UNIVERSITE DE NANCY I

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

METHODES ET OUTILS DE CONCEPTION SYSTEMATIQUE D'ALGORITHMES D'UNIFICATION DANS LES THEORIES EQUATIONNELLES

THESE DE DOCTORAT D'ETAT EN INFORMATIQUE
PRESENTEE PAR

Claude Kirchner

SOUTENUE LE 21 JUIN 1985

DEVANT LA COMMISSION D'EXAMEN

PRESIDENT	M. NIVAT
RAPPORTEURS	J. HSIANG
	G. HUET
	J.P. JOUANNAUD
EXAMINATEURS	L. BERARD-BERGERY
	P. LESCANNE
	R. MOHR
	G. PLOTKIN

UNIVERSITE DE NANCY I CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

METHODES ET OUTILS DE CONCEPTION SYSTEMATIQUE D'ALGORITHMES D'UNIFICATION DANS LES THEORIES EQUATIONNELLES

THESE DE DOCTORAT D'ETAT EN INFORMATIQUE
PRESENTEE PAR

Claude Kirchner

SOUTENUE LE 21 JUIN 1985

DEVANT LA COMMISSION D'EXAMEN

PRESIDENT	M. NIVAT
RAPPORTEURS	J. HSIANG
	G. HUET
	J.P. JOUANNAUD
EXAMINATEURS	L. BERARD-BERGERY
	P. LESCANNE
	R. MOHR
	G. PLOTKIN

A Hélène, Florent, Johanne et Franz.

Il est certaines actions qui ont une fin
et pas de commencement, alors que d'autres
commencent pour ne pas s'achever.
Tout dépend de la position de celui
qui observe.

Frank Herbert

Remerciements

Je remercie très sincèrement tous les membres du jury :

Lionel Berard-Bergery, qui en tant que mathématicien et ami a bien voulu apporter ses compétences à ce jury,

Jieh Hsiang qui a accepté d'écrire un rapport sur cette thèse rédigée en Français et avec qui il m'a été très fructueux de faire une synthèse de ce travail,

Gérard Huet qui a accepté la rude tâche de rapporteur et dont les travaux sur l'unification sont à l'origine des thèmes de cette thèse ; ses questions m'ont incité à approfondir le champ d'application des résultats de ce travail,

Jean-Pierre Jouannaud qui a dirigé la réalisation de cette thèse. Sans son amicale pression, sa compétence et sa disponibilité, ce travail n'aurait pas vu le jour,

Pierre Lescanne qui en lançant le logiciel REVE a contribué à donner une assise pratique à un travail a priori abstrait, et qui a porté un intérêt constant et amical à ce travail,

Roger Mohr pour l'amical intérêt qu'il a porté à ce travail et qui a bien voulu apporter au jury ses compétences en intelligence artificielle,

Maurice Nivat qui a bien voulu s'intéresser à ce travail et qui me fait l'honneur de présider le jury,

Gordon Plotkin dont les travaux sur l'unification sont à la base de ce travail et qui n'a pas hésité à se déplacer depuis Edimbourg.

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, activement ou implicitement en ont facilité la réalisation.

Je remercie en particulier Jean-Luc Remy pour son amical soutien et sa lecture attentive de ce travail, Jalel Mzali pour sa participation au développement de REVEUR3, Françoise Bellegarde, Isabelle Gnaedig, Emmanuel Kounalis et tous les membres de l'équipe pour leur amical appui.

Je remercie également tous les collègues du département de mathématiques appliquées et tout particulièrement Jean-Pierre Finance, Jacques Guyard, Pierre Marchand, Karol Proch, Alain Quéré et François Schwabb, pour leur amical soutien.

Enfin, je tiens à remercier tout particulièrement Hélène pour sa précieuse collaboration et ses encouragements.

Table des matières

INTRODUCTION 1

1. Quels problèmes? 2

2. Quels outils? 6

2.1. Vers la conception automatique d'algorithmes d'unification 8

2.2. La surréduction 10

3. Des exemples 12

4. Une réalisation 13

CHAPITRE 1 : NOTIONS FONDAMENTALES : THEORIE EQUATIONNELLE ET SYSTEMES DE REECRITURE 3

1. Théories équationnelles sur des algèbres libres 15

1.1. Algèbres libres sur un ensemble 15

1.2. L'ensemble des arbres étiquetés 18

1.2.1. Arbres, occurrences, sous-arbres 18

1.2.2. Structure de M(F,X) 19

1.2.3. Opérations sur les arbres 20

1.2.4. Endomorphismes de M(F,X) 21

1.3. Théorie équationnelle 23

1.3.1. Equations 23

1.3.2. Problème de validité. Problème des mots 24

1.3.3. Théorie équationnelle 24

1.3.4. Théorème de complétude 25

2. Filtrage et réécriture dans une théorie équationnelle 25

2.1. Le filtrage 26

2.2. Réécriture équationnelle 29

2.2.1. Définitions 29

2.2.2. Propriété de Church-Rosser modulo E 33

3. Unification équationnelle et surréduction 34

3.1. Unification 34

3.2. L'unification équationnelle : l'acquis 37

3.3. Surréduction 41

4. Complétion équationnelle : une synthèse 41

4.1. Motivations 42

4.2. Paires critiques 45

4.2.1. Théorème de décidabilité de la propriété de Church-Rosser 48

4.3. Procédure de complétion équationnelle 49

4.4. Travaux reliés 52

5. Implementation of a general completion procedure parameterized 53

CHAPITRE 2 : STANDARDISATION DE L'UNIFICATION 71

1. Les unificandes 73

2. A-équivalence 77

3. A-dépendance 81

4. Simplification d'unificandes 88

4.1. La décomposition 88

4.2. Fusion d'un système de multiéquations 94

4.3. Mutation d'un unificande 95

4.4. L'algorithme de simplification 99

5. Résolution d'un système complètement décomposé de multiéquations 100

5.1. Résolution des multiéquations complètement décomposées 101

5.2. La détection des cycles 103

5.2.1. Un ordre sur les multiéquations 103

5.2.2. Application au remplacement compatible 107

5.3. L'algorithme de résolution d'un système complètement décomposé 109

6. L'étude standard de l'unification dans une théorie équationnelle 114

CHAPITRE 3 : LES THEORIES SYNTAXIQUES 115

1. Les théories syntaxiques 117

1.1. Définitions et exemples 117

2. Conditions suffisantes pour qu'une théorie soit syntaxique 121

3. Un algorithme de complétion 128

4. Exemples et applications 130

4.1. Théories constituées d'un nombre fini d'axiomes de commutativité 130

4.2. La transitivité 132

4.3. Transitivité et commutativité 134

4.4. Vers un algorithme d'unification pour une axiomatisation du "si alors sinon" 140

CHAPITRE 4 : UNIFICATION ASSOCIATIVE COMMUTATIVE 145

1. Introduction 145

2. L'opération de mutation en présence d'opérateurs associatif-commutatif 146

2.1. Définitions et notations propres au problème 147

2.2. Simplifications 148

2.3. Cas des systèmes S tels que $p(\bar{S}) > 0$ 149

2.4. Cas des systèmes S tels que $p(\bar{S}) = 0$ 152

2.4.1. Résolution directe de systèmes de multiéquations e telles que $p(e) = 0$ 153

2.4.1.1. Résolution de systèmes d'équations dans le monoide libre commutatif 154

2.4.1.1.1. Résolution des systèmes d'équations diophantiennes linéaires et homogènes 155

2.4.1.1.2. Description de la base de l'ensemble des solutions d'un système d'équations dans le monoide commutatif libre 156

2.4.1.2. Ensemble complet de AC-solutions du système S 157

2.4.1.3. Résolution générale des systèmes S tels que p(S) = 0 158

3. Description de l'algorithme d'unification associative commutative 161

4. Terminaison de l'algorithme d'unification associative commutative standard 161

5. Un exemple développé comparativement 161

6. Conclusion 163

CHAPITRE 5 : UNIFICATION MODULO LA COMMUTATIVITE DROITE 165

1. Détermination de l'opération de mutation 166

1.1. Une structure de terme Cd-aplatie 171

1.2. Le cas variable : résolution des équations de la forme z+Z == y+Y 172

1.3. Résolution des systèmes 174

2. Conclusion 174

CHAPITRE 6 : SURREDUCTION ET SUPERREDUCTION 175

1. La RE-surréduction en tant qu'opération sur les ensembles de A-solutions d'une équation 176

2. Surréduction et unification 183

3. Réduction de l'arbre de surdérivation 186

3.1. La surréduction basique 187

3.1.1. Définitions 189

3.1.2. Complétude de la surréduction basique 191

3.1.3. Une condition suffisante de terminaison de la surréduction 194

3.2. Factorisation de l'arbre de surdérivation 194

3.2.1. Caractérisation des branches infinies rationnelles de l'arbre de surdérivation 195

3.2.2. Définition récursive des A-solutions 198

3.2.3. Exemple de définition récursive des A-solutions 199

3.3. Suppression des branches subsumées 200

3.3.1. Implantation 201

CHAPITRE 7 : UNIFICATION DANS LES MELANGES DE THEORIES 203

1. Combinaison d'algorithmes de complète décomposition 204

1.1. Structuration des systèmes de multiéquations 204

1.2. Une stratégie de complète décomposition dans les mélanges de théories 208

1.3. Terminaison de l'algorithme COMP-DEC 210

2. Applications 219

CONCLUSION 223

1. Des approches complémentaires 223

2. Exemples d'approches et de limitations 226

2.1. Etude de l'unification modulo la commutativité des crochets de Lie 226

2.2. Unification dans les groupes abéliens 227

3. Problèmes ouverts, perspectives de recherches et d'applications 230

BIBLIOGRAPHIE 233

INTRODUCTION

L'objectif de cette thèse est de présenter des outils pour l'étude et la conception d'algorithmes d'unification dans les théories équationnelles puis de les mettre en oeuvre sur des exemples significatifs.

Ces outils sans être universels constituent, à notre connaissance, la première tentative d'étude systématique de moyens utilisés pour concevoir et réaliser des algorithmes d'unification dans des théories équationnelles. L'approche unifiée qu'ils justifient, permet de faire de l'unification modulo des mélanges de théories.

Nous montrons comment ils permettent en particulier d'étudier avec succès des théories constituées d'axiomes permutatifs tels la commutativité, la transitivité ou encore l'associativité commutativité ou la distributivité droite ou gauche, mais aussi des théories non permutatives telle la théorie minus.

Par ailleurs nous montrons comment étendre la surréduction à toutes les réécritures équationnelles connues à ce jour et ce en définissant de façon abstraite la réécriture équationnelle sur les termes ainsi que la

surréduction associée.

Enfin une partie de ce travail est constituée de la description du laboratoire de réécriture équationnelle REVEUR3, dans lequel le travail que nous décrivons ici joue un rôle central.

1. Quels problèmes?

L'unification équationnelle est la résolution d'équations dans des théories équationnelles telles les monoides, monoides commutatif ou groupes.

Par exemple si on considère les termes

$$t_1 = (y*b) + x$$

$$t_2 = (b*x) + z$$

quelle valeur faut-il donner aux variables x, y et z pour que les deux expressions précédentes soient égales syntaxiquement? Il suffit de prendre x = b, z = b et y = b. On dit que la substitution $\alpha = \{(x \leftarrow b), (y \leftarrow b), (z \leftarrow b)\}$ est un unificateur de t_1 et t_2 ou encore que c'est une solution de l'équation $t_1 == t_2$.

Si on suppose de plus que * est un symbole commutatif c'est à dire tel que

$$x*y = y*x$$

l'équation $t_1 == t_2$ a en plus de la solution précédente, l'unificateur $\beta = \{(x \leftarrow x'), (y \leftarrow x'), (z \leftarrow x')\}$ où x' est une nouvelle variable. Mais β est plus générale que α puisqu'il existe une substitution $\rho = \{(x' \leftarrow b)\}$ telle $\rho\beta = \alpha$. α se déduisant de β, il suffira de savoir calculer β c'est à dire en fait un ensemble générateur de l'ensemble des substitutions solution de l'équation considérée.

Le problème devient plus difficile lorsque l'ensemble des unificateurs

n'est plus fini. Prenons par exemple la théorie $minus_2$ [Kirchner1982] constituée des axiomes

$$-(-x) = x$$
$$-(x+y) = (-y)+(-x)$$

l'équation x = -x a alors une infinité de solution de la forme

$\sigma_1 = \{(x \leftarrow y + (-y))\}$,

$\sigma_2 = \{(x \leftarrow (y + (-y)) + (y + (-y)))\}$, ... mais toutes sont engendrées par la première, l'ensemble des solutions de x == -x est donc engendré par $\{\sigma_1\}$, donc par une partie finie, en l'occurrence minimale.

Mais la partie génératrice n'est pas nécessairement finie. Prenons en effet la théorie $minus_3$ (loc.cit.) constituée des axiomes

$$-(-x) = x$$
$$-(x+y) = (-y)+(-x)$$
$$-f(x,y,z) = f(-z,-y,-x)$$

l'équation x == -x a alors pour solution

$\sigma_1 = \{(x \leftarrow z + (-z))\}$

$\sigma_2 = \{(x \leftarrow f(y,z + (-z),-y)\}$,

$\sigma_3 = \{(x \leftarrow f(y,f(y',z + (-z),-y'),-y)\}$,

$\sigma_4 = \{(x \leftarrow f(y,f(y',f(y'',z + (-z),-y''),-y'),-y)\}$,

qui sont toutes indépendantes c'est à dire non comparables.

On retrouve la même situation si on suppose par exemple que * est un opérateur associatif avec l'équation x * a == a * x.

On peut chercher à minimiser l'ensemble générateur de l'ensemble des solutions, ce qui peut se faire par simple énumération d'une partie génératrice dans le cas où il en existe une finie. Sinon c'est un problème difficile qui peut ne pas avoir de solution comme le met en évidence l'exemple suivant [Fages1983] : Si on considère les axiomes

$f(o,x) = x$
$g(f(x,y)) = g(y)$

L'équation $g(x) == g(a)$ a pour solution dans cette théorie

$\sigma_1 = \{(x \leftarrow a)\}$

$\sigma_2 = \{(x \leftarrow f(x_1,a))\}$

$\sigma_3 = \{(x \leftarrow f(x_2,f(x_1,a)))\}$

or σ_1 est moins générale que σ_2 elle même moins générale que σ_3 etc...
Une partie génératrice minimale n'existe donc pas dans cette théorie pour cette équation.

On voit sur ces quelques exemples que les situations sont variées et qu'elles ne sont pas toujours simples.

L'intérêt pour ce type de problème vient du fait que l'unification joue un rôle fondamental dans de nombreux domaines. Citons les suivants sans vouloir être exhaustif :

- Les bases de données : Les règles d'inférences utilisées dans les bases de données déductives [Gallaire1981] tout comme la réalisation de machine base de données [Sabbatel1985] dépendent fondamentalement de l'unification.

- Les langages de programmation : d'une part l'étude de langages de haut niveau dont le mécanisme de passage de paramètres par reconnaissance de motifs motive l'implantation de reconnaisance de schémas dans des structures telles que les listes, les chaînes, les ensembles ou multiensembles. Mais aussi les langages dits de 5ième génération comme Prolog, fondés sur la logique des prédicats du premier ordre [Kowalski1974] ou Qute [Sato1984].

- En algèbre bien évidement : citons le problème du monoide c'est à dire le problème de la décidabilité de la résolution d'équation modulo l'associativité qui a été résolu par Makanin [Makanin1977] ou l'unification dans les groupes abéliens [Lankford1984].

- Dans tous les domaines de l'intelligence artificielle, vision par ordinateur, compréhension du langage naturel, systèmes experts dont en particulier la synthèse de circuits logiques, où la donnée de règles d'inférences ou de spécifications équationnelles requièrent la connaissance d'algorithmes d'unification ad hoc.

- Enfin, et ce n'est pas la moindre des applications, la preuve de théorèmes. Depuis Herbrand [Herbrand1930] qui le premier évoque l'unification non équationnelle jusqu'aux démonstrateurs automatiques de théorèmes basés sur la résolution[Robinson1965]. Une approche particulièrement prometteuse a été initialisée par Knuth et Bendix [Knuth1970] avec la première procédure de complétion, généralisée par la suite de manière à tenir compte d'axiomes non orientables en règles de réécriture [Huet1980,Pedersen1985,Jouannaud1984,Peterson1981]. Les deux dernières approches nécessitant de connaitre des algorithmes d'unification spécialisés.

Les solutions proposées pour construire un algorithme d'unification dans une théorie équationnelle donnée étaient jusqu'à présent indépendantes. La construction d'un algorithme d'unification pour une nouvelle théorie devait donc se (re)faire sans aucun acquis. Par ailleurs la diversité des approches utilisées rendait difficile la combinaison des algorithmes découverts. Nous proposons dans ce travail des outils permet-

tant une approche unifiée et nous les mettons en oeuvre sur plusieurs exemples.

2. Quels outils?

L'idée de base que nous développons consiste à transformer un problème de A-unification (i.e. de résolution d'équation modulo les axiomes de A) a priori compliqué, en un ensemble de problèmes simples tout en restant capable de décrire un ensemble générateur de l'ensemble des solutions, aussi appelé ensemble complet de A-unificateurs, du problème de départ.

L'approche algébrique que nous développons généralise celle de Martelli et Montanari [Martelli1982] à des théories équationnelles. Un problème d'unification ou unificande, est décomposé en un système d'équations, ou si cela est nécessaire en une disjonction de systèmes. Par exemple si on suppose que le symbole * est commutatif l'équation $(a * x) + y == (b * y) + a$ sera décomposée en le système

$S_1 = \{(1)\ (a * x == b * y),$

$(2)\ (y == a)\}$

car le symbole + n'a pas de propriétés particulière. La résolution de l'équation (1) est plus délicate car * ayant une propriété, ici la commutativité, il ne suffit pas en général de la décomposer comme précédement. En fait il est simple de voir que (1) est équivalent, c'est à dire a même ensemble de solution modulo la commutativité, que les deux systèmes d'équations

$S_2 = \{(a == b), (x == y)\}$

$S_3 = \{(a == y), (x == b)\}$

S_2 n'ayant pas de solution, S_1 est équivalent à

$S_4 = \{(a == y), (x == b)\}$

que l'on sait facilement résoudre.

Cette approche a le mérite de permettre d'expliciter complètement les opérations dont on a besoin sur les systèmes d'équations mais également de standardiser l'approche pour toutes les théories équationnelles donc de permettre l'étude de théories dans lesquelles des ensembles de symboles disjoints satisfont des propriétés différentes, par exemple si *, + et = sont des symboles respectivement commutatif, associatif commutatif et transitif.

Pour fonder cette approche, nous aborderons les points suivants :

- D'une part l'étude des transformations conservant les ensembles complets éventuellement minimaux d'unificateurs. Deux types de transformations vont retenir notre attention. D'une part celles qui conservent les ensembles de solutions et par conséquent les ensembles complets de A-unificateurs, et celles qui bien que ne préservant pas les ensembles de solutions permettent de déduire un ensemble générateur de l'ensemble des A-solutions de l'unificande initial à partir d'un ensemble complet de A-solutions de l'unificande transformé.

  Trois transformations des unificandes jouent un rôle fondamental

  * la décomposition qui, comme on l'a vu dans l'exemple précédent, permet de simplifier les unificandes dès que le symbole de tête des expressions ne possède pas de propriétés applicables,

  * la fusion qui a pour fonction de regrouper les contraintes sur les variables,

  * la mutation qui contrairement aux deux précédentes dépend de la théorie et par conséquent échappe à toute standardisation sauf sur des

classes restreintes de théories.

L'enchaînement de ces trois transformations s'il termine conduira à un unificande dont toutes les équations seront de la forme x == t où x est une variable et t un terme.

- C'est ce qui nous amène à étudier les unificandes, et en particulier les systèmes d'équations complètement décomposées c'est à dire de la forme x == t. Comme l'exemple de l'équation x == -x le laisse soupçonner, ce problème n'est pas toujours simple. En effet pour connaître un ensemble complet de A-solutions d'un système complètement décomposé, il faudra savoir résoudre chacune des équations qui le composent, mais également savoir comment combiner ces solutions pour obtenir les solutions du système lui même. C'est pourquoi nous donnons une condition suffisante sur la théorie pour assurer l'existence de A-solutions d'un système de multiéquations complètement décomposé, condition qui s'applique en particulier aux théories permutatives mais également à des théories comme $minus_2$ où $minus_3$.

- Le mécanisme de résolution étant identique pour chaque théorie, cela nous permettra d'aborder l'étude des @i(mélanges de théories), c'est à dire de donner des conditions sous lesquelles les opérations de mutation pour des théories disjointes peuvent s'appliquer sur un même unificande sans que l'on perde la propriété de terminaison du processus de décomposition-fusion-mutation.

2.1. Vers la conception automatique d'algorithmes d'unification

La recherche d'opération de mutation, c'est à dire de transformation permettant de faire décroire la complexité d'un problème d'unification dans

le cas où on peut appliquer des axiomes de la théorie en tête de l'équation, nous a conduit à étudier les théories dans lesquelles on peut décomposer une équation uniquement en suivant la forme syntaxique des axiomes. C'est ce que l'on fait pour un axiome de commutativité comme nous l'avons vu plus haut, mais aussi par exemple pour la distributivité droite :

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

en effet on peut montrer que toute équation $x_1*x_2 == y_1+y_2$ est équivalente au système

$$S = \{(x_1 == x),\ (x_2 == y+z),\ (y_1 == x*y),\ (y_2 == x*z)\}.$$

Lorsque cette décomposition suivant la forme des axiomes est correcte, on peut l'utiliser comme opération de mutation. Il reste à prouver, et c'est souvent difficile, que le processus de décomposition-fusion-mutation termine.

Nous étudierons dans ce travail les théories, que nous appelons syntaxiques, dans lesquelles cette décomposition suivant la forme des axiomes est correcte.

- Nous donnons d'abord des conditions suffisantes pour que la présentation d'une théorie soit syntaxique,

- Puis nous donnons un algorithme de complétion unificationnelle : il permet de compléter "à la Knuth et Bendix", un ensemble d'axiomes en une présentation qui soit syntaxique.

2.2. La surréduction

Une autre approche universelle, que l'on peut également voir comme étant une opération de mutation particulière, est la surréduction, aussi appelée narrowing outremer.

Elle est basée sur la notion de réécriture de termes : au lieu de s'autoriser à utiliser les axiomes de la théorie dans les deux sens, on les oriente en règle de réécriture. On perd alors en général en puissance d'expression d'où l'idée de complèter le système de réécriture initial de façon à garder les même possibilités de déduction axiomatique. La première procédure de complétion donnée par Knuth et Bendix [Knuth1970] réalise cette idée. Elle a depuis été étendue aux cas où certains axiomes comme la commutativité ne peuvent être orientés sans perdre la propriété de terminaison finie de la réécriture. Pour réaliser cette extension on peut soit travailler avec la réécriture standard comme le fait G. Huet [Huet1980] pour des systèmes de réécriture linéaires gauche, soit introduire des relations de réécriture plus générales telles celles de Peterson et Stickel [Peterson1981], de Pedersen [Pedersen1985] et celle de Jouannaud [Jouannaud1983]. Par ailleurs Jean Pierre Jouannaud et Hélène Kirchner (loc.cit.) et [Kirchner1985] ont fondu les procédures de Knuth et Bendix, Huet, Peterson et Stickel en un moule unique. Nous exprimerons ces résultats de façon complètement abstraite en introduisant une définition formelle de la réécriture appelée RE-réécriture.

Intuitivement, surréduire un terme c'est instancier convenablement ses variables pour qu'il devienne réductible par la relation de réécriture considérée. L'unification est donc pour la surréduction ce que le filtrage est pour la réécriture. L'intérêt de la surréduction réside dans le fait

qu'elle permet de décrire des ensembles complets d'unificateurs modulo la théories équationnelle engendrée par le système de réécriture, pourvu que celui-ci soit Church Rosser et à terminaison finie.

Afin de décrire de façon unifiée toutes les relations de surréductions sur les termes, il est naturel d'associer à la RE-réécriture une relation abstraite de surréduction sur les termes, la RE-surréduction. L'étude de cette relation nous permettra d'étendre les résultats connus sur toutes les relations de surréduction.

Nous aborderons les points suivants :

- Etude algébrique de la relation de RE-surréduction associée à une relation quelconque de RE-réécriture noethérienne et confluente. Application à la RE-superréduction qui est la combinaison d'un pas de RE-surréduction suivi de la mise en forme irréductible pour la réécriture considérée.

- Malheureusement, l'algorithme général d'unification auquel conduit la RE-surréduction ne termine pas en général, y compris dans les cas ou les ensembles générateurs minimaux existent et sont finis. Une approche, la surréduction basique, a été prouvée complète par J.M. Hullot [Hullot1980] puis généralisée à la réécriture de Peterson et Stickel dans [Jouannaud1983]. Nous montrons comment l'étendre à la RE-surréduction.
  Nous donnons également une autre méthode consistant à détecter dans l'arbre de RE-surréduction les branches qui sont instances d'une autre ce qui généralise [Rety1985].

- Une autre amélioration qui peut être apportée à la surréduction en tant que processus de résolution d'équations est la détection des boucles. C'est ce que nous appelons la factorisation des arbres de surréductions rationnels. Dans ce cas on obtient une description finie d'ensembles complets infinis de A-solutions.

3. Des exemples

Les résultats théoriques obtenus sont appliqués d'une part à des problèmes d'unification ouverts, mais aussi à des théories comme l'associativité commutativité pour lesquelles des algorithmes sont déjà connus. Nous le faisons afin de montrer que ces théories peuvent être traitées dans notre formalisme. Nous donnons des algorithmes d'unification pour les théories suivantes :

- Un ensemble fini de symboles commutatifs. Cela constitue une généralisation de [Siekmann1979] à plusieurs symboles commutatifs.

- Un ensemble fini de symboles satisfaisant

  $$(x \text{ eq } y) \hat{} (y \text{ eq } z) = (x \text{ eq } y) \hat{} (x \text{ eq } z)$$

  ou eq et ^ peuvent être commutatifs ou non.

- Pour les axiomes du type $c(x,x,y) = c(x,t,y)$ et $c(x,y,x) = c(x,y,f)$. Ils interviennent par exemple dans l'axiomatisation du "si alors sinon" de Bloom et Tindell [Bloom1983].

- L'axiome de commutativité droite (ou gauche)

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

  vérifié par exemple par la division.

- L'associativité-commutativité, pour laquelle nous donnons un nouvel algorithme parallélisable.

L'approche générale suivie nous permettra de donner un algorithme unique d'unification modulo l'ensemble de ces théories par exemple.

4. Une réalisation

Nous terminons le chapitre 1 de cette thèse par un exposé de l'étude et la réalisation du laboratoire de réécriture REVEUR3 que nous avons réalisé en collaboration avec Hélène Kirchner. Ecrit en CLU, il implante les résultats théoriques sur la complétion équationnelle de J.P. Jouannaud et Hélène Kirchner [Jouannaud1984,Kirchner1985] ainsi que les résultats du présent travail en ce qui concerne l'unification. Le filtrage est basé sur les travaux de Jalel Mzali [Mzali1985].
Après une description des fondements théoriques, nous donnons des exemples de complétion avec REVEUR3.

CHAPITRE 1

NOTIONS FONDAMENTALES: THEORIE EQUATIONNELLE ET SYSTEME DE REECRITURE

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 de F-algèbre.
Puis nous présentons une synthèse des différentes relations de réécriture sur les termes et de leur algorithmes de complétion associé et nous terminons par la description du laboratoire de réécriture équationnel REVEUR3.

Ce chapitre a également pour but de fixer les notations et la terminologie utilisées dans cette thèse.

1. Théories équationnelles sur des algèbres libres

1.1. Algèbres libres sur un ensemble

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.

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.
On notera $F_M = \{f_M \mid f \text{ appartient à } F\}$. ○

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

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

$$h(f_M(d_1,d_2,\ldots,d_n)) = f_{M'}(h(d_1),h(d_2),\ldots,h(d_n)).$$

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

Définition 3 : Soit $\aleph$ une classe quelconque de F-algèbres et X un ensemble. On appelle **F-algèbre $\aleph$-libre** engendrée par X (on dit aussi sur X) toute F-algèbre $M=(D_M,F_M)$ tel que

1) M appartient à $\aleph$

2) D_M contient l'ensemble X

3) pour toute F-algèbre $N=(D_N,F_N)$ de $\aleph$ 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. ○

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

Si M_1 et M_2 sont deux F-algèbres $\aleph$-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 $\aleph$-libre engendrée par X.

Le résultat suivant, cas particulier d'un théorème dû à Birkhoff [Birkhoff1935], nous permet de définir l'ensemble des termes.

Théorème 1 : Soit $\aleph$ la classe de toutes les F-algèbres pour F fixé et X un ensemble dont les éléments sont appelés variables. On supposera toujours que soit X soit l'ensemble des constantes est non vide. Alors la F-algèbre $\aleph$-libre (ou plus simplement la F-algèbre libre) engendrée par X existe.

Définition 4 : Dans les conditions du théorème précédent, on appelle ensemble des **termes** du premier ordre, ou plus simplement termes, les éléments de la F-algèbre $\aleph$-libre engendrée par X. ○

Notation : Les termes seront désignés dans la suite par u, v, w, t, t', ..., g, d,... .

Cette définition des termes étant particulièrement peu explicite et peu parlante à notre intuition, nous allons en donner une représentation

dans le paragraphe suivant.

1.2. L'ensemble des arbres étiquetés

1.2.1. Arbres, occurrences, sous-arbres

Soit $\mathbb{N}_+$ l'ensemble des entiers strictement positifs, $\mathbb{N}_+^*$ le monoide libre engendre, ε le mot vide et . l'opération de concaténation.

Définition 5 : Soit une application t de $\mathbb{N}_+^*$ dans F ∪ X et Dom(t) son domaine de définition, c'est-à-dire l'ensemble des m appartenant à $\mathbb{N}_+^*$ tels que t(m) soit défini. Alors t est un **arbre** sur F ∪ X si et seulement si

1) ε est élément de Dom(t)

2) m.i appartient à Dom(t) si et seulement si m appartient à Dom(t) et i est compris entre 1 et l'arité de t(m). ○

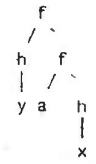
Notation : Le domaine d'un arbre t est aussi appelé ensemble des **occurrences** de t. Il sera noté Dom(t). Les éléments de Dom(t) sont les noeuds de l'arbre, le noeud est la racine et m.i est le i-ème fils du noeud m.

Un arbre est dit fini si son domaine l'est.

Exemple 2 : Soit F = {f, g, h, a} avec ar(f) = 2, ar(g) = ar(h) = 1 et ar(a) = 0 et V = {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))).

Définition 6 : Soit t un arbre et m un élément de Dom(t). Le **sous-arbre** de t à l'occurrence m, noté $t_{|m}$ est l'arbre t' défini par t'(m')=t(m.m') pour tout m' appartenant à $\mathbb{N}_+^*$. ○

Définition 7 : Pour tout sous-arbre u de l'arbre t, $t^{-1}(u)$ est l'ensemble des occurrences de u dans t, noté **O(u,t)**. Quand cet ensemble est fini, son cardinal est noté #(u,t). ○

Définition 8 : Deux noeuds m et m' sont disjoints si et seulement si il n'existe pas m" $\in \mathbb{N}_+^*$ tels que m=m'.m" ou m'=m.m". ○

Nous désignerons par **M(F,X)** l'ensemble des arbres construits sur l'ensemble des symboles F et des variables X.

1.2.2. Structure de M(F,X)

Lemme 1 : M(F,X) peut être muni d'une structure de F-algèbre, et c'est la F-algèbre libre engendrée par X.

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

Afin de pouvoir faire des preuves par récurrence structurelle dans M(F,X), nous allons voir qu'il est possible de construire cet ensemble de facon inductive de la façon suivante :

Soit $M_n(F,X)$ la famille de parties définies par:

-- $M_0(F,X) = X \cup \{c^* \mid c \text{ appartient à } F_0\}$

-- $M_{n+1}(F,X) = M_n(F,X) \cup \{f^*(t_1,\ldots,t_k)$ tel que f appartient à F_k et $t_1,\ldots,t_k$ appartiennent à $M_n(F,X)\}$

Lemme 2 : $M(F,X)$ est la réunion pour tous les n positifs ou nul des $M_n(F,X)$.

