nécessite de mettre en évidence le lien entre RE-surdérivation et RE-dérivation. Des versions particulières de ce résultat sont données dans la thèse de Hullot [Hullot1980a] pour E = ø et la réécriture standard, [Kirchner1982] et [Jouannaud1983a] pour la surréduction associée à la réécriture de Peterson et Stickel.

Proposition 11 : Soit t_0 un terme et α une substitution RE-normalisée telle que

$$\alpha(t_0) \text{ -}\hat{}\text{->}RE[u,g\text{->}d]\ t'_1$$

Alors il existe une substitution σ et une substitution β telles que :

- t_0 -^->$RE[u,\sigma,g\text{->}d]\ t'_1$
- $\beta(t_1) \sim^E t'_1$
- $D(\beta) = V(t_1)$

$- \alpha \sim^{E} \beta.\sigma\ [V(t_0)]$

Lemme 25 : Si (R,E) est un système de réécriture équationnelle convergent et si A = (RUE), $U_A(t,t')$ est A-inclus dans $U_E(t,t') \cup SURSU_A(t,t')$.

Proposition 12 : Pour les théories A = (RUE) telles que (R,E) soit un système de réécriture équationnelle convergent,

$$U_A(t,t') \sim^{A} U_E(t,t') \cup SURSU_A(t,t')$$

Par conséquent les A-solutions d'une équation e=(t==t') sont obtenues soit comme E-solutions de e soit comme produit d'une A-solution de l'équation surréduite composée avec la substitution surréductrice.

Ce résultat permet de caractériser la surréduction au niveau des ensembles de substitutions qu'elle manipule. Le résultat analogue pour la superréduction se justifie simplement par le fait que toute relation de réécriture conserve l'ensemble des A-solutions.

Proposition 13 : Pour les théories A = (RUE) telles que (R,E) soit convergent,

$$U_A(t,t') \sim^{A} U_E(t,t') \cup SUPERSU_A(t,t')$$

Il faut également noter que les résultats précédents restent valides si on se limite aux substitutions surréductrices (ou superréductrices) RE-normalisées.

Corollaire 7 : Soit une théorie A = (RUE) telle que (R,E) soit un système de réécriture équationnelle convergent. Soit S l'ensemble des substitutions σ telles qu'il existe une RE-surdérivation issue de t==t'

$$t==t' \text{ -^->}RE[\sigma_1]\ t_1== t'_1 \ \ldots\ t_{n-1}==t'_{n-1} \text{ -^->}RE[\sigma_n]\ t_n==t'_n$$

avec t_n et t'_n E-unifiables,

$\sigma = \beta.\sigma_n.\ldots.\sigma_1$

et β dans $ECU_E(t_n,t'_n)$.

Alors S est un ensemble complet de A-unificateurs de t et t'.

Ce résultat permet de construire un algorithme complet de A-unification pour une équation (t==t'): il suffit en effet de construire progressivement les surdérivations issues de (t==t'). Chaque fois qu'une équation a ses deux membres unifiables dans la théorie E, on obtient une solution dans la théorie RUE en composant toutes les substitutions surréductrices effectuées, avec un unificateur modulo E dans un ensemble complet de E-unificateurs. La puissance de la méthode repose sur le fait qu'elle est incrémentale, puisqu'elle permet à partir d'une procédure d'unification dans une théorie E dobtenir une procédure d'unification dans la théorie RUE.

La surréduction comme la superréduction permettent de construire de façon systématique des algorithmes de A-unification pour des classes de théories très large, mais en contre-partie de cette généralité, elles ont de gros inconvénients: en effet, la procédure de superréduction ne termine pas nécessairement. Nous en donnons des exemples dans les paragraphes suivants. Par ailleurs si elle termine, l'ensemble des A-unificateurs retourné n'est pas forcément minimal et est souvent redondant.

3. Améliorations de la procédure de superréduction

Illustrons par un exemple un premier problème de divergence qui peut se poser.

Exemple 61 : Soit R le système de réécriture convergent:

$$-(-x) \to x$$
$$-(x + y) \to (-y) + (-x)$$

et E = ø.

Considérons l'équation e=(x==-y). Un ensemble minimal complet de A-unificateurs est {(x←-y)}. Par la procédure issue du corollaire 7, on obtient, outre la solution (x←-y) qui unifie x et (-y), la solution (y @<- -x) obtenue par

(x == -y) -^->RE[1, y ← -x] (x == x)

et la solution $(x \leftarrow (-x_2)+(-x_1))$ $(y \leftarrow x_1+x_2)$ obtenue par

(x == -y) -^->RE[2, y ← x_1+x_2] (x == $(-x_2)+(-x_1)$).

∇

Remarque : Il est clair sur cet exemple que la procédure ne s'arrête pas alors qu'un critère simple d'arrêt pourrait être utilisé: si une surdérivation débouche sur une équation e dont on connait un ensemble complet de A-unificateurs, s'arrêter en retournant cet ensemble composé à gauche par le produit des substitutions surréductrices qui ont permis d'aboutir à e.

Dans l'objectif d'étudier des améliorations de la procédure précédente, nous allons introduire formellement la notion d'arbre de surréduction ou de superréduction.

Définition 93 : On appelle **arbre de surréduction** associé à un terme t (qui peut être une équation) l'arbre noté Arb-sur(t) défini par

Arb-sur(t)(ε) = t

Si Arb-sur(t)(u) = t' alors Arb-sur(t)(u.i) = t" pour σ_i dans Surred(t') et t' -^->RE[σ_i] t".

On définit de façon analogue un **arbre de superréduction** en remplaçant la notion de surréduction par celle de superréduction. ○

Les noeuds de cet arbre sont étiquetés par les équations surréduites et les arêtes sont étiquetées par les substitutions surréductrices. Aussi est-il équivalent de se donner l'occurrence d'un noeud e dans Arb-sur(e_0) et la suite des substitutions surréductrices permettant de surréduire e_0 en e.

Les améliorations de la procédure de surréduction visent toutes à réduire et/ou factoriser l'arbre de surréduction sans perdre la complétude du processus: celui-ci doit toujours fournir, s'il termine, un ensemble complet de A-solutions pour l'équation considérée. Citons trois améliorations qui vont dans ce sens.

- La première est la surréduction basique introduite par Hullot (loc.cit.), qui permet de réduire l'arbre de surréduction en enlevant les branches qui correspondent en fait à des surréductions sur des sous-termes provenant de substitutions appliquées à une étape précédente. Elle est généralisée par C.Kirchner (loc.cit.) dans le cadre de la réécriture équationnelle et elle n'est pas développée ici.

- La seconde amélioration est la factorisation de certaines branches infinies lorsque l'arbre de surréduction est un arbre infini rationnel

(i.e. a un nombre fini de sous-arbres distincts). Cela permettra de décrire finiment des ensembles complets infinis de A-unificateurs.

● Enfin, la suppression des branches subsumées de l'arbre de surréductions c'est-à-dire des branches qui sont, en un certain sens que nous définissons, instance d'une autre branche, permettra également de réduire la taille de l'arbre de surréduction sans perdre la complétude des ensembles de A-solutions.

Toutes ces améliorations sont présentées pour un arbre de surréduction mais sont valables pour un arbre de superréduction.

3.1. Factorisation de l'arbre de surréduction

Dans ce paragraphe, nous montrons comment factoriser un arbre de surréduction infini rationnel. L'exemple suivant d'un tel arbre infini est dû à F.Fages et G.Huet [Fages1983a].

Exemple 62 : On considère le système de réécriture R suivant

g(f(x,y)) -> g(y)

E = ø et l'équation e=(g(x) == g(o)). On a par conséquent

$$e = (g(x) == g(o)) \quad -\hat{}->[2,\ x \leftarrow f(x_1,x_2)]\ e_1 = (g(x_2) == g(o))$$

e_1 est égale à e à un renommage des variables près. Arb-sur(e) est donc un arbre infini rationnel (au renommage des variables près). ∇

Nous allons montrer comment décrire finiment l'ensemble des A-solutions de e, correspondant à un tel arbre.

3.1.1. Caractérisation des branches infinies rationnelles de l'arbre de surréduction

Dans ce qui suit nous supposons que nous résolvons l'équation $e_0 = (t_0 == t'_0)$ et que Arb-sur(e_0) est un arbre rationel. Nous pouvons donc considérer que les équations dérivées de e_0 par surréduction sont numérotées par un entier compris entre 1 et n. Nous appellerons noeuds de succès de Arb-sur(e_0) les noeuds étiquetés par une équation e=(t==t') telle que $ECU_E(t,t')$ ne soit pas vide.

Nous définissons maintenant les objets qui vont nous être nécessaires pour décrire finiment l'arbre de surréduction.

- Soit Loop-Eq l'ensemble des équations e de Arb-sur(e_0) dont est issue une boucle :

$$\text{Loop-Eq} = \{\ e \mid e\ -\hat{}+->\ e \text{ et } e_0\ -\hat{}*->\ e\}$$

- Soit Fin_i l'ensemble des substitutions qui étiquettent un chemin sans boucle et qui termine sur une équation qui a un ensemble de E-unificateurs non vide :

Fin_i = {σ | il existe une surdérivation

$$e_i\ -\hat{}->[a_1]\ e_i^1\ \ldots\ -\hat{}->[a_n]\ e_i^n$$

telle que

* $e_i^n = (t == t')$ avec $ECU_E(t,t')$ non vide
* pour tout k dans $i_1,\ldots,i_n$, e_k n'appartient pas à Loop-Eq
* $\sigma = \gamma.a_n \ldots a_1$ avec $\gamma \in ECU_E(t,t')$}

Un élément de Fin_i est donc un chemin issu du noeud étiqueté par e_i, ne passant pas par une équation donnant lieu à une boucle, mais éventuellement issu d'une telle équation et terminant sur un noeud de succès.

- Soit $Loop_p$ l'ensemble des substitutions obtenues à partir d'une boucle dans une surdérivation issue de e_p.

$Loop_p$ = {σ | il existe une surdérivation

$$e_0 \text{ -^*^-> } e_p \text{ -^-}[\sigma_1]\text{-> } e_p^1 \text{ -^-> } \dots \text{ -^-}[\sigma_k]\text{-> } e_p^k = e_p$$

telle que

* pour tout i et j dans 1,...,k, i ≠ j implique $e_p^i \neq e_p^j$ (pas de boucle)

* $\sigma = \sigma_k \dots \sigma_1$}

Un élément de $Loop_p$ correspond donc à un chemin décrivant une boucle issue du noeud étiqueté par e_p appartenant à Loop-Eq.

- Soit $Prem_p$ l'ensemble des substitutions obtenues dans la première partie d'une surdérivation issue de e_0, entre e_0 et la première équation e_p de Loop-Eq rencontrée:

$Prem_p$ = {σ | il existe une surdérivation

$$e_0 \text{ -^-}[\sigma_1]\text{-> } e_0^1 \text{ -^-> } \dots e_0^{n-1} \text{ -^-}[\sigma_n]\text{-> } e_0^n$$

telle que

* $e_0^n = e_p$

* $e_p \in$ Loop-Eq

* $\sigma = \sigma_n \dots \sigma_1$

* pour tous i et j dans 1,...,n, i ≠ j implique $e_0^i \neq e_0^j$ (pas de boucle)}

Un élément de $Prem_p$ correspond donc à un chemin sans boucle issu de la racine et arrivant sur un noeud d'où part une boucle.

- Pour toutes les équations dont est issue une boucle, $Inter\text{-}loop_{pq}$ décrit les substitutions permettant de passer d'une boucle à une autre :

$Inter\text{-}loop_{pq}$ = {σ | il existe une surdérivation

$$e_p \text{ -^-}[\sigma_1]\text{-> } e_p^1 \text{ -^-> } \dots \text{ -^-}[\sigma_k]\text{-> } e_p^k$$

telle que

* $e_p^k = e_q$

* e_p et e_q appartiennent à Loop-Eq

* p ≠ q

* pour tous i et j dans 1,...,k, i ≠ j implique $e_p^i \neq e_p^j$ (pas de boucle)}

Un élément de $Inter\text{-}loop_{pq}$ correspond donc à un chemin ne contenant pas de boucle et joignant deux noeuds d'où part une boucle.

Nous pouvons maintenant décrire un ensemble complet de A-solutions de l'équation e_0.

- Soient S_i les ensembles de substitutions associés à chaque équation origine d'une boucle et qui sont définis récursivement comme suit.

* S_i contient $Prem_i$,

* Pour toute substitution β dans $Loop_i$ et α dans S_i, β.α appartient à S_i,

* Pour tout entier j et toutes substitutions γ dans $Inter\text{-}loop_{ji}$ et α dans S_i, α.γ appartient à S_i.

Un élément de S_i correspond donc à un chemin possible de la racine de l'arbre de surréduction jusqu'à un noeud étiqueté par e_i, après avoir éventuellement bouclé à des noeuds étiquetés par un ej.

3.1.2. Définition récursive des A-solutions

Soit S l'ensemble des substitutions défini par:

* S contient Fin_0,

* pour toute substitution α dans S_i et β dans Fin_i, β.α appartient à S.

En d'autres termes un élément de S est ou bien un chemin sans boucle depuis la racine jusqu'à un noeud de succès, ou bien est composé d'un chemin de la racine jusqu'à un noeud e_i d'où part une boucle, suivi d'un chemin débouchant sur un noeud de succès.

Théorème 17 : S est un ensemble complet de A-solutions de l'équation e_0.

Preuve: C'est une conséquence des définitions que nous venons de donner.

- D'une part tout élément σ de S est une A-solution de e_0 car σ est le produit de substitutions surréductrices, étiquetant un chemin menant à un noeud de succès e_n, et d'une E-solution de e_n.

- Réciproquement, soit $\sigma = \gamma.a_n.....a_1$ une substitution obtenue par la surdérivation

$$e_0 \text{ -}\hat{}*\text{->}[a_1]\ e_1 \ \ldots\ \text{-}\hat{}*\text{->}[a_n]\ e_n = (t_n == t'_n)$$

avec γ élément de $Unif_E(t_n,t'_n)$ et $e_1,\ldots,e_n$ toutes les équations de cette surdérivation d'où est issue une boucle (i.e. éléments de Loop-Eq). On vérifie facilement que chacune des substitutions a_i appartient à l'un des ensembles $Prem_i$, Fin_i, $Loop_i$ ou $Inter\text{-}loop_{ji}$. Donc σ appartient à S. □

Pour un arbre de surréduction rationnel, la description que nous venons de donner de S est finie; on peut donc modifier la procédure de surréduction de telle sorte qu'elle termine dans ce cas. Il s'agit de détecter des boucles éventuelles dans l'arbre de surréduction. Il suffit pour cela de garder l'histoire de la formation de l'arbre. On obtient ainsi une procédure qui retourne une définition récursive finie des A-solutions

de l'équation e_0.

Exemple 63 : Reprenons l'exemple de F.Fages et G.Huet donné au début de ce paragraphe. L'équation g(x) == g(a) ne peut être surréduite que par l'unique règle g(f(x,y))->g(y) et par conséquent

* pour tout entier i, $Fin_i = Fin_0 = \{(x \leftarrow 0)\}$

* l'ensemble S des A-solutions de l'équation trouvé ici sera donc récursivement défini par

$\{(x \leftarrow a)\} \subseteq S$
si $(x \leftarrow t) \in S$ alors $(x \leftarrow f(y,t)) \in S$.

∇

3.2. Suppression des branches subsumées

L'objectif est ici de montrer comment enlever certaines branches de l'arbre de surréduction qui n'apporteront que des A-solutions instances d'autres A-solutions.

Exemple 64 : Considérons le système de réécriture convergent

x + x -> x
(a + x) + (x + a) -> a + x

et l'équation (x + y) + (y + x) == t où t est un terme quelconque.

On a parmi d'autres superdérivations possibles:

(x + y) + (y + x) == t -^^->[y←x] x == t

(x + y) + (y + x) == t -^^->[x←a] a + y == t -^^->[y←a] a == t

La seconde superdérivation est une instance de la première en ce sens où (a==t) est une instance de (x==t) par la substitution $\beta = (x \leftarrow a)$ et $\beta.(y \leftarrow x) = (x \leftarrow a)(y \leftarrow a)$. Les A-solutions trouvées par la première

surperdérivation sont plus générales que celles trouvées par le seconde. Il est donc inutile de poursuivre la superréduction de a == t. ∇

L'idée de base est donc qu'il est inutile de calculer les A-solutions d'une équation qui est instance d'une autre à condition que les chemins de l'arbre de surréduction qui mènent à chacune de ces équations soient également instances l'un de l'autre.

Théorème 18 : Soient e_0, $e_1=(t_1==t'_1)$ et $e_2=(t_2==t'_2)$ trois équations telles que e_0 -^*^->$[\sigma_1]$ e_1 et e_0 -^*^->$[\sigma_2]$ e_2.
Si il existe β telle que $\beta(e_1) \sim^E e_2$ et $\beta.\sigma_1 \sim^E \sigma_2$
alors $U_A(t_2,t'_2).\sigma_1$ est E-inclus dans $U_A(t_1,t'_1).\sigma_1$.

Preuve: Soit $a \in U_A(t_2,t'_2)$, $a(t_2) \sim^A a(t'_2)$ d'où
$a.\beta(t_1) \sim^E a(t_2) \sim^A a(t'_2) \sim^E a.\beta(t'_1)$
et par conséquent $a.\beta$ est une A-solution de e_1 donc
$U_A(t_2,t'_2).\beta \subseteq U_A(t_1,t'_1)$.
Ce qui, compte tenu de la seconde hypothèse, implique
$U_A(t_2,t'_2).\sigma_2 \sim^E U_A(t_2,t'_2).\beta.\sigma_1 \subseteq U_A(t_1,t'_1).\sigma_1$ □

Une conséquence immédiate de ce résultat est que l'on peut abandonner une branche de l'arbre de surréduction instance d'une autre sans perdre la complétude.

4. Complétion et procédure de superréduction

Soient t_0 et t'_0 les deux termes à unifier et W la réunion de leurs variables notées $\{x_1,x_2,...x_n\}$. Outre le prédicat ==, on introduit de

nouvelles constantes, vrai et boucle_e, ainsi qu'un symbole de fonction solution dont l'arité dépend de l'équation à résoudre. En effet, l'arité de solution est le cardinal de W.

Définition 94 : Une **règle-but** (l,r) est une paire de termes orientée telle que r admet comme symbole de tête le symbole solution et que l soit admet comme symbole de tête le symbole ==, soit est l'une des constantes vrai ou boucle_e. ○
La règle-but initiale s'écrit (t_0 == t'_0, solution(x_1,...,x_n)). Elle s'interprète comme la requête suivante: quelles valeurs faut-il attribuer aux variables $x_1,...,x_n$ pour que le terme (t_0 == t'_0) s'évalue en vrai après instanciation des variables, à l'aide du système de réécriture équationnelle (R,E)?

Notation : Le membre droit d'une règle-but de la forme solution($t_1,...,t_n$) sera plus brièvement noté solution(σ) où σ est la substitution $(x_1 \leftarrow t_1)...(x_n \leftarrow t_n)$.

4.1. Procédure de superréduction

Les paramètres de la procédure sont

- un système de réécriture équationnelle convergent constant (R_1,E) où R_1 est obtenu en ajoutant à R la règle (x == x) -> vrai.
- un ensemble de règle-buts C que nous considèrerons comme des paires de termes orientées.
- un ensemble de paires de termes P qui sont des paires critiques obtenues en superposant les règles de R_1 dans les paires orientées de C. Ces superpositions se font avec la méthode d'unification associée à chaque règle de R. Par contre, la superposition de la règle (x == x) -> vrai dans une

règle-but se fait toujours modulo E.

Chaque règle-but de C est obtenue en normalisant par ->RE une paire critique de P. Son membre gauche peut être interprété comme l'étiquette d'un noeud de l'arbre de superréduction et son membre droit comme le chemin qui conduit à ce noeud. Une règle-but est marquée si et seulement si toutes les superpositions des règles de R_1 et des équations de E dans son membre gauche ont été calculées.

Nous supposerons toujours que la stratégie permettant de choisir une règle-but non marquée de C satisfait l'hypothèse d'équité suivante: pour toute règle-but (l,r), il existe un appel récursif de la procédure tel que

- soit la règle-but (l,r) est une instance de la nouvelle règle-but introduite à cet appel
- soit la règle-but (l,r) est sélectionnée et les superpositions de (l,r) avec les règles de R_1 sont calculées.

Cette hypothèse est nécessaire pour assurer la complétude de la procédure.


```
Initialisation
R1 = R ∪ {(x == x) -> vrai}, C = {(t0 == t'0, solution(x1,...,xn))}
P = ø

NARROWER(P, R1, E, C)
   SI P est non vide
   ALORS choisir une paire (p,q) dans P
         g := p↓R1E, d := q↓R1E

         C := SIMPLIFICATION(C,(g,d))
         NARROWER(P-{(p,q)}, R1, E, C∪{(g,d)})

   SINON SI toutes les règle-buts de C sont marquées
         ALORS ARRET en SUCCES, RETOURNER l'ensemble des solutions
         SINON choisir une règle-but non marquée (l,r) dans C
               en respectant l'hypothèse d'équité
               P := PAIRES-CRITIQUES((l,r), R1, E)
               NARROWER(P, R1, E, C)
         FIN SI
   FIN SI
FIN NARROWER
```

La procédure PAIRES-CRITIQUES calcule toutes les paires critiques obtenues en superposant les règles de R_1 dans le membre gauche l de la règle-but (l,r).

Remarquons que le mode de calcul des paires critiques utilisé dans la procédure implique un parcours de l'arbre de superréduction en largeur et non en profondeur d'abord comme le ferait un interprète PROLOG.

La procédure SIMPLIFICATION implante les factorisations des branches et la suppression des branches subsumées. Lorsqu'on introduit une nouvelle règle-but dans C, il est intéressant d'effectuer des simplifications. Néanmoins toute simplification n'est pas autorisée, sous peine de perdre la complétude de l'ensemble des solutions.

- Nous avons vu dans l'étude théorique précédente que l'on peut supprimer toute règle-but qui est une instance de la nouvelle règle-but calculée. De même si cette dernière est une instance d'une règle-but déjà existante, il

est inutile de l'introduire dans l'ensemble des règles-buts.

- L'autre type de simplification est la détection des bouclages. Si la nouvelle règle-but calculée est de la forme

(t_i == t'_i,solution(σ)) et que l'on détecte un bouclage, c'est-à-dire qu'il existe déjà une règle-but

(t_i == t'_i,solution(α)), avec σ = β.α,

on remplace la règle-but calculée par la règle-but

boucle_e -> solution(β.α).

où boucle_e est une nouvelle constante associée à l'équation (t_i == t'_i). Cette règle-but correspond à un noeud de l'arbre de surdérivation d'où part une boucle. Elle a pour signification de mémoriser que si une solution est du type γ.α, alors toute substitution γ.$(\beta)^n$.α est aussi une solution pour toute valeur de l'entier n.

Remarque : Puisque (R,E) est supposé convergent, (R_1,E) l'est aussi. De plus R_1 n'est pas modifié par NARROWER.

4.2. Preuve de la procédure

La preuve de la procédure se fait en deux étapes: on prouve d'abord la correction puis la complétude. La correction s'exprime par le fait que à chaque règle-but dans C, on peut associer une superdérivation issue de (t_0==t'_0), ou encore un noeud de l'arbre de superréduction. La complétude traduit le fait que réciproquement, chaque équation obtenue à partir de (t_0==t'_0) par superréduction est le membre gauche d'une règle-but de C; plus précisément à chaque noeud de l'arbre de superréduction correspond une règle-but engendrée par la procédure. Ces deux résultats impliquent que NARROWER simule correctement la procédure de superréduction.

Notation : Notons C_i la valeur de l'argument C lors du i-eme appel récursif de NARROWER et soit H la réunion des C_i.

On définit sur H la relation < par: r < r' si et seulement si

$r \in C_i$, $r' \in C_j$, $i < j$ et

r' est obtenu par superposition d'une règle g->d de R_1 sur r, puis normalisation du résultat.

L'ensemble H est partiellement ordonné par la relation < et possède un plus petit élément r_0.

Nous établissons dans un premier temps la bijection entre les noeuds de l'arbre de superdérivation non factorisé et l'ensemble de règle-buts H. Dans un deuxième temps, nous exprimerons les solutions à l'aide de la factorisation de l'arbre de superréduction proposée précédemment.

4.2.1. Correction

Dans la suite, chaque règle-but r_j appartenant à H est notée r_j = (b_j,σ_j).

Lemme 26 : (Lemme de correction)

Soit r_n une règle-but de H distincte de r_0. Il existe une superdérivation issue de b_0

$$b_0 \text{ -^^->RE}[\sigma_1] \ b_1 \text{ -^^->RE}[\sigma_2] \ldots \text{ -^^->RE}[\sigma_n] \ bn$$

telle que ou bien

r_n s'écrit b_n = (t_n == t'_n) -> solution(σ'_n)

ou bien

r_n s'écrit vrai -> solution(σ'_n) et

t_{n-1} et t'_{n-1} sont E-unifiables par l'unificateur σ_n avec $\sigma'_n = (\sigma_n.\sigma_{n-1}....\sigma_1)\downarrow_{RE}$

Preuve: Prouvons par récurrence que r_n est de l'une des formes données. Puisque r_n est distincte de r_0, elle est issue d'une superposition d'une règle de R_1 sur une règle-but r_{n-1} de H. Par hypothèse de récurrence appliquée à $r_{n-1} < r_n$:

- soit r_{n-1} s'écrit (vrai, solution(σ'_n)), ce qui est impossible car aucune règle de R_1 ne peut se superposer sur la constante vrai,
- soit r_{n-1} s'écrit ((t_{n-1} == t'_{n-1}), solution(σ'_n)), donc r_n est issue d'une superposition d'une règle g->d de R_1 sur b_{n-1}, à l'occurrence u, avec la substitution σ_n. σ_n est un unificateur de g et $b_{n-1|u}$. Cette superposition crée la paire critique $(p,q) = (\sigma_n(b_{n-1}[u\leftarrow d]), solution(\sigma_n.\sigma'_n))$.