Ce résultat permet de prouver dans $M(F,X)$ des propriétés de la forme: "Pour tout t appartenant à $M(F,X)$, $P(t)$" par récurrence sur la structure de t. Il met en évidence le lien existant entre les raisonnements par récurrence structurelle et par récurrence sur les entiers.
Un autre exemple de son utilisation est la définition suivante que l'on peut donner de la taille d'un arbre:

Définition 9 : La taille d'un arbre t, notée $|t|$, est définie par:

-- si t appartient à $X \cup F_0$ alors $|t|=0$

-- si $t=f(t_1,\ldots,t_k)$ alors $|t|=1+\sum_{i=1}^{i=k} |t_i|$. ○

1.2.3. Opérations sur les arbres

Outre les opérations de la F-algèbre et celle qui à un arbre et à une occurrence m de son domaine associe le sous-arbre issu du noeud m, nous en utiliserons une autre qui est le remplacement d'un sous-arbre par un arbre.

Définition 10 : Soient t et t' deux arbres et m un noeud de t. Remplacer dans t le sous-arbre $t_{|m}$ par t', c'est définir un nouvel arbre t" noté $t_{[m\leftarrow t']}$ tel que

$Dom(t") = Dom(t) - Dom(t_{|m}) + m.Dom(t')$

$t"(m') = t(m')$ si m' appartient à $Dom(t)-Dom(t_{|m})$

$= t'(m")$ si $m'=m.m"$.

t et m étant donnés, on notera t_m l'application de l'ensemble des arbres dans lui-même, qui à un arbre t' associe l'arbre $t_m(t') = t_{[m\leftarrow t']}$. ○

1.2.4. Endomorphismes de $M(F,X)$

Définition 11 : L'ensemble **Var(t)** des variables d'un arbre t est récursivement défini par:

-- si t est une constante alors $Var(t) = \emptyset$

-- si t est une variable x alors $Var(t) = \{x\}$

-- si $t = f(t_1,\ldots,t_k)$ alors $Var(t) = Var(t_1) \cup \ldots \cup Var(t_k)$.

C'est l'ensemble des variables ayant au moins une occurrence dans le terme. L'ensemble des symboles qui ne sont pas des variables et qui ont une occurrence dans le terme t est notée **Symb(t)**. ○

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

Exemple 3 : si $t = f(f(f(x,a),g(x)),g(y))$, $Var(t) = \{x, y\}$, $Symb(t) = \{f,a,g\}$ et $O(t) = \{\varepsilon,1,11,112,12,2\}$.

Définition 12 : Une **substitution** est une application σ de X dans $M(F,X)$ telle que $\sigma(x)=x$ presque partout, c'est-à-dire sauf sur un sous-ensemble fini de X appelé le domaine de σ et noté $D(\sigma)$.

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

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

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

Le caractère libre de la F-algèbre permet de dire qu'une telle application

σ se prolonge de facon unique en un endomorphisme de M(F,X). Il vérifiera donc pour tout f d'arité k de F, pour tous $t_1,\ldots,t_k$ de M(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, ξ, ... 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 $(x \leftarrow \sigma(x))$.

L'ensemble des substitutions de M(F,X) sera noté **Subst**.

Définition 13 : La composée de deux substitutions α et β est la substitution notée $\alpha\beta$ et définie pour toute variable x par

$$\alpha\beta(x) = \alpha(\beta(x))$$

○

Définition 14 : La restriction d'une substitution σ à un sous-ensemble U de X est notée σ_U et définie par ;

$\sigma_U(x) = \sigma(x)$ si x appartient à U

$= x$ sinon

Définition 15 : Pour toutes substitutions α β telles que $\forall\, x \in D(\alpha) \cap D(\beta)$ $\alpha(x) = \beta(x)$, alors on peut définir $\alpha+\beta$ comme la substitution telle que :

$(\alpha+\beta)(x) = \alpha(x)$ si $x \in D(\alpha)$

$(\alpha+\beta)(x) = \beta(x)$ si $x \in D(\beta)$

$(\alpha+\beta)(x) = x$ sinon.

○

Nous utiliserons librement des propriétés suivantes des substitutions.

- Pour tout sous-ensemble V de X, pour tout terme t et pour toute substitution σ : $Var(t) \subseteq V \Rightarrow \sigma(t) = \sigma_{|V}(t)$.
- Pour toute théorie équationnelle A, tout terme t et toute substitution σ : $D(\sigma) \cap Var(t) = \emptyset \Rightarrow \sigma(t) =_A e$. La réciproque est vraie si $A = \emptyset$.
- Pour toutes substitutions α, β et tous ensembles de variables V_1 et V_2, si $\alpha \leq_A \beta\ [V_1]$ et si $V_2 \subseteq V_1$ alors $\alpha \leq_A \beta\ [V_2]$.

1.3. Théorie équationnelle

1.3.1. Equations

Définition 16 : Une **équation** est une paire de termes $\{t_1,t_2\}$ notée $(t_1 = t_2)$. ○

Définition 17 : Une F-algèbre $M = (D_M,F_M)$ valide l'équation $(t_1 = t_2)$ si et seulement si pour tout homomorphisme h de M(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 $Var(t_1) \cup Var(t_2)$. ○

Définition 18 : Une **variété équationnelle de F-algèbres** est une classe de F-algèbres H pour laquelle il existe un ensemble d'équations A tel que H soit la classe de toutes les algèbres validant les équations de A. H est aussi appelée classe des modèles de A et est notée M(A). ○

G.Birkhoff [loc.cit.] a prouvé que si une classe de F-algèbres est une variété, alors la F-algèbre M(A)-libre engendré par X existe pour tout ensemble X.

1.3.2. Problème de validité. Problème des mots

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) ou problème des mots pour la F-algèbre M(A)-libre. Nous allons voir que ce problème peut s'énoncer en termes de théorie équationnelle.

1.3.3. Théorie équationnelle

Définition 19 : 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:

$$t_1 \; R \; t'_1 , \ldots, t_n \; R \; t'_n \Rightarrow f(t_1,\ldots,t_n) \; R \; f(t'_1,\ldots,t'_n).$$

○

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

Définition 20 : Soit A un ensemble d'équations. La plus petite congruence sur M(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 est notée $=_A$.

Deux termes tels que $t_1 =_A t_2$ sont dit A-égaux.

On parlera aussi de la **théorie équationnelle** engendrée ou présentée par A

pour désigner l'algèbre quotient $M' = (M(F,X)/=_A, F)$. ○

Dans toute la suite de ce travail on supposera qu'aucun des axiomes de la théorie est de la forme x = y avec x et y ∈ X. Cela évitera à la théorie équationnelle d'être réduite à un élément.

Définition 21 : On dit qu'une théorie équationnelle est **régulière** ssi il existe une présentation de A dont tous les axiomes $t_1 = t_2$ satisfont $Var(t_1) = Var(t_2)$, elle est dite **permutative** si les classes d'équivalence sont finies et elle est dite **potente** ssi il existe une présentation de A qui contient au moins un axiome de la forme x = t où x ∈ X et t ∉ X. ○

Il est clair qu'une théorie permutative est régulière et n'est pas potente.

Exemple 4 : La théorie réduite à l'axiome d'idempotence est potente. La théorie constituée d'un ensemble fini de symboles associatif-commutatif est permutative. La théorie constitué de l'axiome x*0 = 0 n'est pas régulière.

1.3.4. Théorème de complétude

Théorème 2 : Etant donné un ensemble d'équations A, une équation $(t_1 = t_2)$ est valide dans la variété M(A) si et seulement si $t_1 =_A t_2$.

Ce théorème dû à G.Birkhoff [loc.cit.] permet d'étudier le problème de la validité dans la variété M(A) de manière purement syntaxique par l'intermédiaire de la relation $=_A$.

2. Filtrage et réécriture dans une théorie équationnelle

Cette section va être consacrée à l'étude de la semi-unification ou filtrage et à son application à la réécriture.

Filtrer le terme t_1 vers le terme t_2 c'est résoudre l'équation $t_1 =_E t_2$ en considèrant que le terme t_2 ne contient pas de variable. Le problème du filtrage est particulièrement important puisque c'est le moteur de la réécriture. C'est pour cette raison que nous le présentons ici de façon abstraite afin de pouvoir décrire de manière unifiée les différentes réécritures sur les termes.

Habituellement un filtre est défini comme une substitution appliquant un terme sur un autre. Un point de vue différent 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.

2.1. Le filtrage

Définition 22 : Soit E un ensemble d'axiomes, on appelle **filtre modulo E** ou **E-filtre** du terme t vers le terme t' toute substitution σ telle que $\sigma(t) =_E t'$. Le ø-filtre de t vers t', s'il existe, est unique et peut être trouvé par un algorithme récursif simple (voir par exemple [Huet1976]). L'opération de filtrage dans la théorie équationnelle $=_E$ est une application $Filtre_E$ de TxT à valeur dans P(Subst) telle que

* pour toute substitution σ appartenant à $Filtre_E(t,t')$, $\sigma(t) =_E t'$,
* le ø-filtre de t vers t', s'il existe, appartient à $Filtre_E(t,t')$. ○

Exemple 5 :

1) On retrouve la notion habituelle de filtrage lorsque $Filtre_ø$ 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 $F_ø$ 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 E-filtres de t vers t'. On notera F_E cette application.

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

4) A la suite de Pedersen [Pedersen1985], on peut prendre pour valeur de $Filtre_E(t,t')$ l'ensemble des substitutions σ telle que $\sigma(t) =_E t'$, les axiomes de E n'étant appliqués dans la E-égalité précédente qu'en dessous d'une occurrence de variable de Var(t). Avec les notations de Pedersen (loc.cit.) 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 domaine donné [Kirchner1985].

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

D'une manière générale, ce formalisme permet 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 le plus souvent

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

La notion de filtrage permet de définir un préordre sur les termes, dit **préordre de filtrage** :

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

Lorsqu'il sera utile de préciser la substitution σ utilisée pour comparer t et t', on notera $t \leqq^{\sigma} t'$.

Ce préordre est compris entre le préordre de filtrage dans la théorie vide noté $\leqq_\emptyset$ défini par :

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

et le préordre de E-filtrage défini par

$t \leqq_E t'$ si et seulement si il existe σ telle que $\sigma(t) =_E t'$.

Ce préordre s'étend aux substitutions :
$\alpha \leqq \beta$ sur un ensemble de variables V inclus dans X, si et seulement si il existe une substitution γ telle que pour tout terme t tel que $Var(t) \subseteq V$, γ appartient à $Filtre_E(\alpha(t),\beta(t))$.

Définition 23 : Le préordre $\leqq$ est **conservé sur les sous-termes** si et seulement si $t \leqq t'$ implique pour toute occurrence u appartenant à Dom(t), $t_{|u} \leqq t'_{|u}$. ○

Exemple 6 :

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

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

est conservé sur les sous-termes.

2.2. Réécriture équationnelle

La notion générale de filtrage que nous venons d'introduire permet de définir 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.

2.2.1. Définitions

Définition 24 : Soit A une théorie, R et E des ensembles disjoints d'axiomes tels que $A = R \cup E$. Les axiomes $g = d$ de R sont orientés en règle de réécriture notées $g \to d$. La **RE-réécriture** notée $\to_{RE}$ est la réécriture associée à R et à $Filtre_E$ et définie par :

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

si et seulement si σ appartient à $Filtre_E(g,t_{|u})$ et $t' = t[u \leftarrow \sigma(d)]$.
Si on suppose que les règles de R sont numérotées, et si la règle $g \to d$ porte le numéro k alors $t \to_{RE[\sigma,u,g\to d]} t'$ est aussi noté $t \to_{RE[\sigma,u,k]}$

t'.

Une suite de RE-réécritures est appelée **RE-dérivation**. 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 σ(x) est RE-normalisé. ○

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

Exemple 7 : En reprenant dans l'ordre les cas de l'exemple précédent, on retrouve

1) La réécriture habituelle notée $\rightarrow R$ associée à $F_{\emptyset}$.

2) La réécriture de Peterson et Stickel, qui sera notée $\rightarrow R,E$, associée à F_E.

3) La réécriture $\rightarrow(R_1 \cup R_{n1},E)$ introduite dans [Jouannaud1983] qui consiste à soit utiliser un filtre de $F_{\emptyset}$ pour réécrire par une règle linéaire à gauche, soit utiliser un E-filtre de F_E pour réécrire par une règle non linéaire.

4) La réécriture de Pedersen définie par

$$t \rightarrow RE[\sigma,u,g\rightarrow d]\ t'$$

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

5) La réécriture à l'aide d'un ensemble de règles et méta-règles définie dans [Kirchner1985]

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.

Certaines approches de la réécriture équationnelle telle celle décrite dans [Lankford1977] utilisent la réécriture notée $\rightarrow R/E$ définie par $=_E.\rightarrow R.=_E$. Cette relation simule la réécriture induite par R dans les classes d'équivalence modulo $=_E$ 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. On a bien sûr, pour toute théorie A = (RUE), l'inclusion des relations:

$$\rightarrow R \subseteq \rightarrow RE \subseteq \rightarrow R,E \subseteq \rightarrow R/E$$

Définition 25 : La relation $\rightarrow RE$ est **E-noethérienne** ou **noethérienne modulo E** si et seulement si $\rightarrow R/E$ est noethérienne. ○

Définition 26 : Une paire de termes (t_1,t_2) est dite **R/E-confluente** et on note $t_1 \downarrow_{R/E} t_2$, si et seulement si il existe t tel que t_1 -*->R/E t et t_2 -*->R/E t. ○

La notion de réécriture est étroitement liée à celle de l'ordre induit par le filtrage utilisé, par le fait que si t est réductible par la règle $g\rightarrow d$ et le filtre σ à l'occurrence u, $g \leq^{\sigma} t_{|u}$. Pour généraliser les résultats obtenus dans [Jouannaud1984], il est nécessaire d'introduire deux propriétés de la réécriture par rapport au préordre ≤.

Définition 27 : La RE-réécriture est **compatible** avec le préordre associé à $Filtre_E$ sur le couple de termes (t,t') si et seulement si

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

=>

t' RE-réductible en t'1 et $t'1 \downarrow_{R/E} \sigma(t_1)$.

○

Définition 28 : La RE-réécriture est **compatible au sommet** avec le préordre associé à $Filtre_E$ sur le couple de termes (t,t') si et seulement si

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

=>

t' RE-réductible en t'1 et $t'1 \downarrow_{R/E} \sigma(t_1)$.

○

Exemple 8 :

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

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

3) $\rightarrow_{R_1 \cup R_{n1},E}$ est compatible avec le préordre associé sur tous les couples de termes (t,t') tels que $Filtre_E(t,t') = F_{\emptyset}(t,t')$. $\rightarrow_{R_1 \cup R_{n1},E}$ est compatible au sommet avec l'ordre 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 $t \leq^{\mu} t'$, t' appartient à $\{\mu_E(t)\}$. Ce qui signifie que $t' = \mu'(t)$ avec $\mu'(x) =_E \mu(x)$ 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 $\sigma'(x) =_E \sigma(x)$ pour toute variable x de g. On en déduit que $\mu'(t_{|u}) = \mu'.\sigma'(g)$ et $\mu'.\sigma'(x) =_E \mu'.\sigma(x)$ pour toute variable x de g. Donc t' est réductible par g en t'1 = $t'[u \leftarrow \mu'.\sigma(d)]$. Par conséquent, $t' = \mu'(t)$ est réductible en $\mu'(t_1) =_E \mu(t_1)$ puisque $Var(t_1)$ est inclus dans Var(t).

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

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

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

2.2.2. Propriété de Church-Rosser modulo E

Définition 29 : Une paire de termes (t_1,t_2) est dite **RE-confluente modulo E** et notée $t_1 \downarrow t_2$ si et seulement si il existe des termes t'1 et t'2 tels que

t_1 -*->RE t'1, t_2 -*->RE t'2 et $t'1 =_E t'2$

- $\rightarrow$RE est **Church-Rosser modulo E** si et seulement si pour tous termes t_1, t_2,
  $t_1 =_A t_2$ implique $t_1 \downarrow t_2$

- $\rightarrow$RE est **confluente modulo E** si et seulement si pour tous termes t, t_1, t_2,
  t -*->RE t_1 et t -*->RE t_2 implique $t_1 \downarrow t_2$

● $\rightarrow_{RE}$ est **cohérente modulo E** si et seulement si pour tous termes t, t_1, t_2,

$t \text{ -*->}_{RE} t_1$ et $t =_E t_2$ implique il existe t_2' tel que $t_2 \rightarrow_{RE} t_2'$ et $t_1 \downarrow t_2'$.

○

On obtient le résultat fondamental suivant :

Théorème 3 : (Théorème de Church-Rosser modulo)

Si $\rightarrow_{RE}$ est E-noethérienne, les propriétés suivantes sont équivalentes:

(1) $\rightarrow_{RE}$ est Church-Rosser modulo E

(2) $\rightarrow_{RE}$ est confluent modulo E et cohérent modulo E

(3) pour tous termes t, t', $t =_A t'$ si et seulement si $t\downarrow_{RE} =_E t'\downarrow_{RE}$.

Preuve: Elle est donnée dans [Kirchner1985]. □

3. Unification équationnelle et surréduction

Nous présentons maintenant ce qu'est la résolution d'équations dans une théorie équationnelle. De même que pour le filtrage, nous adoptons un point de vue synthétique nous permettant d'englober les travaux existants en particulier sur la surréduction.

3.1. Unification

Définition 30 : Soit E un ensemble d'axiomes et $UNIF_E$ une application de TxT à valeur dans P(Subst) telle que pour tous termes t et t',

(1) pour toute σ dans $UNIF_E(t,t')$, $\sigma(t) =_E \sigma(t')$

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

(3) pour toute substitution ρ, pour tous termes g et t tels que $\rho \in Filtre_E(t,t')$, $D(\rho) \cap Var(g) = \emptyset$, $\gamma \in UNIF(g, \rho(t)) \Rightarrow \gamma\rho \in UNIF(g,t)$. ○

Remarque : La condition (2) assure la cohérence entre le filtrage utilisé et l'unification, la condition (3) permet de s'assurer que l'ensemble des unificateurs est suffisament stable pour que l'on puisse retrouver les propriétés classiques.

Exemple 9 : 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 $\sigma(t) =_E \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap (Var(t) \cup Var(t'))$.

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 domaine donné.

6) On peut également faire varier les axiomes avec lesquels on unifie suivant un critère sur les termes. Cela généralise le point 3.

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

Définition 31 : Soit E un ensemble d'axiomes et $Unif_E$ une application de TxT 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 $Var(t) \cup Var(t')$ et $I(\sigma)$ est choisi d'intersection vide avec W, où W est un ensemble de variables qui contient $Var(t) \cup Var(t')$.

(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 $\sigma \leq_E \sigma'$ $[Var(t) \cup Var(t')]$ O

Il est important de noter que l'ordre utilisé pour comparer les substitutions de $UNIF_E(t,t')$ est l'ordre 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 en entrée et retourne un ensemble complet d'unificateurs. 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.

3.2. L'unification équationnelle : l'acquis

Nous décrivons dans cette section l'état des connaissances en unification équationnelle.

Rappelons tout d'abord la définition des différents axiomes et théories dont nous parlerons dans la suite.

A(f)	Associativité	f(f(x,y),z)=f(x,f(y,z))
C(f)	Commutativité	f(x,y)=f(y,x)
Dd(f,g)	Distributivité droite	f(g(x,y),z)=g(f(x,z),f(y,z))
Dg(f,g)	Distributivité gauche	f(z,g(x,y))=g(f(z,x),f(z,y))
D(f,g)	Distributivité	Dd U Dg
E(h,*)	Endomorphisme	h(x*y)=h(x)*h(y)
AE(h,*)	Anti-endomorphisme	h(x*y)=h(y)*h(x)
H(h,*,+)	Homomorphisme	h(x*y)=h(x)+h(y)
I(f)	Idempotence	f(x,x)=x
Iv(h)	Involution	h(h(x))=x
InvD(*,e)	Inverse à droite	x*i(x)=e
InvG(*,e)	Inverse à gauche	i(x)*x=e
Sd(f,g,h)	Simplification droite	f(g(x,y),h(y))=x
Sg(f,g,h)	Simplification gauche	f(h(y),g(y,x))=x
Nd(*,e)	Neutre à droite	x*e=x
Ng(*,e)	Neutre à gauche	e*x=x
Cd(f)	Commutativité droite	f(f(x,y),z)=f(f(x,z),y)
Cl(f)	Commutativité gauche	f(x,f(y,z))=f(y,f(x,z))
L(*)	Crochets de Lie	(x*y)*z=z*(y*x)
T(f,g)	Transitivité	f(g(x,y), g(y,z))= f(g(x,y),g(x,z))

Nous utiliserons également les théories suivantes

AG	Groupe abélien	A(*), C(*), InvD(*,e), Nd(*,e)

QG	Quasi groupe	Sl(.,`,h), Sl(.,`,h), Sr(.,/,h), Sr(.,/,h) avec h = identité
	Minus	AE(-,+), Iv(-)
	Equal	C(eq), T(et, eq)
ABS	Arbres binaires signés	AE(-,+), Iv(-), Sd(+,+,-), Sg(+,+,-)
FH		Ng(*,e), f(x*y) = f(y)
DgAN		Dg(f,g), A(g), Nd(f,e), Ng(f,e)

A la suite de Siekmann et Szabo [Siekmann1984,Szabo1982], nous introduisons la notion de type d'une théorie.

Définition 32 : On dit qu'une théorie A est de **type unitaire** (noté 1) si toute équation a un ensemble minimal complet de A-solutions contenant au plus un élément :

$$t(A) = 1 \Leftrightarrow \forall\, t, t' \in M(F,X)\ |ECUM(t==t',A)| \leq 1$$

Elle de **type fini** (noté +) si toute équation a un ensemble complet et fini de A-unificateurs.

$t(A) = + \Leftrightarrow \forall\, t, t' \in M(F,X)$, il existe un ECU de t et t' de cardinal fini

Elle est de **type infinitaire** (noté ∞) ssi toute équation a un ensemble minimal complet de A-solutions et il existe des équations qui ont un ensemble minimal complet de A-unificateurs de cardinal non fini.

$t(A) = \infty \Leftrightarrow \forall\, t, t' \in M(F,X)$ ECUM(t==t',A) existe

et $\exists\, t, t' \in M(F,X)$ avec $|ECUM(t==t',A)| = \infty$

Si elle n'est d'aucun des types précédents alors on dit qu'elle est de type zéro, noté $\odot$. ○

Dans le tableau récapitulatif suivant, qui tient compte des résultats obtenus dans la suite de ce travail, nous désignons par déci toute théorie pour laquelle la A-unification est décidable.

Théorie	Type	Déci	Remarques et références
ø	1	oui	Des algorithmes ont été proposés par Herbrand [Herbrand1930], Robinson [Robinson1965,Robinson1971], Huet [Huet1976], Martelli et Montanari [Martelli1982], Paterson et Wegman [Paterson1978], Corbin et Bidoit [Corbin1983], Fages [Fages1983].
A(f)	∞	oui	Une procédure générant un ensemble minimal complet de A-unificateurs a été donnée par Plotkin [Plotkin1972] et la décidabilité a été montrée par Makanin [Makanin1977].
C(f)	+	oui	Siekmann [Siekmann1979]. Voir le chapitre sur les théories syntaxiques.
I(f)	+	oui	Raulefs and Siekmann [Raulefs1978] ou par surréduction [Hullot1980].
A+C	+	oui	Stickel [Stickel1981] donne un algorithme dont Fages a prouvé la terminaison [Fages1984], pour un ensemble fini de symboles AC. Voir également [Livesey1976] et [Hullot1980]. Nous donnons dans le chapitre 4 un algorithme standard.
A(f)+I(f)	?	oui	Pour la décidabilité voir Szabo [Szabo1982].
C(f)+I(f)	+	oui	Raulefs et Siekmann [Raulefs1978] ou par R,E-surréduction [Jouannaud1983]. Voir également le chapitre sur la RE-surréduction dans ce travail.
A+C+I	+	oui	Livesey et Siekmann [Livesey1976] ou par R,E-surréduction. Voir aussi [Stickel1981] et [Fages1984].

Théorie	Type	Déci	Remarques et références
Dg ou Dd	+	oui	Arnborg et Tiden [Arnborg1985] donnent un algorithme standard dont la preuve de terminaison est intéressante.
DgAN	?	non	L'indécidabilité a été prouvée par Arnborg et Tiden [Arnborg1985].
D(f,g)	∞	?	Szabo [Szabo1982].
D(f,g)+A(f)	∞	non	Szabo [Szabo1982]
D(f,g)+C(f)	∞	?	Szabo [Szabo1982]
D(f,g)+A(f)+I(f)	?	oui	Szabo [Szabo1982]
D(f,g)+A(f)+C(f)	∞	non	Szabo [Szabo1982]
H(f,*,+)	1	oui	Vogel [Vogel1978]
H(f,*,+)+A(f)	∞	oui	Vogel [Vogel1978]
H(f,*,+)+A(f)+C(f)	∞	oui	Vogel [Vogel1978]
E+A+C	∞	?	Vogel [Vogel1978]
Minus	∞/+	oui	Kirchner [Kirchner1984]. Quand il existe seulement des symboles d'arité impaire, le type est fini.
T(f,g)	+	oui	Chapitre 3 de ce travail.
T(f,g)+C(f)	+	oui	Chapitre 3 de ce travail.
T(f,g)+C(f)+C(g)	+	oui	Chapitre 3 de ce travail.
Cr(f)	+	oui	Chapitre 5 de ce travail. Jeanrond [Jeanrond1980] donne un algorithme sans preuve de correction ni de terminaison.
QG	+	oui	Hullot par surréduction [Hullot1980]
AG	+	oui	Lankford, Butler and Brady[Lankford1984].
ABS	∞/+	oui	Kirchner C. et Kirchner H. par R,E-surréduction [Kirchner1982]. Quand il n'y a que des symboles d'arité impaire le type est fini.
FH	⊙	?	Fages et Huet montrent dans [Fages1983] que l'équation f(x) =? f(a) n'a pas d'ensemble minimal complet de FH-unificateurs.

3.3. Surréduction

La surréduction est à l'unification ce que la réécriture est au filtrage. Nous verrons dans la suite le rôle important que joue cette relation dans la résolution d'équations.

Définition 33 : Soit A une théorie, (R,E) un système de réécriture équationnel tel que A = RUE. Les règles de réécriture de R sont notées $g \to d$. On note -^->RE la **surréduction** associée à R et à $Unif_E$, définie par :

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

ssi σ appartient à $Unif_E(t_{|m}, g)$ et $t' = \sigma(t[m \leftarrow d])$.

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 $Unif_E$ pourront être implicites. ○

Notons que si t -^->RE[σ,m,$g \to d$] t' alors σ(t) ->RE[σ,m,$g \to d$] t'.

Exemple 10 :

1) Si on prend comme méthode d'unification $UP_\emptyset$, on retrouve la notion habituelle de surréduction [Fay1978,Hullot1980].

2) Si on choisit $Unif_E = ECU_E$, on obtient la notion de R,E-surréduction [Jouannaud1983].

4. Complétion équationnelle : une synthèse

4.1. Motivations

A l'origine, la procédure de complétion de Knuth et Bendix a pour but de calculer un système de réécriture R qui génère la même équivalence sur les termes qu'un ensemble donné d'axiomes A. De plus le système résultant a la propriété de Church-Rosser, ce qui permet de décider de l'égalité de deux termes à l'aide de réécritures uniquement, à condition d'avoir la propriété de terminaison de R. Une méthode de preuve de la A-égalité de deux termes consiste alors à calculer leur 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 perdre la propriété de terminaison. 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 équationnel (c'est-à-dire un couple (R,E) d'un ensemble de règles R et d'un ensemble d'équations E) équivalent à A. Citons quelques approches particulières de ce problème: celle de Huet, restreinte à des ensembles de règles linéaires à gauche, celle de Peterson et Stickel, pour laquelle l'ensemble des axiomes non orientables E doit être composé d'équations linéaires, enfin celle de Jouannaud qui permet de s'affranchir de toutes ces conditions de linéarité, généralisant par là les deux approches précédentes. 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: les restrictions apparaissent non pas dans le type de règles ou d'équations, mais sur 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 la réécriture d'une classe d'équivalence, on définit une réécriture sur les termes tenant compte des équations de E. Grâce à la vision unifiée de ces différentes réécritures sur les termes que nous avons introduite, une propriété de Church-Rosser modulo E, paramétrée par la relation de réécriture, a été définie : 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 $T\downarrow$ qui 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. Un algorithme de complétion équationnel a pour objectif de calculer un système de règles possédant ces deux propriétés et équivalent à 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. J.P. Jouannaud et H. Kirchner ont donné un cadre général permettant

de les englober [Jouannaud1984]. Nous donnons à la fin de cette section une procédure de complétion, généralisant leurs travaux, et obtenue en collaboration avec Hélène Kirchner. Elle est prouvée dans sa thèse [Kirchner1985]. Pour une notion de réécriture sur les termes donnée, elle permet d'engendrer à partir d'un ensemble d'équations E et d'un ensemble de paires de termes R, un système de réécriture équationnel 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 les symboles de fonction 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 equations E constitue une théorie prédéfinie de REVEUR 3.
- Le système permet de faire des expérimentations, en ce sens qu'une complétion est paramétrisée par le type de réécriture choisi par l'utilisateur. Celui-ci peut donc essayer de compléter une 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 en ajoutant des extensions de règles plus ou moins systématiquement. L'autre porte sur la façon de superposer deux règles entre elles et permet par exemple de choisir de traiter les plus petites d'abord.

Les expérimentations réalisées jusqu'à présent ont mis en évidence

l'importance du choix de la réécriture, à la fois pour la **terminaison** de la complétion et pour la rapidité de celle-ci.

Nous présentons maintenant une procédure de complétion équationnelle faisant la synthèse des travaux dans le domaine. Les notions essentielles utilisées dans la suite sont définies. Les preuves des résultats énoncés dans cette partie peuvent être trouvées dans le thèse d'Hélène Kirchner [loc.cit.]. Enfin en annexe, nous donnons un article décrivant l'implantation de la procédure de complétion équationnelle dans le laboratoire de réécriture REVEUR3.

4.2. Paires critiques

A toute notion d'unification correspond une notion adaptée de paires critiques.

Définition 34 : 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 T de superpositions si et seulement si $T = Unif_E(t,t'|_u)$.
Etant données deux règles $g \to d$ et $l \to r$ telles que Var(g) et Var(l) soient disjoints, et que l se superpose dans g à l'occurrence u avec l'ensemble complet de superpositions T, alors l'ensemble

$$\{(p,q) \mid p=\tau(d),\ q=\tau(g[u \leftarrow r]) \text{ pour tout } \tau \text{ dans } T\}$$

est un **ensemble complet de paires critiques** de la règle $l \to r$ dans la règle $g \to d$ à l'occurrence u. A chaque superposition τ de T est associé le terme superposé $\tau(g)$. ○

Exemple 11 :

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 $Unif_E$

- 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 $\rightarrow_{R_1 \cup R_{n1}, E}$.

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 $\sigma(t) =_E \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap (Var(t) \cup Var(t'))$.

- si t (ou symétriquement t') est un membre gauche d'équation:

l'ensemble des E-unificateurs σ de t et t' tels que $\sigma(t) =_E \sigma(t')$, les axiomes de E n'étant appliqués qu'en dessous des occurrences de variables de $D(\sigma) \cap Var(t')$.

On définit ainsi une notion de paire critique adaptée à la réécriture de Pedersen.

Définition 35 : 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 3 : Supposons que

- soit $t \rightarrow_{R[\varepsilon,\sigma,l\rightarrow r]} t_1$ ou $t \mid-_{E[\varepsilon,\sigma,l\rightarrow r]} t_1$ et $t \rightarrow_{RE[m,\sigma,g\rightarrow d]} t_2$ où m est une occurrence de non variable dans Dom(l),
- soit $t \rightarrow_{RE[\varepsilon,\sigma,l\rightarrow r]} t_1$ et $t \mid-_{E[m,\sigma,g\rightarrow d]} t_2$ où m est une occurrence de non variable dans Dom(l) différente de ε. Supposons de plus que le préordre associé à $\rightarrow_{RE}$ est conservé sur les sous-termes.

Alors il existe une paire critique (p,q) dans un ensemble complet de paires critiques de $g \rightarrow d$ dans $l \rightarrow r$ à l'occurrence m telle que $p \leq_E t_1$ et $q \leq_E t_2$, où $\leq_E$ est le préordre associé à la méthode de filtrage utilisée dans l'application de $\rightarrow_{RE}$.

Preuve: Dans les deux premiers cas, σ appartient à $Filtre_E(g, t_{|m})$ et $t = \sigma(l)$. Donc, par les conditions (2) et (3) de la définition de $UNIF_E$, σ appartient à $UNIF_E(g, l_{|m})$. Il existe alors α appartenant à $Unif_E(g, l_{|m})$ tel que $\alpha \leq_E \sigma$ [V] avec $V = Var(l) \cup Var(g)$. Il existe une substitution γ telle que pour tout terme u tel que $Var(u) \subseteq V$, γ appartient à $Filtre_E(\alpha(u), \sigma(u))$. $(p,q) = (\alpha(r), \alpha(l[m \leftarrow d]))$ est une paire critique de $g \rightarrow d$ dans $l \rightarrow r$ à l'occurrence m telle que, par définition γ appartienne à $Filtre_E(p, t_1)$ et à $Filtre_E(q, t_2)$.