Donc b_{n-1} -^->RE[u,σ_n,g->d] p. r_n est obtenu à partir de (p,q) par normalisation, puis orientation:

r_n: $(\sigma_n(b_{n-1}[u\leftarrow d]))\downarrow_{RE}$, solution($\sigma.\sigma'_n$)$\downarrow_{RE}$

Donc $b_n = (\sigma_n(b_{n-1}[u\leftarrow d]))\downarrow_{RE}$

$\sigma'_n = (\sigma_n.\sigma'_n)\downarrow_{RE}$

et b_{n-1} -^^->RE[σ_n] b_n.

Comme g->d appartient à R_1, on distingue deux cas:

- cas 1: Si g->d appartient à R, puisque b_{n-1} a pour symbole de tête ==, u n'est pas l'occurrence ε, et b_n peut s'écrire b_n = (t_n==t'_n).
- cas 2: Sinon g->d est la règle x==x -> vrai. Dans ce cas, u est l'occurrence ε, et σ_n est un E-unificateur de b_{n-1} avec x==x. Donc σ_n est un E-unificateur de t_{n-1} et t'_{n-1}. r_n s'écrit:

vrai -> solution(σ'_n). □

4.2.2. Complétude

Lemme 27 : (Lemme de complétude)

Soit une superdérivation issue de $b_0 = (t_0 == t'_0)$

$$b_0 \text{ -^^->RE}[\sigma_1] \ b_1 \text{ -^^->RE} \ ... \ \text{-^^->RE}[\sigma_n] \ b_n=(t_n==t'_n)$$

alors il existe une règle-but r_n de H telle que r_n s'écrit

- soit (t_n == t'_n, solution(σ'_n))
- soit (vrai, solution(σ'_n))

avec $\sigma'_n = (\sigma_n,...\sigma_1)\downarrow_{RE}$.

Preuve: Par récurrence sur la longueur n de la superdérivation. Supposons le résultat vrai pour toute longueur inférieure à n:

$b_{n-1} = (t_{n-1} == t'_{n-1})$ -^^->RE[u,σ_n,g->d] $b_n = (t_n == t'_n)$.

Par hypothèse de récurrence, il existe une règle-but r_{n-1} de H s'écrivant (b_{n-1}, solution(σ'_{n-1})).

D'après l'hypothèse d'équité, il existe un appel récursif j tel que r_{n-1} soit choisi et PAIRES-CRITIQUES(r_{n-1},R_1,E) soit appelée. Comme $b_{n-1|u}$ et g sont unifiables par un E-unificateur σ_n appartenant à un ensemble complet de E-unificateurs, la paire critique $(\sigma_n(b_{n-1}[u\leftarrow d]), solution(\sigma_n.\sigma'_{n-1}))$ est engendrée. NARROWER normalise et oriente toutes les paires de P avant d'en engendrer d'autres. Il existe donc un appel k > j tel que C_k contienne

$r_n = (\sigma_n(b_{n-1}[u\leftarrow d])\downarrow_{RE}, solution((\sigma_n.\sigma'_{n-1})\downarrow_{RE}))$.

$= (b_n, \sigma'_n)$ en posant $\sigma'_n = (\sigma_n.\sigma'_{n-1})\downarrow_{RE}$.

Si g est (x==x), u est l'occurrence ε et le membre gauche de r_n est la constante vrai. □

4.2.3. Validité

Puisque l'ensemble des règle-buts de H est en bijection avec les noeuds de l'arbre de superréduction, nous allons pouvoir donner de l'ensemble des solutions trouvées par l'algorithme, une définition récursive analogue à celle donnée dans le paragraphe précédent. Un noeud de succès correspond à une règle-but (vrai, solution(σ)) et un noeud dont est issue une boucle correspond à une règle-but (boucle_e, solution(σ)).

On peut à partir de H, définir, comme dans le paragraphe 3.1.2, un ensemble S de substitutions.

Définition 95 : On appelle solution de la procédure NARROWER toute substitution σ de l'ensemble S construit à partir de H. ○

Théorème 19 : Théorème de validité: L'ensemble des solutions de la procédure NARROWER appliquée à $(t_0==t'_0)$ est un ensemble complet de RUE-unificateurs de t_0 et t'_0.

Preuve: - Soit S l'ensemble des solutions de NARROWER, et soit θ appartenant à S. D'après le lemme de correction il existe une superdérivation issue de b_0:

$$b_0 \text{ -^^->RE}[\sigma_1]\ b_1 \text{ -^^->RE ... -^^->RE}[\sigma_n]\ b_n=(t_n==t'_n)$$

telle que t_n et t'_n soient E-unifiables par σ et $\theta = (\sigma.\sigma_n...\sigma_1)\downarrow_{RE}$. $\sigma.\sigma_n...\sigma_1$ est un RUE-unificateur de t_0 et t'_0, donc

θ est un RUE-unificateur de t_0 et t'_0.

- Soit α un RUE-unificateur de t_0 et t'_0. Par complétude de la superréduction, il existe une superdérivation

$$b_0 \text{ -^^->RE}[\sigma_1]\ b_1 \text{ -^^->RE ... -^^->RE}[\sigma_n]\ b_n=(t_n==t'_n)$$

telle que t_n et t'_n soient E-unifiables par σ et $\theta=\sigma.\sigma_n...\sigma_1 \leq_A \alpha$. Donc $\theta\downarrow_{RE} \leq_A \alpha$. D'après le lemme de complétude, $\theta\downarrow_{RE}$ est une solution de NARROWER. □

4.3. Comparaison avec la procédure de complétion

Comme la procédure de complétion équationnelle, la procédure de superréduction est basée sur les deux opérations de superposition et réduction et fournit une procédure de preuve de la satisfiabilité d'une équation dans l'algèbre libre d'une classe équationnelle d'algèbres. En ce sens elle s'intègre à la famille des preuves par complétion présentées dans cette thèse.

Néanmoins la procédure de superréduction et la procédure de complétion équationnelle ne sont pas identiques; nous allons passer en revue les différences entre les deux procédures et justifier en même temps l'introduction de ces différences.

On introduit ici un ensemble de règle-buts. Leur définition en tant que paires orientées permet de mettre en évidence le parallèle avec la superposition entre règles que fait habituellement la procédure de complétion. Cependant il faut noter qu'on ne peut pas utiliser les règle-buts comme des règles de réécriture et ceci pour plusieurs raisons:

- elles ne peuvent pas être utilisées pour réduire les paires critiques de P. En effet cela provoquerait la disparition éventuelle de solutions car cela reviendrait à abandonner certaines branches de l'arbre de superréduction.

- elles ne peuvent pas être simplifiées sans précaution par une nouvelle règle-but calculée, toujours sous peine de perdre la complétude de l'ensemble des solutions.

- elles ne satisfont pas en général la condition que l'ensemble des variables de leur membre gauche contienne l'ensemble des variables de leur membre droit. Pour pallier à cette difficulté, une solution proposée par F.Fages est de remplacer une règle-but

$(t_i == t'_i,\ solution(u_1,\ldots,u_n))$ par la règle

$*(t_i,t'_i,u_1,\ldots,u_n) \rightarrow solution(u_1,\ldots,u_n)$ où * est un nouveau symbole d'arité card(W)+2. L'inconvénient de cette solution est que, pour simuler correctement la superréduction, il ne faut pas faire de superpositions dans les sous-termes $u_1,\ldots,u_n$. On pourrait envisager une implantation dans laquelle les sous-termes $u_1,\ldots,u_n$ du membre gauche seraient partagés et marqués de manière à ne jamais s'autoriser de superposition sur ces sous-termes, apportés par une substitution surréductrice. Une telle implantation réaliserait la procédure de superréduction basique.

Parallèlement à l'introduction de l'ensemble de règle-buts, il faut spécialiser un système de réécriture R_1. L'ensemble de règles R_1 est utilisé pour réduire les paires critiques. C'est aussi l'ensemble des règles qui se superposent dans les règle-buts. Remarquons que les règle-buts ne se superposent jamais dans les règles à cause de leur symbole de tête ==.

Pour s'adapter au problème de la superréduction, les deux procédures de simplification et d'orientation des règles ont été modifiées.

- Lorsqu'on introduit une nouvelle règle-but dans C, il est toujours intéressant d'effectuer certains types de simplifications. Celles décrites dans le paragraphe 3.2 sont autorisées.

- La procédure d'orientation d'une règle-but consiste simplement ici en un test syntaxique. Le membre gauche a toujours comme symbole de tête le symbole solution. Le membre droit peut être de la forme (t == t') ou bien être la constante vrai ou une constante boucle_e. Remarquons qu'ainsi la procédure NARROWER ne s'arrête jamais en échec, contrairement à la procédure de complétion.

4.4. Un exemple

Pour illustrer la procédure NARROWER, nous donnons un exemple de problème de coloriage de carte. C'est un problème classique en programmation logique. La carte est représentée sur la figure 1.

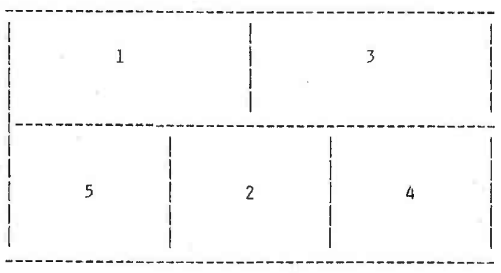


Figure 1 : la carte

Le problème posé à NARROWER était de colorier la carte, sachant que la partie 1 est "b" et la partie 2 est "j". L'univers du problème est présenté

sous la forme suivante à NARROWER:

```
% les opérations sur les booléens

x * 0 == 0
0 * x == 0
1 * x == x
x * 1 == x
(x * ( y * z)) == ((x * y) * z)


% description des couleurs (b,j,m,r toutes distinctes)

next(b, j) == 1
next(b, m) == 1
next(b, r) == 1

next(m, j) == 1
next(m, b) == 1
next(m, r) == 1

next(j, b) == 1
next(j, m) == 1
next(j, r) == 1

next(r, j) == 1
next(r, m) == 1
next(r, b) == 1

next(x, x) == 0


% description de la carte

carte(x1, x2, x3, x4, x5) == next(x1, x2)*next(x1, x3)*
next(x1, x5)*next(x2, x3)*next(x2, x4)*next(x2, x5)*next(x3, x4)


% le problème à résoudre

carte(b,j,x3,x4,x5) == 1
```

NARROWER trouve 8 solutions possibles:

% les solutions

1. $x_3 \leftarrow m$, $x_4 \leftarrow b$, $x_5 \leftarrow m$
2. $x_3 \leftarrow m$, $x_4 \leftarrow b$, $x_5 \leftarrow r$
3. $x_3 \leftarrow m$, $x_4 \leftarrow r$, $x_5 \leftarrow m$
4. $x_3 \leftarrow m$, $x_4 \leftarrow r$, $x_5 \leftarrow r$
5. $x_3 \leftarrow r$, $x_4 \leftarrow m$, $x_5 \leftarrow m$
6. $x_3 \leftarrow r$, $x_4 \leftarrow m$, $x_5 \leftarrow r$
7. $x_3 \leftarrow r$, $x_4 \leftarrow b$, $x_5 \leftarrow m$
8. $x_3 \leftarrow r$, $x_4 \leftarrow b$, $x_5 \leftarrow r$

5. Surréduction contrainte et méta-surréduction

Cette section étend les résultats obtenus sur la complétion de méta-règles à la résolution d'équations. Nous envisageons ici trois types de problèmes:

● Supposons que la théorie équationnelle dans laquelle on veut résoudre des équations possède un système de réécriture convergent infini. La surréduction habituelle devient alors impossible, ou du moins elle ne peut être utilisée que pour semi-décider de la satisfiabilité d'une équation. Encore faut-il adopter une stratégie particulière, consistant à explorer l'arbre de surréduction en largeur d'abord, puis en profondeur à partir de la première surréduction possible, et éventuellement à faire des retours en arrière en cas d'échec, pour être sûr de trouver une solution s'il en existe. Le danger qui subsiste est de partir dans une branche infinie. Cependant, s'il est possible de trouver un système de méta-règles permettant de décider de la théorie équationnelle, celui-ci peut aussi être util-

isé pour faire une forme de surréduction appelée surréduction contrainte. Nous montrons que ce processus permet de calculer des ensembles complets d'unificateurs.

● L'autre problème est à nouveau celui de divergence du processus de surréduction. Tout comme pour la complétion, il peut se produire deux cas de divergence, en profondeur et en largeur. La même notion de schématisation s'applique encore, mais il faut noter que cette fois-ci, ce sont des familles de règles-but qui sont schématisées par des méta-règles. La surréduction et la superréduction sur des méta-termes sont alors définies pour un système de réécriture équationnelle convergent. Nous montrons qu'à partir des méta-substitutions calculées par ce processus, il est possible d'obtenir, par instanciation des méta-variables, un ensemble complet d'unificateurs.

● Un troisième cas encore plus complexe est celui où, disposant d'un système de méta-règles permettant de décider de la théorie équationnelle $\sim^A$, le processus de surréduction contrainte diverge lui aussi. La méta-surréduction et la méta-superréduction sur des méta-termes sont alors définies pour un système de méta-règles. Ces deux derniers problèmes se traitent dans un formalisme commun.

5.1. Surréduction contrainte

Supposons donné un ensemble de méta-règles MR qui schématise correctement un système de réécriture infini engendré par complétion d'un ensemble d'axiomes A. Rappelons que dans ce cas, la relation de réécriture -->>> est convergente et satisfait <<<-*->>> = $\sim^A$. A la notion de réécriture contrainte -->> est associée une relation -^->> utilisant de l'unification contrainte au lieu du filtrage contraint.

L'unification contrainte est de l'unification modulo la congruence ~, avec vérification des contraintes sur les images des méta-variables.

Définition 96 : Soit UNIF-contraint l'application de T(F,XUMX)xT(F,XUMX) dans l'ensemble des parties de l'ensemble des substitutions de T(F,XUMX) qui a un couple (T,T') associe

- l'ensemble des substitutions $\sigma^{\sim}$ satisfaisant:
  -- $\sigma^{\sim}$ est un unificateur modulo ~ de T et T'
  -- pour toute méta-variable Z de MV(T), $\sigma^{\sim}(Z)$ appartient à DOM

si T est un méta-terme et T' un terme .

- l'ensemble vide sinon.

Tout élément de UNIF-contraint(T,T') est appelé un **unificateur contraint** de T et T'. ○

On définit, comme dans le chapitre 2, la notion d'ensemble générateur de l'ensemble des unificateurs contraints.

Définition 97 : Soit Unif-contraint l'application qui associe à tout couple (T,T') de T(F,XUMX)xT(F,XUMX) un **ensemble complet d'unificateurs contraints** i.e. un ensemble d'unificateurs contraints, complet relativement à UNIF-contraint et à Filtre-contraint. ○

Nous sommes maintenant en mesure de définir la surréduction contrainte.

Définition 98 : La relation de **surréduction contrainte** est définie par:

$$t \text{ -}\hat{}\text{->>}[u,\sigma^{\sim},G\text{->}D]\ t'$$

si et seulement si

- $\sigma^{\sim}$ appartient à Unif-contraint($G, t_{|u}$)
- $t' = t[u <- \sigma^{\sim}(D)]$.

$\sigma^{\sim}$ est appelée **substitution surréductrice contrainte**. ○

Remarque : Notons que si σ appartient à Unif-contraint(T,T'), $\sigma(T)$ et $\sigma(T')$ sont des termes de $T(F,X)$. Par conséquent t' est encore un terme de $T(F,X)$.

Remarque : Il est facile de voir que

si t -^->>$[u,\sigma^{\sim},G->D]$ t', alors $\sigma^{\sim}(t)$ -->>$[u,\sigma^{\sim},G->D]$ $\sigma^{\sim}(t')$.

Le processus de surréduction contrainte permet de calculer un ensemble complet de A-unificateurs de deux termes t et t'.

Théorème 20 : Soit S l'ensemble des substitutions σ telles qu'il existe une surdérivation contrainte issue de t==t'

$$t==t' \text{ -^->>}[\sigma_1]\ t_1==t'_1 \text{ -^->> } \ldots\ t_{n-1}==t'_{n-1} \text{ -^->>}[\sigma_n]\ t_n==t'_n$$

avec $\sigma = \beta.\sigma_n.\ldots.\sigma_1$

et β dans un ensemble complet d'unificateurs modulo ~ de t_n et t'_n.

Alors S est un ensemble complet de A-unificateurs de t==t'.

Preuve: Elle reprend point par point la preuve du résultat du corollaire 7.

● 1ere étape: s'assurer que la surréduction contrainte ne fait pas sortir de l'ensemble des A-solutions. Il faut montrer que pour toute substitution surréductrice contrainte σ qui surréduit (t==t') en ($t_1==t'_1$), et tout A-unificateur α de t_1 et t'_1, alors $\alpha.\sigma$ est une A-solution de t==t'. Or, par définition de ces deux substitutions, on a

$$\alpha.\sigma(t) \text{ -*->> } \alpha(t_1) \sim^A \alpha(t'_1) \text{ <<-*- } \alpha.\sigma(t')$$

et par conséquent $\alpha.\sigma$ est une A-solution de t==t' puisque <<<-*->>> coincide avec $\sim^A$.

● 2eme étape: la preuve de la réciproque nécessite de mettre en évidence le lien entre surréduction contrainte et réduction contrainte. Soit t_0 un terme et ρ une substitution normalisée pour -->>> telle que $\rho(t_0)$ -->>$[u,G->D]$ t'_1.

Alors il existe des substitutions contraintes σ et μ telles que :

* t_0 -^->>$[u,\sigma,G->D]$ t_1
* $\mu(t_1) \sim t'_1$
* $D(\mu) = V(t_1)$
* $\rho \sim \mu.\sigma\ [V(t_0)]$

Ce qu'on peut schématiser par

$$\begin{array}{ccc} t_0 & \text{-^->>}[u,\sigma,G->D] & t_1 \\ \downarrow & & \downarrow \\ \rho(t_0) & \text{-->>}[u,G->D] & t'_1 \end{array}$$

Montrons d'abord que t_0 est surréductible pour -^->>. $\rho(t_0)$ étant réductible par -->> à l'occurrence u par la méta-règle G->D, que l'on supposera telle que $V(G) \cap V(\rho(t_0)) = \emptyset$, il existe une substitution γ telle que

$$\gamma \in \text{Filtre-contraint}(G, \rho(t_0)_{|u}).$$

Mais ρ étant normalisée pour -->>> et $V(G) \cap V(\rho(t_0)) = \emptyset$,

$$\gamma \in \text{Filtre-contraint}(G, \rho(t_{0|u})) .$$

Comme le domaine de ρ ne contient pas de méta-variables, et que γ et ρ peuvent être choisies de telle sorte que leurs domaines soient

disjoints,

$$\gamma.\rho \in \text{UNIF-contraint}(G, t_{0|u}).$$

Par conséquent il existe σ dans Unif-contraint$(G, t_{0|u})$ et une substitution δ telles que

$$\delta.\sigma \sim \gamma.\rho \ [MV(G) \cup V(t_0)]$$

et t_0 est bien surréductible par -^->>.

$$t_0 \ -\hat{}->>[u,\sigma,G->D] \ t_1 = \sigma(t_0[u \leftarrow D])$$

Définissons maintenant μ par $\mu = \delta_{|V(t_1)}$. Le domaine de μ ne contient donc pas de méta-variable et on a bien

$$\rho \sim \mu.\sigma \ [V(t_0)]$$

Montrons enfin que $\mu(t_1) \sim t'_1$. En effet,

$\mu(t_1) = (\mu.\sigma)(t_0[u \leftarrow D]) = \mu.\sigma(t_0)[u \leftarrow \mu.\sigma(D)] \sim \rho(t_0)[u \leftarrow \rho(D)] \sim t'_1$.

● 3eme étape: montrons que tout A-unificateur α de t et t' est soit un unificateur modulo $\sim$, soit est une A-solution obtenue par surréduction contrainte à partir de $(t==t')$. Soit ρ la normalisée de α pour -->>>. Puisque <<<-*->>> coincide avec $\sim^A$, ρ est bien encore une A-solution de $(t==t')$.

Si $\rho(t) \sim \rho(t')$ la conclusion est claire.

Sinon, il faut montrer que, pour toute A-solution α de $t==t'$, il existe une substitution σ qui surréduit $t==t'$ en $t_1==t'_1$ par -^->> et une A-solution μ de $t_1==t'_1$ telle que $\alpha \sim^A \mu.\sigma$.

Puisque $\rho(t) \sim^A \rho(t')$ implique $\rho(t)$ <<<-*->>> $\rho(t')$ et que -->>> a

la propriété de Church-Rosser, au moins l'un des deux termes $\rho(t)$ ou $\rho(t')$ doit être réductible pour -->>. C'est à dire que

$$\rho(t)==\rho(t') \ -->> \ t_0==t'_0.$$

On peut donc appliquer la propriété précédente sur le terme $\rho(t)==\rho(t')$. Il existe donc une substitution normalisée μ telle que

$$t==t' \ -\hat{}->> \ t_1==t'_1$$

avec $\mu(t_1) \sim t_0$ et $\mu(t'_1) \sim t'_0$ et $\alpha \sim^A \rho \sim \mu.\sigma$. Comme de plus

$$\mu(t_1) \sim t_0 \sim^A \rho(t) \sim^A \rho(t') \sim^A t'_0 \sim \mu(t'_1)$$

μ est une A-solution de $t_1==t'_1$.

● 4eme étape: on est maintenant en mesure de prouver le résultat annoncé.

Si σ est un élément de S, il est clair que c'est une A-solution de $t==t'$.

Si α est une A-solution finie de $t==t'$, montrons par l'absurde que α s'écrit sous la forme d'un produit fini

$$\alpha \sim^A \beta.\sigma_{n-1}\cdots\sigma_0$$

avec β unificateur modulo $\sim$ de t_n et t'_n. Si ce n'était pas le cas, en posant

$$\sigma = \prod_{i=1}^{\infty} \sigma_i$$

$\sigma(t==t')$ serait réductible une infinité de fois par -->>, ce qui contredit le fait que -->>> est noethérien.

Soit ρ appartenant à un ensemble complet d'unificateurs modulo $\sim$ de

$(t_n == t'_n)$ tel que $\rho \leq \beta$ $[V(t_n == t'_n)]$. On a donc

$$\rho.\sigma_{n-1}\cdots\sigma_0 \leq \beta.\sigma_{n-1}\cdots\sigma_0 =_A \alpha$$

Ce qui prouve que S est un ensemble complet de A-solutions de t==t'. □

Exemple 65 : Associativité et idempotence

Reprenons l'exemple de la théorie définie par les axiomes suivants:

f(f(x,y),z) = f(x,f(y,z))
f(x,x) = x

L'équivalence ~ est celle engendrée par le système de réécriture

R_0 : f(f(x,y),z) -> f(x,f(y,z))

L'ensemble des structures est l'ensemble des "peignes" a' variables toutes distinctes:

$$S = \{x_1,\ f(x_1,x_2),\ f(x_1,f(x_2,x_3)),\ f(x_1,f(x_2,f(x_3,x_4)))\ldots\}$$

Le système de réécriture infini engendré est correctement schématisé par la méta-règle f(X,X) -> X. Puisque -->>> est convergente, il est donc possible de résoudre dans cette théorie des équations par surréduction contrainte. Ici la surréduction contrainte coincide avec la surréduction modulo R_0 puisque la méta-règle est sans contraintes. Malheureusement, il existe des ensembles minimaux d'unificateurs modulo l'associativité qui sont infinis et suivant le type d'équation que l'on veut résoudre, il est possible que la procédure d'unification associative engendre de tels ensembles. Il sera donc nécessaire dans de tels cas de schématiser si possible l'ensemble des solutions.

Considérons par exemple l'équation

f(f(x,y),a) == f(b,z)

La surréduction contrainte

f(f(x,y),a) == f(b,z) -^->>[(X←f(y,a))(x←f(y,a))] f(y,a) == f(b,z)

conduit à la solution (x←f(b,a))(y←b)(z←a).

Une deuxième surréduction contrainte

f(f(x,y),a) == f(b,z) -^->>[(X←y)(x←y)] f(y,a) == f(b,z)

conduit à la solution (x←b)(y←b)(z←a). ∇

5.2. Méta-surréduction

Supposons maintenant que l'on dispose d'un système de réécriture équationnelle convergent (R,E) équivalent à la théorie $\sim^A$, ou d'un système de méta-règles MR schématisant le système de réécriture (équationnelle) engendré par complétion de l'ensemble d'axiomes A. La première situation peut bien évidemment être considérée commme un cas particulier de la seconde.

Lorsque la procédure de superréduction engendre une infinité d'équations (t==t') qui se répartissent en un nombre fini de suites d'équations schématisables, on introduit comme pour la complétion un ou plusieurs types de méta-variables avec leur domaines associés. Le but de la méta-surréduction est de schématiser l'arbre de surréduction en y remplaçant une infinité d'équations par une méta-équation, i.e. une équation sur des méta-termes.

Rappelons que Méta-UNIF(T,T') est l'ensemble des méta-unificateurs de T et T', et que Méta-Unif(T,T') est un ensemble complet de méta-unificateurs de T et T'.

Lemme 28 : Soient T et T' deux méta-termes et $\sigma^{\sim}$ un méta-unificateur de T et T'. Pour toute instanciation principale ψ' de $\sigma^{\sim}(T)$ et $\sigma^{\sim}(T')$, il existe une instanciation principale ψ de T et T' et une substitution σ telles que $\sigma.\psi \sim \psi'.\sigma^{\sim}$ sur $MV(T) \cup MV(T')$.