Dans le dernier cas, σ appartient a' $Filtre_E(l,t)$ et σ appartient à $Filtre_E(g,t_{|m})$. Puisque l'ordre est conservé sur les sous-termes, σ appartient à $Filtre_E(l_{|m},t_{|m})$. Donc σ appartient à $UNIF_E(g,l_{|m})$ et on conclut comme dans le cas précédent. □

Exemple 12 :

1) Si on choisit pour $\rightarrow_{RE}$ la relation $\rightarrow_R$, (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 $\rightarrow_{RE}$ la relation $\rightarrow_{R,E}$, (p,q) satisfait la condition suivante: il existe une substitution σ telle que $\sigma(p) =_E t_1$ et $\sigma(q) =_E t_2$.

3) Si on choisit pour $\rightarrow_{RE}$ 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 $\{\sigma_E(q)\}$.

4.2.1. 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 équationnels s'énonce ainsi:

Théorème 4 : Soit R un ensemble de règles E-noethérien, et E une théorie telle que $Unif_E$ existe et que les classes d'équivalence modulo $=_E$ soient finies. Supposons que $\rightarrow_{RE}$ soit la réunion d'un nombre fini n de relations de réécriture équationnelles $\rightarrow_{R_iE}$ telles que

* pour tout $i \in [1..k]$, $\rightarrow_{R_iE}$ est compatible sur $M(F,X)\times M(F,X)$ avec l'ordre associé $\leq^i$ qui est conservé sur les sous-termes,

* pour tout $i \in [k+1..n]$, $\rightarrow_{R_iE}$ est compatible au sommet avec l'ordre associé $\leq^i$ sur $M(F,X)\times M(F,X)$.

Alors $\rightarrow_{RE}$ est Church-Rosser modulo E si et seulement si

- toutes les paires critiques de confluence sont R/E-confluentes modulo E

- toutes les paires critiques de cohérence (p,q) obtenues par superposition d'une règle dans un axiome sont telles qu'il existe p' tel que $p \rightarrow_{RE} p'$ et (p',q) soit R/E-confluente.

- toutes les paires critiques de cohérence (p,q) obtenues par superposition d'un axiome dans une règle de $\bigcup_{i=1}^{k} R_i$ sont telles qu'il existe q' tel que $q \rightarrow_{RE} q'$ et (p,q') soit R/E-confluente.

Exemple 13 : Il est donc possible de mélanger la réécriture standard, la réécriture de Pedersen et celle de Peterson et Stickel en définissant $\bigcup_{i=1}^{k} \rightarrow_{R_iE}$ comme la réunion de la réécriture standard avec un ensemble de règles R1 et de la réécriture à la Pedersen avec un ensemble différent de règles R2 (et dans ce cas k = 2). En effet ces deux types de réécriture ont les mêmes propriétés de compatibilité avec leur ordre associé qui est conservé sur les sous-termes. $\bigcup_{i=k+1}^{n} \rightarrow_{R_iE}$ est réduit à la réécriture de Peterson et Stickel avec un ensemble de règles R_3 (et n = 3 dans ce cas).

4.3. 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'axiomes 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 $=_E$ et tel que l'ordre strict associé soit noethérien).

La procédure de complétion est décrite de façon détaillée dans l'annexe à ce 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 annexe

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 nécessaires, et ajoute des protections ou des extensions si c'est nécessaire, comme cela est décrit dans l'annexe. Rappelons 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, est définie de la façon suivante:

Définition 36 : Posons $\leq = \bigcup_{i=1}^{n} \leq^i$. $l' \rightarrow r'$ est **simplifiable** par $l \rightarrow r$ si et seulement si les deux conditions suivantes sont satisfaites :

1) - ou bien il existe une occurrence u de Dom(l') différente de ε et $l \leq l'|_u$
- ou bien $l \leq^i l'$ et $\rightarrow R+E$ est compatible avec $\leq^i$ sur le couple (l,l').

2) - l'->r' n'est protégée que par des règles elle-mêmes simplifiables.

○

En d'autres termes, ou bien l' est R+E-réductible par l->r à une occurrence différente de ε, ou bien l' est R+E-réductible au sommet par l->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 ->R.

Nous pouvons maintenant décrire la procédure de complétion équationnelle.

```
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' >E q' ALORS l = p', r = q';
                    (P,R) = SIMPLIFICATION(P-{(p,q)},R,l→r)
                    R=E-COMPLETION(P,R∪{l→r},E,>)
           q' > p' ALORS l = q', r = p';
                    (P,R) = SIMPLIFICATION(P-{(p,q)},R,l→r)
                    R=E-COMPLETION(P,R∪{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 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
```

Considérons alors le système de règles R_∞ engendré par la procédure.

Théorème 5 : Soit (R,E) un système de réécriture équationnel 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 $\rightarrow R_\infty E$ est Church-Rosser modulo E.

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 est développée en détail dans l'annexe à ce chapitre dans le cas de la réécriture $\rightarrow_{R_1 \cup R_{n1}, E}$.

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 thèse d'Hélène Kirchner).

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 équationnel tel que

- RUE est équivalent à A
- R est E-noethérien
- R est R/E-Church-Rosser
- les règles de R sont interréduites.

Un tel système est obtenu en normalisant pour la relation ->R/E le système de réécriture retourné par la procédure de complétion équationnelle.

4.4. 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. Citons en particulier les travaux de Winkler et Buchberger [Winkler1985,Winkler1983] et 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 dernier article améliore ces résultats en donnant un critère plus facilement implantable et montre que l'efficacité de la procédure de complétion est considérablement améliorée. Il serait intéressant d'étudier l'extension de leurs critères à la procedure de complétion équationnelle générale.

5. Implementation of a general completion procedure parameterized by built-in theories and strategies

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

Claude Kirchner and Hélène 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 system, 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 [Lescanne1983] 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 E denoted $=_E$, rewriting rule, rewriting system, normal form of a term t denoted hereafter $t\downarrow$, 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 ->(R1URnl,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<-*- t'.

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 powerfull 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' <-*- ql <- 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↓ or an extension. The extension introduced for a non left linear rule l->r is defined as g[u\l]->g[u\r]↓. 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, R left linear rules — only left linear rules are allowed and put in R1

Peterson and Stickel's method:
E linear axioms, R rules — all rules are put in R2

Jouannaud's method:
any E non empty, R1 left linear rules, Rn1 non left linear rules — left linear rules are put in R1; non left linear rules are put in R2

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 Rnl,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 Rnl,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 Rnl,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 ->(RlURnl,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) * (zl ** y)) * z) -> (((x * zl) ** 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

CHAPITRE 2

STANDARDISATION DE L'UNIFICATION

Dans ce chapitre, nous donnons les définitions et résultats généraux permettant de situer le cadre formel dans lequel se situe la suite de ce travail.

Nous considérons la recherche des solutions d'un problème d'unification comme la recherche d'une transformation qui à l'équation de départ associe un ensemble de systèmes d'équations qui soient complètement décomposées, c'est à dire de la forme x == t.
Cette démarche est typique de la résolution d'équations du premier degré dans l'ensemble des réels par exemple, mais n'avait pas à notre connaissance été appliquée à l'unification dans les théories équationnelles, excepté la théorie vide pour laquelle Martelli et Montanari [Martelli1982] ont initialisé la démarche que nous suivons ici. Nous avons appelé cette approche "standard" dans la mesure où elle permet de standardiser la résolution des problèmes d'unification équationnelle.

Cela permettra en particulier la construction aisée de bibliothèques d'algorithmes d'unification qui sont nécessaire dans de nombreuses applications telle l'algorithme de complétion généralisé REVEUR3 décrit dans la suite et issu des travaux théoriques de [Jouannaud1984].

Cela permet également de résoudre le problème du mélange de "bonnes" théories en donnant un cadre unifié et des outils pour la construction d'algorithmes d'unification modulo des propriétés équationnelles faisant intervenir des ensembles disjoints de symboles. Nous pourrons donc traiter par exemple l'unification modulo la transitivité d'un ensemble de symboles F_1 et la commutativité d'un ensemble de symboles F_2 disjoint de F_1.

Nous allons successivement définir les notions de multiéquations, de systèmes et de disjonctions de systèmes de multiéquations. Ces objets seront aussi appelés unificandes. Puis nous étudierons deux types de transformations d'un unificande U1 en un unificande U2 : l'une qui conserve les ensembles d'unificateurs, l'autre qui permet de déduire d'un ensemble générateur des solutions de U2 un ensemble générateur des solutions de U1. Les transformations de ce second type sont nécessaires pour tenir compte de l'introduction ou de l'élimination de variables dans les unificandes.

Comme dans la théorie vide nous pouvons alors définir la décomposition d'une équation en un système d'équations et la fusion qui permet de regrouper les contraintes sur les variables qui sont déjà isolées dans le système. La mutation est le dernier élément de ce triptyque, et contrairement aux deux transformations précédentes elle dépend étroitement de la théorie considérée. C'est en elle que se retrouvent cristallisées les difficultés essentielles de l'unification équationnelle, et c'est pour justifier les opérations de mutation propres aux théories étudiées par la suite que nous introduisons les diverses transformations telles que la généralisation et le remplacement compatible.

Après la phase de simplification d'un unificande, c'est à dire sa transformation en un unificande ne comportant que des équations que l'on sait résoudre directement, nous étudions comment déterminer à partir des ensembles de A-solutions de chaque équation, un ensemble complet de A-solutions de l'unificande de départ. Pour des théories suffisament régulières nous donnons alors un algorithme de résolution des unificandes complètement décomposés.

Nous illustrerons souvent les définitions et résultats par des exemples pris dans les théories suivantes sur M(F,X) (elles sont définies dans le chapitre précédent) : ac, minus et asb (arbres signes binaires).

1. Les unificandes

Définition 37 : On appelle **multiéquation** tout multiensemble non vide de termes e. Nous noterons Var(e) l'ensemble des variables ayant une occurrence dans un terme de la multiéquation, V(e) l'ensemble des termes réduits à une variable et Term(e) l'ensemble des termes non variables de e. Une **équation** est une multiéquation réduite à deux termes. Une substitution α est **A-solution** de la multiéquation e si et seulement si pour tout élément r et s de e on a $\alpha(s) =_A \alpha(r)$. L'ensemble de toutes les substitutions qui sont A-solutions de e est noté ES(e,A).

Exemple 14 : Avec e = {{ x, x, y, -x, a+(-z) }} on a V(e) = {x, y}, Var(e) = {x, y, z}, Term(e) = {-x, a+(-z)}.

e sera aussi notée : e = (x == x == y == -x == a+(-z)) ou encore e = (x == x == y == {-x, a+(-z))}.

La substitution σ = (x ← a+(-a), y ← a+(-a), z ← a, w ← a) est une minus-solution de e. Noter qu'ici le domaine de σ n'est pas inclus dans Var(e). Nous n'avons aucune restriction à ce sujet dans la définition.

Définition 38 : Un **système** de multiéquations est un multiensemble S de multiéquations. L'ensemble des variables ayant une occurrence dans le système S est noté Var(S). On a donc $Var(S) = \bigcup_{e \in S} Var(e)$. Une substitution σ est A-solution du système S si et seulement si σ est A-solution de chaque élément de S. L'ensemble de toutes les substitutions qui sont A-solutions du système S est noté ES(S,A). On a donc $ES(S,A) = \bigcap_{e \in S} ES(e,A)$.

Notation : Le système S contenant les multiéquations $e_1, \ldots, e_n$ sera noté $S = \{e_1, \ldots, e_n\}$ ou encore $\{e_i\}_{i=1\ldots n}$.

La notion de disjonction de systèmes que nous allons définir maintenant s'introduit dès que l'on souhaite travailler dans des théories aussi simples que la commutativité. En effet, dans cette dernière théorie par exemple, on souhaite pouvoir formaliser le rapport entre l'équation (a+b == x+y) et la disjonction des systèmes d'équations S_1 = {x==a, y==b} et S_2 = {x==b, y==a}. Cela nécessite donc de définir l'objet $[S_1, S_2]$ ainsi que les opérations nécessaires sur cet objet.

Définition 39 : Un multiensemble U de systèmes de multiéquations est appelé une **disjonction de systèmes.** L'ensemble des variables ayant une occurrence dans l'un des systèmes appartenant à U est noté Var(U) : $Var(U) = \bigcup_{S \in U} Var(S)$. Une substitution σ est A-solution de la disjonction de

systèmes U si et seulement si σ est A-solution d'au moins un système de U. En notant ES(U, A) l'ensemble des substitutions qui sont A-solutions de U, on a par définition : $ES(U, A) = \bigcup_{S \in U} ES(S,A)$.

Notation : La disjonction de systèmes U contenant les systèmes $S_1, \ldots, S_n$ sera noté $U = [S_1, \ldots, S_n]$ ou encore $[S_i]_{i=1\ldots n}$.

Exemple 15 : Soient les systèmes S_1 = {x==y==a+(x+b), x==-x} et S_2 = { a+(a+z)==x+z }. $U = [S_1, S_2]$ est une disjonction de systèmes.

Définition 40 : On appelle **unificande** un problème quelconque d'unification, c'est à dire une multiéquation, un système ou une disjonction de systèmes. Une **A-solution** d'un unificande U est donc une substitution σ qui est A-solution du problème d'unification U considéré. ○

Un unificande vide a un ensemble vide de A-solution.

Nous introduisons maintenant la notion désormais classique de système générateur d'un ensemble de A-unificateurs d'un unificande : les ensembles complets, éventuellement minimaux, de A-unificateurs. La définition que nous en donnons est proche de celle de [Plotkin1972] reprise par G.Huet [Huet1976] et J.M.Hullot [Hullot1980].

Définition 41 : Soit U un unificande et W un ensemble de variables contenant Var(U). Σ est un **ensemble complet de A-unificateurs** de U en dehors de W si et seulement si :

(1) $\forall \sigma \in \Sigma$, $D(\sigma) \subseteq Var(U)$ and $I(\sigma) \cap W = \emptyset$

(2) $\forall \sigma \in \Sigma$, $\sigma \in ES(U,A)$

(3) $\forall \sigma' \in ES(U,A)$, $\exists \sigma \in \Sigma$ tel que $\sigma \leq_A \sigma' [Var(U)]$

De plus Σ est dit minimal si la condition suivante est satisfaite :

(4) $\forall \sigma, \sigma' \in \Sigma, \sigma \leq_A \sigma'$ [Var(U)] => $\sigma = \sigma'$

Un ensemble complet de A-unificateurs en dehors de W sera noté $ECU^W(U,A)$, si de plus il est minimal on le notera $ECUM^W(U,A)$. Nous omettrons W lorsque cela ne prête pas à confusion. ○

Montrons tout d'abord comment calculer un ensemble complet d'unificateurs d'une disjonction de systèmes en fonction des ECU des systèmes qui le composent.

Lemme 4 : Soit $U = [Si]_{i\in I}$ et W un ensemble de variables contenant Var(U). Alors $\bigcup_{i\in I} ECU^W(Si,A)$ est un ensemble complet de A-unificateurs de U en dehors de W.

Preuve: (1) La condition (1) est clairement satisfaite.

(2) on a $\bigcup_{i\in I} ECU^W(Si,A) \subseteq ES(U,A)$

(3) $\forall \sigma \in ES(U,A)$, $\exists i \in I$ tel que $\sigma \in ES(Si,A)$.

Donc $\exists \alpha \in ECU^W(Si,A)$ tel que $\alpha \leq_A \sigma$ [Var(Si)]. Mais $D(\alpha) \subseteq Var(Si)$ et par conséquent $\alpha \leq_A \sigma$ [Var(U)], ce qui assure que α appartient à $ECU^W(U,A)$. □

Remarque : Malheureusement on ne peut pas étendre la proposition précédente à des ensemble minimaux complets de A-unificateurs comme le montre l'exemple suivant. Soit A une théorie équationnelle quelconque et

D = [S_1, S_2] avec

S_1 = {x == y} et

S_2 = {x == a,

y == a}

S_1, S_2 et D ont respectivement pour ensemble complet minimal de A-unificateurs {(x←z), (y←z)}, {(x←a), (y←a)} et {(x←z), (y←z)}. On n'a donc pas ECUM(D,A) = ECUM(S_1,A) ∪ ECUM(S_2,A).

2. A-équivalence

Les transformations d'unificandes les plus simples que l'on utilise couramment sont celles qui conservent l'ensemble des A-solutions.

Définition 42 : Deux unificandes U1 et U2 sont **A-équivalents** si et seulement si ils ont même ensemble de A-solutions. Cela sera noté U1 $\Leftrightarrow_A$ U2 ou U1 ⇔ U2 s'il n'y a pas d'ambiguité sur la théorie A. ○

Exemple 16 : Si le symbole + est commutatif, l'équation a+b == x+y est équivalente à la disjonction de système [{(a==x), (b==y)}, {(a==y), (b==x)}].

Les résultats suivants donnent des exemples importants d'unificandes A-équivalents. Le premier montre le rôle du remplacement des variables dans une équation.

Lemme 5 : Soit e = (V(e) == {t}) une multiéquation qui ne possède qu'un seul terme non variable et au moins un terme variable. Soit x une variable de V(e) et ξ la substitution de domaine V(e) définie par : $\xi(z)$ = x pour tout z dans V(e). Alors e et é = (V(e) == ξ(t)) sont A-équivalentes.

Preuve: Si V(e) est réduit à une variable, c'est évident. Sinon on peut supposer sans perte de généralité que V(e) est réduit à 2 variables V(e) = {x,y}. Soit σ une A-solution de e. On a donc $\sigma(x) =_A \sigma(y)$ et par conséquent $\sigma(t) =_A \sigma(\xi(t))$, donc σ est également A-solution de

é. Réciproquement, si σ est A-solution de é, σ est A-solution de x == y donc σ(t) = σ(t'). Ce qui assure que σ est A-solution de e. □

Exemple 17 : e = (x == y == z == f(x,f(x,y))) a le même ensemble de A-solutions que é = (x == y == z == f(x,f(x,x))).

Pour certaines théories, comme l'associativité-commutativité, cela peut permettre de réduire substantiellement l'espace de recherche des A-solutions en ne travaillant plus que sur une seule variable. Pour d'autres théories, comme les arbres signés binaires, pour lesquelles on sait mieux résoudre les multiéquations linéaires par exemple, on peut utiliser la réciproque.

D'autre part, la A-équivalence est également compatible avec la réécriture.

Lemme 6 : Soit R un système de réécriture tel que $=_R \subseteq =_A$. Si t_1 -*->R t_1' et t_2 -*->R t_2' alors $t_1 == t_2 \Leftrightarrow_A t_1' == t_2'$.

Preuve: Pour toute A-solution σ de $t_1 == t_2$ on a

$$\sigma(t_1') =_R \sigma(t_1) =_A \sigma(t_2) =_R \sigma(t_2')$$

donc σ est une A-solution de $t_1' == t_2'$. Réciproquement, si σ est une A-solution de $t_1' == t_2'$ par le même type de raisonnement on en déduit que σ est une A-solution de $t_1 == t_2$. □

Remarque : Dans le lemme précédent, on ne suppose rien sur la réécriture : ni terminaison, ni convergence ou autre. Ce résultat permet de chercher les

solutions sur des formes particulières d'équations obtenues après réécriture préalable de l'équation de départ par une relation de réécriture appropriée incluse dans $=_A$.

Si un symbole d'opération f d'arité 2 est régulier à gauche par exemple, c'est à dire tel que pour tous termes u, v et w, f(u,v) $=_A$ f(u,w) => v $=_A$ w, alors toute équation est simplifiable à gauche dans le sens suivant :

Définition 43 : Une équation e = (f(u,v) == f(u,w)) est **simplifiable à gauche** ssi e $\Leftrightarrow_A$ (v == w). ○

Lemme 7 : Si f est un symbole régulier à gauche alors toute équation e = (f(u,v) == f(u,w)) est simplifiable à gauche.

Preuve: Il faut montrer que f(u,v) == f(u,w) $\Leftrightarrow_A$ v == w.

Si α est une A-solution de v == w, il est clair que c'est une A-solution de e.

Réciproquement, si α(f(u,v)) $=_A$ α(f(u,w)) alors puisque f est régulière on a α(v) $=_A$ α(w). □

Nous avons un résultat similaire pour la simplifiabilité à droite qui se définit de façon évidente.

Par ailleurs, l'intérêt fondamental de la A-équivalence est qu'elle conserve les ensembles complets, éventuellement minimaux, de A-solutions.

Proposition 1 : Soient U et U' deux unificandes, Z = Var(U) ∩ Var(U'), W un ensemble de variables contenant Var(U) ∪ Var(U'), et ECU un ensemble complet de A-solutions de U en dehors de W. U est A-équivalent à U' si et seulement si

$$\Sigma = \{\sigma_{|Z} \mid \sigma \in ECU\}$$

est un ensemble complet de A-solutions de U et U'. De plus, si on suppose que ECU est un ensemble complet minimal de A-solutions de U alors Σ est lui aussi minimal.

Preuve: C'est une généralisation à des unificandes de la proposition 1.2 de [Kirchner1982].

Notons tout d'abord que si Σ est un ECU commun à U et U' alors U et U' sont clairement A-équivalents.

Réciproquement, si U est A-équivalent à U' vérifions que Σ est bien un ensemble complet de A-solutions de U et U'. Pour cela remarquons qu'il suffit de montrer que Σ est un ensemble complet de A-solutions de U'.

(1) est trivialement vérifiée compte tenu des hypothèses.

(2) Si σ est un A-unificateur de U alors $\sigma_{|Var(U)}$ est également A-solution de U donc de U' par hypothèse donc $\sigma_{|Z}$ également. Donc si U et U' sont A-équivalents $\sigma \in ES(U,A) \Rightarrow \sigma_{|Z} \in ES(U,A) \cap ES(U',A)$. Donc $\Sigma \subseteq ES(U',A)$.

(3) Soit α une A-solution de U'. Donc $\alpha_{|Z} \in ES(U,A)$. Il existe donc σ dans ECU tel que $\sigma_{|Var(U)} \leq_A \alpha_{|Z}$, d'où

$$\sigma_{|Z} \leq_A \sigma_{|Var(U)} \leq_A \alpha_{|Z} \leq_A \alpha_{|Var(U')}$$

ce qui prouve la complétude.

Si on suppose par ailleurs que ECU est un ensemble complet minimal de A-solutions de U, montrons qu'il en est de même de Σ pour U'. Soient $\alpha = \sigma_{|Z}$ et $\alpha' = \sigma'_{|Z}$ avec σ et σ' dans ECU, deux éléments

de Σ tels que $\alpha \leq_A \alpha'$ [Var(U')]. On a donc aussi $\alpha_{|Var(U)} \leq_A \alpha'_{|Var(U)}$ et

$$\alpha'_{|Var(U)} = \alpha' = \sigma'_{|Z} \leq_A \sigma'_{|Var(U)}$$

Comme de plus α est élément de ES(U,A) il existe τ dans ECU tel que $\tau_{|Var(U)} \leq_A \alpha_{|Var(U)}$. D'où

$$\tau_{|Var(U)} \leq_A \alpha_{|Var(U)} \leq_A \sigma_{|Var(U)}$$

et comme ECU est minimal

$$\tau_{|Var(U)} =_A \alpha_{|Var(U)} =_A \sigma_{|Var(U)}$$

car σ et τ sont éléments de ECU. D'où

$$\sigma_{|Var(U)} = \alpha_{|Var(U)} \leq_A \alpha'_{|Var(U)} \leq_A \sigma'_{|Var(U)}$$

comme σ et σ' sont dans ECU, elles sont égales sur Var(U), ce qui entraine l'égalité de α et α' sur Var(U) donc sur Z. □

Ce résultat est important car il permet de construire un ensemble complet de A-solutions d'un unificande en construisant un autre unificande à partir du premier par des transformations successives conservant la A-équivalence. Mais comme nous allons le voir dans la suite les transformations considérées ne préservent pas toujours la A-équivalence mais plutôt la A-dépendance que nous introduisons maintenant.

3. A-dépendance

La A-équivalence qui préserve les ensembles de A-solutions n'est pas suffisante pour étudier certaines transformations des unificandes telle que

la généralisation. En fait ce qui nous intéresse n'est pas tant l'ensemble des A-solutions qu'une base de cet ensemble, d'où la notion de A-dépendance.

Définition 44 : Soient U1 et U2 deux unificandes et W un ensemble de variables contenant Var(U2). U2 est dit **A-dépendant** de U1 en dehors de W ou encore U1 **A-étend** U2 en dehors de W, si et seulement si

(1) $(Var(U1) - Var(U2)) \cap W = \emptyset$

(2) Pour tout ensemble complet de A-solutions Σ de U1, $\Sigma_{|Var(U2)}$ est un ensemble complet de A-solutions de U2.

Cela sera noté $U1 \Rightarrow_A^W U2$. W pourra être omis. ○

La première condition n'est qu'une condition technique permettant de s'assurer que les "nouvelles" variables introduites le sont en dehors de celles qui sont déjà utilisées. Il faut également remarquer que toutes les variables de U2 ne figurent pas nécessairement dans U1.

Exemple 18 : L'équation e = ((x+a)+f(z,b)) == z+b) se A-étend dans le système S = { ((x+a)+y == z+b), y == f(z,b) } et ceci pour une théorie arbitraire.

Etudions maintenant les propriétés de la A-dépendance. Tout d'abord montrons que moyennant de bonnes hypothèses sur les ensembles de variables $\Rightarrow_A$ est un préordre dont $\Leftrightarrow_A$ est l'équivalence associée.

Lemme 8 :

(1) Si $U1 \Rightarrow_A^{W_1} U2$ et $U2 \Rightarrow_A^{W_2} U3$ avec $W2 \subseteq W1$ alors $U1 \Rightarrow_A^{W_2} U3$.

(2) Si U1 et U2 sont deux disjonctions de systèmes alors

$$(U1 \Leftrightarrow_A U2) \Leftrightarrow ((U1 \Rightarrow_\lambda U2) \text{ et } (U2 \Rightarrow_A U1))$$

Preuve: (1) $\Rightarrow_A$ est "transitive" : Soit Σ un ensemble complet de A-unificateurs de U1. Par hypothèse on a donc $\Sigma_{|Var(U2)}$ ECU de U2 et par conséquent $\Sigma_{|Var(U2)|Var(U3)} = \Sigma_{|Var(U1)\cap Var(U2)\cap Var(U3)}$ est un ECU de U3.

Mais par ailleurs les conditions $(Var(U1) - Var(U2)) \cap W = \emptyset$, $(Var(U2) - Var(U3)) \cap W = \emptyset$, $Var(U2) \subseteq W1$ et $Var(U3) \subseteq W2$ impliquent que $Var(U1) \cap Var(U2) \cap Var(U3) = Var(U1) \cap Var(U3)$.
En effet $\forall x \in Var(U1) \cap Var(U3)$ si $x \notin Var(U2)$ alors $x \in X-U2$.
Comme d'autre part $x \in Var(U3) \Rightarrow x \in W2 \Rightarrow x \in W1$, on a $x \in U1 \cap X-U2 \cap W2$, ce qui contredit $(Var(U1) - Var(U2)) \cap W = \emptyset$.

Donc $\Sigma_{|Var(U1)\cap Var(U2)\cap Var(U3)} = \Sigma_{|Var(U1)\cap Var(U3)} = \Sigma_{|Var(U3)}$ est un ECU de U3, ce qui prouve le premier résultat.

Si $U1 \Rightarrow_A U2$ et si Σ est un ECU de U1 alors $\Sigma_{|Var(U2)}$ est un ECU de U2. Mais comme $U2 \Rightarrow_A U1$, $\Sigma_{|Var(U2)|Var(U1)} = \Sigma_{|Var(U1)\cap Var(U2)}$ est un ECU de U1.

$\Sigma_{|Var(U1)\cap Var(U2)}$ est donc un ECU commun à U1 et à U2. En appliquant le théorème sur les unificandes A-équivalents on en déduit que U1 et U2 sont A-équivalents. □

Nous allons maintenant voir des exemples importants de systèmes A-dépendants qui seront souvent utiles dans la suite. Tout d'abord nous étudions la relation entre un unificande et son ensemble de A-solutions considéré lui même comme un unificande.

Définition 45 : Soit Σ un ensemble de substitutions, on désigne **Eq(Σ)** l'unificande définie par

$$Eq(\Sigma) = [\ \{\ (x == \sigma(x))\ |\ x \in D(\sigma)\}\ |\ \sigma \in \Sigma]$$

Si Σ est un ensemble complet de A-unificateurs de l'unificande U en dehors de l'ensemble de variables W alors $Eq(\Sigma)$ sera noté $Eq^W(U)$. ○

Exemple 19 : Avec

$$\Sigma = \{\sigma = ((x \leftarrow f(a,b)), (y \leftarrow a)),$$
$$\sigma' = ((x \leftarrow f(x,x)), (z \leftarrow y))\}$$

on a

$$Eq(\Sigma) = [\ \{(x == f(a,b)), (y == a)\},$$
$$\{(x == f(x,x)), (z == y)\}]$$

Lemme 9 : Pour tout unificande U et tout ensemble de variables W contenant Var(U) on a : $Eq^W(U) \Rrightarrow_A^W U$.

Preuve: Par définition de $Eq^W(U)$, la condition sur les variables est bien vérifiée.

Notons Σ l'ensemble complet de A-unificateurs dont $Eq^W(U)$ est issue.

Soit α est une A-solution quelconque de U, il existe σ dans Σ telle que $\sigma \leq_A \alpha\ [Var(U)]$. Soit Θ un ECU quelconque de $Eq^W(U)$, σ étant A-solution de $Eq^W(U)$ par définition, il existe μ tel que $\mu \leq_A \sigma\ [Var(Eq^W(U))]$.

Mais comme par définition $D(\sigma) \subseteq Var(U)$ et $D(\mu) \subseteq Var(Eq^W(U))$ on a

$\mu_{|Var(Eq^W(U))\cap Var(U)} = \mu_{|Var(U)}$.

D'où $\mu_{|Var(U)} = \mu_{|Var(Eq_W(U))\cap Var(U)}$

$\leq_A \sigma_{|Var(Eq_W(U))\cap Var(U)} \leq_A \sigma_{|Var(U)}$

$\leq_A \alpha$,

ce qui prouve que $\Theta_{|Var(U)}$ est un ECU de U. □

Remarque : La réciproque $U \Rrightarrow_A Eq^W(U)$ est en général fausse à cause, par exemple, de nouvelles variables introduites.

Un deuxième exemple important d'utilisation de la A-dépendance est la généralisation des termes constituant une équation. Ceci est utilisé par exemple par M.Stickel [Stickel1981] dans son algorithme d'unification associative commutative.

Définition 46 : Soit $f(t_1,\ldots t_n) == f'(t_1',\ldots,t_p')$ une équation e. Soit W un ensemble de variables tel que $Var(e) \subseteq W$. Soient $x_1,\ldots,x_n,x_1',\ldots,x_p'$ des variables distinctes prises en dehors de W. Soit enfin le système suivant :

$$S = \{f(x_1,\ldots,x_n) == f(x_1',\ldots,x_p'),$$
$$x_1 == t_1,$$
$$\ldots$$
$$x_n == t_n,$$
$$x_1' == t_1',$$
$$\ldots$$
$$x_p' == t_p'\}$$

S s'appelle le **généralisé** de e en dehors de W et se note $Gen^W(e)$. ○

Lemme 10 : Pour toute équation e on a $Gen^W(e) \Rrightarrow_A^W e$.

Preuve: Soit α une A-solution quelconque de e. Soit β la A-solution de $S = Gen^W(e)$ définie par

$$\beta = \alpha_{|Var(e)} \circ (x_1 \leftarrow \alpha(t_1)) \ldots (x_p' \leftarrow \alpha(t_p'))$$

on a bien sur $\alpha_{|Var(e)} = \beta_{|Var(e)}$. Soit enfin Θ un ECU quelconque de S. Il existe donc μ dans Θ tel que $\mu \leq_A \beta$ [Var(S)], donc $\mu \leq_A \beta$ [Var(e)] d'où $\mu \leq_A \alpha$ [Var(e)], ce qui montre que $\Theta_{|Var(e)}$ est un ECU de e. □

Nous allons maintenant justifier les remplacements que l'on peut faire dans un unificande, en remplaçant un unificande par un autre unificande qui lui est A-équivalent ou qui le A-généralise.

Lemme 11 : Soit S un système et W un ensemble de variables tel que $Var(S) \subseteq W$. Si une multiéquation e du système S se A-étend en dehors de W en (respectivement est A-équivalente à) la disjonction de systèmes $U = [S_j]_{j\in J}$ alors S se A-étend en dehors de W en (resp. est A-équivalent à) la disjonction de systèmes $U' = [S_j \cup (S-\{e\})]_{j\in J}$.