Preuve: Par définition d'un méta-unificateur, $\psi'(\sigma^{\sim}(T)) \sim \psi'(\sigma^{\sim}(T'))$. Notons ϕ la substitution $\psi'.\sigma^{\sim}$. ϕ est une instanciation des méta-variables de T et T' et par décomposition d'une instanciation (chapitre 3, corollaire 1), appliquée au couple (T,T'), il existe une substitution γ de T(F,X) et une instanciation principale ψ des méta-variables de T et T' telles que $\phi \sim \gamma.\psi$ $[MV(T) \cup MV(T')]$. On peut alors écrire $\psi'.\sigma^{\sim}(Z) \sim \gamma.\psi(Z)$ pour tout Z de $MV(T) \cup MV(T')$. D'où en posant $\sigma = \gamma$, $\psi'.\sigma^{\sim}(Z) \sim \sigma.\psi(Z)$ pour tout Z de $MV(T) \cup MV(T')$. □

Comme dans le cas du filtrage, il existe un lien entre Méta-Unif et UNIF-contraint: à tout méta-unificateur correspond une infinité d'unificateurs contraints.

Proposition 14 : Soit t un terme et T un méta-terme. Soit $\sigma^{\sim}$ appartenant à Méta-Unif(T,t). Pour toute instanciation principale ψ' de $\sigma^{\sim}(T)$, $\psi.\sigma^{\sim}$ appartient à UNIF-contraint(T,t).

Preuve: Puisque $\sigma^{\sim}(T) \equiv \sigma^{\sim}(t)$, pour toute instanciation principale ψ' de

$\sigma^{\sim}(T)$, $\psi'.\sigma^{\sim}(T) \sim \psi'.\sigma^{\sim}(t)$. De plus d'après le lemme du méta-unificateur, $\psi'.\sigma^{\sim}(Z) \sim \sigma.\psi(Z)$ et donc $(\psi'.\sigma^{\sim}(Z))$ appartient à DOM. On en déduit que $\psi'.\sigma^{\sim}$ appartient à UNIF-contraint(T,t). □

Remarquons que si t et t' sont tous deux des termes et si $\sigma^{\sim}$ appartient à Méta-Unif(t,t'), alors $\psi.\sigma^{\sim}$ est un unificateur modulo ~ de t et t'.

Comme dans le cas de la complétion de méta-règles, nous élargissons le problème à l'ensemble des méta-termes. Ici, ce n'est pas une propriété de Church-Rosser qui nous intéresse, mais le problème de l'unification dans la théorie engendrée par MR sur les méta-termes.

Définition 99 : $T \,|\equiv|_{MR}\, T'$ si et seulement si il existe G=D dans MR considéré comme ensemble d'équations, une occurrence u de $dom(T) \cap dom(T')$ et une substitution $\sigma^{\sim}$ dans Méta-Filtre(G,$T_{|u}$), telles que $T_{|u} \equiv \sigma^{\sim}(G)$ et $T'_{|u} \equiv \sigma^{\sim}(D)$.

On note $\equiv_{MR}$ la fermeture réflexive, symétrique et transitive de $|\equiv|_{MR}$. ○

Lemme 29 : Si $T \equiv_{MR} T'$, pour toute instanciation principale ψ, $\psi(T) \sim^{A} \psi(T')$.

Preuve: Elle résulte d'une part du fait que pour toute G->D appartenant à MR et toute instanciation principale ψ, $\psi(G) \sim^{A} \psi(D)$. □

Définition 100 : Une substitution $\sigma^{\sim}$ de $T(F, X \cup MX)$ est un **MR-méta-unificateur** de T et T' si et seulement si $\sigma^{\sim}(T) \equiv_{MR} \sigma^{\sim}(T')$ et pour toute méta-variable Z de $D(\sigma^{\sim})$, pour toute instanciation principale ψ, $\psi.\sigma^{\sim}(Z)$ appartient à DOM.

On dira aussi que $\sigma^{\sim}$ est une **MR-méta-solution** de l'équation T==T'. O

Remarque : Si $\sigma^{\sim}$ est un MR-méta-unificateur de T et T', alors pour toute instanciation principale ψ, $\psi.\sigma^{\sim}(T) \sim^{A} \psi.\sigma^{\sim}(T')$.

On associe à la relation de méta-réduction sur les méta-termes une relation de méta-surréduction dans laquelle les notions d'égalité, filtrage et unification sont remplacées par celles de méta-égalité, méta-filtrage, méta-unification.

Définition 101 : La relation de **méta-surréduction** est définie par:

$$T \; =\hat{}=>[u,\sigma^{\sim},G->D] \; T'$$

si et seulement si

- $\sigma^{\sim}$ appartient à Méta-Unif(G,$T_{|u}$)
- T' = T[u<-$\sigma^{\sim}$(D)]. O

A partir d'une MR-méta-solution d'une équation (t==t'), où t et t' sont des termes de T(F,X), on retrouve par instanciation des méta-variables un ensemble complet de A-solutions de l'équation t = t'.

Théorème 21 : Soit S l'ensemble des substitutions $\sigma^{\sim}$ telles qu'il existe une méta-surdérivation issue de t==t'

$$t==t' \; =\hat{}=>[\sigma_1] \; T_1==T'_1 \; =\hat{}=> \; \ldots \; =\hat{}=>[\sigma_n] \; T_n==T'_n$$

avec T_n et T'_n méta-unifiables,

$\alpha^{\sim}$ dans Méta-Unif($T_n==T'_n$),

et $\sigma^{\sim} = \alpha^{\sim}.\sigma_n.....\sigma_1$.

Alors $\Psi(S) = \{\psi.\sigma^{\sim}$ pour toute $\sigma^{\sim}$ dans S et toute instanciation principale ψ

des méta-variables de $I(\sigma^{\sim})\}$ est un ensemble complet de A-unificateurs de t et t'.

Preuve: La technique de preuve est encore similaire à celle du résultat du corollaire 7.

● lere étape: Il faut montrer que pour toute substitution surréductrice $\sigma^{\sim}$ qui méta-surréduit (T==T') en ($T_1==T'_1$), et tout MR-méta-unificateur $\alpha^{\sim}$ de T_1 et T'_1, alors $\alpha^{\sim}.\sigma^{\sim}$ est une MR-méta-solution de T==T'. Or par définition de ces deux substitutions on a

$$\alpha^{\sim}.\sigma^{\sim}(t) \; =*=> \; \alpha^{\sim}(t_1) \equiv MR \; \alpha^{\sim}(t'_1) \; <=*= \; \alpha^{\sim}.\sigma^{\sim}(t').$$

D'autre part, pour toute méta-variable Z et pour toute instanciation ψ, $\psi.\alpha^{\sim}.\sigma^{\sim}(Z)$ appartient à DOM car $\psi.\alpha^{\sim}.\sigma^{\sim}(Z) = \phi.\sigma^{\sim}(Z)$ où ϕ est une instanciation qui, par décomposition d'une instanciation, peut se mettre sous la forme $\sigma.\psi'$. Donc $\psi.\alpha^{\sim}.\sigma^{\sim}(Z) \sim \sigma.\psi'.\sigma^{\sim}(Z)$ appartient à DOM. Par conséquent $\alpha^{\sim}.\sigma^{\sim}$ est une MR-méta-solution de T==T'.

● 2eme étape: Soit T_0 un méta-terme et $\rho^{\sim}$ une substitution normalisée pour ==>>, satisfaisant pour toute méta-variable Z et toute instanciation principale ψ, $\psi.\rho^{\sim}(Z)$ appartient à DOM. Si $\rho^{\sim}(T_0)$ ==>[u,G->D] T'_1, alors il existe des méta-substitutions $\sigma^{\sim}$ et $\mu^{\sim}$ telles que :

* $T_0 \; =\hat{}=>[u,\sigma^{\sim},G->D] \; T_1$
* $\mu^{\sim}(T_1) \equiv T'_1$
* $D(\mu^{\sim}) = MV(T_1)$
* $\rho^{\sim} \equiv \mu^{\sim}.\sigma^{\sim} \; [MV(T_0)]$
* $\mu^{\sim}$ est mormalisée pour ==>> et satisfait la condition: pour

toute méta-variable Z et toute instanciation principale ψ, $\psi.\mu^{\sim}(Z)$ appartient à DOM.

Ce qu'on peut schématiser par

$$\begin{array}{ccc} T_0 & =\hat{}=>[u,\sigma^{\sim},G->D] & T_1 \\ \downarrow & & \downarrow \\ \rho^{\sim}(T_0) & ==>[u,G->D] & \equiv T'_1 \end{array}$$

Montrons d'abord que T_0 est méta-surréductible. $\rho^{\sim}(T_0)$ étant méta-réductible à l'occurrence u par la méta-règle G->D que l'on supposera telle que $MV(G) \cap MV(\rho^{\sim}(t_0)) = \emptyset$, il existe une substitution $\gamma^{\sim}$ telle que

$$\gamma^{\sim} \in \text{Méta-Filtre}(G,\rho^{\sim}(T_0)_{|u}).$$

Mais $\rho^{\sim}$ étant normalisée pour ==>> et $MV(G) \cap MV(\rho^{\sim}(T_0)) = \emptyset$,

$$\gamma^{\sim} \in \text{Méta-Filtre}(G,\rho^{\sim}(T_{0|u}))$$

Comme $\rho^{\sim}$ et $\gamma^{\sim}$ satisfont chacune la condition: pour toute méta-variable Z, pour toute instanciation principale ψ, $\psi(\rho^{\sim}(Z))$ et $\psi(\gamma^{\sim}(Z))$ appartient à DOM, et que $\gamma^{\sim}$ et $\rho^{\sim}$ peuvent être choisies de telle sorte que leurs domaines soient disjoints,

$$\gamma^{\sim}.\rho^{\sim} \in \text{Méta-UNIF}(G,T_{0|u}).$$

Par conséquent il existe $\sigma^{\sim}$ dans $\text{Méta-Unif}(G,T_{0|u})$ et une substitution $\delta^{\sim}$ telles que

$$\delta^{\sim}.\sigma^{\sim} \equiv \gamma^{\sim}.\rho^{\sim} \ [MV(G) \cup MV(T_0)]$$

De plus $\delta^{\sim}$ satisfait aussi la condition: pour toute méta-variable Z

et toute instanciation principale ψ, $\psi(\delta^{\sim}(Z))$ appartient à DOM. T_0 est bien méta-surréductible.

$$T_0 =\hat{}=>[u,\sigma^{\sim},G->D] \ T_1 = \sigma^{\sim}(T_0[u\leftarrow D])$$

Définissons maintenant $\mu^{\sim}$ par $\mu^{\sim} = \delta_{|MV(T_1)}$. On obtient

$$\rho^{\sim} \equiv \mu^{\sim}.\sigma^{\sim} \ [MV(T_0)] \text{ et } \gamma^{\sim} \equiv \mu^{\sim}.\sigma^{\sim} \ [MV(G)]$$

Notons que $\mu^{\sim}$ est normalisée pour ==>> puisque $\rho^{\sim}$ l'est. Montrons enfin que $\mu^{\sim}(T_1) \equiv T'_1$. En effet,

$\mu^{\sim}(T_1) = (\mu^{\sim}.\sigma^{\sim})(T_0[u\leftarrow D]) = \mu^{\sim}.\sigma^{\sim}(T_0)[u\leftarrow\mu^{\sim}.\sigma^{\sim}(D)]$

$\equiv \rho^{\sim}(T_0)[u\leftarrow\gamma^{\sim}(D)] = T'_1$.

● 3eme étape: montrons que tout MR-méta-unificateur $\rho^{\sim}$ normalisé pour ==> de T et T' est soit un méta-unificateur, soit une MR-méta-solution obtenue par méta-surréduction à partir de (T==T').

Si $\rho^{\sim}(T) \equiv \rho^{\sim}(T')$, la conclusion est claire.

Sinon, puisque $\rho^{\sim}(T) \equiv_{MR} \rho^{\sim}(T')$ et que ==>> a la propriété de Church-Rosser, au moins l'un des deux termes $\rho^{\sim}(T)$ ou $\rho^{\sim}(T')$ doit être méta-réductible. C'est à dire que

$$\rho^{\sim}(T)==\rho^{\sim}(T') ==> T_0==T'_0.$$

On peut donc appliquer la propriété précédente sur le terme $\rho^{\sim}(T)==\rho^{\sim}(T')$. Il existe donc une méta-substitution normalisée $\mu^{\sim}$ telle que

$$T==T' =\hat{}=> T_1==T'_1$$

avec $\mu^{\sim}(T_1) \equiv T_0$ et $\mu^{\sim}(T'_1) \equiv T'_0$ et $\rho^{\sim} \equiv \mu^{\sim}.\sigma^{\sim}$. Comme de plus

$$\mu^{\sim}(T_1) \equiv T_0 \equiv_{MR} \rho^{\sim}(T) \equiv_{MR} \rho^{\sim}(T') \equiv_{MR} T'_0 \equiv \mu^{\sim}(T'_1),$$

$\mu^{\sim}$ est une MR-méta-solution de $T_1 == T'_1$.