Preuve: Posons $S' = S - \{e\}$. Soit Σ un ECU de U'.

a) Montrons d'abord que $\Sigma_{|Var(S)} \subseteq ES(S,A)$.

$\forall \sigma \in \Sigma$, $\exists j \in J$ tel que σ est A-solution de $S' \cup S_j$, donc $\Sigma_{|Var(S)}$ est A-solution de S'. Il reste à montrer que $\Sigma_{|Var(S)}$ est aussi A-solution de e.

Or comme σ est A-solution de S_j, Θ étant un ECU quelconque de U, il existe $\mu \in \Theta$ et une substitution α tels que

$$\alpha.\mu_{|Var(S)} = \sigma_{|Var(S)}$$

or $D(\mu) \subseteq Var(U)$ donc

$$\alpha.\mu_{|Var(e)} = \sigma_{|Var(S)\cap Var(e)}$$

Comme $\mu_{|Var(e)}$ est A-solution de e par hypothèse, $\sigma_{|Var(S)\cap Var(e)}$ est également A-solution de e donc $\sigma_{|Var(S)}$ est également A-solution de e.

b) Montrons maintenant que $\Sigma_{|Var(S)}$ est un ECU de S. Soit α une A-solution de S et Θ un ECU de U. On peut sans manquer de généralité supposer que $D(\alpha) \subseteq Var(S)$.

$$\alpha \in ES(e,A) \Rightarrow \exists\, \mu \in \Theta$$

et une substitution λ tels que $\lambda.\mu_{|Var(e)} = \alpha_{|Var(e)}$ Comme $D(\lambda\mu) \cap \alpha \subseteq Var(e)$ et que ces 2 applications sont égales sur Var(e), on peut considérer l'application $\alpha+\lambda\mu$. $\alpha+\lambda\mu$ étant A-solution de U', il existe σ dans Σ et une substitution τ tels que

$$\tau.\sigma_{|Var(U')} = (\alpha+\lambda\mu)_{|Var(U')} = \alpha_{|Var(U')} + \lambda\mu_{|Var(U')}$$

or le domaine de σ est précisément Var(U') donc $\sigma_{|Var(U')} = \sigma$, donc

$$\tau.\sigma_{|Var(U')\cap Var(S)} = \tau.\sigma_{|Var(S)} =$$

$$\alpha_{|Var(U')\cap Var(S)} + \lambda\mu_{|Var(U')\cap Var(S)}$$

or

$$\begin{aligned} &Var(U') \cap Var(S) \cap D(\mu) \\ &\subseteq Var(U') \cap Var(S) \cap Var(U) \\ &\subseteq Var(U') \cap Var(e) \subseteq Var(e). \end{aligned}$$

Par conséquent on a

$$\lambda\mu_{|Var(U')\cap Var(S)} \leq_A \lambda\mu_{|Var(e)}$$

et donc

$$\tau.s|_{Var(S)} \leq_A \sigma|_{Var(S)} + \lambda\mu|_{Var(S)} = \sigma|_{Var(S)}$$

□

La propriété précédente s'étend aux remplacements de systèmes par des disjonctions de systèmes, la preuve en est omise :

Lemme 12 : Soit U1 une disjonction de systèmes et W un ensemble de variables tel que $Var(U1) \subseteq W$. Si un système S de U1 se A-généralise en dehors de W en la disjonction de systèmes (respectivement est A-équivalent à) $U = [S_j]_{j\in J}$ alors U1 se A-généralise en la disjonction de systèmes (resp. est A-équivalent à) $U' = U \cup (U1 - S)$.

4. Simplification d'unificandes

Dans cette section nous allons étudier ce que nous pensons être les trois premières étapes fondamentales d'un algorithme d'unification (tous les algorithmes de A-unification dans une théorie équationnelle à notre connaissance font appel à ces 3 phases de façon plus ou moins évidente). Leur objectif commun est de simplifier l'unificande de départ de façon à obtenir en un temps fini un unificande A-équivalent ou A-dépendant, dont les multiéquations soient complètement décomposées. Les systèmes ainsi obtenus sont aussi appelés par B.Courcelle systèmes réguliers [Courcelle1984]. Leur étude constituera la matière de la section suivante.

Dans toute la suite de cette partie, A désigne une théorie équationnelle quelconque.

4.1. La décomposition

Nous généralisons ici le concept de décomposition introduit par Martelli et Montanari dans le cas de la théorie vide.

Tout d'abord, introduisons un sous-ensemble important de l'ensemble des symboles de l'algèbre considérée : les symboles décomposables. Ce sont les symboles dont l'apparition en tête d'une équation permet de simplifier l'équation sans faire référence aux axiomes de la théorie.

Définition 47 : L'ensemble des **symboles décomposables** est le plus grand sous ensemble F_d de F tel que pour tous éléments f et f' de F_d, pour tous termes $t = f(t_1,\ldots,t_n)$ et $t' = f'(t_i',\ldots,t_p')$ on ait

(1) $f \neq f' \Rightarrow ES(t==t', A) = \emptyset$

(2) $f = f' \Rightarrow (t == t' \Leftrightarrow_A \{t_i==t_i'\}_{i=1,\ldots,n})$.

Les éléments de F_d sont appelés symboles décomposables. ○

En fait la condition de maximalité imposée dans la définition précédente n'est pas essentielle. En pratique on peut prendre comme ensemble de symboles décomposables tout sous ensemble de F_d, mais plus l'ensemble des symboles décomposables sera grand, plus la décomposition associée sera efficace.

Il n'est pas simple de donner une construction systématique de F_d mais on peut noter que :

(1) F_d contient tous les symboles qui n'apparaissent pas comme symbole de tête d'un axiome de la théorie A. Par exemple si l'unique axiome de la théorie est $((x + y) + (- y)) == x$ alors - appartient à F_d.

(2) F_d contient bien sûr tous les symboles qui n'interviennent dans aucun axiome de la théorie.

Nous introduisons maintenant la terminologie nécessaire dans la suite.

Définition 48 : On appelle **top_F_d_terme** tout terme dont le symbole de tête est un symbole décomposable.

L'ensemble des termes variables d'une multiéquation e est noté **V(e)**. L'ensemble des top_F_d_termes de e est noté **T(e)** et l'ensemble des autres éléments de e est noté **P(e)**. P(e) est donc l'ensemble des termes non variables de e et qui ne sont pas des top_F_d_termes. La multiéquation e sera souvent notée e=(V(e)==P(e)==T(e)).

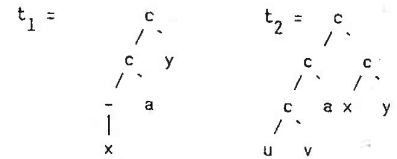
Une multiéquation e est dite **décomposable** si et seulement si T(e) contient au moins deux éléments. Dans le cas contraire e est dite **indécomposable**.

○

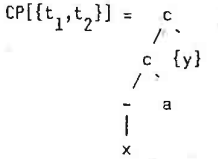
Exemple 20 : Si on suppose que Fd = {f,g} alors pour la multiéquation e = {x, x, y, -x, -z, f(-x, -y), g(a,b)} on a V(e) = {x, x, y}, P(e) = {-x, -z} et T(e) = {f(-x, -y), g(a,b)}.

L'idée de base de la décomposition est que si e a des A-solutions et si la multiéquation est décomposable alors T(e) a une partie commune et un système d'équations aux différences aussi appelé frontière.

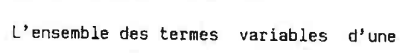
Exemple 21 : Dans le cas de la théorie minus prenons par exemple les termes



leur partie commune est alors



et leur frontière



Définition 49 : Soit M = $\{t_1,\ldots,t_n\}$, un multiensemble non vide de termes ; le top_F_d_terme appelé **partie commune** de M noté CP[M] et le **système d'équations aux différences** ou **frontière** de M noté FR[M] sont récursivement définis par :

SI pour tous i dans [1..n], $t_i(\varepsilon) = f$ avec $f \in F_d$

ALORS CP[M] = $f(CP[\{t_1/1,\ldots,t_n/1\}],\ldots,CP[\{t_1/p,\ldots,t_n/p\}])$
et
FR[M] = $\bigcup_{(j\in[1..p])}$ FR[{$t_1/j,\ldots,t_n/j$}]

SINON SI il existe deux top_F_d_termes t_i et t_j dans M tels que $t_i(\varepsilon) \neq t_j(\varepsilon)$

ALORS ni la partie commune ni la frontière n'existent. (On dit qu'il y a **collision** (ou clash) de symbole. Ceci se propage aux appels englobants).

SINON (il existe i dans [1..n] tel que t_i soit une variable ou ne soit pas un top_F_d_terme)

CP[M] = t_i (cette définition n'est pas déterministe)
et
FR[M] est le système réduit à la multiéquation M.

○

Lemme 13 : Soit e une multiéquation ayant une A-solution σ. Alors e a une partie commune et une frontière et, pour tout terme t de e, $\sigma(t) =_A \sigma(CP[e])$ et σ est A-solution de FR[e].

Preuve: * Si la partie commune et la frontière de e n'existaient pas cela viendrait d'une collision de symboles et par conséquent il n'y aurait pas de A-solution.

* Montrons la seconde partie du lemme par récurrence structurelle

sur CP[e].

Si CP[e] = x avec x ∈ X, alors par définition de la partie commune, x est un terme de e. Par conséquent on a bien $\sigma(x) =_A \sigma(t)$ pour tout t de e. De plus dans ce cas FR[e] = e et donc σ est A-solution de FR[e].

Même raisonnement que dans le cas précédent si CP[e] = $f(t_1,\ldots,t_n)$ et $f \notin F_d$.

Sinon CP[e] = $f(t_1,\ldots,t_n)$ où f est un symbole décomposable ; par définition de la partie commune et pour tout i dans [1..n] t_i est la partie commune de $T_i = \{t/i \mid t \in e\}$. Par définition de F_d toute A-solution de e est également A-solution de T_i et par hypothèse de récurrence σ est donc A-solution des équations $w == t_i$ ou w est un élément quelconque de T_i. Donc pour tout terme t de e $\sigma(t) =_A \sigma(CP[e])$}.

Dans ce cas, FR[e] est la réunion des frontières des T_i qui ont tous σ comme A-solution, donc σ est également A-solution de FR[e]. □

De façon à simplifier au maximum les multiéquations et à détecter le plus rapidement possible les cas d'échec, on va calculer non pas la partie commune de toute la multiéquation mais seulement la partie commune de T(e).

Définition 50 : Soit e une multiéquation, on appelle **décomposition** de e et on note **Dec(e)** le système de multiéquations défini par :

$$Dec(e) = \{(V(e) = P(e) = CP[T(e)])\} \cup FR[T(e)]$$

○

Exemple 22 : Dans la théorie minus, si e=(x==y==-x==t_1==t_2) où t_1 et t_2 sont les termes de l'exemple précédent, alors

Dec(e) = {(x==y==-x==c(c(-x,a),y))} ∪ {(-x==c(u,v)), (y==c(x,y))}

Proposition 2 : Soit e une multiéquation. Ou bien Dec(e) n'existe pas auquel cas e n'a pas de A-solution, ou bien Dec(e) $\Leftrightarrow_A$ e.

Preuve: Si Dec(e) n'existe pas cela provient d'une collision de symboles et par conséquent e n'a pas de A-solution.

Dans le cas contraire, si T(e) est vide ou réduit à un élément, le résultat est immédiat.

Si e = T(e) le lemme précédent permet de conclure.

Plaçons nous donc dans le cas ou P(e) ∪ V(e) ≠ ø et soit u un élément de cet ensemble. Pour toute A-solution σ de e et pour tout élément t de T(e) on a donc $\sigma(t) =_A \sigma(u)$. Par le lemme précédent on a donc $\sigma(u) =_A \sigma(CP[T(e)])$ et donc σ est une A-solution de (V(e) == P(e) == CP[T(e)]) et de FR[T(e)].

Réciproquement, par construction de Dec(e) toute A-solution de Dec(e) est A-solution de e. □

Définition 51 : On appelle **décomposition** d'un système S et on note Dec(S), le système obtenu en remplaçant un élément e de S par sa décomposition dans S : Dec(S) = (S-{e}) ∪ Dec(e). ○

Cette définition est indéterministe mais le choix de la multiéquation à décomposé est indifférent.

Comme conséquence de la proposition précédente et de la conservation de la

A-équivalence par remplacement on obtient :

Corollaire 1 : Pour tout système S, soit il existe une multiéquation e dont la décomposition n'existe pas, auquel cas le système n'a pas de A-solutions. Soit S et Dec(S) sont A-équivalents.

4.2. Fusion d'un système de multiéquations

La fusion d'un système de multiéquations correspond au regroupement des contraintes sur les variables qui sont apparues au cours de la simplification du système.

Paradoxalement, la fusion, conceptuellement très simple comme on va pouvoir le constater dans cette courte section, pose des problèmes d'implantation qui justifient pour une bonne part les différences importantes d'efficacité suivant le type de représentation choisie.

Définition 52 : La **fusion** des multiéquations e et é, notée **Fus(e,é)** est définie par :

* si $V(e) \cap V(é) = \emptyset$ alors Fus(e,é) = {e, é}
* sinon Fus(e,é) = ((V(e) ∪ V(é)) == (P(e) ∪ P(é)) == (T(e) ∪ T(é))). ○

Exemple 23 : Dans la théorie minus, si e=(x==y==-x==-z==c(a,x)==c(a,a)) et é=(x==c(a,a)) alors Fus(e,é) = (x==y==-x==-z==c(a,x)==c(a,a)==c(a,a)).

Lemme 14 : Pour toutes multiéquations e et é, {e, é} $\Leftrightarrow_A$ Fus(e, é)

Preuve: Si $V(e) \cap V(é) = \emptyset$ c'est évident. Sinon soit x une variable de V(e) ∩ V(é). Si σ est une A-solution de {e, é} alors pour tout terme t de e ∪ é, $\sigma(t) =_A \sigma(x)$ et par conséquent, σ est A-solution de Fus(e, é). La réciproque est claire. □

Nous étendons maintenant la notion de fusion à des unificandes.

Définition 53 : Pour tout système S, on appelle fusion de S le système Fus(S) obtenu en remplacant toutes les paires de multiéquations par leur fusion. La fusion d'une disjonction de systèmes U est obtenue en fusionnant chacun des systèmes qui la compose on la note Fus(U). ○

Nous obtenons donc comme corollaire du lemme précédent et des résultats généraux sur la A-équivalence :

Corollaire 2 : Pour tout unificande U, U $\Leftrightarrow_A$ Fus(U).

4.3. Mutation d'un unificande

L'enchaînement des opérations de décomposition et de fusion va permettre d'obtenir à partir d'un unificande quelconque un unificande dont les multiéquations seront des types suivants :

(1) $V(e) == \{p_1,\ldots,p_n\} == \{u\}$ avec $n > 0$,

(2) $V(e) == \{p_1,\ldots,p_n\}$ avec $n > 1$,

(3) V(e) == {t},

(4) V(e).

Définition 54 : Dans les deux derniers cas les multiéquations sont dites **complètement décomposées.** ○

Dans cette partie, nous ne nous intéressons qu'aux deux premiers type de multiéquations. Nous allons donner plusieures façons de continuer la simplification de telles multiéquations, de telle sorte que l'on puisse réitérer les opérations de décomposition et de fusion jusqu'à n'avoir que des équations de type (3) et (4).

Pour cela nous introduisons une nouvelle transformation des unificandes

appelée mutation.

Mais avant de donner les définitions formelles nous allons voir sur des exemples quelques caractéristiques de cette transformation.

Exemple 24 : Dans la théorie vide mutation et décomposition se confondent i.e. on peut considérer la décomposition comme la mutation de la théorie vide.

Exemple 25 : Dans la théorie minus, l'équation e = (x == -x == c(a,b)) est indécomposable et non complètement décomposée. Pour la transformer on va considérer le système équivalent S = {(x == c(a,b)), (-x == c(a,b)))}. L'équation (-x == c(a,b)) est équivalente à l'équation (x == c(-b,-a)). Par conséquent e est équivalent au système S' = {(x == c(a,b)), (x == c(-b,-a)))} qui fusionne en S" = {(x == c(a,b) == c(-b,-a))} pour lequel on peut poursuivre les opérations de décomposition et fusion.

Exemple 26 : Soit maintenant la théorie ou le symbole + est commutatif. L'équation e = ((a*b)+x == (x*b)+y) est indécomposable et non complètement décomposée. On montrera dans la suite qu'elle est équivalente à la disjonction des systèmes

```
U = [{a * b == x * b,
      x == y},
     {x == x * b,
      y == a * b}]
```

pour lesquels on peut poursuivre le processus de simplification.

En l'absence de méthode générale nous allons donner des outils de base permettant d'aborder ce problème.

Définition 55 : Un unificande U est dit **complètement décomposé** si toutes

les multiéquations qui le compose sont complètement décomposées. U est dit **indécomposable** s'il est fusionné et si aucune des multiéquations qui le compose n'est décomposable. ○

De façon à ne pas pouvoir enchaîner une infinité d'opération de mutation, nous allons imposer à cette transformation de faire décroitre une certaine mesure de complexité :

Définition 56 : Soit μ une mesure de complexité des unificandes (c'est à dire une application de la classe des unificandes dans $\mathbb{N}$), relative à la théorie A.
Un unificande fusionné indécomposable et non complètement décomposé U est **mutable** ssi il existe un unificande fusionné **Mut(U)** tel que

* $Mut(U) \Rightarrow_A U$
* $\mu(Mut(U)) < \mu(U)$.

La transformation consistant à remplacer un unificande U par Mut(U) est appelée mutation. Il faut noter que Mut(U) n'est pas en général unique.

On dira qu'une théorie est **mutable** si tout unificande indécomposable et non complètement décomposé est mutable dans cette théorie. ○

Les théories vide, AC, minus, commutative, sont des théories mutables puisque ce sont des théories pour lesquelles existe un algorithme fini d'unification.

Voici quelques méthodes de mutation d'un unificande U indécomposable et non complètement décomposé.

[1] La **décomposition suivant la forme des axiomes.** On prend en compte la forme des axiomes pour décomposer l'équation.

C'est ce qui est utilisé pour traiter la commutativité par exemple et d'une façon plus générale pour les théories que nous appelons syntaxiques dans la suite de ce travail.

[2] La **généralisation d'une équation de l'unificande suivie de sa résolution** :

- On choisit une multiéquation e = $(f(t_1,\ldots,t_n) == f'(t_1',\ldots,t_p'))$ dans U, non complètement décomposée et indécomposable.
- Soit $Gen^W(e)$ le système généralisé issu de e. $Gen^W(e)$ contient par définition l'équation é = $(f(x_1,\ldots,x_n) == f'(x_1',\ldots,x_p'))$. En général il est plus simple de résoudre cette équation plutôt que e. C'est ce qu'on fait dans cette méthode, en remplaçant é par Eq^W(é). L'unificande ainsi obtenu est la mutation de U.

Cette méthode est utilisée par exemple dans l'unification associative commutative pour muter certains types de systèmes comme nous le verrons dans la suite.

[3] La **transformation d'une multiéquation non complètement décomposée**, par des propriétés propres à la théorie considérée, en une multiéquation avec laquelle on peut poursuivre la décomposition.

Ceci est utilisé par exemple pour la théorie minus.

[4] La **surréduction**. Cette méthode générale de résolution d'équations peut, en étant appliquée partiellement, donner la mutation d'un unificande.

[5] La **simplifiabilité**. Si un symbole f non décomposable est régulier alors une équation du type f(u,v) == f(u,w) est A-équivalente à v == w.

Nous reviendrons en détail sur ces outils et nous les appliquerons à la découverte de plusieurs algorithmes d'unification.

4.4. L'algorithme de simplification

L'application répétée des 3 phases de décomposition fusion et mutation va conduire soit à un échec de l'unification, soit à un unificande complètement décomposé.

Bien sur, la stratégie d'enchaînement des opérations de simplification appliquée dans l'algorithme ci-dessous n'est pas absolue.

Nous utiliserons les notations suivantes :

DEC(e,S) retourne le système S[e <- Dec(e)],

FUS(S) retourne la fusion du système S,

MUT(S) retourne la mutation du système S.

```
DEC-FUS-MUT = proc(U : unificande) RETOURNE(unificande)
                                   SIGNALE(échec)
    SI U est complètement décomposé
    ALORS RETOURNER(U)
    SINON soit S un système non complètement décomposé de U
        soit e une multiéquation non complètement décomposée de S
        SI |T(e)| > 1
        ALORS SI CP[T(e)] existe
              ALORS RETOURNER(DEC-FUS-MUT(FUS(U[S <- Dec(e,S)])))
              SINON SIGNALER(échec(clash de symboles))
              FSI
        SINON SI Mut(S) existe
              ALORS RETOURNER(DEC-FUS-MUT(FUS(U[S <- MUT(S)])))
              SINON SIGNALER(échec(mutation))
              FSI
        FSI
  FSI
FIN DEC-FUS-MUT
```

De ce qui précède on déduit le résultat suivant :

Théorème 6 : Soit A une théorie mutable et U un unificande, si DEC-FUS-MUT(U) termine en échec alors U n'a pas de A-solutions. Sinon, si la

procédure termine alors elle retourne un unificande complètement décomposé U' tel que U' $\Rightarrow_A$ U.

Remarque : Si la procédure de décomposition-fusion-mutation ne termine pas aucune conclusion n'est possible. Le lecteur pourra se reporter au chapitre sur l'unification modulo la commutativité droite où nous donnons un exemple de système n'ayant pas de solution et pour lequel la procédure ne termine pas. De même pour des systèmes ayant des solutions la procédure peut ne pas terminer. En fait nous touchons là l'une des principales difficulté de la conception d'un algorithme d'unification : sa preuve de terminaison. Le dernier chapitre de cette thèse sera consacré à une approche modulaire de ce problème.

En supposant que les problèmes de la mutation et de la terminaison de l'algorithme de simplification soient résolus pour une théorie donnée A, les résultats précédents permettent de se ramener au problème de la résolution d'unificandes complètement décomposés dans cette théorie. C'est à la résolution de ce problème que nous consacrons la prochaine section.

5. Résolution d'un système complètement décomposé de multiéquations

L'objectif de cette section est de donner des outils permettant de résoudre des systèmes complètement décomposés. Cela permettra de résoudre des disjonctions de systèmes complètement décomposés puisque l'ensemble des A-solutions d'une telle disjonction est la réunion des ensembles des A-solutions de chacun des systèmes la composant. Aussi ne parlerons nous plus dans cette section que de systèmes et de multiéquations.

5.1. Résolution des multiéquations complètement décomposées

De la même manière que dans ce qui précède, notre approche sera basée sur la détermination de l'ensemble des A-solutions (ou d'un ensemble complet de A-solutions) d'un système à partir des ensembles de A-solutions ou des ECU des multiéquations qui le composent. Afin de résoudre un système complètement décomposé, nous cherchons donc d'abord à résoudre indépendamment chacune des multiéquations qui le compose.

Le système étant complètement décomposé, les multiéquations qui le composent sont du type

(1) e = V(e),

(2) e = (V(e) == t) avec $V(e) \cap Var(t) = \emptyset$,

(3) e = (V(e) == t) avec $V(e) \cap Var(t) \neq \emptyset$.

Les deux premiers cas sont faciles de résoudre. Le dernier dépend de la théorie et est indécidable en général, aussi nous restreindrons nous à certaines classes de théories.

Lemme 15 : Soit e =(V(e) == t) une multiéquation telle que $V(e) \cap Var(t) = \emptyset$ et W un ensemble de variables contenant Var(e). Pour de telles multiéquations un ensemble complet de A-solutions en dehors de W, réduit à un élément et par conséquent minimal, est donné par

$$\Sigma = \{\sigma\} \text{ avec } \sigma = \prod_{x \in V(e)} (x \leftarrow Renom^W(t))$$

Preuve: En effet σ est de façon claire A-solution de e et pour toute autre A-solution α de e on a

$$\forall x \in V(e) : \alpha\sigma(x) = \alpha(t) =_A \alpha(x),$$

$\forall x \in V(t) : a\sigma(x) = a(x)$, et par conséquent $\sigma \leq_A a\ [Var(e)]$. □

De la même façon on peut prouver :

Lemme 16 : Soit e =(V(e)) une multiéquation réduite à des variables et W un ensemble de variables contenant Var(e). Pour de telles multiéquations un ensemble complet de A-solutions en dehors de W, réduit à un élément et par conséquent minimal est donné par

$$\Sigma = \{\sigma\} \text{ avec } \sigma = \prod_{x \in V(e)} (x \leftarrow z)$$

où z est une variable prise en dehors de W.

Définition 57 : Les théories A pour lesquelles il existe une méthode de résolution complète c'est à dire calculant un ensemble complet de A-solutions pour toute équation du type (3) sont dites **complètes** et sont encore appelées des C-théories. A est dite **fortement complète** s'il existe pour toute équation x = t un ensemble complet de A-solution dont chaque élément a pour domaine {x}. ○

Exemple 27 : Les théories vide, AC, commutative, minus sont des théories fortement complètes.

Remarque : Soit A = {a+b = a}, l'équation x+y == x a pour ensemble complet minimal de A-solution {((x ← a), (y ← b))}. Cette théorie est donc complète mais n'est pas fortement complète.
De même les théories non régulières ne sont pas fortement complètes. Si on prend par exemple A = {x * 0 = 0} alors l'équation x == x*y a pour seule A-solution la substitution {(x ← 0), (y ← 0)}.

Lemme 17 : Soit A une théorie fortement complète et e = (V(e) == t) une multiéquation telle que V(e) ∩ Var(t) contienne la variable x. Soit Σ un ensemble complet de A-solutions de l'équation x == t. Alors,

$$\Delta = \{ \prod_{z \in V(e)} (z \leftarrow \sigma(x)) \mid \sigma \in \Sigma \}$$

est un ensemble complet de A-solutions de e.

Preuve: Soit ξ la substitution de domaine V(e) telle que pour tout z de V(e), ξ(z) = x. Soit é = (V(e) == ξ(t)), nous avons montré que e $\Leftrightarrow_A$ é. Or il est clair, compte tenu de l'hypothèse de forte complétude, que Δ est un ECU de é et par conséquent puisque Var(e) = Var(é), c'est également un ECU de e. □

5.2. La détection des cycles

L'objectif est ici de déterminer, par un critère si possible simple, si un système de multiéquations a des A-solutions finies. Une étape ultérieure construira ces A-solutions éventuelles.

5.2.1. Un ordre sur les multiéquations

D'une manière qui est classique dans la théorie vide, la détection des cycles dans une substitution solution d'un système se fait en utilisant la relation suivante sur les multiéquations du système.

Définition 58 : Soient e et é deux multiéquations. e < é ssi il existe x dans V(e) et un terme t dans Term(é) tels que x appartienne à Var(t). ○

La fermeture transitive <+ de cette relation doit être un ordre strict sur l'ensemble des multiéquations du système pour que celui ci ait des solutions finies. Pour l'algorithme de G.Huet par exemple, cela est vérifié après la construction de la solution potentielle. Martelli et Montanari

ont montré comment construire cet ordre au cours des phases de décomposition et fusion (les seules nécessaires dans le cas de la théorie vide), en associant à chaque multiéquation un compteur. Cela permet de détecter plus rapidement des échecs dûs à des cycles, mais l'initialisation des compteurs est lourde et une vérification a postériori est souvent préférable.

Lemme 18 : ([Martelli1982]) Dans la théorie vide, si un système de multiéquations S a des solutions alors <+ est un ordre strict sur S.

Nous avons prouvé un résultat similaire dans la théorie équationnelle minus [Kirchner1982], mais ce n'est pas vrai en général comme le montre l'exemple suivant.

Exemple 28 : Dans la théorie des arbres binaires signés (abs) le système

$$S = \{\ e_1 = (x == f(z,a)),\ e_2 = (z == f(x,-a))\ \}$$

a en particulier pour solution $(x \leftarrow f(a,a))(z \leftarrow a)$ et pourtant on a $e_1 < e_2 < e_1$.

Même phénomène avec

$$S' = \{\ e_1 = (x == f(z,a)),\ e_2 = (z == f(-z,x))\ \}$$

qui a aussi pour solution $(x \leftarrow f(a,a))(z \leftarrow a)$ et pour lequel on a encore $e_1 < e_2 < e_1$ mais aussi $e_2 < e_2$.

La méthode usuelle utilisée dans le cas de la théorie vide ne peut donc pas s'appliquer en général, mais nous allons voir que le lemme précédent peut être étendu à une classe importante de théories contenant en particulier les théories permutatives.

Définition 59 : On appelle théorie **stricte** ou S-théorie toute théorie A telle que pour tout système S, si ce système a des A-solutions alors <+ est un ordre strict sur S. ○

Une théorie stricte est régulière. En effet si elle n'est pas régulière certain axiomes introduisent de nouvelles variables et par conséquent des équations de la forme $x = t$ avec $x \in Var(t)$ peuvent avoir des solutions donc la théorie n'est pas stricte.

Exemple 29 : Si A = {0 = x*0} l'équation e = (z == y*z) a pour A-solution {(z ← 0)} et e < e.

Une théorie permutative est stricte. En effet si une théorie n'est pas stricte, il existe des équations e = (x == t) telles que $x \in Var(t)$ et ayant des A-solutions. Mais dans ce cas les classes ne sont pas finies, donc la théorie n'est pas permutative.

Proposition 3 : Soit A une théorie telle qu'il existe un pré-ordre [irréflexif qui contienne la relation de sous terme strict, c'est à dire tel que pour tout terme t, t' [t si t' est un sous terme strict de t, et qui soit compatible avec la congruence définie par A, c'est à dire tel que pour tous termes t, t' et t" tels que $t =_A t'$ et t' [t" alors t [t".
Alors A est une théorie stricte.

Preuve: Montrons par l'absurde que pour tout élément e de S e <+ e est faux. Si il existe une multiéquation e de S telle que e <+ e, il existe une suite de multiéquations $(e_i)_{i=1...n}$ de S telle que

$$e = e_1 < e_2 < \ldots < e_n = e$$

Donc par définition de <, il existe des suites $(x_i)_{i=1...n-1}$ et $(t_i)_{i=2...n}$ avec x_i élément de $V(e_i)$ et t_i élément de $Term(e_i)$ telles que pour tout i dans [1..n-1], x_i soit une variable de t_{i+1}. Donc σ étant une A-solution de S on a

$$\forall\, i \in [1..n-1],\ \sigma(x_i) =_A \sigma(t_i) \text{ et } \sigma(x_i) \text{ sous terme de } \sigma(t_{i+1})$$

Par conséquent pour la relation [donnée par l'hypothèse de l'énoncé nous avons,

$$\sigma(t_i) \ [\ \sigma(t_{i+1}) \text{ pour } i = 1,\ldots,n-1.$$

d'où

$$\sigma(t_1) \ [\ \sigma(t_2) \ [\ \ldots \ [\ \sigma(t_{n-1}) \ [\ \sigma(t_n)$$

avec $t_1 = t_n$ puisque $e_1 = e_n$. Or $\sigma(t_1) \ [\ \sigma(t_1)$ est impossible par définition de [. Donc <+ est un ordre strict. □

Ce résultat est applicable à une vaste classe de théories. En particulier aux théories qui conservent la taille comme la commutativité ou l'associativité-commutativité. En effet, il suffit alors de prendre t [t' $\Leftrightarrow |t| < |t'|$.

Remarque : On peut étendre la notion de théorie stricte en utilisant un ordre < sur des classes de multiéquations comme cela est fait par exemple pour la théorie minus dans [Kirchner1982] en prennant comme relation d'équivalence : e S é <=> e = miroir(é).

5.2.2. Application au remplacement compatible

Le but de ce paragraphe est de montrer comment définir et justifier les remplacements des variables par les valeurs qui leurs sont déjà assignées dans un système de multiéquations.

Nous avons maintenant les outils nécessaires à une telle étude. En effet, le remplacement des variables apparaissant dans certaines parties d'un système par les termes qui leur sont affectés par une multiéquation de ce système est une opération dont il est simple de prouver qu'elle conserve la A-équivalence. Mais le remplacement des variables d'une partie d'un système par les valeurs qui leurs sont imposées par une autre partie de ce système nécessite de définir la substitution explicitant le remplacement de façon cohérente (en particulier pour qu'il n'y ait pas de cycle). Cela va être possible pour les théories strictes. De plus, nous allons imposer au remplacement d'être compatible avec certaines conditions, c'est pourquoi nous l'appelons compatible.

Définition 60 : Soit $S = S_1 \cup S_2 \cup S_3$ une partition d'un système fusionné, telle que $S_1 = \{e_1, e_2, \ldots, e_n\}$ est un système complètement décomposé avec $i < j \Rightarrow e_i \not< e_j$ (i.e. ordonné par ordre décroissant). Soit C un prédicat sur les multiéquations. On appelle remplacement **C-compatible** (ou compatible avec C, ou compatible si C est clair dans le contexte) dans S_2 par S_1 le système $S_1^C[[S_2]]$ récursivement défini par

* si n = 0, alors S_2.
* si n > 0, on désigne par t le terme non variable de e_1 si il existe, sinon on prend pour t l'une quelconque des variables de $V(e_1)$. Soit σ la substitution $\prod_{x \in V(e_1)-t} (x \leftarrow t)$.

$$S_1^C[[S_2]] = (S_1 - \{e_1\})^C[[\ \{\sigma(me) \mid me \in S_2 \text{ et } C(\sigma(me)) = vrai\}$$

$$\cup \{me \mid me \in S_2 \text{ et } C(\sigma(me) = faux\}\ ]].$$

○

Exemple 30 : Nous dirons qu'un terme est homogène si tous ses symboles de fonctions d'arité au moins un sont identiques. Soit C la condition imposant à tous les termes non variables d'une multiéquation d'être homogènes. Considérons les systèmes

$S_1 = \{ x == f(u,v),\ u == f(a,b),\ v == g(a,z) \}$

$S_2 = \{ f(x,z) == g(v,u),\ g(x,u) == g(u,v) \}$

on a alors,

$S_1^C[[S_2]] = \{f(f(f(a,b),v),v) == g(g(a,z),u),\ g(x,u) == g(u,g(a,z))\}$.

En effet la valeur de x peut être substituée dans f(x,z) car le terme obtenu est homogène, par contre on ne peut remplacer u par f(a,b) dans g(x,u) sans perdre l'homogenéité du terme.

Lemme 19 : Avec les notations précédentes et si A est une théorie stricte, $S \Leftrightarrow_A S_1 \cup S_1^C[[S_2]] \cup S_3$.

Preuve: Montrons tout d'abord par récurrence sur n avec $S_1 = \{e_1, e_2, \ldots, e_n\}$ où $i < j \Rightarrow e_i \not< e_j$, que toute A-solution de S est A-solution de $S' = S_1 \cup S_1^C[[S_2]] \cup S_3$.

Si n = 0 c'est clair.

Sinon pour n > 0, soit σ la substitution $\prod_{x \in e_1 - t} (x \leftarrow t)$. Par définition, $S_1^C[[S_2]] = (S_1 - \{e_1\})^C[[\ \{\sigma(me) \mid me \in S_2 \text{ et } C(\sigma(me)) = vrai\} \cup \{me \mid me \in S_2 \text{ et } C(\sigma(me) = faux\}\ ]]$.

Si a est une A-solution de S, a est A-solution de e_1 donc pour toute variable x de $D(\sigma)$ on a

$$(*) \quad a(x) =_A a(t) = a\sigma(x)$$

et par conséquent a est A-solution de toute multiéquation de $S_1^C[[S_2]]$ donc de S'.

Réciproquement, si a étant A-solution de S' est également A-solution de S_1 donc de e_1 et par conséquent (*) est encore valable et permet de conclure. □

Remarque : Si dans la proposition précédente on prend n = 1, il est inutile de supposer la théorie stricte.

5.3. L'algorithme de résolution d'un système complètement décomposé

Pour une sous classe des théories complètes et strictes nous allons donner un algorithme qui retourne à partir d'un système complètement décomposé l'ensemble des A-solutions du système.

Soit $Q = \{e_1, \ldots, e_n\}$ une suite de multiéquations complètement décomposées ordonnée par < de telle sorte que $i < j$ implique $e_i \not< e_j$ (i.e. triée topologiquement par ordre décroissant). Si e_i contient un terme non variable nous le noterons t_i. W est un ensemble de variables qui contient Var(Q) et en dehors duquel seront prises les nouvelles variables.

```
SUBST = proc(Q : suite ordonnée de multiéquations) RETOURNE(ECU)
  SU := {Id}
  POUR j DE 1 A n FAIRE
    CAS (1) Term(e_j) = ø ALORS
          (soit y dans V(e_j) et z une nouvelle variable)
          SU := ∪_{β∈SU} [ ⊓_{x∈V(e_j)} (x←z)] . β
        (2) Term(e_j) = {t_j} ET Var(t_j) ∩ V(e_j) = ø ALORS
          SU := ∪_{β∈SU} [ ⊓_{x∈V(e_j)} (x←Renom^W(t_j))] . β
        (3) Term(e_j) = {t_j} et ∃ x ∈ Var(t_j) ∩ V(e_j) ALORS
          (Soit Σ un ensemble complet de A-solutions de x == t_j en
           dehors de W)
          SU := ∪_{σ∈Σ} [ ∪_{β∈SU} [ ⊓_{y∈V(e_j)} (y←σ(t_j))] . β
    FINCAS
  FINPOUR
  RETOURNER(SU)
FIN_SUBST
```

Théorème 7 : Soit A une théorie stricte et fortement complète. Pour tout système complètement décomposé S, si on peut ordonner les éléments de S en une suite de multiéquations $Q = \{e_1,\ldots,e_n\}$ telle que $i < j$ implique $e_j \not< e_i$ et si l'algorithme SUBST termine sur Q alors il retourne un ensemble complet de A-solutions de S. Si Q n'existe pas alors S n'a pas de A-solutions.

Preuve: Si Q n'existe pas puisque A est supposée stricte, le système S n'a pas de A-solutions.

Montrons maintenant que si SUBST termine avec SU comme résultat alors SU est un ensemble de A-solutions de S.

Si on suppose que $Q=(e_j)_{j=1\ldots n}$, tout élément σ de SU s'écrit σ_n .

$\ldots . \sigma_2 . \sigma_1$, ou chacun des σ_j est une A-solution de la multiéquation e_j. Calculons

$$\sigma_n\ldots\sigma_1(x_j) \qquad \sigma_n\ldots\sigma_1(t_j).$$

pour x_j dans $V(e_j)$ et t_j dans $Term(e_j)$ si ce dernier n'est pas vide. Notons que puisque la théorie est supposée fortement complète, le domaine de chacune des substitutions σ_i est $V(e_i)$ et puisque par hypothèse S est fusionné, les domaines des σ_i sont tous disjoints. D'autre part par définition de Q les variables de t_j peuvent seulement apparaitre dans $V(e_k)$ pour les multiéquations e_k telles que $j \leq k \leq n$. donc les expressions ci dessus sont égales à

$$\sigma_n..\sigma_j(x_j) \qquad \sigma_n..\sigma_j(t_j)$$

qui sont A-égales par définition de σ_j.

Montrons maintenant la completude de SU par récurrence sur la longueur n de la suite Q. Nous nous réfèrerons aux parties de l'algorithme SUBST par leur numero dans le CAS.

Si $Q = (e_1)$ alors dans les trois cas il est clair que SU est un ensemble complet de A-solution de Q donc de S.

Si $Q=(e_1, e_2, \ldots, e_n)$, soit α un A-unificateur de Q. α est donc une A-solution de e_1 et par définition de σ_1, il existe ρ tel que

$$\rho\sigma_1 =_A \alpha \qquad [Var(e_1)]$$

Soit ρ' définie par

(1) $\rho' = \rho\ [I(\sigma_1)]$

(2) $\rho' = \alpha \ [X\text{-}I(\sigma_1)]$

ρ' est un A-unificateur de $\{e_2,\ldots,e_n\}$ car comme $I(\sigma_1) \cap Var(S) = \emptyset$ on a par définition de ρ'

$$\rho' =_A \alpha \ [Var(e_2) \cup \ldots \cup Var(e_n)]$$

de plus on a également

$$\rho'\sigma_1 =_A \alpha \qquad [Var(e_1)]$$

On applique l'hypothèse de récurrence sur $Q' = \{e_2,\ldots,e_n\}$. Il existe donc μ telle que

$$\mu\sigma_n\ldots\sigma_2 = \rho' \ [Var(e_2) \cup \ldots \cup Var(e_n)]$$

Soit μ' définie par

(1) $\mu' = \mu \ [I(\sigma_n\ldots\sigma_2)]$

(2) $\mu' = \rho' \ [X\text{-}I(\sigma_n\ldots\sigma_2)]$

on a

$$\mu'\sigma_n\ldots\sigma_2 = \rho' \ [X - I(\sigma_n\ldots\sigma_2)]$$

Compte tenu que les A-unificateurs de chacune des multiéquations e_k est pris en dehors de W donc de Var(S), en regroupant les égalités précédentes on obtient

$$\mu'\sigma_n\ldots\sigma_2\sigma_1 =_A \alpha \ [Var(S)]$$

ce qui termine la preuve de complètude. □

Dans le cas de la théorie vide le théorème précédent fournit une preuve de l'algorithme de Martelli et Montanari ainsi que de l'unicité de l'unificateur principal quand il existe, telle qu'elle est donnée dans [Jouannaud1982] ou indépendamment dans [Raoult1982].

Remarque : Il est important de comparer cet algorithme aux algorithmes classiques d'unification "à la Robinson". Dans de tels algorithmes dès que l'unification d'une partie de l'équation détermine une solution celle-ci est répercutée dans les deux termes par instanciation (avec copie comme dans l'algorithme de Robinson ou sans copie comme dans les algorithmes de Corbin-Bidoit ou Fages). On n'a pas dans ce cas à prendre en compte une hypothèse comme la forte complétude. Par exemple pour A = {a = a+a}, l'équation f(y,x+y) == f(z,x) se décompose immédiatement et on obtient (y == z) < (x == x+y). Mais A n'étant pas fortement complète, on ne peut conclure par le théorème précédent, en effet la résolution de x == x+y imposerait (y ← a) et la résolution de y == z, (y ← z) par exemple, il faudrait donc une étape suplémentaire permettant d'unifier les contraintes y == a et y == z, ce qui peut présenter en général (contrairement au cas présent) des difficultés.

Par contre avec un algorithme "à la Robinson" comme celui proposé par Yelick [Yelick1985] et en reprennant le même exemple, on obtient la contrainte (y ← z) qui par instanciation dans x+y == x amène à la résolution de x+z == x dont la A-solution combinée à (y ← z) donne la A-solution de l'équation {(x ← a), (y ← a), (z ← a)}.

On peut donc considérer l'approche de décomposition-fusion-mutation suivie de la résolution des systèmes par SUBST comme une méthode "sans instanciations", par rapport aux algorithmes "à la Robinson", ceci ayant les

conséquences que nous venons de voir sur les hypothèses nécessaires.

6. L'étude standard de l'unification dans une théorie équationnelle

Après avoir montré comment obtenir des disjonctions de systèmes complètement décomposés, nous venons de voir comment résoudre les systèmes de multiéquations complètement décomposés obtenus, pour de "bonnes" théories c'est-à-dire pour les théories fortement complètes et strictes.

Pour ce type de théories nous obtenons donc un schéma unifié d'étude de l'unification. La décomposition et la fusion ne dépendent pas de la théorie tandis que les opérations qui dépendent de la théorie sont maintenant "factorisées" dans la mutation. Cela permet d'implémenter facilement des bibliothèques d'algorithmes d'unification comme celle qui est construite dans REVEUR3 [Kirchner1985].

En conclusion, pour décrire un nouvel algorithme standard d'unification pour une théorie A il faut :

- Déterminer l'opération de mutation d'un système et prouver sa correction,
- Faire la preuve de terminaison du processus de décomposition fusion mutation,
- Montrer que A est une théorie fortement complète et stricte.

Nous appliquerons cette approche dans les études de théories qui vont suivre.

CHAPITRE 3

LES THEORIES SYNTAXIQUES

La construction d'algorithmes d'unification pour une théorie équationnelle A est une opération artisanale et cela restera toujours vrai en général puisque l'unification équationnelle est indécidable comme l'a montré P.Szabo dans sa thèse [Szabo1982] pour A réduit aux axiomes d'associativité et distributivité, ou encore pour l'associativité commutativité distributivité. Plus récemment Arnborg et Tiden [Arnborg1985] ont montré que l'unification dans $Dg(*,+) \cup A(+) \cup Ng(*,1) \cup Nd(*,1)$ était également indécidable.

Néanmoins, nous allons voir dans ce chapitre comment le formalisme que nous avons développé va nous permettre, par les outils qu'il fournit, de construire automatiquement les algorithmes d'unification associés à certaines théories équationnelles. A terme, outre les cas où la méthode de construction automatique que nous allons développer permet de conclure, les outils que nous proposons, associés à la RE-surréduction que nous étudierons plus

loin, permettront la réalisation de systèmes assistant l'utilisateur dans la construction d'un algorithme d'unification.

Pour déterminer un algorithme d'unification la première partie difficile est de construire l'opération de mutation associée à la théorie. Or l'expérience montre que dans bon nombre de cas on peut déduire de façon simple l'opération de mutation de la forme des axiomes de la théorie.

Prenons par exemple la théorie réduite à la commutativité : x+y = y+x. Il est clair que d'une part les équations du type

$$e = (u*v==u'+v')$$

avec $* \neq +$ n'ont pas de A-solutions et que d'autre part si $*=+$, alors e est A-équivalent à la disjonction des systèmes

$$S_1 = \{u = u', \; v = v'\}$$
$$S_2 = \{u = v', \; v = u'\}.$$

S_1 est obtenu en considérant qu'il n'y a pas d'application de l'axiome en tête, alors que S_2 est le système qui résulte d'une application de l'axiome en tête. L'opération de mutation étant ainsi déterminée et le processus de décomposition-fusion-mutation terminant puisque la taille des équations diminue strictement par mutation et décomposition, nous avons un algorithme complet et fini d'unification modulo la commutativité du symbole +. Le même raisonnement permettra également de conclure pour un ensemble fini de symboles commutatif.

Sur cet exemple simple (que nous justifierons complètement dans la suite) nous voyons que pour certaines théories, la mutation peut se déduire facilement, en fait syntaxiquement, de la forme des axiomes. Il n'en est

pas de même pour toutes les théories, comme on peut le constater avec l'associativité-commutativité!

1. Les théories syntaxiques

Nous introduisons tout d'abord les notations propres à cette partie. Afin d'éclairer le formalisme, nous développerons parallèlement l'exemple de la théorie que nous noterons C, constituée d'un nombre fini de symboles commutatif.

1.1. Définitions et exemples

Dans toute cette partie nous supposerons que les ensembles d'axiomes considérés sont potents. Cela exclut dont des ensembles d'axiomes contenant par exemple l'idempotence $x + x = x$ ou l'involution $-(-x) = x$.

Définition 61 : Soit T(A) l'ensemble des têtes d'axiomes de A défini par :

$$T(A) = \{\{f,f'\} \mid \exists \{g,d\} \in A \text{ tel que } top(g)=f \text{ et } top(d) = f'\}.$$

Soit A(f,f') l'ensemble des axiomes de la théorie dont les symboles de têtes sont f et f', c'est à dire :

$$A(f,f') = \{\{g = d\} \mid top(g) = f \text{ et } top(d) = f'\}.$$

Noter que si f = f' chaque axiome apparait 2 fois dans A(f,f').

Nous dirons qu'un ensemble d'axiomes A est **résolvant** ssi pour tous termes $t = f(t_1,\ldots,t_n)$ et $t' = f'(t_1',\ldots,t_p')$ tels que $\{f,f'\} \in T(A)$:

$t =_A t'$

<=>

$* \; f = f'$ et $\forall j \in [1..n]$, $t_j =_A t_j'$

ou

* il existe un élément g=d de A(f,f') tel que Var(g) ∪ Var(d) soit disjoint de Var(t) ∪ Var(t') (ce qui est toujours possible par un renommage des variables) et une substitution σ tels que :

- $\forall j \in [1..n]$, $t_j =_A \sigma(g_{|j})$

et

- $\forall k \in [1..p]$, $t_k' =_A \sigma(d_{|k})$.

On appelle **théorie syntaxique** toute théorie équationnelle pour laquelle il existe un ensemble fini et résolvant d'axiomes qui l'engendre. ○

L'intérêt des théories syntaxiques est de permettre la généralisation, par la forme des axiomes, d'une équation. Nous définissons cette généralisation qui peut être considérée comme une forme de décomposition.

Définition 62 : Soit A un ensemble d'axiomes potent. Soient $e = (f(t_1,\ldots t_n) == f'(t_1',\ldots,t_p'))$ une équation et W un ensemble de variables dites protégées, contenant Var(e).

Si f=f' on note $Dec_1(e)$ le système $\{t_j == t_j' \text{ pour } j \text{ dans } [1..n]\}$. (Cela prolonge la définition de la décomposition à un niveau à des équations dont le symbole de tête n'est pas décomposable.)

On désigne par **Gen_ax(e,A,W)** la disjonction de systèmes contenant exactement tous les systèmes $\{t_j == g_{|j} \text{ pour } j \text{ dans } [1..n],\ t_k' == d_{|k} \text{ pour } k \text{ dans } [1..p]\}$ pour g = d dans A(f,f') et où l'on suppose que toutes les variables de tous les axiomes g=d ont été renommées de telle sorte que $(Var(g) \cup Var(d)) \cap W = \emptyset$.

La **généralisation de e par les axiomes de A** est définie comme étant l'unificande

Gen(e,A,W) = Gen_ax(e,A,W) si f ≠ f'

= Gen_ax(e,A,W) ∪ $[Dec_1(e)]$ si f = f'. ○

Exemple 31 : En reprenant l'exemple de l'introduction avec l'axiome x + y = y + x et l'équation u + v == u' + v' :

Gen(e,A,W) = [{u == u', v == v'} (= $Dec_1(e)$)

{x == v, y == u, x == u', y == v'} (y + x = x + y)

{x == u, y == v, x == v', y == u'} (x + y = y + x)

Si on considère la théorie A contenant le seul axiome -(x+y) = (-y) + (-x) et l'équation e=(-f(a,b) == f(a+z) + b) alors

Gen(e,A,W) = {f(a,b) == x+y, f(a+z) == -y, b == -x}

La généralisation par les axiomes est une transformation conservant la complétude des unificandes pour les théories syntaxiques. C'est ce qu'exprime le résultat suivant :

Théorème 8 : Si A est un ensemble résolvant d'axiomes non potents, alors pour toute équation indécomposable $e = (f(t_1,\ldots,t_n)==f'(t_1',\ldots t_p'))$, $Gen(e,A,W) \Rightarrow_A^W e$.

Preuve: Posons U = Gen(e,A,W), il faut montrer que pour tout ensemble complet de A-solutions Σ de U, $\Sigma_{|Var(e)}$ est un ensemble complet de A-solutions de e.

1) Montrons tout d'abord que $\Sigma_{|Var(e)}$ est un ensemble de A-solutions de e.

Soit σ une A-solution de U. Il existe donc un système S de U dont σ est A-solution.

Si ce système est $Dec_1(e)$, il est clair que σ est bien A-solution de e.

Sinon il existe g = d dans A(f,f') tel que σ soit A-solution du système $S = \{t_j == g_{|j}$ pour $j \in [1..n]$, $t_k' == d_{|k}$ pour $k \in [1..p]\}$. On a donc

$\sigma(f(t_1,\ldots,t_n)) = f(\sigma(t_1),\ldots,\sigma(t_n))$

$=_A f(\sigma(g_{|1}),\ldots,\sigma(g_{|n})) = \sigma(g)$

$=_A \sigma(d) = f(\sigma(d_{|1}),\ldots,\sigma(d_{|p}))$

$=_A \sigma(f(t_1',\ldots,t_p'))$.

Ce qui prouve que σ est bien A-solution de e.

2) Montrons maintenant la complétude de $\Sigma_{|Var(e)}$:

$\forall \alpha \in ES(e,A)\ \exists \sigma \in \Sigma$ tel que $\sigma \leq_A \alpha\ [Var(e)]$.

L'équation e sera notée t == t'. $\alpha \in ES(e,A) <=> \alpha(t) =_A \alpha(t')$, de deux choses l'une,

a) f = f' et pour tout j de [1..n] on a $\alpha(t_j) =_A \alpha(t_j')$, ce qui assure que α est une A-solution de U,

b) il existe un élément g=d de A(f,f') et une substitution μ tels que $D(\mu) \cap Var(e) = \emptyset$ et $\forall j \in [1..n]$, $\alpha(t_j) =_A \mu(g_{|j})$ et $\forall k \in [1..p]$, $\alpha(t_k') =_A \mu(d_{|k})$.

$\mu+\alpha$ a un sens puisque μ et α ont des domaines disjoints et c'est un A-unificateur de U puisque c'est une A-solution de l'un de ses systèmes.

Posons $\lambda = \alpha$ dans le cas a) et $\lambda = \alpha+\mu$ dans le cas b).

On a donc $\lambda_{|Var(e)} = \alpha$ dans tous les cas. Comme Σ est un ECU de U et que λ est A-solution de U, il existe une substitution σ de Σ telle que

$$\rho.\sigma =_E \lambda\ [Var(U)]$$

Ce qui implique puisque $Var(e) \subseteq Var(U)$

$$\rho.\sigma =_E \lambda\ [Var(e)]$$

D'où la conclusion. □

Nous obtenons donc en généralisant e par Gen(e,A,W) une opération de mutation pour les théories syntaxiques. Le problème qui se pose alors est de prouver qu'une théorie est syntaxique et ce si possible de façon automatique. C'est ce problème que nous abordons maintenant.

2. Conditions suffisantes pour qu'une théorie soit syntaxique

Nous allons donner des conditions suffisantes permettant de s'assurer qu'une théorie est syntaxique. La première est simplement une condition sur les occurrences d'application des axiomes dans la preuve d'égalité de deux termes. Nous en déduirons une condition sur les paires critiques d'axiomes et un algorithme de complétion d'un ensemble d'axiomes A en un ensemble d'axiomes A' qui soit résolvant.

Proposition 4 : Soit A un ensemble d'axiomes tels que pour tous termes $t = f(t_1,\ldots,t_n)$ et $t' = f(t_1',\ldots,t_p')$ A-égaux il existe une preuve

$$t = v_0\ |-|[m_0]\ v_1\ |-|\ \ldots\ |-|[m_{n-1}]\ v_n = t'$$

avec au plus l'un des m_j égal à ε, pour j dans [0..n-1]. Alors A est résolvant.

Preuve: Soient t et t' deux termes quelconques tels que $t =_A t'$. Par hypothèse, il existe une preuve

$$t = v_0\ |-|[m_0]\ v_1\ |-|\ \ldots\ |-|[m_{n-1}]\ v_n = t'$$

avec au plus l'un des m_j égal à ε.

Si aucun des m_j n'est ε, la preuve ne comportant aucune application d'un axiome en tête, on a nécessairement f = f' et pour tout j de [1..n], $t_j =_A t_j'$.

S'il existe k dans [0,n-1] tel que $m_k = \varepsilon$, et si l'axiome appliqué est g=d, on a

$$t = v_0 \ |-|[m_0]\ v_1 \ \ldots\ |-|\ v_k\ |-|[\varepsilon,g=d]\ v_{k+1}\ |-|\ \ldots\ [m_{n-1}]\ v_n = t'$$

où v_k et v_{k+1} s'écrivent nécessairement

$v_k = f(v_1^k,\ldots,v_n^k)$, $v_{k+1} = f'(v_1^{k+1},\ldots,v_p^{k+1})$ et par conséquent,

pour tout j de [1..n] : $t_j =_A v_j^k = g_{|j}$,

pour tout l de [1..p] : $t'l =_A v_l^k = d_{|l}$. □

Nous verrons sur un exemple (la commutativité droite) que la réciproque est fausse : il existe des ensembles d'axiomes résolvants pour lesquels certaines preuves contiennent forcément plusieurs applications d'axiomes en tête.

Dans ce qui suit nous allons donner des conditions suffisantes locales, permettant de s'assurer qu'un ensemble d'axiomes est résolvant.

Nous noterons **$t\ |-*-|[\neq\varepsilon]\ t'$** si dans cette chaine d'application d'axiomes aucune application n'a lieu en ε.

Définition 63 : On dit qu'un ensemble A d'axiomes est **ε-confluent** ssi les deux conditions suivantes sont satisfaites pour tous termes t_0, t_1, t_2.

1) Si

$$t_1\ |-|[\varepsilon]\ t_0\ |-|[\varepsilon]\ t_2$$

alors il existe des termes w et w' tels que

$$t_1\ |-*-|[\neq\varepsilon]\ w\ |-\delta-|[\varepsilon]\ w'\ |-*-\ [\neq\varepsilon]\ t_2$$

avec δ = 0 ou 1.

2) Si

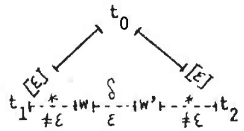
$$t_1\ |-|[\varepsilon]\ t_0\ |-+-|[\neq\varepsilon]\ t_2,$$

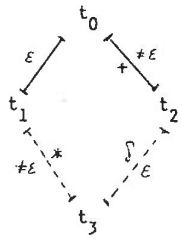
alors il existe t_3 tel que

$$t_1\ |-*-|[\neq\varepsilon]\ t_3\ |-\delta-|[\varepsilon]\ t_2$$

avec δ = 0 ou 1.

Ce qu'on peut schématiser, en représentant l'hypothèse par des traits pleins et la conclusion en pointillés, par :





○

La ε-confluence peut s'exprimer également de la manière suivante :

Lemme 20 : Un ensemble d'axiomes A est ε-confluent ssi pour tous termes t, t_1, t_2, t_3 et pour toute occurrence m tels que

$$t_1\ |-|[\varepsilon,g=d]\ t\ |-|[m,g'=d']\ t_2\ |-n-|[\neq\varepsilon]\ t_3$$

avec $n \in \mathbb{N}$, il existe un terme w tel que

$$t_1\ |-*-|[\neq\varepsilon]\ w\ |-\delta-|[\varepsilon]\ t_3 \text{ avec } \delta = 0 \text{ ou } 1.$$

Preuve: Conséquence immédiate de la définition. □

Lemme 21 : Si A est un ensemble d'axiomes ε-confluent alors A est résolvant.

Preuve: Pour tous termes t et t' tels que $t =_A t'$, nous allons montrer qu'il existe une preuve de $t =_A t'$ qui ne fait intervenir qu'une seule application d'axiome à l'occurrence ε, cela nous permettra de conclure en appliquant le résultat précédent.

Raisonnons par récurrence sur le nombre n d'applications d'axiomes à l'occurrence ε d'une des preuves de $t =_A t'$.

Si n = 0 ou 1, c'est clair.

Sinon, s'il existe deux applications d'axiomes consécutives à l'occurrence ε, on applique 1) et on conclut par récurrence, dans le cas contraire on applique 2) de façon à se ramener au cas précédent. C'est ce que nous détaillons maintenant.

$$t = v_0 \ |-|[m_0]\ v_1\ |-|\ \ldots\ |-|[m_{k-1}]\ v_k = t'$$

Soient m_j et m_l les deux premières occurrences de cette preuve égale à ε. On a donc

$$v_j\ |-|[m_j]\ v_{j+1}\ |-q-|\ v_l\ |-|[m_l]\ v_{l+1}$$

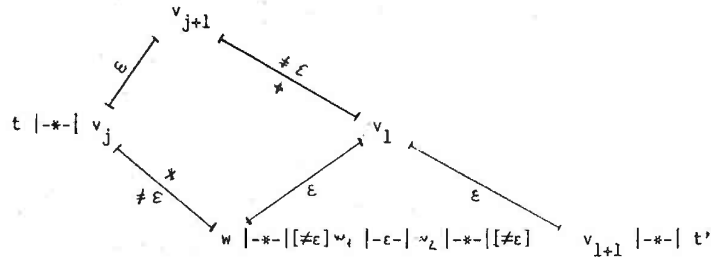
Si q = 0, alors la condition 1 de ε-confluence et l'hypothèse de récurrence permettent de conclure puisqu'on a la preuve suivante :

$$t\ |-*-|\ v_j\ |-*-|\ v_{l+1}\ |-*-|\ t'$$

qui ne comporte plus que n-1 applications d'un axiome à

l'occurrence ε.

Si q ≠ 0 les conditions 1 et 2 permettent de conclure en suivant le schéma suivant :



Formellement,

$$v_j\ |-|[\varepsilon]\ v_{j+1}\ |-+-|[\neq\varepsilon]\ v_l$$

implique par la condition 2 de ε-confluence l'existence d'un terme w tel que

$$v_j\ |-*-|[\neq\varepsilon]\ w \text{ et } v_l\ |-|[\varepsilon]\ w.$$

En sachant que

$$w\ |-|[\varepsilon]\ v_l\ |-|[\varepsilon]\ v_{l+1}$$

la condition 1 de ε-confluence permet de conclure à l'existence d'une preuve

$$w\ |-*-|[\neq\varepsilon]\ w1\ |-|[\varepsilon]\ w2\ |-*-|[\neq\varepsilon]\ v_{l+1}.$$

La preuve de $t =_A t'$

$$t\ |-*-|[\neq\varepsilon]\ v_j\ |-*-|[\neq\varepsilon]\ w1\ |-|[\varepsilon]\ w2\ |-*-|[\neq\varepsilon]\ v_{l+1}\ |-*-|\ t'$$

ne comporte plus que n-1 applications d'axiome à l'occurrence ε et

l'hypothèse de récurrence permet de conclure. □

Le lecteur familier avec les preuves localisées sur des axiomes ou des règles de réécriture [Knuth1970,Huet1980,Jouannaud1984] pensera bien sûr à localiser sur des paires critiques adéquates la vérification des conditions 1 et 2 du résultat precédent. C'est ce que nous allons faire maintenant.

Définition 64 : Soit A un ensemble d'axiomes. Rappellons que (p,q) est une **paire critique** ssi il existe deux axiomes g=d et g'=d' de A, une occurrence oc de O(g) et un ø-unificateur σ de $g_{|oc}$ et g' tels que p = σ(d) et q = $\sigma(g_{[oc \leftarrow d']})$.

On dira qu'une paire critique (p,q) est **ε-confluente** ssi

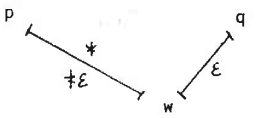
soit $p \mid\!-\!*\!-\!\mid_{[\neq\varepsilon]} q$,

ou bien,

il existe un terme w tel que $p \mid\!-\!*\!-\!\mid_{[\neq\varepsilon]} w \mid\!-\!\mid_{[\varepsilon]} q$.

Ce qu'on peut schématiser par les diagrammes

$p \mid\!-\!*\!-\!\mid_{[\neq\varepsilon]} q$ ou



○

Proposition 5 : Un ensemble d'axiomes A est ε-confluent si et seulement si toute paire critique (p,q) est ε-confluente.

Preuve: Il est facile de vérifier que la ε-confluence implique la ε-confluence des paires critiques.

Réciproquement, soit t, t_1, t_2, t_3 des termes tels que

$$t_1 \mid\!-\!\mid_{[\varepsilon,g=d]} t \mid\!-\!\mid_{[m,g'=d']} t_2 \mid\!-n-\!\mid_{[\neq\varepsilon]} t_3.$$

Raisonnons par récurrence sur n.

Si n = -1 c'est trivialement vrai.

Pour n ≥ 0.

Si $m \notin O(g)$ (donc $m \neq \varepsilon$) alors il n'y a pas de paire critique sous jacente et par conséquent il existe t_4 tel que

$$t_1 \mid\!-\!\mid_{[m,g'=d']} t_4 \mid\!-\!\mid_{[\varepsilon,g=d]} t_2 \mid\!-n-\!\mid_{[\neq\varepsilon]} t_3$$

ce qui permet de conclure en appliquant l'hypothèse de récurrence sur la preuve de $t_4 =_A t_3$.

Si au contraire $m \in O(g)$ alors soient α et β les substitutions de domaines respectivement inclus dans Var(g) et Var(g') où l'on suppose que $Var(g) \cap Var(g') = \emptyset$ (il est toujours possible de renommer les variables des axiomes) et telles que $t = \alpha(g)$ et $t_{|m} = \beta(g')$. On a donc

$$(\alpha+\beta)(g_{|m}) = \alpha(g)_{|m} = t_{|m} = \beta(g') = \alpha+\beta(g').$$

Donc α+β est un ø-unificateur de $g_{|m}$ et de g'. Soit σ l'unificateur principal de $g_{|m}$ et g', il existe donc μ tel que μs = α+β. Soit (p,q) la paire critique telle que p = σ(d) et $q = \sigma(g_{[m \leftarrow d']})$. On a donc

$$t_1 = \mu(p) \mid\!-\!\mid_{[\varepsilon,g=d]} \sigma(g) \mid\!-\!\mid_{[m,g'=d']} \mu(q) = t_2$$

Or par hypothèse sur la m-paire critique (p,q),

soit $p \mid\!-\!*\!-\!\mid_{[\neq\varepsilon]} q$ auquel cas

$$t_1 = \mu(p) \mid\!-\!*\!-\!\mid_{[\neq\varepsilon]} \mu(q) = t_2$$

ce qui permet de conclure indépendamment de n,

soit il existe un terme w tel que p $|-*-|[\neq\varepsilon]$ w $|-|[\varepsilon]$ q d'où,

$$t_1 = \mu(p) \ |-*-|[\neq\varepsilon] \ \mu(w) \ |-|[\varepsilon] \ \mu(q) = t_2 \ |-n-|[\neq\varepsilon] \ t_3$$

Ce qui permet de conclure en appliquant l'hypothèse de récurrence sur la preuve de $\mu(w) =_A t_3$. □

On obtient donc une condition suffisante permettant de tester si un ensemble d'axiomes est résolvant :

Corollaire 3 : Tout ensemble d'axiomes A tel que toute paire critique soit ε-confluente est résolvant.