● 4eme étape: Si $\rho^{\sim}$ est une MR-méta-solution finie de T==T', montrons par l'absurde que α s'écrit sous la forme d'un produit fini

$$\rho^{\sim} \equiv \beta^{\sim}.\sigma_{n-1}\cdots\sigma_0$$

avec $\beta^{\sim}$ appartenant à Méta-UNIF(T_n,T'_n). Si ce n'était pas le cas, en posant

$$\sigma = \prod_{i=1}^{\infty} \sigma_i$$

$\sigma(T==T')$ serait réductible une infinité de fois par ==>, ce qui contredit le fait que ==>> est noethérien.

Soit $\delta^{\sim}$ un élément de Méta-Unif(T_n,T'_n) satisfaisant

$\delta^{\sim} \leq \beta^{\sim}$ [MV(T_n) $\cup$ MV(T'_n)]. On a donc

$$\delta^{\sim}.\sigma_{n-1}\cdots\sigma_0 \leq \beta^{\sim}.\sigma_{n-1}\cdots\sigma_0 \equiv \rho^{\sim}$$

● 5eme étape: nous sommes maintenant en mesure de prouver le résultat annoncé.

Si $\sigma^{\sim}$ est un élément de Σ, il est clair que c'est une MR-méta-solution de t==t'. Donc pour toute instanciation principale ψ, $\psi.\sigma^{\sim}(t) \sim^A \psi.\sigma^{\sim}(t')$.

Si α est une A-solution finie de t==t', soit ρ la normalisée de α pour ==>> ou de façon équivalente pour -->>>. D'après ce qui précède, $\rho \equiv \gamma^{\sim}.\sigma^{\sim}$ [V(t) $\cup$ V(t')] avec $\sigma^{\sim}$ appartenant à S. Donc pour toute instanciation principale ψ, $\alpha \sim^A \rho \sim^A \psi.\gamma^{\sim}.\sigma^{\sim}$ [V(t) $\cup$ V(t')]. De plus $\psi.\gamma^{\sim}$ est une instanciation des méta-variables de $\sigma^{\sim}(t)$ et $\sigma^{\sim}(t')$, et par décomposition d'une

instanciation, est ~-équivalente à $\gamma.\psi'$ sur MV($\sigma^{\sim}(t)$) $\cup$ MV($\sigma^{\sim}(t')$). Donc $\alpha \sim^A \gamma.\psi'.\sigma^{\sim}$ [V(t) $\cup$ V(t')], ce qui prouve la complétude de $\Psi(S)$. □

Cette technique a été utilisée avec succès pour résoudre les équations dans les arbres signés.

Exemple 66 : La résolution par surréduction modulo E de l'équation

f(x,f(x,f(a,y)))=x

dans la théorie des arbres signés avec

E : -(-(x)) = x
-(f(x,y)) = f(-(y),-(x))
-(h(x)) = h(-(x))

R : f(-x,f(x,y)) -> y
f(f(y,x),-x) -> y

engendre par superposition modulo E de la règle f(-x,f(x,y)) -> y dans

f(x,f(x,f(a,y))) == x

la famille infinie d'équations:

f(a,y) == $f(x_1,-x_1)$

f(a,y) == $h(f(x_1,-x_1))$

f(a,y) == $h(h(f(x_1,-x_1)))$

f(a,y) == $h(h(h(f(x_1,-x_1))))$

...

l'ensemble des structures est l'ensemble des solutions minimales de l'équation x=-x modulo la théorie E des arbres signés. La congruence ~ est ici l'égalité engendrée par E. Un algorithme de méta-unification a été

proposé dans [Kirchner1982]. La méta-surréduction

f(x,f(x,f(a,y))) == x =^=>[(x←X)] f(a,y) == X =^=>[(y←f(-a,z)] z == X

conduit à la méta-solution (x←X)(z←X)(y←f(-a,z)) qui donne par instanciation principale de X une infinité de A-solutions.

Notons qu'une autre solution est obtenue par

f(x,f(x,f(a,y))) == x =^=>[(x←-a)] f(-a,y) == -a =^=>[(y←f(a,z)] z == -a

C'est la solution (x←-a)(z←-a)(y←f(a,-a)). ∇

Donnons un autre exemple, utilisant l'unification associative.

Exemple 67 : Supposons que l'on veuille résoudre des équations dans la théorie définie par les axiomes suivants:

```
(x*z)+(a*z) = (x+a)*z
(x*y)*z = x*(y*z)
```

où a est une constante. Orientées de gauche à droite, ces équations engendrent un système de réécriture infini, à cause de la superposition de la règle d'associativité dans la règle de distributivité. Les premières règles sont:

```
(x*z)+(a*z) -> (x+a)*z
(x*(y*z))+(a*z) = ((x*y)+a)*z
(x*(y*(y'*z)))+(a*z) = ((x*(y*y'))+a)*z
 ...
```

Les techniques dévéloppées au chapitre 3 permettent de shématiser ce système en prenant pour ~ l'équivalence engendrée par l'associativité, pour domaine l'ensemble des termes tout entier et pour ensemble de méta-règles

```
(X*z)+(a*z) -> (X+a)*z
```

Comme cet ensemble de méta-règles est sans contraintes, on peut utiliser la

complétion équationnelle (modulo ~) pour prouver sa convergence.

Supposons maintenant que l'on veuille résoudre l'équation

```
y+(x*a) == (a+a)*x
```

dans cette théorie. La surréduction contrainte coincide ici avec la surréduction équationnelle (modulo l'associativité) puisque les méta-règles sont sans contraintes. En unifiant modulo l'associativité les termes y+(x*a) et (X*z)+(a*z), on obtient un ensemble complet infini d' unificateurs

```
Σ = {(x←a)(y←X*a)(z←a),
     (x←a*a)(y←X*(a*a))(z←a*a),
     (x←a*(a*a))(y←X*(a*(a*a)))(z←a*(a*a))
      ... }.
```

On introduit donc un nouvel ensemble de méta-variables MY, dont le domaine est l'ensemble des solutions de l'équation (x*a == a*x) modulo l'associativité et dont l'ensemble des structures est

```
S = {a, a*a, a*(a*a), a*(a*(a*a)), ... }.
```

La méta-surréduction de (y+(x*a) == (a+a)*x) fournit alors l'équation ((X+a)*Y == (a+a)*Y) avec la méta-substitution (x←Y)(y←X*Y)(z←Y). Les deux membres de l'équation se méta-unifient par (X←a). On obtient alors la méta-solution (y←a*Y)(x←Y), qui engendre par instanciations principales de Y, les solutions

```
{(y←a*a)(x←a),
 (y←a*(a*a))(x←a*a),
 (y←a*(a*(a*a)))(x←a*(a*a)),
 ... }
```

∇

6. Extensions

Il est naturel d'envisager l'extension de ce language de programmation restreint à d'autres prédicats. Il donne alors naissance à ce que Dershowitz appelle la programmation par "rewrite program" [Dershowitz1984,Dershowitz1984a]. Un "rewrite program" est un ensemble fini de règles de réécriture convergent qui décrit équationnellement l'univers du problème. Un tel système est utilisé comme un programme déterministe pour calculer un ensemble de valeurs de sortie qui satisfont un but donné. Plus précisément, étant donné un prédicat P, on cherche un ensemble [t] de valeurs (qui sont des termes clos irréductibles) telles que, pour tout ensemble [s] de valeurs données, P([s],[t]) soit vrai. P est codé par un symbole fonctionnel p tel que, si P([s],[t]) est vrai, alors p([s],[t]) s'évalue en vrai à l'aide du "rewrite program". Cette propriété est appelée propriété de correction par Dershowitz. La satisfiabilité de P pour [s] donné revient à trouver les solutions de l'équation p([s],[z]) = vrai dans la théorie équationnelle définie par les règles du "rewrite program". Calculer avec un "rewrite program" consiste à appliquer une procédure de complétion linéaire à ce programme et à une règle-but qui s'écrit p([x],[z]) -> ans([z]), où p est le terme d'appel contenant les valeurs d'entrée [x] qui sont des termes clos irréductibles, [z] est un vecteur de variables de sortie et le symbole de fonction ans est utilisé pour stocker le résultat, c'est-à-dire les valeurs des variables satisfaisant le prédicat P. Les règles du programme sont superposées dans la règle-but et dans celles qui en sont ainsi dérivées. Ni les règles dérivées, ni les règles du programme ne sont superposées entre elles.

Ces travaux montrent la puissance de la procédure de complétion et son

intérêt pour la construction d'un interprète d'un langage de programmation basé sur la logique équationnelle. Cependant le fait d'avoir à coder les prédicats est souvent un obstacle à la compréhension et à l'écriture des programmes. De plus il faut savoir vérifier la correction du codage ce qui, en pratique, peut être difficile. Enfin se posent les problèmes de divergence de la complétion qui constituent un obstacle majeur pour un tel langage. Les diverses solutions que nous avons proposées pour réduire l'arbre de recherche des solutions, prennent là toute leur importance.

CONCLUSION

CONCLUSION

L'étude des systèmes de réécriture a pris ces dernières années un essor important et leur champ d'application s'est considérablement élargi, en particulier dans le domaine de la démonstration automatique [Buchberger1985]. Face à cette diversité, il s'agit de faire le bilan de notre apport personnel et de dégager les perspectives ouvertes par ce travail.

Nous sous sommes restreints, dans cette thèse, aux preuves en logique équationnelle. Notre apport majeur dans ce domaine est d'élargir la classe des théories équationnelles dans lesquelles les preuves par complétion s'appliquent. Cet élargissement s'est fait à la fois par l'incorporation des équations non orientables dans le processus de complétion, et par la proposition d'outils permettant de traiter les divergences.

Un système basé sur la complétion se heurte toujours à deux problèmes majeurs, celui de l'orientation des paires de termes et celui de la non terminaison du processus.

- Les paires de termes non orientables sont des équations permutatives (possédant les mêmes ensembles de variables dans les deux membres). En pratique, ces équations engendrent le plus souvent des classes d'équivalence finies. Dans le cas où l'on connait un algorithme complet d'unification, la complétion équationnelle permet d'incorporer ces équations dans les opérations de filtrage et d'unification. Cela met en évidence l'importance de savoir construire de façon systématique des algorithmes de filtrage et unification dans les théories équationnelles. Mais en outre apparait alors le problème de la preuve de terminaison de la réécriture équationnelle, c'est-à-dire le problème de trouver un ordre permettant d'orienter les

règles en tenant compte des équations.

- Dans le cas de la génération d'un ensemble infini de règles, nous avons tenté une première approche pour le résoudre par l'introduction de méta-règles schématisant des familles de règles. Pour découvrir les méta-règles, nous avons proposé une méthode d'inférence, mais il semble possible d'en trouver d'autres. Par ailleurs, le problème de la complétion de méta-règles reste ouvert dans le cas général. Nous savons le résoudre dans le cas de méta-règles sans contraintes, où nous avons montré qu'il est équivalent dans de nombreux cas, à une complétion équationnelle. Cependant il faudrait, pour traiter en pratique plus de cas de divergence, prendre en compte, dans la complétion équationnelle, des équations engendrant des classes d'équivalence infinies. Soulignons par ailleurs que le traitement des divergences à l'aide de méta-règles permet aussi de travailler avec des théories dans lesquelles il existe des ensembles complets infinis d'unificateurs, comme par exemple l'associativité.

Mais les preuves par complétion ont encore bien des perspectives intéressantes d'extension:

- les résultats que nous avons obtenus dans le domaine de la logique équationnelle ont aussi des répercussions dans le domaine des preuves en logique du premier ordre.

- un atout majeur des preuves par complétion est qu'elles s'implantent de manière uniforme, tout en présentant une grande diversité de fonctionnalités.

- en ce sens, les preuves par complétion constituent des outils intéressants dans les langages de programmation logique, car leur mécanisme d'inférence est à la base à la fois du prouveur de théorèmes et de

l'interprète.

● Extension à la preuve en logique du premier ordre

Lorsqu'on veut sortir du cadre assez limité des raisonnements équationnels, il est naturel de chercher à faire des preuves en logique des prédicats du premier ordre.

Une première idée développée par N.Dershowitz[Dershowitz1984], est de coder les prédicats par des symboles de fonctions à valeur booléennes et de se ramener ainsi à la logique équationnelle. Bien que séduisante pour nous, cette idée n'est pourtant pas très réaliste dans le cadre d'un système de preuve interactif, car ce codage exige de l'utilisateur une grande expertise. De plus, pour avoir assez de puissance d'expression, il faut pouvoir utiliser des équations et des règles conditionnelles [Dershowitz1984b], dont l'intégration dans un mécanisme de complétion est un problème encore mal résolu à ce jour [Remy1984,Kaplan1984].