L'intérêt de ce résultat est de donner le moyen de calculer automatiquement une condition suffisante pour qu'un ensemble d'axiomes soit résolvant. Cela permet donc de déterminer automatiquement pour les ensembles d'axiomes dont toutes les paires critiques sont ε-confluentes l'opération de mutation. Pour ce type de théories nous savons par conséquent construire automatiquement un algorithme de A-unification dont il ne restera plus qu'à prouver la terminaison, ce qui d'ailleurs n'est pas toujours simple.

Avant de montrer sur des exemples l'efficacite de cette approche, terminons cette partie par un algorithme de complétion.

3. Un algorithme de complétion

Comme dans tout algorithme de complétion équationnelle [Knuth1970,Jouannaud1984], si la propriété requise sur les paires critiques n'est pas satisfaite, on complète l'ensemble des objets considérés, ici l'ensemble des axiomes, par le ou les objets permettant d'assurer, au moins

localement, la propriété.

Ici nous allons donc calculer les paires critiques d'axiomes et vérifier si elles sont ε-confluentes. Si oui, il n'y a pas de problème. Sinon, on ajoute l'axiome qui permet de rendre la paire critique ε-confluente.

D'où la procédure de complétion :

```
Complétion_unificationnelle(A : ensemble d'axiomes)
            RETOURNE(ensemble d'axiomes)
   B : ensemble d'axiomes := ø
   TANTQUE A est non vide FAIRE
      Choisir un axiome g=d dans A
      Calculer les paires critiques de g=d sur lui même et avec les
        éléments de B.
      Ajouter toute paire critique non ε-confluente pour A U B dans A.
      Ajouter g=d dans B
   REFAIRE
   RETOURNER(B)
FIN complétion_unificationnelle
```

Si cette procédure termine elle retourne un ensemble fini et résolvant d'axiomes. Sinon moyennant une hypothèse de choix équitable de l'axiome g=d dans A, l'ensemble infini d'axiomes engendré est résolvant.

L'implantation de cette procédure (re)pose le problème du calcul de la A-égalité à partir d'axiomes. Si on impose que cette A-égalité se fasse en une seule application d'axiomes il n'y a pas de difficultées mais dès que l'on s'autorise des chaines de E-égalités, ce que nous faisons ici lors du calcul de ε-confluence des paires critiques, il faut s'assurer que le processus ne boucle pas.

Une implantation de réécriture "non terminante" qui possède de nombreux points communs avec ce problème a été réalisée par R. Gobel dans le projet SEKI [Gobel1983]. Elle est basée sur une détection des boucles lors des réécritures. Une implantation de l'algorithme de complétion

unificationnelle pourra s'en inspirer. Notons que si les classes d'équivalence sont finies, la vérification se limite à un parcours de la classe.

4. Exemples et applications

Nous allons appliquer les résultats généraux obtenus, à la construction d'algorithmes d'unification pour des théories syntaxiques.

4.1. Théories constituées d'un nombre fini d'axiomes de commutativité

Nous considérons ici le cas d'un ensemble C d'axiomes de commutativité de symboles de l'algèbre.

$$C = \{x +_i y = y +_i x \mid i \in I\}$$

Ce problème a été étudié par J. Siekmann [Siekmann1979] pour I réduit à un singleton et avec une approche particulière à la théorie. Des améliorations de l'approche "naive", de décomposition suivant la forme des axiomes, sont proposées. Il ne nous semble pas qu'elles apportent un réel gain d'efficacité.

Nous appliquons donc les résultats de ce chapitre à C.

Tout d'abord, il est clair que l'ensemble des symboles décomposables est $F-\{+_i \mid i \in I\}$.

Etudions maintenant la mutation.

Proposition 6 : C est un ensemble résolvant.

Preuve: C'est évident car toute les paires critiques sont trivialement ε-confluentes. □

Toute équation $e = (u +_i v == u' +_i v')$ se généralise donc dans la disjonction de système :

$$U = [\ \{ u == u', v == v'\},\ \{ x == v, y == u, x == u', y == v' \}\ \{ x == u, y == v, x == v', y == u' \}\ ]$$

qui fusionne en

$$U' = [\ \{ u == u', v == v'\},\ \{ x == v == u', y == u == v' \}\ \{ x == u == v', y == v == u' \}\ ]$$

Mais comme les variables x et y de U' n'apparaissent pas dans e et qu'elles n'apparaissent pas dans les termes de U', la disjonction de systèmes

$$U'' = [\ \{ u == u', v == v'\},\ \{ v == u', u == v' \}\ \{ u == v', v == u' \}\ ] = [\ \{ u == u', v == v'\},\ \{ v == u', u == v' \}\ ]$$

déduite de U', généralise également e : U" =>C e. On pose donc pour toute équation indécomposable et non complètement décomposée : Mut(e) = U". Pour tout système S et toute équation e de S non complètement décomposée et indécomposable : Mut(S) = S[e ← Mut(e)].

Lemme 22 : L'algorithme de C-unification standard termine pour l'opération de mutation que nous venons de définir.

Preuve: La décomposition et la mutation font strictement décroitre la taille des termes alors que la fusion conserve cette taille, cela assure donc les terminaison du processus de décomposition fusion mutation pour les théories C. □

Cela justifie l'exemple donné dans l'introduction de ce chapitre.

4.2. La transitivité

Nous étudions dans cette section l'unification modulo l'axiome permutatif impliquant la transitivité :

$$T(*,eq) : (x\ eq\ y) * (y\ eq\ z) = (x\ eq\ y) * (x\ eq\ z)$$

eq est souvent interprété comme l'égalité et $*$ comme le "et" booléen. L'unification modulo cet axiome est noté comme un problème ouvert dans [Paul1984]. Nous allons montrer que la théorie engendrée par l'ensemble

$$T = \{T(*_i,eq) \mid i \in I\}$$

est syntaxique et que l'algorithme standard d'unification qui en découle termine.

Tout d'abord, il est clair que l'ensemble des symboles décomposables est $F-\{*_i \mid i \in I\}$ car eq n'apparait pas en tête d'axiome.
Etudions maintenant la mutation.

Proposition 7 : T est un ensemble résolvant.

Preuve: En effet, les paires critiques sont trivialement ε-confluentes. □

Toute équation $u *_i v == u' *_i v'$ se généralise donc dans la disjonction de système

U = [

Decl(e)

{ u == u', v == v'}

décomposition par $(x\ eq\ y)*_i(y\ eq\ z) = (x\ eq\ y)*_i(x\ eq\ z)$

{ u == x eq y, v == y eq z, u' == x eq y, v' == x eq z}

décomposition par $(x\ eq\ y)*_i(x\ eq\ z) = (x\ eq\ y)*_i(y\ eq\ z)$

{ u == x eq y, v == x eq z, u' == x eq y, v' == y eq z}]

Nous définissons donc la mutation d'une équation e par Mut(e) = U et la mutation d'un système par extension, Mut(S) = S[e ← Mut(e)].

Proposition 8 : L'algorithme de T-unification standard termine pour l'opération de mutation que nous venons de définir.

Preuve: Pour tout terme t et tout symbole f de l'algèbre considérée, rappelons que nous notons $|t|_f$ le nombre d'occurrence de f dans le terme t. Pour toute multiéquation e, tout système S et toute disjonction de systèmes U on pose

$$|e|_f = \sum_{t \in e} |t|_f \quad \text{et}$$

$$|S|_f = \sum_{e \in S} |e|_f$$

$$\mu(S) = \sum_{i \in I} |S|_{*_i}$$

$$\mu(U) = \sum_{S \in U} \mu(S)$$

La mutation fait strictement décroitre la mesure de complexité μ sur les systèmes tandis que la fusion comme la décomposition la conservent. Cela assure la terminaison du processus de décomposition fusion mutation. □

4.3. Transitivité et commutativité

Nous allons donner un algorithme d'unification dans le cas où on suppose en plus de la transitivité de * que le symbole eq est commutatif. Pour simplifier les preuves nous ne considèrerons que la théorie engendrée par les axiomes T(*,eq) et C(eq), pour deux symboles eq et * de F. L'extension à des théories comportant plusieurs couples de symboles différents vérifiant ces deux axiomes ne présente pas de difficultés particulières. On pourra également la voir comme une conséquence de notre étude des mélanges de théories.

Soit T.C l'ensemble des axiomes

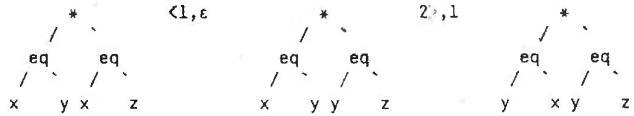
(1) T(*,eq) : (x eq y) * (y eq z) = (x eq y) * (x eq z)
(2) C(eq) : x eq y = y eq x.

L'ensemble des symboles décomposables est F-{*, eq}.

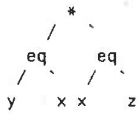
Nous allons calculer toutes les paires critiques de (1) et (2) pour vérifier leur ε-confluence. Afin d'expliciter les calculs nous représentons les paires critiques par des diagrammes dans lesquels l'indication " i>,m " signifie que l'axiome i a été appliqué à l'occurrence m et de gauche à droite par rapport à l'écriture qui le définit.

Les paires critiques de T(*,eq) ou C(eq) sur eux-mêmes sont trivialement ε-confluentes.

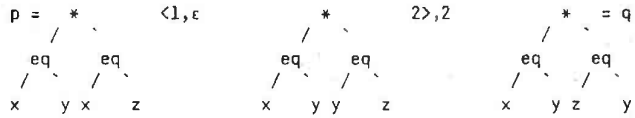
Pour les autres :



2>,1 1>,ε



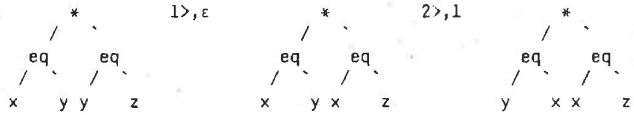
et

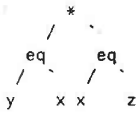


Or on ne peut pas appliquer un axiome en tête sur q sans confondre des variables et la commutativité de eq ne suffit pas à montrer que p = q. On va donc ajouter l'axiome

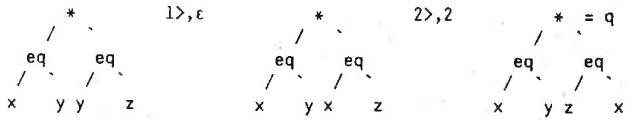
(3) (x eq y) * (x eq z) = (x eq y) * (z eq y)

La paire critique précédente (p,q) est alors trivialement ε confluente.





Cette paire critique est donc ε-confluente, mais



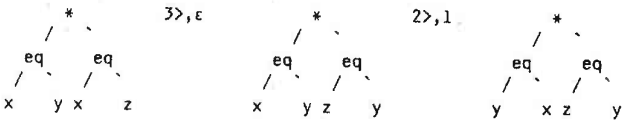
n'est pas ε-confluente car aucune application de (1) ou (3) sur q ne permet de ε-confluer.

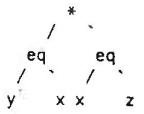
On ajoute donc l'axiome

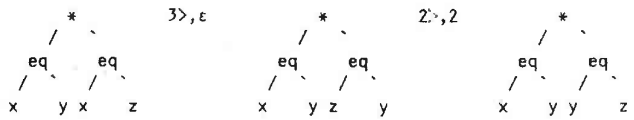
(4) (x eq y) * (y eq z) = (x eq y) * (z eq x)

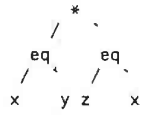
ce qui rend la paire critique précédente trivialement ε-confluente.

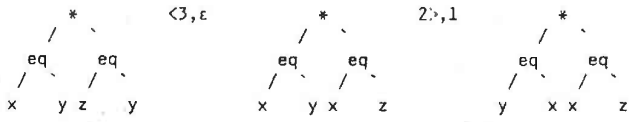
Les paires critiques de (3) et (4) sur eux même sont clairement ε-confluentes. Calculons les autres.

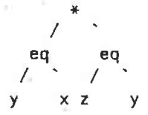


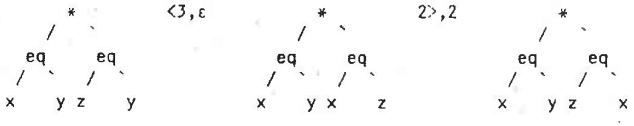


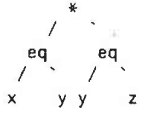




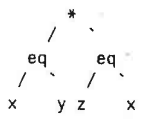




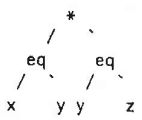




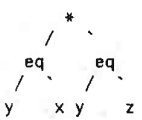
Les paires critiques de (4) avec (2)



<4,ε

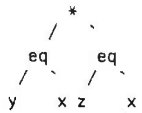


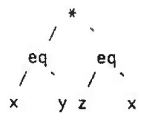
2>,1



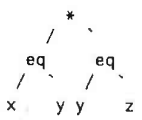
2>,1

<3,ε

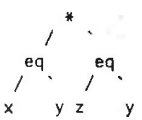




<4,ε

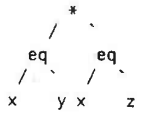


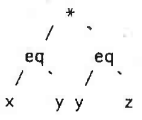
2>,2



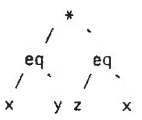
2>,2

3>,ε

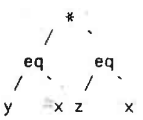




4>,ε

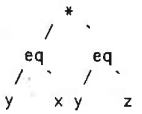


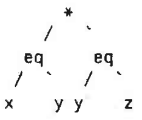
2>,1



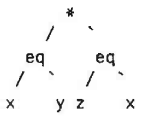
2>,1

3>,ε

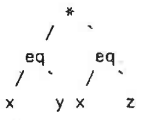




4>,ε

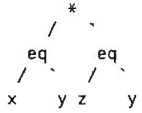


2>,1



2>,2

<3,ε



Il ne reste plus qu'à calculer les paires critiques de (1) (3) et (4) entre eux. Nous laissons le lecteur se convaincre qu'elles sont ε-confluentes.

Nous pouvons donc conclure

Proposition 9 : L'ensemble des axiomes

(1) (x eq y) * (y eq z) = (x eq y) * (x eq z)
(2) x eq y = y eq x
(3) (x eq y) * (x eq z) = (x eq y) * (z eq y)
(4) (x eq y) * (y eq z) = (x eq y) * (z eq x)

est résolvant pour la théorie engendrée par T.C.

Cela nous permet de construire l'opération de mutation d'une équation:

Si e = (u * v == u' * v') alors,

```
Mut(e) = [ { u == u', v == v'},
        décomposition suivant (x eq y) * (y eq z) = (x eq y) * (x eq z)
          { u == x eq y, v == y eq z, u' == x eq y, v' == x eq z }
        décomposition suivant (x eq y) * (x eq z) = (x eq y) * (y eq z)
          { u == x eq y, v == x eq z, u' == x eq y, v' == y eq z }
        décomposition suivant (x eq y) * (x eq z) = (x eq y) * (z eq y)
          { u == x eq y, v == x eq z, u' == x eq z, v' == z eq y }
        décomposition suivant (x eq y) * (z eq y) = (x eq y) * (x eq z)
          { u == x eq y, v == z eq y, u' == x eq y, v' == x eq z }
        décomposition suivant (x eq y) * (y eq z) = (x eq y) * (z eq x)
          { u == x eq y, v == y eq z, u' == x eq y, v' == z eq x }
        décomposition suivant (x eq y) * (z eq x) = (x eq y) * (y eq z)
          { u == x eq y, v == z eq x, u' == x eq y, v' == y eq z } ]
```

Si e = (u eq v == u' eq v') alors en reprenant ce que nous avons vu pour la commutativité :

$$Mut(e) = [\ \{ u == u', v == v'\},\ \{ u == v', v == u'\}]$$

Si e = (u eq v == u' * v'), Mut(e) = ø.

Par ailleurs, la mutation fait strictement décroitre la mesure

$$\mu = \sum_{e \in S} (|e|_{*} + |e|_{eq})$$

ce qui assure la terminaison du processus de décomposition-fusion-mutation pour la théorie T.C.

On voit que pour travailler modulo les axiomes précédents auquels on ajoute la commutativité de *, il est préférable d'automatiser la complétion. On obtiendra dans ce cas un algorithme d'unification pour la théorie constituée des axiomes T(*,eq), C(eq) et C(*).

4.4. Vers un algorithme d'unification pour une axiomatisation du "si alors sinon"

Le "si alors sinon", est une instruction fondamentale dans la plupart des langages évolués. Bloom et Tindell [Bloom1983] ont trouvé un ensemble d'axiomes (Cond) tels que l'ensemble des assertions valides dans la classe C des algèbres où un symbole de fonction est toujours interprété comme un test "si alors sinon", coincide exactement avec celui des assertions prouvables par l'ensemble d'axiomes (Cond) suivant :

(1) c(Ω,x,y) = Ω
(2) c(x,Ω,Ω) = Ω
(3) c(t,x,y) = x
(4) c(f,x,y) = y
(5) c(x,x,y) = c(x,t,y)
(6) c(x,y,x) = c(x,y,f)
(7) c(x,c(x,y,z),w) = c(x,y,w)
(8) c(x,y,c(x,z,w)) = c(x,y,w)
(9) c(c(x,y,z),u,v) = c(x,c(y,u,v),c(z,u,v))
(10) c(x,c(y,z,u),c(y,v,w)) = c(y,c(x,z,v),c(x,u,w))

Formellement, soit F = {Ω, t, f, c} un ensemble de symboles d'arité respective 0, 0, 0, 3. Soient tt et ff l'interprétation de t et f dans toute F-algèbre I. C est la classe des F-algèbres I satisfaisant pour tous d, d', d" du domaine de I :

$$c_I(d,d',d'') = d' \text{ si } d = tt$$
$$= d'' \text{ si } d = ff$$
$$= \Omega_I \text{ sinon}$$

Mais, C n'étant pas fermée par produit [Guessarian1977], C n'est pas une variété équationnelle. Cependant comme l'ont montré Bloom et Tindell t=t' est valide dans C si et seulement si $t =_{Cond} t'$. Ce résultat a été étendu à un alphabet F quelconque par C. Benoit [Benoit1984]. Pour décider de la Cond-égalité on souhaiterait compléter Cond en un système de RE-réécriture canonique pour un bon choix de E. Il faut donc pouvoir travailler modulo les axiomes permutatif de Cond et donc disposer d'un algorithme d'unification modulo ces axiomes. C'est dans cette optique que nous allons donner un algorithme d'unification modulo les axiomes (5) et (6). Il restera alors à traiter avec l'axiome (10), ce qui sera tenté dès qu'une implantation de l'algorithme de complétion unificationnelle existera.

Théorème 9 : Soient les axiomes

(5) c(x,x,y) = c(x,t,y)

(6) c(x,y,x) = c(x,y,f)

(5') c(x,x,f) = c(x,t,x)

(6') c(x,x,x) = c(x,t,f).

L'ensemble de ces axiomes est ε-confluent.

Preuve: Il suffit de calculer les ε-paires critiques. □

Nous disposons donc d'une opération de mutation pour la théorie constituée des axiomes (5) et (6). Elle est définie par :

Si e = (c(u, v, w) == c(u', v', w') alors,

```
Mut(e) = [ { u == u', v == v', w == w'},
         décomposition suivant c(x,x,y) = c(x,t,y)
           { u == v == u' == x, w == w' == y, v' = t}
         décomposition suivant c(x,t,y) = c(x,x,y)
           { u == u' == v' == x, v == t, w == w' == y}
         décomposition suivant c(x,y,x) = c(x,y,f)
           { u == u' == w == x, v == v' == y, w' == f}
         décomposition suivant c(x,y,f) = c(x,y,x)
           { u == u' == w' == x, v == v' == y, w == f}
         décomposition suivant c(x,x,f) = c(x,t,x)
           { u == v == u' == w' == x, w == f, v' == t}
         décomposition suivant c(x,t,x) = c(x,x,f)
           { u == w == u' == v' == x, v == t, w' == f}
         décomposition suivant c(x,x,x) = c(x,t,f)
           { u == v == w == u' == x, v' == t, w' == f}
         décomposition suivant c(x,t,f) = c(x,x,x)
           { u == u' == v' == w' == x, v == t, w == f} ]
```

La terminaison du processus de décomposition-fusion-mutation est claire du fait que l'opération de mutation que nous venons de décrire fait strictement décroître la taille des termes considérés dans les multiéquations.

Un algorithme d'unification modulo l'ensemble des axiomes de Cond pourra éventuellement être obtenu en combinant de l'unification suivant la méthode "syntaxique" que nous avons décrite, pour les axiomes de cond de type purement permutatif, avec la RE-surréduction obtenue à partir des

axiomes restant dans Cond. Cela pourra être obtenu à partir d'une extension du système REVEUR3.

CHAPITRE 4

UNIFICATION ASSOCIATIVE COMMUTATIVE

1. Introduction

L'unification associative commutative (en abrégé AC) est la résolution d'équations modulo les propriétés d'associativité-commutativité d'un nombre fini de symboles d'opérations de l'algèbre considérée. Cette théorie équationnelle a été certainement la plus étudiée après la théorie vide. Ceci pour deux raisons. D'une part les propriétés d'associativité-commutativité sont associées dans de très nombreuses structures algébriques. D'autre part, dans les applications basées sur les techniques de réécriture on ne peut les dissocier l'une de l'autre en orientant par exemple l'associativité en règle de réécriture et en réécrivant modulo la commutativité car $\rightarrow_{R/E}$ n'a plus alors la propriété de terminaison finie. En effet si on a $(x+y)+z \rightarrow x+(y+z)$ et $x+y = y+x$, alors $(x+y)+z \rightarrow x+(y+z) =_C (z+y)+x \rightarrow z+(y+x) =_E (x+y)+z$. Il faut donc travailler modulo ces deux axiomes.

Par ailleurs, la résolution d'équations modulo l'associativité-commutativité est un problème difficile, contrairement par exemple à ce qui se passe pour la commutativité seule. Les difficultés résident à deux niveaux. D'une part la découverte d'algorithme d'unification efficace et d'autre part la preuve de terminaison de cet algorithme. Historiquement, les premiers algorithmes ont été donnés indépendemment par Livesey et Siekmann [Livesey1976] et Stickel [Stickel1975], repris par J.M.Hullot [Hullot1980]. Mais la terminaison des algorithmes proposés reste alors un problème ouvert. Ce n'est qu'une dizaine d'années plus tard que F.Fages [Fages1984] donnera une preuve de terminaison de ces algorithmes.

Des implantations ont été réalisées par M.Stickel ainsi que dans le système KB [Fages1984] et dans REVEUR3 décrit dans l'annexe et dans [Kirchner1985], afin de rendre possible la complétion de systèmes de réécriture modulo l'associativité-commutativité.

Il ne reste plus aujourd'hui qu'une difficulté : celle de permettre l'unification modulo l'associativité commutativité en présence d'autres propriétés sur d'autres opérateurs, et la preuve de terminaison associée. Cela sera une conséquence de l'étude de l'unification dans les mélanges de théories qui est faite dans la suite. Nous allons ici montrer comment le processus de décomposition-fusion-mutation agit sur les systèmes comportant des symboles AC, puis nous montrerons comment résoudre les systèmes de multiéquations comportant un ou plusieurs symboles AC et des symboles sans propriété.

2. L'opération de mutation en présence d'opérateurs associatif-commutatif

Nous allons appliquer ici les résultats généraux obtenus dans les

chapitres précédents. Il suffit donc de déterminer l'opération de mutation dès que l'on se trouve en présence de symboles de fonction associatif-commutatif, en abrégé AC.

2.1. Définitions et notations propres au problème

On désigne dans toute la suite par Fac l'ensemble des symboles associatif commutatif de l'algèbre considérée. On ne considère dans cette partie que des symboles soit sans propriété, soit AC.

Tout terme sera considéré dans cette section sous sa forme aplatie.

Définition 65 : La forme **ac-aplati** d'un terme t est un couple (f, args) noté apla(t) où f est un symbole de fonction et args un multiensemble fini de forme aplatie de termes, récursivement définie par :

- si $t \in V \cup F_0$ alors apla(t) = (t, {})
- si $t = f(t_1, \ldots, t_n)$ alors apla(t) = (f, args) avec, en notant pour j dans [1..n] $apla(t_j) = (f_j, args_j)$,
  - * $apla(t_j) \in args$ si et seulement si $f_j \notin Fac$ ou $f_j \neq f$
  - * $args_j \subseteq args$ si et seulement si $f_j = f$ et $f \in Fac$. ○

La notion de terme aplatie est proche de celle de terme construit avec des symboles d'arité variable, il faut cependant noter que l'ordre des fils d'un symbole n'importe pas ici dans la structure de multiensemble.

Exemple 32 : En supposant que + et * sont des symboles AC soit

t = (a * (x + b)) + ((x + y) + a). On a alors :

apla(t) = (+, ((*, ((a, ()), (+, ((x, ()), (b, ()))))), (x, ()), (y, ()), (a, ())))

Notation : Nous utiliserons les notations plus agréables suivantes :

(t, ()) est noté t,

(f, (t_1, ..., t_n)) est noté $f(t_1,...,t_n)$ ou t_1 f t_2 ... f t_n si f a une notation infixée. La notion d'occurrence s'étend sans difficulté autre que notationnelle aux termes aplatis.

Par conséquent dans l'exemple précédent, apla(t) sera noté (a*(x+b))+x+y+a et on a $t(\varepsilon)$ = + et t(1) = *.

Afin d'étudier la terminaison du processus de décomposition-fusion-mutation dans le cas AC, nous introduisons les applications suivantes:

Définition 66 : Pour tout terme t en représentation aplatie on pose

$$p(t) = \sum_{m \text{ tel que } t(m) \in Fac} |m|$$

Pour toute multiéquation e,

$$p(e) = \max_{t \in e} p(t)$$

pour tout système S,

$$p(S) = \max_{e \in S} p(e)$$

et pour toute disjonction de système U,

$$p(U) = \max_{S \in U} p(S).$$

○

Exemple 33 : En poursuivant l'exemple précédent, p(t) = 0 + 1 + 2 = 3

2.2. simplifications

Si un symbole + est AC, il est régulier à droite et à gauche. Cette propriété classique permet de simplifier les équations, comme nous l'avons montré dans l'étude générale de l'équivalence. Ce qui se traduit, pour tout symbole f AC et pour tous termes t, t_k, t_k' (éventuellement variables) et compte tenu de la mise sous forme aplatie, par l'équivalence :

$$f(t,t_1,...,t_n) == f(t,t_1',...,t_p') \Leftrightarrow f(t_1,...,t_n) == f(t_1',...,t_p').$$

La première étape dans la mutation AC est par conséquent la simplification des termes apparaissant dans les deux membres de l'équation.

2.3. Cas des systèmes S tels que p(S) > 0

On peut maintenant introduire la première transformation utile dans le cas AC. Elle généralise la transformation Gen^W étudiée précédemment.

Soit e=($(f(t_1,..., t_p) == f(t_{p+1},..., t_{n+1})$) une équation en représentation aplatie (c'est à dire telle que les termes qui la composent soient en représentation aplatie) telle que, f soit un symbole AC. Soit W un ensemble de variables tel que Var(e) ⊆ W. On généralise tout d'abord cette équation de façon minimale, en un système d'équations : à tout terme non variable t_j (pour $1 \leq j \leq n+1$) est associé une "nouvelle" variable en dehors de W, où le renommage est fait de telle sorte que si deux termes non variables sont égaux, ils sont généralisés en la même variable. On ne renomme donc pas les termes variables t_j. Le système obtenu est noté $Gen^W_{min}(e)$ et peut être représenté par

$$\begin{array}{l} f(x_1,..., x_p) == f(x_{p+1},..., x_{n+1}) \\ \quad ... \\ x_i == t_j \\ \quad ... \end{array}$$

Exemple 34 : Si f est AC alors f(a, g(a, b), x) == f(g(a, b), x, y, b) est généralisé dans le système

$$\begin{array}{l} S = \{ f(x_1, x_2, x) == f(x_2, x, y, x_3), \\ \qquad x_1 == a, \\ \qquad x_2 == g(a,b), \\ \qquad x_3 == b\} \end{array}$$

On pourra remarquer au cours de ce qui suit qu'il n'est en fait pas utile de généraliser les sous termes qui ne contiennent pas de symboles AC, comme par exemple les constantes.

Une preuve similaire à celle faite pour Gen^W montre que $Gen^W_{min}(e) \Rrightarrow_{AC} e$.

Comme l'on montré M.Stickel et J.M.Hullot, on sait résoudre l'équation é = $(f(x_1,\ldots, x_p) == f(x_{p+1},\ldots, x_{n+1}))$. Soit R un ensemble complet de AC-solutions de cette équation et $[Eq(r)]_{r\in R}$ la disjonction de système associé à R. On note

$$mut_r(e) = Fus(Eq(r) \cup (Gen^W_{min}(e)\text{-é}))$$

et par conséquent,

$$mut(e) = [mut_r(e)]_{r\in R} \Rrightarrow_{AC} e$$

Notation : Soit e une multiéquation du système S, telle p(e) > 0. On notera $mut_r(S)$ le système dans lequel e a été remplacée par $mut_r(e)$: $mut_r(S) = S_{[e\leftarrow mut_r(e)]}$. D'où la définition de la première mutation d'un système :

$$mut(S) = [mut_r(S)]_{r\in R}$$

Lemme 23 : Soit e une multiéquation telle que p(e) > 0 alors si e a des solutions, quelque soit é dans $mut_r(e)$ on a p(é) < p(e).

Preuve: Pour toutes les équations issues de généralisation et qui sont donc du type $x_i == t_j$, on a $p(x_i == t_j) < p(e)$ car t_j est un sous terme d'un terme de e. D'autre part on a pour tout e" dans Eq(r), p(e") = 0. □

Définition 67 : A chaque système S on associe le p(S)-uplet d'entiers suivant:

$$m(S) = (s_{p(S)}, s_{p(S)-1}, \cdots, s_0)$$

ou s_i désigne le nombre de termes t de S tels que p(t) = i. U étant une disjonction de systèmes, les n-uplet d'entiers étant ordonnés par l'ordre lexicographique, on pose $m(U) = \max_{S\in U} m(S)$. ○

L'ensemble des suites finies sur les entiers étant muni de l'ordre lexicographique, m définit donc une mesure sur les systèmes de multiéquations.

Lemme 24 : Soit S un système tel que m(S) > 0. Alors,

- mut fait décroitre strictement m.
- Fus conserve m.
- Dec fait décroitre m au sens large.

Preuve: - mut fait décroitre strictement m, c'est une conséquence du lemme précédent.

- Fus conserve m car la fusion conserve les termes
- Dec fait décroire m au sens large car la décomposition de deux termes dont l'un au moins contient un symbole AC fait strictement décroitre m, mais la décomposition de deux termes sans symboles AC ne modifie pas m. □

On en déduit donc :

Corollaire 4 : Soit U une disjonction de système telle que p(U) > 0. En itérant un nombre fini de fois les étapes de décomposition fusion et de

première mutation mut définie ci-dessus, on obtient une disjonction de systèmes U' tels que p(U') = 0

Il reste donc à montrer que l'algorithme de complète décomposition termine pour tout système S non complètement décomposé tel que p(S) = 0, c'est à dire tel que les symboles AC n'apparaissent dans les termes en représentation aplatie du système, qu'à l'occurrence ε.

2.4. Cas des systèmes S tels que p(S) = 0

Contrairement à ce qu'on pourrait imaginer de prime abord l'opération de première mutation (mut) que nous venons de décrire pour les systèmes S tels que p(S) > 0 ne permet pas de terminer pour les systèmes tels que p(S) = 0. En voici un exemple.

Soit le système S = { x + y == u + v, x + y' == u + v'}. En appliquant mut sur ces deux équations on obtient une disjonction de systèmes dont voici l'un des éléments après fusion :

$$\begin{aligned}
&\{ x == x_1 + x_2 == x_1' + x_2',\\
&\quad u == x_1 + x_3 == x_1' + x_3',\\
&\quad y == x_3 + x_4,\\
&\quad v == x_2 + x_4,\\
&\quad y' == x_3' + x_4',\\
&\quad v == x_2' + x_4'\}
\end{aligned}$$

Or les deux premières équations de ce système ne sont que celles de S à un renomage des variables près.

La première idée qui va permettre d'éviter ce phénomène est la résolution simultanée de plusieurs multiéquations. Dans l'exemple précédent cela revient à transformer S directement en un système complètement décomposé. Cette démarche est en fait tout à fait naturelle dans le contexte des systèmes de multiéquations, dans lequel nous nous trouvons.
Nous explicitons maintenant cette démarche.

2.4.1. résolution directe de systèmes de multiéquations e telles que p(e) = 0

Nous montrons dans ce paragraphe comment trouver des ensembles complets de AC-unificateurs pour des systèmes de multiéquations du type $p_1 == p_2$ avec p_1 et p_2 tels que $top(p_1) = top(p_2) \in Fac$ et $p(p_1 == p_2) = 0$.

Nous généralisons donc ici les travaux de M.Stickel et J.M.Hullot aux systèmes d'équations. Il n'y a pas de d'originalité théorique par rapport à ce qu'ils ont fait; nous montrons simplement comment utiliser leurs résultats pour résoudre notre problème.

Rappelons en les termes. Soit $S = \{e_j\}_{j\in J}$ un système d'équations de la forme

$$e_j = f(x_1^j, \ldots, x_{n_j}^j) == f(x_{n_j+1}^j, \ldots, x_{p_j}^j)$$

où $f \in Fac$.

La structure de terme étant ici réduite à un niveau, le problème se ramène à la résolution de ces mêmes équations dans le monoide libre abélien, ce dernier problème se ramenant lui même à la résolution d'équation diophantienne linéaire et homogène dans N.

Nous étudions maintenant ces différentes étapes.

2.4.1.1. résolution de systèmes d'équations dans le monoide libre commutatif

Un exposé concernant la résolution des équations peut être trouvé dans [Hullot1980].

Définition 68 : On appelle monoide libre abélien M, la F-algèbre AC-libre sur V, où F est l'ensemble réduit au symbole + d'arité 2. Les éléments de M sont appelés mots abéliens, l'opération + étant alors vue comme la concaténation, le mot vide est noté ^. M est symétrisable dans un groupe que nous noterons G : (G, +) est le plus petit groupe contenant M l'opposé d'un élément o étant noté -o. G peut être muni classiquement d'une structure de Z-module à gauche. Nous noterons multiplicativement l'opération de Z sur G. ○

Définition 69 : Soit J un ensemble fini d'entiers et a_k des entiers relatifs. Un élément $O = (o_1, \ldots, o_n)$ de M est solution du système d'équations

$$S = \{ \sum_{k=i_j}^{k=n_j} a_k x_k = \hat{} \}_{j=1\ldots n} \qquad \text{(equ.monoide)}$$

si et seulement si

$$\forall\, j=1\ldots n : \sum_{k=i_j}^{k=n_j} a_k o_k = \hat{}$$

○

Au système S on peut associer un système d'équations diophantiennes lineaires homogènes D_S. Or on a le résultat suivant :

Théorème 10 : L'ensemble des solutions de l'équation diophantienne linéaire homogène

$$\sum_{i\in I} a_i x_i = 0$$

avec $\forall\, i \in I\ a_i \in Z$, forme un monoide commutatif sur $\mathbb{N}^n$. Cet ensemble peut donc être engendré par une base finie [Clifford1967, Redei1963].

Par conséquent l'ensemble des solutions du système S est un sous monoide de base finie B car intersection des monoides solutions de chacune des équations de S. Toute solution du système S est donc de la forme

$$N = (v_1, \ldots, v_p) = G \circ C$$

où G est un élément quelconque de $\mathbb{N}^q$ et C la matrice q*p dont chacune les lignes est un vecteur de la base B.

Il ne reste plus qu'à donner un algorithme qui permette de calculer non pas la base d'une équation diophantienne [Huet1978], mais une base du système D_S.

2.4.1.1.1. Résolution des systèmes d'équations diophantiennes linéaires et homogènes

Deux méthodes sont possibles :

Soit en utilisant les résultats de G. Huet sur la résolution d'une équation et en résolvant le système de la même façon que dans un espace vectoriel.

- Soit en généralisant les résultats de G. Huet à des systèmes d'équations. Il faut alors déterminer quelles sont les bornes à utiliser pour trouver directement une base du système.

Dans les deux cas, la résolution des systèmes d'équations diophantiennes sera plus efficace que n résolutions des équations e qui composent le système S comme cela est fait dans l'algorithme de Stickel. En effet l'espace des solutions engendré sera plus restreint du fait de l'accumulation des contraintes.

2.4.1.1.2. Description de la base de l'ensemble des solutions d'un système d'équations dans le monoide commutatif libre

Théorème 11 : Soit C la matrice q*p dont chaque ligne est un vecteur de la base B de l'ensemble des solutions du système d'équations diophantiennes D_S issue de (equ. monoide). Alors l'ensemble des solutions de S dans M est $\{ToC \mid T \in M^q\}$.

Preuve: La preuve est la même que celle donnée par Hullot □

Pour chaque AC-solution du système S, chaque variable du système doit être instanciée (éventuellement en elle même). De plus, il est intéressant pour des raisons d'efficacité, d'éliminer d'emblée les solutions qui produiront un échec à la décomposition. Il faut tenir compte de ces faits pour déterminer, à partir de la description d'une base des solutions du système D (utilisant éventuellement le mot vide) un ensemble complet de AC-solutions du système S.

Pour ce faire, J.M. Hullot montre dans sa thèse comment décrire une base de l'ensemble des solutions du système D lorsque l'on a fixé des contraintes c sur la forme des solutions. L'ensemble Σ_M^c des solutions de S satisfaisant ces contraintes peut alors se décrire à l'aide de s matrices C_j par

$$\Sigma_M^c = \{ T \circ C_1, \ldots, T \circ C_s \mid T \in (M-\{\hat{}\})^r \}$$

Nous renvoyons à [Hullot1980] pour davantage de détails.

2.4.1.2. Ensemble complet de AC-solutions du système S

Soit $C' = (c_{i,j})$ avec $1 \leq i \leq r$ et $1 \leq j \leq p$, l'une des matrices, utilisée dans la description ci dessus, des solutions sous contraintes de S. Soit W un ensemble de variables contenant $Var(S) = \{x_1, \ldots, x_p\}$ et (z_i), $i \in N$ une suite de variables distinctes en dehors de W. Soit σ la substitution de domaine Var(S) définie pour $1 \leq k \leq p$ par

$$\sigma(x_k) = f(z_1^{c_{k,1}}, \ldots, z_r^{c_{k,r}})$$

Soit $\Sigma = \{\sigma_1, \ldots, \sigma_2\}$ l'ensemble de ces substitutions pour toutes les matrices C_i.

Théorème 12 : Σ est un ensemble complet de AC-solution du système S en dehors de W.

Preuve: La preuve ne pose pas de difficultés et peut être trouvée dans [Stickel1975] □

Par conséquent, si S est un système d'équations tel que p(S) = 0 et dont les éléments sont de la forme $p_1 = p_2$, avec $top(p_1) = top(p_2) \in Fac$. Pour résoudre ce système on procède de la façon suivante :

* si tous les termes ont le même symbole ac alors on applique le théorème précédent,

* sinon on partitionne l'ensemble des équations suivant leur symbole de tête, on résoud indépendamment chacune des partitions et on regroupe les contraintes pour terminer.

Plutôt que de donner une description formelle d'une telle transformation, nous en donnons maintenant un exemple.

Exemple 35 : Soit S = { x + y == u + v,

x + y' == u + v',

x * u == a * v}

La résolution du système réduit au deux premières équations donne 27 solutions, la résolution de la troisième équation, 4 solutions. En regroupant les contraintes on trouve 13 solutions au système S.

2.4.1.3. Résolution générale des systèmes S tels que p(S) = 0

On pourrait penser que moyennant la résolution en bloc des multiéquations indécomposables et mutables, on assure la terminaison du processus de mutation. Il n'en est rien comme le montre l'exemple suivant.

Exemple 36 : Soit

S = { x + y == z + t,

x == z + u}

L'un des systèmes obtenu par mutation de la première équation puis fusion est S' = { x == z + t == $x_1 + x_2$,

z == $x_1 + x_3$,

y == $x_3 + x_4$,

t == $x_2 + x_4$}

On reconnait dans les 2 premières multiéquations de S' un renommage des équations de S.

Afin d'éviter ce phénomène de bouclage, nous allons reporter les contraintes de la forme x == z+u de l'exemple précédent, dans les équations à muter. Dans l'exemple cela permettra de simplifier par z et donc de

terminer.

Définition 70 : Pour tout système S indécomposable, fusionné et tel que p(S) = 0 on note **cd(S)** l'ensemble des multiéquations complètement décomposées de S et **am(S)** le complémentaire de cd(S) dans S, c'est à dire l'ensemble des multiéquations de S qui sont indécomposables mais non complètement décomposées. ○

On supposera dans ce qui suit pour plus de clarté, que les systèmes considérés sont uniquement composés d'équations. Si cela n'est pas le cas, il est simple de se ramener à un système équivalent qui vérifie cette propriété.

Soit S un système non complètement décomposé. Puisque AC est une théorie stricte, soit cd(S) a une plus petite solution σ, soit < n'est pas un ordre strict sur cd(S) et par conséquent S n'a pas de solution. On notera S' le système obtenu par remplacement compatible avec p(S) = 0 de cd(S) dans am(S) : S' = cd(S) ∪ cd(S)[[am(S)]], et S'' le système obtenu par simplification de S'.

On a par définition S'' = cd(S) ∪ am(S''). Soit W un ensemble de variables tel que Var(S'') ⊆ W. On a vu dans le paragraphe précédent comment résoudre directement le système am(S''). Soit $[Si]_{i\in I} = Eq^W(am(S''))$ la disjonction de systèmes associée à l'ensemble complet de AC-solutions en dehors de W de am(S'').

Soit MUT(S) = $Fus([Si \cup cd(S)]_{i\in I})$

Lemme 25 : Avec les notations précédentes, MUT(S) $\Rightarrow_{AC}$ S

Exemple 37 : Si on suppose que + et * sont AC,

$S = \{x + b == z + a,$

$x == a + u,$

$v == x * b\}$

on a

$cd(S) = \{x == a + u,$

$v == x * b\}$

et

$am(S) = \{x + b == z + a\}$

$S' = \{a + u + b == z + a,$

$x == a + u,$

$v == x * b\}$

et

$S'' = \{u + b == z + a,$

$x == a + u,$

$v == x * b\}$

$Eq^W(am(S'')) = [\ \{u == a, z == b\},$

$\{u == a + x_1, z == b + x_1\}\ ]$

et par conséquent

$MUT(S) = [\ \{ u == a,$

$z == b,$

$x == a + u,$

$v == x * b\},$

$\{ u == a + x_1,$

$z == b + x_1,$

$x == a + u,$

$v == x * b\}\ ]$

3. Description de l'algorithme d'unification associative commutative

Pour appliquer les résultats généraux il suffit de préciser l'opération de mutation d'un système S. Celle ci est définie de la façon suivante :

Si p(S) > 0, la mutation de S est mut(S), sinon MUT(S).

Il faut noter que l'on pourrait définir la mutation uniquement par MUT. Mais cette opération est intrinsèquement plus complexe que mut. En effet il faut analyser cd(S) pour détecter un bouclage éventuel puis faire un remplacement compatible. Or l'expérience prouve que dans la plupart des cas, mut permet d'aboutir à une disjonction de systèmes complètement décomposés et donc de résoudre un système de façon extrèmement efficace.

4. Terminaison de l'algorithme d'unification associative commutative standard

Théorème 13 : Le processus de décomposition fusion et AC-mutation termine pour toute équation.

Preuve: Le cas où il n'y a qu'un seul symbole AC est simple. Dans le cas contraire, le théorème est une conséquence du résultat géneral sur la terminaison dans les mélanges de théories que nous donnons au dernier chapitre. □

5. Un exemple développé comparativement

Nous allons pour terminer comparer sur un exemple volontairement simple l'algorithme de Stickel et l'algorithme standard dont nous venons de

décrire l'opération de mutation.

Soit + un symbole AC, f un symbole sans propriété et l'équation

$$e = (f(x+y,x+b) == f(u+v,u+a))$$

a) Par l'algorithme standard : Le symbole f étant décomposable,

$$e \Leftrightarrow S = \{x+y == u+v,\ x+b == u+a\}.$$

La résolution directe de ce dernier système donnant les 4 solutions suivantes :

$\sigma_1 = \{x \leftarrow a,\ u \leftarrow b,\ v \leftarrow a,\ y \leftarrow b\}$

$\sigma_2 = \{x \leftarrow a,\ u \leftarrow b,\ v \leftarrow (nv_1 + a),\ y \leftarrow (nv_1 + b)\}$

$\sigma_3 = \{u \leftarrow (b + nv_3),\ x \leftarrow (a + nv_3),\ v \leftarrow a,\ y \leftarrow b\}$

$\sigma_4 = \{u \leftarrow (b + a),\ x \leftarrow (a + a),\ v \leftarrow (nv_1 + a),\ y \leftarrow (nv_1 + b)\}$

On obtient donc après résolution d'un système de deux équations diophantiennes l'ensemble des solutions de e.

b) Par l'algorithme de Stickel :

Le symbole f n'ayant pas de propriété on résoud tout d'abord l'équation x+y == u+v, ce qui donne les 7 solutions suivantes :

$\sigma_1 = \{v <- x,\ u <- y\}$
$\sigma_2 = \{u <- (y+nv_3),\ x <- (v+nv_3)\}$
$\sigma_3 = \{u <- x,\ v <- y\}$
$\sigma_4 = \{v <- (y+nv_2),\ x <- (nv_2+u)\}$
$\sigma_5 = \{u <- (nv_2+x),\ y <- (v+nv_2)\}$
$\sigma_6 = \{v <- (nv_1+x),\ y <- (nv_1+u)\}$
$\sigma_7 = \{u <- (nv_2+nv_4),\ v <- (nv_1+nv_3),\ x <- (nv_3+nv_4),\ y <- (nv_1+nv_2)\}$

On va alors instancier successivement par chacune de ces sept substitutions les termes de l'équation x+b == u+a puis résoudre les équations obtenues. Il faudra donc résoudre 7 équations diophantiennes qui seront en général plus complexe que les équations issues de S.

Par exemple il faudra résoudre l'équation $\sigma_7(x+b) == \sigma_7(u + a)$, c'est-à-dire

$$nv_3 + nv_4 + b == nv_2 + nv_4 + a$$

et on trouve les deux solutions suivantes :

$\sigma_8 = \{nv_3 <- a,\ nv_2 <- b\}$

$\sigma_8 = \{nv_2 <- (b + nnv_3),\ nv_3 <- (a + nnv_3)\}$

de même il faudra résoudre $\sigma_6(x+b) == \sigma_6(u+a)$, c'est-à-dire x+b == u+a qui a pour solution :

$\sigma_9 = \{x <- a,\ u <- b\}$.

On voit clairement sur cet exemple que la résolution de systèmes d'équations permet d'accélérer notablement la résolution du problème. Il faut également noter que la résolution de chacun des systèmes peut se faire en parallèle.

6. Conclusion

Nous avons présenté un algorithme d'unification AC conséquence des outils que nous avons développé auparavent. Nous aurions pu nous limiter à l'étude de l'unification dans le cas d'un seul symbole AC et appliquer les résultats généraux sur les mélanges de théories donnés dans la suite. Cela conduit à considérer la décomposition comme l'opération de mutation de la théorie vide. Nous avons préféré montrer d'une part comment nos outils permettaient d'approcher le problème et d'autre part comment la décomposition pouvait, en étant intégrée au processus, accélérer la résolution. Par ailleurs, le parallélisme possible dans la résolution d'un problème d'AC-unification étant clairement mis en évidence il sera intéressant d'étudier le coût de l'algorithme standard AC-unification sur n processeurs en parallèle.

CHAPITRE 5

UNIFICATION MODULO LA COMMUTATIVITE DROITE

Nous nous intéressons ici au cas où l'ensemble des axiomes est réduit au seul axiome de commutativité droite

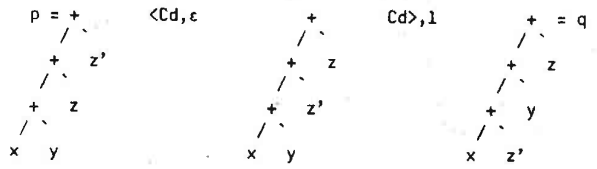
$$Cd(+) : (x + y) + z = (x + z) + y.$$

C'est un axiome qui intervient dans de nombreuses structures algébriques. Rappelons que Cd est un axiome que satisfait par exemple la division ou l'exponentiation dans l'ensemble des réels.
Un algorithme technique de Cr-unification a été donné par Jeanrond [Jeanrond1980] sans preuve de correction ni de terminaison. Il ne nous parait pas simple d'en faire la preuve.

Il est clair que l'ensemble des symboles décomposables est $F-\{+\}$. Nous allons donc uniquement chercher à résoudre modulo Cd un système d'équations dans l'algèbre $M(\{+\},X)$.

1. Détermination de l'opération de mutation

Si nous cherchons à montrer que cet axiome est résolvant en calculant les paires critiques on obtient :



qui est la seule possibilité de confluence de la paire critique (p,q), mais qui malheureusement ne satisfait pas les hypothèses requises pour que la paire critique soit ε-confluente.

On pourrait pas conséquent essayer de complèter l'ensemble des axiomes en un ensemble qui soit ε-confluent. Nous allons plutôt montrer directement que {Cd} est résolvant. Pour cela nous utiliserons la relation suivante :

Définition 71 : Soit » la relation définie sur les couples de termes par

(u,v) » (u',v')

<=>

(a) u |-p-| v et u' |-q-| v'

où p et q sont les longueurs des plus courtes preuves de Cd-égalité avec q < p

ou bien

(b) u' $=_{Cd}$.sub.$=_{Cd}$ u et v' $=_{Cd}$.sub.$=_{Cd}$ v

(où rappelons le, "t_1 sub t_2" ssi t_1 est un sous terme strict de t_2).

○

La relation » est noethérienne car Cd conserve la taille des termes. En effet, si il existait une chaine infinie de couples en relation par », Cd conservant la taille on ne pourra pas avoir une infinité de cas (b) puisque la relation de sous terme diminue strictement la taille. Il faudrait donc qu'il y ait une chaine infinie de cas (a) ce qui est également impossible.

Proposition 10 : {Cd(+)} est un ensemble résolvant, c'est à dire, pour tous termes t = a+b et t' = c+d, si t $=_{Cd}$ t' alors ou bien a $=_{Cd}$ c et b $=_{Cd}$ d ou bien il existe un terme w tel que a $=_{Cd}$ w + d et c $=_{Cd}$ w + b.

Preuve: Soit

(*) a+b |-|[m_1] a1+b1 |-|[m_2] ... c'+d' |-|[m_k] c+d

une plus courte preuve de a+b $=_{Cd}$ c+d.

Si toutes les occurrences d'application d'axiome dans cette preuve sont plus grande que ε (∀ i ∈ [1..k], $m_i \neq \varepsilon$) alors il est clair que a $=_{Cd}$ c et b $=_{Cd}$ d.

Sinon, utilisons la relation » définie plus haut pour faire la preuve par récurrence noetherienne.

Dans (*) désignons par u_1, u_2, u_3 les termes a+b, c'+d' et c+d respectivement. Par hypothèse on a (u_1,u_3) » (u_1,u_2) ; nous pouvons donc appliquer l'hypothèse de récurrence sur (u_1,u_2) : ou bien a $=_{Cd}$ c' et b $=_{Cd}$ d' ou bien il existe un terme w tel que a $=_{Cd}$ w+d' et c' $=_{Cd}$ w+b.

Si $m_k \neq \varepsilon$ alors $c =_{Cd} c'$ et $d =_{Cd} d'$ d'où la conclusion.

Si $m_k = \varepsilon$ alors il existe un terme w' tel que $c' = w'+d$ et $c = w'+d'$. Si on est dans le cas où $a =_{Cd} c'$ et $b =_{Cd} d'$, la conclusion est immédiate.

Sinon on a : $a =_{Cd} w+d'$ et $c' =_{Cd} w+b$, et par conséquent

$w+b =_{Cd} c'$ sub $c'+d =_{Cd} a+b = u_1$ et

$w'+d = c'$ sub $c'+d = u_2$ donc

(u_1,u_2) R $(w+b,w'+d)$, on peut alors appliquer l'hypothèse de récurrence au couple $(w+b,w'+d)$. Il existe donc w" tel que $w =_{Cd} w"+d$ et $t =_{Cd} w"+b$. D'où, en remplaçant w et w' par leurs valeurs dans les expressions de a et c:

$a =_{Cd} (w"+d)+d' =_{Cd} (w"+d')+d$ et $c =_{Cd} (w"+b)+d' =_{Cd} (w"+d')+b$

ce qui fournit la conclusion en prenant $w = w"+d'$. □

Comme conséquence des résultats précédents, nous pouvons donc préciser maintenant une généralisation possible de toute équation indécomposable. Toute équation $e = (u+v == u'+v')$ est généralisée dans la disjonction de systèmes

$$
\begin{aligned}
U = [\ &\{\ u == u',\ v == v'\},\\
&\quad \text{décomposition par } (x + y) + z = (x + z) + y.\\
&\{\ u == x + y,\ v == z,\ u' == x + z,\ v' == y\ \},\\
&\quad \text{décomposition par } (x + z) + y = (x + y) + z.\\
&\{\ u == x + z,\ v == y,\ u' == x + y,\ v' == z\ \}\ ].
\end{aligned}
$$

On a donc

$$U \Rightarrow_{Cd} e.$$

Il existe un généralisé plus simple, en effet on peut facilement montrer que $U' \Rightarrow_{Cd} e$ avec

$$
\begin{aligned}
U' = [\ &\{\ u == u',\ v == v'\},\\
&\{\ u == x + v',\ u' == x + v\ \}\ ]
\end{aligned}
$$

Nous allons voir que cette dernière transformation n'est pas suffisante pour donner une opération de mutation qui permette la terminaison du processus de décomposition-fusion-mutation. En effet, certaines équations se généralisent mal par les résultats précédents, car la généralisation fait apparaître une autre équation aussi complexe.

Par exemple, l'équation $e = (x+u == x+v)$ où x est une variable et u un terme, se généralise (en prenant la définition de U' donnée plus haut) en $[\{u == v\}, \{x = z+v == z+u\}]$ où z est une nouvelle variable. En fait il suffit de remarquer que le symbole + est simplifiable à droite et à gauche pour résoudre ce premier problème. Cela découle des résultats suivants :

Lemme 26 : Pour tous termes u, v, w on a $u+v =_{Cd} u+w$ si et seulement si $v =_{Cd} w$ (+ est régulière à gauche).

Preuve: Si $v =_{Cd} w$, alors la conclusion est claire. Montrons la réciproque par récurrence sur la taille de u.

Si $|u| = 1$ alors aucune application de l'axiome Cd n'est possible en tête et par conséquent $v =_{Cd} w$.

Pour un terme u de taille $n > 1$, $u+v =_{Cd} u+w$ implique par les résultats précédents l'existence d'un terme t tel que $u =_{Cd} t+w$ et $u =_{Cd} t+v$ donc $u =_{Cd} t+w =_{Cd} t+w$. On peut appliquer l'hypothèse de récurrence sur cette dernière relation car $|w| < n$ puisque Cd est un axiome qui conserve la taille. Donc $v =_{Cd} w$. □

Lemme 27 : Pour tous termes u, v, w on a $u+v =_{Cd} w+v$ si et seulement si

$u =_{Cd} w$ (+ est régulière à droite).

Preuve: Si $u =_{Cd} w$ alors la conclusion est claire.

Sinon, en appliquant le lemme sur la structure des termes Cd-équivalents, soit $u =_{Cd} w$ auquel cas la preuve est terminée, soit il existe un terme t tel que $u =_{Cd} t+v$ et $w =_{Cd} t+v$ donc par transitivité, $u =_{Cd} w$. □

On en déduit les Cd-équivalences d'équations associées, pour tous termes u, v, w :

$$u+v == u+w \Leftrightarrow_{Cd} v == w$$
$$v+u == w+u \Leftrightarrow_{Cd} v == w$$

La simplifiabilité permet de résoudre le problème des équations du type x+u == x+v par exemple, mais ce n'est pas suffisant.

Exemple 38 : Soit l'équation (x+a)+d == (x+c)+b. Elle se généralise, d'après de que nous venons de voir en une disjonction de système qui contient S = { x+a == y+b, x+c == y+d }. Si on généralise indépendamment les deux équations de S, on obtient entre autre système :

S' = { x == x'+b, y == x'+a, x == y'+d, y == y'+c}, qui fusionne en

S' = { x == x'+b == y'+d, y == x'+a == y'+c }.

Et par conséquent la généralisation que nous avons utilisée ne convient pas puisque nous obtenons un système similaire au système S de départ.

L'exemple précédent est typique des théories A pour lesquelles la connaissance de l'ensemble des A-solutions de toute équation de la forme u+v == u'+v' ne permet pas de résoudre un système, comme cela est le cas par exemple pour les théories commutatives. C'est un phénomène analogue à celui rencontré pour les théories associatives commutatives. Nous allons voir

qu'il faut avoir davantage d'informations sur la structure profonde des termes à unifier ainsi que sur l'environnement dans lequel se fait cette unification, c'est à dire sur le système auquel appartient l'équation que l'on veut généraliser.

1.1. Une structure de terme Cd-aplatie

De même que dans le cas associatif commutatif il est naturel d'introduire une représentation aplatie des termes :

Définition 72 : On appelle représentation Cd-aplatie d'un terme u de symbole de tête + le triplet $\Delta(u) = (+, t, T)$ où t est un terme tel que $t(\varepsilon) \neq +$ et T un multi-ensemble de termes en représentation Cd-aplatie, tels que

* si $u = (((..((t+t_1)+t_2)+...+t_p)$ alors $T = \{\{r_1,...,r_p\}\}$ avec pour tout $k \in [1..p]$ $\Delta(t_k) = r_k$.
* si u = t alors $T = \emptyset$.

$\Delta(u)$ se notera plus facilement $t+\{\{t_1,...,t_p\}\}$. ○

Exemple 39 : Si u = ((a+g(x,y))+(a+x)) alors $\Delta(u)$ = (+, a, {{ g(x,y), (+,a,{{x}}) }}).

La Cd-égalité de 2 termes se traduit simplement sur les représentations Cd-aplaties :

Lemme 28 : Si $\Delta(u) = t+T$ et $\Delta(u') = t'+T'$ alors $u =_{Cd} v$ si et seulement si il existe un bijection π de T sur T' telle que $\forall r \in T$, $\pi(r) =_{Cd} r$.

Preuve: C'est une conséquence des résultats précédents. □

1.2. Le cas variable : résolution des équations de la forme z+Z == y+Y

Nous donnons d'abord l'ensemble complet de Cd-solutions d'une équation dans le cas "variable", c'est à dire pour les équations de la forme z+Z == y+Y où Z et Y sont des multiensembles de variables tels que $Z \cap Y = \emptyset$ et z, y des variables différentes, ce qui est toujours possible puisque + est un symbole régulier à droite et à gauche.

Il faut tout d'abord noter la différence fondamentale avec le cas associatif-commutatif : Z et Y sont des multiensembles dont il ne sera utile d'instancier les éléments que par des variables, du fait que + n'est pas associatif. Par exemple (x+((y+z)+u) ne pourra pas s'écrire x+y+z+u. Les seules variables que l'on pourra "utilement" instancier par des termes de la forme x+X dans l'équation z+Z == y+Y sont donc y et z. La recherche d'un ensemble complet de Cd-unificateurs pour l'équation z+Z == y+Y va donc nécessiter quelques définitions techniques permettant de décrire formellement la correspondance entre Y et Z.

Les relations que nous introduisons maintenant ont leur justification dans le fait qu'à toute Cd-solution α de l'équation z+Z == y+Y on peut associer la relation R^α sur D(Z)xD(Y) (où D(Z) désigne le domaine du multiensemble Z) telle que $z' \ R^\alpha \ y' \Leftrightarrow \alpha(z') =_{Cd} \alpha(y')$.

Soit R une relation quelconque sur D(Z)xD(Y). A chaque élément de Z et Y nous associons l'ensemble des éléments qui se correspondent de la manière suivante, soient :

$\tilde{z}^0 = \{z' \mid z' \in Z$ et $\exists\, y \in Y$ tel que $z \ R \ y$ et $z' \ R \ y\}$

$\tilde{z}^i = \{z' \mid z' \in Z$ et $\exists\, y \in Y$, $\exists\, z'' \in \tilde{z}_j$ avec $j < i$ tel que $z'' \ R \ y$ et $z' \ R \ y\}$

$$\tilde{z} = \bigcup_{i \in N} \tilde{z}^i$$

$\tilde{z}$ est l'ensemble des éléments de D(Z) qui sont connectés par le graphe de la relation R. De même on défini pour tout élément y de Y $\tilde{y}$. On déduit de R la relation $\tilde{R}$ définie par $\tilde{z} \ \tilde{R} \ \tilde{y}$ ssi il existe z_1 dans $\tilde{z}$ et y_1 dans $\tilde{y}$ tels que $z_1 \ R \ y_1$. Nous associons alors à chaque classe $\tilde{z}$ une "nouvelle" variable notée $nv(\tilde{z})$ et à chaque classe $\tilde{y}$ la variable $nv(\tilde{z})$ si il existe $\tilde{z}$ telle que $\tilde{z} \ \tilde{R} \ \tilde{y}$, sinon une nouvelle variable, dans les deux cas nous les notons $nv(\tilde{y})$. A chaque variable de $\tilde{z}$ (resp. $\tilde{y}$) nous associons la variable notée nv(z) égale à $nv(\tilde{z})$ (resp. nv(y) égale à $nv(\tilde{y})$).

A toute relation R sur D(Z)xD(Y) nous pouvons donc associer les substitutions

$$\xi_R^Z = \mathop{o}_{z_1 \in D(Z)} (z_1 \leftarrow nv(z_1))$$

$$\xi_R^Y = \mathop{o}_{y_1 \in D(Y)} (y_1 \leftarrow nv(y_1))$$

Nous avons maintenant les notations nous permettant de décrire un ensemble générateur de l'ensemble des Cd-solutions :

Lemme 29 :

$\Sigma = \{\xi_R^Z.\xi_R^Y.(z \leftarrow x+(\xi_R^Y(Y)-\xi_R^Z(Z))).(y \leftarrow x+(\xi_R^Z(Z)-\xi_R^Y(Y))) \mid \forall\, R \in D(Z)xD(Y)\}$

est un ensemble complet de Cd-unificateurs de l'équation z+Z == y+Y.

Preuve: Il est clair que chaque élément de Σ est un Cd-unificateur de z+Z == y+Y.

Montrons maintenant la complétude.

Soit α une Cd-solution de z+Z == y+Y. On associe à α la relation R_α sur ZxY telle que

$$\forall z \in Z, \forall y \in Y, x R_\alpha y <=> \alpha(x) =_{Cd} \alpha(y)$$

A cette relation R_α on associe un élément σ de Σ qui vérifie par définition $\sigma <_{Cd} \alpha [\{z,y\} \cup Z \cup Y]$, ce qui prouve la complétude.

□

1.3. Résolution des systèmes

Nous pouvons maintenant préciser l'opération de mutation en général.

Si S est un système indécomposable mais non complètement décomposé, composé de termes sur M({+},X) alors, la mutation de ce système est définie comme la disjonction de systèmes obtenue par résolution itérative de S. Rappelons que cela consiste à résoudre les équations les unes après les autres et à remplacer les valeurs trouvées dans les autres équations du système.

2. Conclusion

La théorie Cd(+) étant une théorie permutative, elle est stricte et fortement complète. Par ailleurs nous venons de montrer que si le seul symbole de l'algèbre est +, nous connaissons un algorithme de complète décomposition des systèmes. Par conséquent nous en déduisons un algorithme complet de Cd-unification dans M({+},X). Les résultats sur les mélanges de théories donnés dans la suite permettrons de conclure en général c'est à dire pour une famille quelconque de symboles satisfaisant l'axiome Cd.

CHAPITRE 6

SURREDUCTION ET SUPERREDUCTION

Nous allons étudier dans cette partie, de façon très générale, la surréduction et la superréduction.

La surréduction ou narrowing est une méthode générale de résolution d'équations dans une théorie équationnelle introduite par Slagles [Slagle1974], étudiée par Fay [Fay1978,Fay1979] et complètement étudiée par J.M. Hullot [Hullot1980] dans le cas de la Rø-réécriture. Les résultats de ce dernier ont été étendus au cas de la R,E-réécriture dans [Jouannaud1983].