Par ailleurs, le champ d'application des preuves par complétion a déjà été étendu à la preuve de théorèmes en logique des prédicats du premier ordre, en s'appuyant sur la procédure de complétion équationnelle. Les premiers travaux concernant l'utilisation des techniques de réécriture en logique du premier ordre sont dus à J.Hsiang [Hsiang1983]. Sa méthode procède par réfutation, en niant la formule à prouver, et en la transformant en un ensemble de clauses S={C}, puis en un ensemble de règles {C->vrai}. La preuve de l'inconsistance d'un ensemble de clauses S dans une théorie équationnelle définie par un système de réécriture équationnelle convergent, utilise un système de réécriture équationnelle

convergent pour les algèbres booléennes et une stratégie de calcul de paires critiques dont le but est d'engendrer l'équation (vrai=faux). Citons également les travaux de F.Fages [Fages1983] et de L.Fribourg [Fribourg1985] sur le sujet.

La justification de l'utilisation des techniques de réécriture et de complétion a été clarifiée par ailleurs par un travail de E.Paul [Paul1984a] qui a proposé une nouvelle interprétation du principe de résolution en termes de preuve par complétion.

La comparaison entre résolution et procédure de complétion dans le but de mieux utiliser le mécanisme de réduction, a également été poursuivie, par Kapur et Narendran [Kapur1984a] d'une part et J.Hsiang [Hsiang1985] d'autre part.

D'autres travaux récents de E.Paul [Paul1985] ont montré comment résoudre le problème de l'égalité dans des théories non équationnelles définies par un ensemble de clauses de Horn, à l'aide de la procédure de complétion. Dans le cadre de ces théories, les preuves inductives par complétion s'étendent à des formules universellement quantifiées en logique des prédicats. De même la résolution de requêtes par superréduction trouve dans ce cadre une possibilité d'extension.

Cependant un inconvénient majeur de la méthode de E.Paul est de ne pas traiter les égalités non orientables en règles de réécriture apparaissant par exemple dans des clauses telles : (x=y) => f(x)=f(y).

J.Hsiang (loc.cit) s'inspirant des techniques de Peterson [Peterson1983], propose un résultat encore plus fort dans le cadre général du

calcul des prédicats du premier ordre avec égalité, montrant par là que les techniques de réécriture ont la même puissance que paramodulation et résolution réunies.

Un des grands avantages de ces méthodes est d'utiliser uniquement les deux règles d'inférence de la procédure de complétion, superposition et normalisation, pour simuler une quantité de règles d'inférence: résolution positive unitaire, superréduction, simplification, destruction par subsumption, destruction des tautologies,

Il semble d'autre part que la complétude de ces méthodes soit reliée à l'utilisation de la procédure de complétion en tant qu'algorithme de semi-décision. C'est là un problème intéressant, mais encore ouvert à l'heure actuelle.

● Uniformité de l'implantation, diversité des fonctionalités:

La deuxième idée importante qui se dégage de nos travaux est qu'une même procédure, celle de complétion équationnelle, peut présenter une grande diversité de fonctionalités. Dans les différentes procédures que nous avons étudiées, les arguments sont toujours un ensemble de paires de termes et un ensemble de règles. Mais nous avons montré comment jouer sur ces ensembles pour obtenir des fonctionnalités différentes de la complétion:

- dans la complétion équationnelle, les règles peuvent être réparties en sous-ensembles correspondant à des types de réécriture équationnelle différents. Cela permet d'obtenir plus d'efficacité dans la réécriture, par exemple en n'utilisant filtrage et unification modulo E que lorsque c'est nécessaire. A l'inverse, utiliser une réécriture plus puissante permet

d'éviter certains bouclages. REVEUR-3 est une première réalisation de cette idée dans le cadre de la réécriture équationnelle.

- dans la complétion inductive, la construction hiérarchique des spécifications induit une hiérarchie sur les ensembles de règles et de paires de termes à orienter. Leur subdivision en sous-ensembles correspondant à cette hiérarchie permet d'exploiter à fond la modularité de la construction d'une spécification. De plus cette modularité transparait alors naturellement au niveau de la preuve.

- dans la procédure de superréduction, en différenciant les règle-buts exprimant les équations à résoudre, des règles du système de réécriture convergent décrivant l'univers du problème, on implante les seules superpositions nécessaires à l'obtention des solutions. Il faut noter toutefois que cette restriction n'est possible que parce que l'on dispose d'un résultat de complétude de la superréduction.

- pour les preuves en logique du premier ordre, les superpositions sont également orientées vers un but bien précis, qui est d'engendrer l'équation vrai=faux. Là encore la restriction des superpositions, basée sur les résultats de complétude des techniques de résolution, doit permettre l'implantation de stratégies complètes. C'est à l'heure actuelle un sujet largement étudié, et il est probable que des résultats ne tarderont pas à être obtenus dans ce domaine.

Tout ceci plaide en faveur d'un système élargissant les idées de REVEUR-3 et s'inspirant du système ERIL [Dick1985] dans lequel l'utilisateur peut définir ses ensembles de règles et de paires de termes à orienter en spécifiant quelles superpositions et réductions sont nécessaires pour chacun.

● Preuves par complétion et programmation logique:

Les différentes procédures de complétion décrites dans cette thèse ont été présentées dans une optique de preuve de théorèmes dans les théories équationnelles. Cependant, l'étude de la superréduction a fait apparaitre un autre point de vue possible, celui de la programmation logique. Nous allons développer l'idée que les travaux présentés ici constituent des outils intéressants pour les langages de programmation logique.

La programmation logique est née de l'idée d'utiliser la logique des prédicats du premier ordre comme un language de programmation. Le terme "langage de programmation logique" désigne tout langage possédant une sémantique basée sur un système logique, que ce soit la logique des prédicats du premier ordre, la logique équationnelle (OBJ [Futatsugi1985]), la logique des clauses de Horn (PROLOG [Kowalski1974]), celle des clauses de Horn équationnelles (SLOG [Fribourg1984a]), ou la logique des clauses de Horn avec égalité (EQLOG [Goguen1984] et autres successeurs de PROLOG). Ce style de programmation présente de gros avantages: clarté, simplicité d'utilisation, réutilisation et maintenance faciles, dus en grande partie au fait que ce sont des langages de très haut niveau.

Pour concevoir une nouvelle génération de langages de programmation possédant un fondement logique rigoureux, il apparait nécessaire que la logique sous-jacente permette de prendre en compte un certain nombre de caractéristiques des languages de programmation et de spécification:

- Des mécanismes de structuration sont nécessaires dans les applications réelles pour fournir la possibilité de développer et valider de grands logiciels. Ils sont fournis par l'encapsulation de code exécutable dans des

modules, la possibilité d'importer et d'instancier des modules précédement développés en utilisant la paramétrisation.

- Les types et la vérification de types sont nécessaires pour détecter certaines erreurs.

- Fonctions et prédicats doivent être disponibles dans le langage. Bien qu'il soit possible de coder toutes les fonctions comme des prédicats ou tous les prédicats comme des fonctions à valeurs booléennes, le langage gagne en clarté si les deux sont disponibles. Il semble en effet naturel d'exprimer avec des prédicats les relations entre objets et avec des fonctions leur structure. De même dans un cadre unifié, on satisfait des requêtes sur les prédicats et on résoud des équations sur les fonctions. Combiner fonctions et prédicats permet en outre de disposer à la fois des avantages de la programmation fonctionnelle du premier ordre et de ceux de la programmation logique.

La conception d'un langage de programmation doit tenir compte de ces exigences. Son implantation nécessite une sémantique opérationnelle précise permettant la construction d'interprètes.

Un interprète peut être vu comme un programme qui dérive des conséquences logiques à partir d'un ensemble de données en utilisant des règles d'inférence. Il est important de savoir prouver des propriétés de correction et de complétude, assurant la conformité des résultats avec ceux de la logique sous-jacente. D'autre part, les constructions faites par l'utilisateur doivent être validées par des preuves dans la logique correspondante. Ainsi la conception d'interprètes de langages de programmation logique est étroitement reliée à celle de prouveur de théorèmes.

En logique équationnelle, la complétion est à la fois ur prouveur de théorèmes permettant de décider de l'égalité dans les théories équationnelles, de faire des preuves par induction, de valider l'enrichissement de théories, et un interprète qui permet de satisfaire des requêtes exprimées sous forme d'équations. Ce qui est remarquable dans cette approche, c'est qu'il existe deux règles d'inférence uniquement, la superposition et la réduction, qui sont à la base du prouveur de théorèmes et de l'interprète. De ce fait, ils s'implantent uniformément par une même procédure dont on fait varier les paramètres.

C'est un des apports de cette thèse que de mettre en évidence l'uniformité des procédures permettant de résoudre l'ensemble des problèmes de base d'un langage de programmation logique.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

Ardis1980.

M.A. Ardis, "Data Abstraction Transformations," Technical Report TR-925, University of Maryland, Maryland (USA), 1980.

Bellegarde1985.

F. Bellegarde, "Utilisation des Systèmes de Réécriture d'Expressions Fonctionnelles comme outils de Transformation de Programmes Itératifs," Thèse de doctorat d'Etat, Université de Nancy I, 1985.

Bidoit1981.

Michel Bidoit, "Une Méthode de Présentation Des Types Abstraits: Applications," Thèse de 3eme Cycle, Université de Paris-Sud, Juin 1981.

Birkhoff1935.

G. Birkhoff, "On the Structure of Abstract Algebras," Proc. Cambridge Phil. Soc. 31, pp. 433-454, 1935.

Birkhoff1970.

G. Birkhoff and J.D. Lipson, "Heterogeneous Algebras," Journal of Combinatorial Theory 8, pp. 115-133, 1970.

Boyer1977.

R. Boyer and J. Moore, "A Lemma Driven Automatic Theorem Prover for Recursive Function Theory," 5th International Joint Conference on

Artificial Intelligence, pp. 511-519, 1977.

Buchberger1985.

Bruno Buchberger, "History and Basic Features of the Critical-Pair/Completion Approach," Rewriting Techniques and Application, Dijon (France), May 1985.

Buchberger1976.

B. Buchberger, "A Theorical Basis for the Reduction of Polynomials to Canonicals Forms," Acm-Sigsam Bulletin 39, pp. 19-29, August 1976.

Burstall1977.

R.M. Burstall and J.A. Goguen, "Putting Theories Together to Make Specifications," Proc. 5th IJCAI, 1977.

Cohn1965.

P.M. Cohn, "Universal Algebra," Harper And Row, New York, 1965.

Corbin1983.

J. Corbin and M. Bidoit, "Pour une réhabilitation de l'algorithme d'unification de Robinson," IFIP 1983, 1983.

Cras1983.

J.Y. Cras, "Conception d'un système modulaire traitant le cas de non-convergence de l'algorithme de Knuth-Bendix," Rapport de stage,Ecole Centrale des Arts et Manufactures, Chatenay-Malabry, 1983.

Dershowitz1979.

N. Dershowitz and Z. Manna, "Proving Termination With Multiset Orderings," Comm. of ACM, vol. 22, pp. 465-476, 1979.

Dershowitz1982.

N. Dershowitz and L. Marcus, "Existence And Construction of Rewrite Systems," Research Report, University of Illinois, USA, 1982.

Dershowitz1984.

N. Dershowitz, "Computing With Term Rewriting Systems," Procedings of An NSF Workshop On The Rewrite Rule Laboratory, April 1984.

Dershowitz1984a.

N. Dershowitz, "Equations as programming language," Fourth Jerusalem Conference on Information Technology, pp. 114-123, Jerusalem (Israel), May 1984.

Dershowitz1984b.

N. Dershowitz and D. Plaisted, "Logic Programming cum Applicative Programming," private communication, 1984.

Dick1985.

A.J.J. Dick, "ERIL - Equational reasoning: an interactive laboratory," Proceedings of the EUROCAL Conference, Linz (Austria), 1985.

Fages1983.

F. Fages, "Formes canoniques dans les algèbres booléennes," Thèse de 3eme cycle. Université de Paris 6, 1983.

Fages1983a.

F. Fages and G. Huet, "Unification and Matching in Equational Theories," Proceedings of CAAP 83, vol. 159, pp. 205-220, Springer Verlag, l'Aquilla, Italy, 1983.

Fay1978.

M. Fay, "First-Order Unification in An Equational Theory," Master Thesis, U. of California At Santa Cruz. Tech. Report 78-5-002, May 1978.

Fay1979.

M. Fay, "First-Order Unification in an Equational Theory," Proceedings of the 4th Workshop on Automated Deduction, pp. 161-167, Austin, Texas, 1979.

Fribourg1984.

L. Fribourg, "A narrowing procedure for theories with constructors," Proceedings 7th international Conference on Automated Deduction, Napa Valley (California, USA), 1984.

Fribourg1984a.

L. Fribourg, "Oriented Equational Clauses as a Programming Language," J. Logic Programming, vol. 1:2, pp. 165-177, 1984.

Fribourg1985.

L. Fribourg, "Handling function definitions through innermost superposition and rewriting," 1st Conference on Rewriting Techniques and Applications, Dijon (FRANCE), 1985.

Futatsugi1985.

K. Futatsugi, J.A. Goguen, J.P. Jouannaud, and J. Meseguer, "Principles of OBJ2," Proceedings, 12th ACM Symposium on Principles of Programming Languages Conference, 1985.

Goguen1977.

J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright, "Initial Algebras Semantics And Continuous Algebras," J. of ACM 24, pp. 68-95, 1977.

Goguen1980.

J.A. Goguen, "How to Prove Algebraic Inductive Hypotheses Without Induction, With Applications to the Correctness of Data Type Implementation," Proc. 5th Workshop on Automated Deduction, pp. 356-373, Les Arcs, France, 1980.

Goguen1982.

J.A. Goguen and J. Meseguer, "Completeness of many-sorted equational logic," SRI International Technical Report CSL-135, Menlo Park, California (USA), 1982.

Goguen1984.

J. Goguen and J. Meseguer, "Equality, Types, Modules and Generics for Logic Programming," SRI Internal Publication, Menlo Park, 1984.

Goguen1985.

J. A. Goguen, J. P. Jouannaud, and J. Meseguer, "Operational Semantics for Order-Sorted Algebra," ICALP, Nafplion Greece, 1985.

Guttag1978.

J.V. Guttag and J.J. Horning, "The Algebraic Specification of Abstract Data Types," Acta Informatica 10, pp. 27-52, 1978.

Honda1979.

M. Honda and R. Nakajima, "Modularly structured set of very many

axioms," Proc. 6th Int. Joint Conference on Artificial Intelligence, pp. 400-402 vol.1, Tokyo, 1979.

Hsiang1983.

J. Hsiang and N. Dershowitz, "Rewrite methods for clausal and non-clausal theorem proving," in Proc. Int. Conf. on Automata Language and Programming, LNCS 154, pp. 331-346, Springer Verlag, Barcelona (Spain), 1983.

Hsiang1985.

J. Hsiang, "Two results in term rewriting theorem proving," 1st Conference on Rewriting Techniques and Applications, Dijon (FRANCE), 1985.

Huet1976.

G. Huet, "Résolution d'Equations dans des Langages d'Ordre 1,2, ... ω," Thèse d'Etat, Université de Paris VII, 1976.

Huet1980.

G. Huet, "Confluent reductions: abstract properties and applications to term rewriting systems," J. of ACM, vol. 27, no. 4, pp. 797-821, Oct. 1980.

Huet1980a.

G. Huet and D. Oppen, "Equations and Rewrite Rules: A Survey," in Formal Languages: Perspectives And Open Problems, ed. Book R., Academic Press, 1980.

Huet1982.

G. Huet and J.M. Hullot, "Proofs By Induction in Equational Theories

With Constructors," J. Comp. Sys. Sc., vol. 25, pp. 239-266, 1982.

Hullot1980.

J.M. Hullot, "Canonical Forms And Unification," in Proceedings of the Fifth Conference on Automated Deduction, Lecture Notes in Computer Science, vol. 87, pp. 318-334, Springer Verlag, Les Arcs, France, July 1980.

Hullot1980a.

J.M. Hullot, "Compilation de Formes Canoniques dans les Théories Equationnelles," Thèse de 3ème Cycle, Université de Paris Sud, 1980.

Jouannaud1981.

J.P. Jouannaud and Y. Kodratoff, "Program Synthesis From Example of Behaviour," Proc. of the International Workshop on Program Construction. Chateau De Bonas. Ed. Biermann And Guiho. Reidel Publish, 1981.

Jouannaud1982.

J-P. Jouannaud and P. Lescanne, "On Multiset Orderings," Information Processing Letters, vol. 10, no. 2, pp. 57-63, 1982.

Jouannaud1983.

J.P. Jouannaud, "Church-Rosser computations with equational term rewriting systems," Proc. 4th Conference on Automata, Algebra and Programming, L'Aquila, 1983.

Jouannaud1983a.

J. P. Jouannaud, C. Kirchner, and H. Kirchner, "Incremental Construc-

tion of Unification Algorithms in Equational Theories," in Proceedings of the International Conference On Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 154, pp. 361-373, Springer Verlag, Barcelona Spain, 1983.

Jouannaud1984.

J.P. Jouannaud and H. Kirchner, "Completion of a set of rules modulo a set of equations," Proceedings 11th ACM Conference of Principles of Programming Languages, Salt Lake City (Utah, USA), 1984.

Jouannaud1985.

J.P. Jouannaud and E. Kounalis, "Proofs by induction in equational theories without constructors," CRIN 85-R-042, Nancy, 1985.

Kaplan1984.

S. Kaplan, "Fair Conditional Term Rewriting Systems: Unification, Termination and Confluence," Laboratoire de Recherche en Informatique, Universite d'Orsay (France), Orsay, 1984.

Kapur1984.

D. Kapur and D.R. Musser, "Proof by consistency," General Electric 84GEN008, Schenectady USA, 1984.

Kapur1984a.

D. Kapur and P. Narendran, "An equational approach to theorem proving in first-order predicate calculus," Unpublished manuscript, GE Research Laboratory, 1984.

Kirchner1982.

C. Kirchner and H. Kirchner, "Contribution à la résolution d'équations

dans les algèbres libres et les variétés équationnelles d'algèbres," Thèse de 3ème cycle, Université de Nancy I, 1982.

Kirchner1985.

C. Kirchner, "Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles," Thèse de doctorat d'Etat, Université de Nancy I, 1985.

Knuth1970.

D. Knuth and P. Bendix, "Simple Word Problems in Universal Algebras," Computational Problems in Abstract Algebra Ed. Leech J., Pergamon Press, pp. 263-297, 1970.

Kounalis1985.

E. Kounalis, Validation de spécifications algébriques par complétion inductive, Université de Nancy I, 1985.

Kounalis1985a.

E. Kounalis, "Completeness in data type specifications," Proceedings of Eurocal Conference LNCS, Linz (Austria), 1985.

Kowalski1974.

R.A. Kowalski, "Predicate Logic As Programming Language," Proc. Ifip 74, North Holland, pp. 569-574, 1974.

Kuchlin1985.

W. Kuchlin, "A confluence criterion based on the generalized Newman lemma," Proceedings of the EUROCAL Conference, Linz (Austria), 1985.

Lankford1977.

D.S. Lankford and A.M. Ballantyne, "Decision Procedures for Simple Equational Theories With Permutative Axioms: Complete Sets of Permutative Reductions," Report Atp-37, Dept. of Mathematics And Computer Science U.Texas, Austin, April 1977.

Lankford1977a.

D.S. Lankford and A.M. Ballantyne, "Decision Procedures for Simple Equational Theories With Commutative-Associative Axioms: Complete Sets of Commutative-Associative Reductions," Report Atp-39, Dept. of Mathematics And Computer Science U.Texas, Austin, Aug. 1977.

Lankford1977b.

D.S. And Ballantyne A.M. Lankford, "Decision Procedures for Simple Equational Theories With Commutative Axioms: Complete Sets of Commutative Reductions," Report Atp-35, Dept. of Mathematics And Computer Science U.Texas, Austin, March 1977.

Lankford1979.

D.S. Lankford, "A Unification Algorithm for Abelian Group Theory," Report Mtp-1, Math. Dept., Louisiana Tech. University, Jan. 1979.

Lankford1979a.

D.S. Lankford and A.M. Ballantyne, "The Refutation Completeness of Blocked Permutative Narrowing And Resolution," fourth Conference on Automated Deduction, Austin, pp. 53-59, Feb. 1979.

Lankford1981.

D.S. Lankford, "A simple explanation of inductionless induction," Louisiana Tech.University,Depart.of Mathematics,Memo MTP-14, Ruston

(Louisiana), 1981.

Martelli1982.

A. Martelli and U. Montanari, "An efficient unification algorithm," ACM T.O.P.L.A.S, vol. 4, no. 2, pp. 258-282, 1982.

Meseguer1983.

J. Meseguer and J.A. Goguen, "Initiality, induction and computability," CSL Technical Report 140, SRI International, Menlo Park (California), 1983.

Musser1980.

D. L. Musser, "On Proving Inductive Properties of Abstract Data Types," Proceedings of the 7th Annual Acm Symposium on Principles of Programming Languages, pp. 154-162, Las Vegas, 1980.

Padawitz1985.

P. Padawitz, "Parameter preserving data type specifications," CAAP'85 Trees in Algebra and Programming, 1985.

Paterson1978.

M.S. Paterson and M.N. Wegman, "Linear Unification," J. of Computer And Systems Sciences, vol. 16, pp. 158-167, 1978.

Paul1984.

E. Paul, "Proof by induction in equational theories with relations between constructors," Proc.9th Colloquium on trees in Algebra and Programming, Bordeaux, 1984.

Paul1984a.

E. Paul, "A new interpretation of the resolution principle," Proceedings 7th international Conference on Automated Deduction, Napa Valley (California, USA), 1984.

Paul1985.

E. Paul, "On solving the equality problem in theories defined by Horn clauses," Proc.of Eurocal Conference, Linz (Austria), 1985.

Pedersen1985.

J. Pedersen, "Obtaining complete sets of reductions and equations without special unification algorithms," Proceedings of the EUROCAL Conference, Linz (Austria), 1985.

Peterson1981.

G. Peterson and M. Stickel, "Complete sets of reduction for equational theories with complete unification algorithms," J. of ACM, vol. 28, no. 2, pp. 233-264, 1981.

Peterson1983.

G.E. Peterson, "A Technique for Establishing Completeness Results in Theorem proving with equality," J.ACM, vol. 28, pp. 233-264, 1983.

Plotkin1972.

G. Plotkin, "Building-In Equational Theories," Machine Intelligence, vol. 7, pp. 73-90, 1972.

Puel-Cormier1983.

L. Puel-Cormier, "Preuves dans l'algèbre terminale. Algorithmes de complétion," Thèse de 3eme cycle, Université de Paris VII, 1983.

Remy1982.

J.L. Remy, "Etude de la Réécriture Conditionnelle et de ses Applications à la Spécification, à l'Implémentation et à la Certification des Types Abstraits Algébriques," Thèse d'Etat, Université Nancy 1, 1982.

Remy1984.

J.L. Remy and H. Zhang, "REVEUR 4: a System for Validating Conditional Algebraic Specifications of Abstract Data Types," Proceedings of the 5th ECAI, Pisa, 1984.

Robinson1965.

J.A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," J. of ACM, vol. 12, pp. 32-41, 1965.

Shoenfield1967.

J.R. Shoenfield, Mathematical Logic, 1967.

Siekmann1978.

J. Siekmann, "Unification And Matching Problems," Ph. D. Thesis, Memo Csm-4-78, U. of Essex, 1978.

Stickel1981.

M. Stickel, "A Complete Unification Algorithm for Associative-Commutative Functions," J. of ACM, vol. 28, no. 3, pp. 423-434, 1981.

Thiel1984.

J.J. Thiel, "Stop losing sleep over incomplete data type specifications," Proceedings 11th ACM Conference of Programming Languages, Salt Lake City (Utah, USA), 1984.

Winkler1983.

F. Winkler and B. Buchberger, "A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm," Colloquium on Algebra, Combinatorics and Logic in Computer Science, Gyor (Hungary), 1983.

Wirsing1982.

Martin Wirsing, "Structured Algebraic Specifications," Colloque Les Mathématiques de l'Informatique, Paris, Mars 1982.

OM DE L'ETUDIANT : KIRCHNER Hélène

NATURE DE LA THESE : DOCTORAT ES SCIENCES

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 14 Juin 1985 20 JUIN 1985 n° 2099

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITE NANCY I
R. MAINARD

RESUME

L'objet de cette thèse est l'étude des systèmes de réécriture et de leur procédures de complétion. Elle fournit en cela des outils puissants pour la programmation logique et la démonstration automatique. Une preuve par complétion est réalisée à l'aide de deux règles d'inférence qui sont la superposition et la normalisation. Cependant, la puissance et la généralité des preuves par complétion sont contrebalancées par le fait qu'elles peuvent ne pas terminer ou au contraire s'arrêter en échec sans que l'on puisse conclure. Le premier objectif de cette thèse est de proposer des solutions à ces deux problèmes.

- L'arrêt de la procédure sur une équation non orientable motive l'introduction et l'étude des systèmes de réécriture équationnels.
- Les cas où la procédure de complétion diverge en engendrant des familles infinies de règles sont traités grâce à l'introduction de méta-règles.

Les preuves par complétion peuvent ainsi s'appliquer à une classe plus large de théories équationnelles.

Le deuxième objectif est d'étudier les applications des preuves par complétion.

- Une procédure de complétion inductive hiérarchique permet de faire des preuves dans les types abstraits algébriques. Elle exploite la construction de la spécification pour structurer ses preuves.
- Une méthode générale et incrémentale de construction d'algorithmes d'unification dans les théories équationnelles et sa réalisation par une procédure de complétion est étudiée.
- Enfin les preuves par complétion trouvent également des perspectives d'application en programmation logique.