L'approche que nous proposons dans cette partie permet de donner une description unifiée et simple des résultats connus à ce jour.
D'autre part les résultats nouveaux que nous obtenons sont les suivants :

* la preuve de correction et de complétude de la RE-surréduction pour toute relation de RE-réécriture qui soit confluente et noethérienne. Cela

étend en particulier les résultats les plus généraux dans le domaine qui avaient été obtenu dans [loc.cit],

* des méthodes complètes de réduction de l'arbre des surréductions :

  - la RE-surréduction basique qui étend les résultats de J.M.Hullot, pourvu que (R,E) satisfasse une condition de forte cohérence,

  - la détection de branches subsumées de l'arbre de RE-surdérivation,

* la description finie d'ensembles complets infinis de A-unificateurs obtenus par RE-surréduction. Ce qui permet de donner un algorithme décrivant un ensemble complet de A-unificateurs d'une équation, qui termine même si l'ensemble à décrire est infini. Ceci sous la condition que l'arbre de RE-surréduction soit rationnel.

Dans toute la suite de cette section, nous considérons l'ensemble des axiomes A partitionné en E et R, R étant utilisé comme système de RE-réécriture ainsi que nous l'avons définie dans les généralités.

Le symbole == utilisé pour décrire les équations est considéré dans cette partie 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.

1. La RE-surréduction en tant qu'opération sur les ensembles de A-solutions d'une équation

Nous allons voir dans cette section en quel sens la RE-surréduction conserve l'ensemble des A-solutions d'une équation.

Tout d'abord rappellons ce qu'est la surréduction et introduisons la superréduction.

Définition 73 : Soit A une théorie, (R,E) un système de réécriture équationnel tel que A = RUE. 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 -^->RE[σ,m,g→d] t' ssi σ appartient à $Unif_E(t_{|m}, g)$ et t' = σ(t[m←d]).

σ s'appelle **substitution RE-surréductrice** de t vers t', RE pourra être omis. 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 $Unif_E$ pourront être implicites. ○

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 [Fay1979] la **RE-superréduction** qui combine une étape de RE-surréduction et la mise en RE-forme normale du terme surréduit. Formellement,

Définition 74 : Soit A une théorie, (R,E) un système de réécriture équationnel tel que A = EUR. On note -^^->RE la **superréduction** associée à R et à $Unif_E$ définie par :

t -^^->RE[σ,m,g→d] t' <=> σ ∈ $Unif_E(t_{|m}, g)$ et t' = $\sigma(t[m\leftarrow d])\downarrow_{RE}$.

σ s'appelle **substitution superréductrice** de t vers t'. On note Superred(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. ○

Nous introduisons maintenant les notations appropriées à la manipulation des ensembles de A-solutions.

Définition 75 : Σ et Σ' étant des ensembles de substitutions et α une substitution, l'ensemble $\Sigma.\alpha$ est défini par $\Sigma.\alpha = \{\sigma.\alpha \mid \sigma \in \Sigma\}$.

On dit que Σ est **E-inclus** dans Σ' ssi

$$\forall\, \sigma \in \Sigma,\ \exists\, \sigma' \in \Sigma' \text{ tel que } \sigma =_E \sigma'$$

Σ et Σ' sont **E-égaux** ssi ils sont E-inclus l'un dans l'autre. ○

Exemple 40 : Si on prend pour E la théorie minus on a

$\{x\} =_E \{-(-x),\ -(-(-(-x)))\}$

Notation : Une méthode de calcul des E-unificateurs $Unif_E$ étant déterminée, soient

$$SURES(t,t',A) = \bigcup_{\sigma \in Surred(t==t',R,Unif_E)} ES(t_1,t_1',A).\sigma$$

$$SUPERES(t,t',A) = \bigcup_{\sigma \in Superred(t==t',R,Unif_E)} ES(t_1,t_1',A).\sigma$$

Exemple 41 : Si on prend $E = \emptyset$, $R = \{x+x \rightarrow x\}$ et $Unif_\emptyset$ comme méthode d'unification alors

$$SURES(y+(x+a)==a,A) = ES(x+a==a,A).(y \leftarrow x+a) \cup ES(y+a==a,A).(x \leftarrow a)$$

Le résultat suivant permet de s'assurer que la surréduction ne fait pas sortir de l'ensemble des A-solutions.

Lemme 30 : SURES(t,t',A) est inclus dans ES(t,t',A)

Preuve: Il faut montrer que pour toute application $Unif_E$, toute substitution surréductrice σ de Surred(t==t', R, $Unif_E$) et tout élément α de $ES(t_1,t_1',A)$, alors $\alpha.\sigma$ est une A-solution de t==t'. Or par définition de ces deux substitutions on a

$$\alpha\sigma(t) \rightarrow_{RE} \alpha(t_1) =_A \alpha(t_1') \ {}_{RE}\!\leftarrow \alpha\sigma(t')$$

et par conséquent $\alpha\sigma$ est une A-solution de t==t' puisque $\rightarrow_{RE}$ est inclus dans $=_A$. □

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 [Hullot1980] pour $E = \emptyset$ et la Rø-réécriture, et [Jouannaud1983] pour la surréduction associée à la réécriture de Peterson et Stickel.

Proposition 11 : Soit t_0 un terme et ρ une substitution RE-normalisée telle que $\rho(t_0) \rightarrow_{RE[m,g \rightarrow d]} t_1'$.

Alors il existe des substitutions σ et μ telles que :

* $t_0 \text{ -^->}_{RE[m,g \rightarrow d,\sigma]} t_1$
* $\mu(t_1) =_E t_1'$
* $D(\mu) = Var(t_1)$
* $\rho =_E \mu\sigma\ [Var(t_0)]$

Ce qu'on peut schématiser par

$$\begin{array}{ccc} t_0 & \text{-^->}_{RE[m,g\rightarrow d,\sigma]} & t_1 \\ \rho \big\downarrow & & \big\downarrow \mu \\ \rho(t_0) & \rightarrow_{RE[m,g\rightarrow d]} & t_1' \end{array} \qquad =_E$$

Preuve: Montrons d'abord que t_0 est RE-surréductible. $\rho(t_0)$ étant RE-réductible à l'occurrence m par la règle $g \rightarrow d$ que l'on supposera telle que $Var(g) \cap Var(\rho(t_0)) = \emptyset$, il existe une substitution γ telle que

$$\gamma \in Filtre_E(g, \rho(t_0)_{|m})$$

mais ρ étant RE-normalisée et $Var(g) \cap Var(\rho(t_0)) = \emptyset$,

$$\gamma \in Filtre_E(g, \rho(t_{0|m})) \subseteq UNIF_E(g, \rho(t_{0|m}))$$

Par la condition (3) de la définition de UNIF on a,

$$\gamma\rho \in UNIF_E(g, t_{0|m})$$

et γ et ρ pouvant être choisies de telle sorte que leurs domaines soient disjoints, on a $\gamma\rho = \gamma+\rho$.

Par conséquent il existe σ dans $Unif_E(g, t_{0|m})$ et une substitution δ telles que

$$\delta.\sigma =_E \gamma+\rho \ [Var(g) \cup Var(t_0)]$$

et t_0 est bien RE-surréductible

$$t_0 \text{ -^->RE}[m, g \rightarrow d, \sigma] \ t_1 = \sigma(t_{0[m \leftarrow d]})$$

Définissons maintenant μ par $\mu = \delta_{|Var(t_1)}$. On a bien par définition de ces objets

$$\rho =_E \mu.\sigma \ [Var(t_0)]$$

Montrons enfin que $\mu(t_1) =_E t_1'$. En effet,

$\mu(t_1) = (\mu.\sigma)(t_{0[m \leftarrow d]})$

$= \mu.\sigma(t_0)_{[m < -\mu\sigma(d)]}$

$=_E \rho(t_0)_{[m < -\rho(d)]} =_E t_1'$ □

Lemme 31 : Si A = (R, E) est RE-Church-Rosser alors ES(t,t',A) est A-inclus dans ES(t,t',E) ∪ SURES(t,t',A).

Preuve: Soit ρ la RE-normalisée de α.

Si $\rho(t) =_E \rho(t')$ la conclusion est claire puisque $\alpha =_A \rho$ et que ρ est dans ce cas une E-solution de t==t'.

Sinon, il faut montrer que pour toute A-solution α de t==t' il existe une substitution σ qui surréduit t==t' en $t_1 == t_1'$ et une A-solution μ de $t_1 == t_1'$ telle que $\alpha =_A \mu\sigma$.

Si $\rho(t) =_A \rho(t')$ sans qu'il y ait E-égalité, puisque le couple (R,E) est supposé RE-Church-Rosser, au moins l'un des deux termes $\rho(t)$ ou $\rho(t')$ doit être RE-réductible. C'est à dire que

$$\rho(t) == \rho(t') \rightarrow_{RE} t_0 == t_0'.$$

On peut donc appliquer la propriété précédente sur le terme $\rho(t) == \rho(t')$. Il existe donc une substitution normalisée μ telle que

$$t == t' \text{ -^->RE } t_1 == t_1'$$

avec $\mu(t_1) =_E t_0$ et $\mu(t_1') =_E t_0'$ et $\alpha =_A \rho =_E \mu\sigma$. Comme de plus on a

$$\mu(t_1) =_E t_0 =_A \rho(t) =_A \rho(t') =_A t_0' =_E \mu(t_1')$$

μ est une A-solution de $t_1 == t_1'$. □

On peut donc déduire des résultats précédents :

Théorème 14 : Si A = (R, E) est RE-Church-Rosser alors

$$ES(t==t',A) =_A ES(t==t',E) \cup SURES(t==t',A)$$

Par conséquent les A-solutions d'une équation e sont obtenues soit comme E-solution de e soit comme le 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 et par conséquent permet de mettre en oeuvre les résultats généraux obtenus précédement. Par exemple, la réduction des termes des équations surréduites utilisée par Fay et appelée narrowing dans [Rety1985] se justifie de façon très simple dans notre formalisme puisque nous avons vu que toute relation de réécriture conserve l'ensemble des A-solutions d'un unificande.

Théorème 15 : Si A = (R, E) est RE-Church-Rosser alors

$$ES(t==t',A) =_A ES(t==t',E) \cup SUPERES(t==t',A)$$

Plus généralement, on peut appliquer les transformations qui conservent les ensembles de A-solutions sur les équations surréduites : décomposition, fusion, ou autre mais aussi celles qui généralisent, donc toute opération de mutation.

Par ailleurs ces résultats permettent de justifier une stratégie de surréduction ou de superréduction où on arrête de surréduire un terme dès que l'on connait non pas un ensemble complet de E-solutions mais un ensemble complet de A-solutions, comme nous l'explicitons dans l'exemple suivant.

Réciproquement, le résultat précédent permet de considérer la surréduction comme une opération de mutation.

Il faut également noter que ces résultats restent valables si on se limite aux substitutions surréductrices (ou superréductrices) RE-normalisée.

2. Surréduction et unification

Nous allons appliquer les résultats précédents à la recherche d'ensemble complets de A-solutions d'une équation donnée.

Corollaire 5 : Soit A une théorie telle que le couple (R,E) soit RE-Church-Rosser et la RE-réécriture noetherienne. Soit Σ l'ensemble des substitutions σ telles qu'il existe une RE-surdérivation issue de t==t'

$$t==t' \text{ -^->}_{RE[\sigma_0]} t_1==t_1' \text{ -^->} \ldots t_{n-1}==t_{n-1}' \text{ -^->}_{RE[\sigma_{n-1}]} t_n==t_n'$$

avec t_n et t_n' E-unifiables, $\sigma = \rho.\sigma_{n-1}\ldots\sigma_0$ et ρ dans $ECU(t_n==t_n',E)$. Alors Σ est un ensemble complet de A-unificateurs de t==t'.

Preuve: Si σ est un élément de Σ 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

$$\alpha =_A \beta.\sigma_{n-1}\ldots\sigma_0$$

avec $\beta \in ES(t_n==t_n',E)$. Si ce n'était pas le cas par application de la proposition précédente, on aurait, en posant

$$\sigma = \prod_{i=1}^{\infty} \sigma_i$$

$\sigma(t==t')$ réductible une infinité de fois, ce qui contredit la noethérianité de $\rightarrow_{RE}$.

Soit ρ un élément de $ECU(t_n==t_n', E)$ tel que $\rho \leq_E \beta \ [Var(t_n==t_n')]$. On a donc

$$\rho.\sigma_{n-1}\cdots\sigma_0 \leq_E \beta.\sigma_{n-1}\cdots\sigma_0 =_A \alpha$$

Ce qui prouve que Σ est un ensemble complet de A-solutions de $t==t'$. □

Ce résultat permet de construire une procédure complete de A-unification pour une équation $t==t'$: il suffit en effet de construire progressivement les surdérivations issues de $t==t'$. Une telle procédure ne terminera pas nécessairement, nous en donnerons des exemples dans le paragraphe suivant. Par ailleurs si elle termine, l'ensemble des A-unificateurs retourné n'est pas forcément minimal.

Exemple 42 : Soit R le système de réécriture confluent :

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

et $E = \emptyset$.

Soit $e=(x==-y)$, un ensemble minimal complet de A-unificateurs est $\{(x \leftarrow -y)\}$. Par la procédure issue du corollaire précédent, on obtient, outre la solution $(x \leftarrow -y)$ qui correspond à la E-solution de $t==t'$, la solution $(y \leftarrow -x)$ obtenue par :

$x==-y \ -\hat{}->[1, y \leftarrow -x] \ x == x$

et la solution $(x \leftarrow (-x_2)+(-x_1))$ $(y \leftarrow x_1+x_2)$ obtenue par :

$x==-y \ -\hat{}->[2, y \leftarrow x_1+x_2] \ x == (-x_2)+(-x_1)$.

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 é 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 é.

Dans l'objectif d'étudier des améliorations de la procédure précédente nous allons introduire formellement la notion d'arbre de surdérivation.

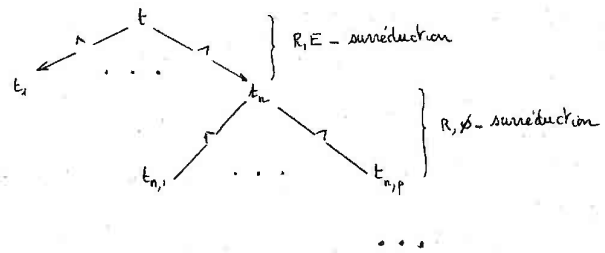
Définition 76 : On appelle **arbre de surdérivation** associé à un terme t (qui peut être une équation) l'arbre noté Arb_sur(t) défini par

- $Arb\_sur(t)(\varepsilon) = t$
- Si $Arb\_sur(t)(m) = t'$ alors $Arb\_sur(t)(m.i) = t''$ pour σ_i dans $Surred(t')$ et $t' \ -\hat{}->[\sigma_i] \ t''$. ○

Un arbre de superdérivation se défini de façon analogue.

Remarque : Il est équivalent de se donner l'occurrence d'un noeud t' dans $Arb\_sur(t)$ et la suite des substitutions surréductrices permettant de surréduire t en t'.

Remarque : Les résultats précédents permettent de faire des surdérivations "hétérogènes" en ce sens où il n'est pas nécessaire de faire le même type de surréduction à chaque étape de surréduction, à condition qu'à une profondeur donnée dans l'arbre de surdérivation le même type de surréduction soit utilisé.



Un arbre de surréduction hétérogène

3. Réduction de l'arbre de surdérivation

La surréduction comme la superréduction permettent de construire de façon systématique des algorithmes de A-unification mais de graves défauts entachent cette généralité. Ce sont la non terminaison du processus de surréduction et la redondance des solutions trouvées. Il faut donc réduire et/ou factoriser l'arbre de surréduction sans perdre la complétude du processus qui doit toujours fournir, s'il termine, un ensemble complet de A-solutions pour l'équation considérée. Nous allons voir trois améliorations qui vont dans ce sens.

D'une part la surréduction basique introduite par Hullot [Hullot1980] qui permet de réduire l'arbre de surdérivation en enlevant les branches qui correspondent en fait à des surréductions sur des sous termes provenant de parties de substitutions appliquées à une étape précédente. Nous généralisons cette approche aux réécritures équationnelles.

D'autre part la factorisation de certaines branches infinies rationnelles de l'arbre de surdérivation. Cela permettra de décrire finiment des ensembles infinis de A-unificateurs et éventuellement, si l'arbre de

surdérivation est infini rationnel, de décrire finiment des ensembles complets infinis de A-unificateurs.

Enfin, la suppression des branches subsumées de l'arbre de surdérivations c'est-à-dire des branches qui sont en quelle sorte instance d'une autre branche, permettra également de réduire la taille de l'arbre de surdérivation sans perdre la complétude des ensembles de A-solutions trouvés.

Nous supposerons dans toute la suite que A est une théorie telle que le couple (R,E) soit RE-church-rosser et à terminaison finie.

3.1. La surréduction basique

La surréduction basique a été définie et étudiée par J.M. Hullot [loc.cit.] pour la surréduction simple (E = ∅) puis généralisée à la surréduction de Peterson et Stickel (R,E-surréduction) dans un travail en collaboration avec J.P.Jouannaud et H.Kirchner [Jouannaud1983].

L'approche que nous proposons ici va nous permettre de définir et de prouver la complétude de la surréduction basique pour une classe de relations de surréduction contenant la surréduction simple et la R,E-surréduction.

L'idée directrice de la surréduction basique prend naissance dans la correspondance entre réduction et surréduction exprimée dans le lemme @@@. Ce lemme met également en évidence l'existence de E-unificateurs pour l'équation surréduite si l'équation réduite correspondante est en A-forme normale modulo E. Une "bonne" stratégie de surréduction va donc consister à calquer une méthode de recherche de la forme normale utilisant une

stratégie correcte. Par exemple réécriture de l'extérieur vers l'intérieur ou de l'intérieur vers l'extérieur ou encore réécriture n'utilisant que des filtres normalisés.

Exemple 43 : Si on considère le système de réécriture

$h(h(x)) \rightarrow x$

$h(a) \rightarrow a$

on a alors en suivant une stratégie de l'intérieur vers l'extérieur

$$h(h(a)) \rightarrow h(a) \rightarrow a$$

et en suivant une stratégie de réduction à filtre normalisé en plus de la réduction précédente on a la possibilité :

$$h(h(a)) \rightarrow [(x \leftarrow a)]\ a$$

La stratégie de l'intérieur vers l'extérieur est donc plus intéressante car moins générale que la stratégie à filtre normalisé tout en restant complète. De plus, chose très importante ici, l'ensemble des occurrences de réductions licites au ième pas de réduction M_i pour la stratégie de l'intérieur vers l'extérieur est fonction de l'ensemble des occurrences licites pour la (i-1)ème réduction, de l'occurrence de réduction m et de la règle utilisée $g \rightarrow d$:

$$M_i = (M_{i-1} - \{m.m' \mid m.m' \in M_{i-1}\}) \cup \{m.m'' \mid m'' \in O(d)\}$$

La surréduction basique va utiliser cet ensemble d'occurrences s'il existe une bonne correspondance entre les occurrences d'application d'une règle dans la réduction et l'occurrence d'application de la surréduction associée. Dans ce cas, la surréduction sera guidée de façon syntaxique

(donc efficace) dans les termes à surréduire.

Définissons maintenant la surréduction basique.

3.1.1. Définitions

Définition 77 : Une RE-réécriture étant choisie, on dit qu'une RE-dérivation

$$t_0 \rightarrow_{RE[m_0,\sigma_0]} t_1 \rightarrow_{RE[m_1,\sigma_1]} \ldots \rightarrow_{RE[m_n,\sigma_n]} t_{n+1}$$

suit une **stratégie à filtre normalisé** ssi chaque filtre utilisé σ est RE-normalisé.

Cette même dérivation suit une **stratégie de l'intérieur vers l'extérieur** ssi à chaque réduction $t_k \rightarrow_{RE[m_{k+1},\sigma_{k+1}]} t_{k+1}$ de la RE-dérivation considérée, il n'existe pas de radical à une occurrence plus grande que m_{k+1}. ○

Définition 78 : Soit t_0 un terme, M_0 un ensemble d'occurrence de $O(t_0)$ clos par préfixe et ρ une substitution RE-normalisée. La RE-dérivation

$$\rho(t_0) \rightarrow_{RE[m_0,k_0]} t_1' \rightarrow_{RE} \ldots \rightarrow_{RE[m_n,k_n]} t_{n+1}'$$

et la RE-surdérivation

$$t_0 \text{ -^->}_{RE[m_0,k_0]} t_1 \text{ -^->}_{RE} \ldots \text{ -^->}_{RE[m_n,k_n]} t_{n+1}$$

sont basées sur M_0 si et seulement si pour tout i dans $[1..n]$, m_i appartient à M_i avec

$$M_{i+1} = (M_i - \{m_i.m' \mid m_i.m' \in M_i\}) \cup \{m_i.m'' \mid m'' \in O(d_{k_i})\}$$

M_{i+1} sera aussi noté **Ocbas**(M_i,m,k_i). La RE-dérivation ou RE-surdérivation

est dite **basique** si $M_0 = O(t_0)$. ○

Exemple 44 : Si nous considérons le même système de réécriture que dans l'exemple précédent, la surdérivation suivante est basique

$$h(y) + y \text{ -^->}[1,(y \leftarrow h(x))]\ h(x) + h(x) \text{ -^->}[1,(x \leftarrow a)]\ h(a) + a$$

alors que la suivante ne l'est pas :

$$h(y) + y \text{ -^->}[1,(y \leftarrow h(x))]\ h(x) + h(x) \text{ -^->}[2,(x \leftarrow a)]\ a + h(a)$$

Nous pouvons maintenant introduire les ensembles de A-solutions calculés par surréduction basique.

Définition 79 : Soient t et t' deux termes, M un ensemble d'occurrences, on note

$$SURES(t,t',A,M) = \bigcup_{\sigma \in Surred(t==t',R,Unif_E,M)} ES(t_1,t_1',A).\sigma$$

où $Surred(t==t',R,Unif_E,M)$ est l'ensemble des substitutions surréductrices issues de t==t' pour la méthode d'unification $Unif_E$ et l'ensemble de règles R, l'unification se faisant à une occurrence de M. R et $Unif_E$ pourront être implicites dans la notation ci dessus.
L'ensemble des A-solutions de t==t', trouvées par RE-surréduction basée sur M est notée BU(t==t',A,M) on a donc

$$BU(t==t',A,M) = ES(t==t',E) \cup \bigcup_{\sigma \in Surred(t==t',M)} BU(t==t',A,Ocbas(M,m,k)).\sigma$$

○

Nous allons chercher sous quelles conditions BU(t==t',A,O(t==t')) est l'ensemble des A-solutions de t==t'.

3.1.2. Complétude de la surréduction basique

Définition 80 : M étant un sous ensemble de O(t==t') clos par préfixe, soit **SU-N(t==t',A,M)** le sous ensemble de l'ensemble des A-solutions normalisées de t==t' défini par :
SU-N(t==t',A,M) = {σ | σ ∈ ES(t==t',A) et il existe une RE-dérivation basée sur M de σ(t==t') vers sa forme normale}. ○

Remarque : Suivant les valeurs de M, SU-N(t==t',A,M) peut être vide ou non. Si t==t' a des A-solutions alors SU-N(t==t',A,O(t==t')) est non vide car si σ est une A-solution RE-normalisée de t==t', il existe une RE-dérivation de l'intérieur vers l'extérieur de σ(t==t') vers sa RE-forme normale en vertu du lemme qui suit.

Lemme 32 : Soient t un terme, ρ une substitution normalisée et t' = ρ(t). Alors toute RE-dérivation de l'intérieur vers l'extérieur issue de t' est basique et il existe une dérivation basique allant de t' vers sa RE-forme normale.

Preuve: La première assertion découle des définitions. La seconde provient du fait qu'il existe toujours une RE-dérivation de t' vers sa RE-forme normale, suivant une stratégie de l'intérieur vers l'extérieur. Il suffit alors d'appliquer la première assertion. □

Pour prouver la complétude de la surréduction basique nous nous limitons aux théories qui vérifient la propriété de **forte cohérence** qui assure une correspondance entre les occurrences d'application d'une RE-réduction et celles d'une RE-surréduction associée.

Définition 81 : On dit qu'une théorie A = (R,E) est **fortement RE-cohérente**

si elle RE-confluente et si pour tout terme t, tout sous ensemble M de O(t) clos par préfixe, toute RE-dérivation issue de t basée sur M et tout terme $u =_E t$ tel que dans cette E-égalité les occurrences d'applications d'axiomes soient plus grande que les éléments de M alors il existe une RE-dérivation issue de u et basée sur M. ○

Exemple 45 :

* Toute théorie R,E-Church-Rosser est fortement cohérente comme cela est prouvé dans [Jouannaud1983].

* Pour la $R_1 \cup R_{nl}$,E-réécriture, il n'y a pas de résultat général comme pour la R,E-réécriture car dans ce cas l'application d'un axiome dans une règle peut modifier la réductibilité du terme. Il est par contre simple de vérifier que le système de $R_1 \cup R_{nl}$,E-réécriture découvert par REVEUR3 dans le cas des groupes abélien, par exemple, est bien fortement cohérent.

Lemme 33 : Soit A = (R,E) une théorie fortement RE-cohérente telle que le couple (R,E) soit Church-Rosser, la RE-réécriture noetherienne, t et t' deux termes et M un sous ensemble de O(t==t'). Alors SU-N(t==t',A,M) est E-inclus dans BU(t==t',A,M).

Preuve: Par récurrence noethérienne en utilisant l'ordre bien fondé induit par la relation de RE-réécriture.

Soit ρ un élément quelconque de SU-N(t==t',A,M). Si ρ(t==t') est RE-normalisé alors ρ est un E-unificateur de t et t', c'est donc par définition un élément de BU(t==t',A,M).

Sinon, ρ(t == t') est RE-réductible en sa forme normale par une dérivation basique dont la premier pas est tel qu'il existe μ tel que,

$\rho(t==t') \rightarrow_{RE[m,k]} t_0 == t_0'$,

$t==t' \text{ -}\hat{}\text{->}_{RE[m,k,\sigma]} t_1==t_1'$,

$\mu(t_1==t_1') =_E t_0 == t_0'$,

$\rho =_E \mu.\sigma$

ce qu'on peut résumer par le diagramme suivant :

$$\begin{array}{ccc} \rho(t) =_A \rho(t') & \rightarrow_{RE[m,k]} & t_0 == t_0' \\ \uparrow \rho & & =_E \uparrow \mu \\ t == t' & \text{-}\hat{}\text{->}_{RE[m,k,\sigma]} & t_1 == t_1' \end{array}$$

La RE-dérivation de $t_0==t_0'$ vers sa RE-forme normale est donc basée sur M'=Ocbas(M,m,k). La théorie étant supposée strictement cohérente, $\mu(t_1==t_1')$ se réduit également en sa RE-forme normale par une RE-dérivation basée sur M'. De plus $\mu(t_1==t_1')$ est un fils pour le bon ordre induit par la relation de RE-réécriture de ρ(t==t'). En appliquant l'hypothèse de récurrence, μ qui est un élément de SU-N($t_1==t_1'$,A,M') appartient donc aussi à BU($t_1==t_1'$,A,M'). Comme $\rho =_E \mu.\sigma$, ρ est donc E-égale à un élément de BU(t==t',A,M). □

Si on applique ce résultat pour M = O(t==t') il vient

$$SU\text{-}N(t==t',A,O(t==t')) \subseteq_E BU(t==t',A,O(t==t')) \subseteq ES(t==t',A) \subseteq_A SU\text{-}N(t==t',A,O(t==t'))$$

ce qui s'énonce :

Corollaire 6 : Pour toute théorie A fortement cohérente BU(t==t',A,O(t==t')) =A ES(t==t',A).

De la même façon que pour la surréduction simple on en déduit une manière de calculer un ensemble complet de A-solutions d'une équation t==t'.

3.1.3. Une condition suffisante de terminaison de la surréduction

La surréduction basique permet de réduire l'espace des surréductions mais en général elle ne termine pas non plus. Nous donnons maintenant une condition suffisante qui généralise celle donnée par J.M. Hullot pour la Rø-surréduction.

Proposition 12 : Si A est une théorie tel que le couple (R,E) soit Church-Rosser, fortement cohérent et noethérien et telle qu'il existe un algorithme fini de E-unification. Si toute RE-surdérivation basique issue d'un membre gauche d'un élément de R termine alors toute surdérivation basique issue d'un terme quelconque termine.

Preuve: C'est essentiellement la même que celle de [Hullot1980] □

Exemple 46 : Le résultat précédent s'applique facilement si les membres droits des règles sont réduits à une variable. La surréduction basique fournie donc un algorithme d'unification fini pour les théories suivantes :

* L'idempotence : $x + x \rightarrow x$
* les quasi groupes [loc.cit.],
* les arbres binaires signés [Kirchner1982],
* Commutativité et idempotence,
* Associativité-commutativité idempotence

etc ...

3.2. Factorisation de l'arbre de surdérivation

Dans cette section nous montrons comment factoriser les branches infinies rationnelles de l'arbre de surdérivation. L'exemple suivant d'un tel arbre infini est inspiré par [Fages1983].

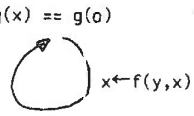
Exemple 47 : On considère le système de réécriture R suivant

(1) $g(f(x,y)) \rightarrow g(y)$

E = ø et l'équation e=(g(x) == g(o)). On a par conséquent

$$e = (g(x) == g(o)) \quad -\hat{}->[1,\ x \leftarrow f(x_1,x_2)] \quad é = (g(x_2) == g(o))$$

é est égale à e à un renomage des variables près, Arb_sur(e) est donc un arbre infini rationnel (au renomage des variables près). On peut le représenter par



Nous allons montrer comment décrire finiment l'ensemble des A-solutions de e issues d'une telle branche infinie.

3.2.1. Caractérisation des branches infinies rationnelles de l'arbre de surdérivation

Dans ce qui suit nous considérons 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 de [1..n]. Nous appellerons **noeuds de succès** de Arb_sur(e_0) les noeuds étiquetés par une équation e telle que ECU(e,E) ne soit pas vide.

Nous définissons maintenant les objets qui vont nous être nécessaires pour décrire finiment l'arbre de surdérivation.

Soit **Loop_Eq** l'ensemble des équations e de Arb_sur(e_0) dont est issue une boucle :

$$Loop\_Eq = \{ e \mid e \text{ -+^-> } e \quad e_0 \text{ -*^-> } e\}$$

Soit Fin_i l'ensemble des substitutions qui étiquetent un chemin sans boucle et qui termine sur une équation qui a un ensemble de E-unificateurs non vide :

$Fin_i = \{ \sigma \mid e_i \text{ -^->}[a_1] e_{i_1} \text{ ... -^->}[a_n] e_{i_n}$

tel que

* $e_{i_n} = (t == t')$ avec $T=ECU(t==t',E) \neq \emptyset$
* $\forall k$ dans $i_1, \ldots, i_n$, e_k n'appartient pas à Loop_Eq
* $\sigma = \tau.a_n \ldots a_1$ avec $\tau \in T.\}$

Un élément de Fin_i est donc un chemin issu du noeud étiqueté par ei, 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.

$Loop_p = \{ \sigma \mid \sigma = \sigma_k \ldots \sigma_1$

tel qu'il existe une surdérivation

* $e_0 \text{ -^*-> } e_p$
* $e_p \text{ -^->}[\sigma_1] e_{p_1} \text{ -^->}[\sigma_2] \ldots \text{ -^->}[\sigma_k] e_{p_k} = e_p$
* $\forall$ i et j dans $1,\ldots,k$, $i \neq j \Rightarrow e_{p_i} \neq e_{p_j}$ (pas de boucle)$\}$

Un élément de $Loop_p$ correspond donc à un chemin décrivant une boucle issue du noeud étiqueté par e_p.

Soit $Prem_p$ l'ensemble des substitutions obtenues comme la première partie

d'une surdérivation issue de e_0, sans contenir de boucle :

$Prem_p = \{ \sigma \mid$ il existe une surdérivation

$$e_0 \text{ -^->}[\sigma_1] e_1^0 \ldots e_{n-1}^0 \text{ -^->}[\sigma_n] e_n^0$$

telle que

* $e_n^0 = e_p$
* $e_p \in Loop\_Eq$
* $\sigma = \sigma_n \ldots \sigma_1$
* $\forall$ i et j dans $1,\ldots,n$, $i \neq j \Rightarrow e_i^0 \neq e_j^0$ (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, $Inter\_loop_{pq}$ décrit les substitutions permettant de passer d'une boucle à une autre :

$Inter\_loop_{pq} = \{\sigma \mid$ il existe une surdérivation

$$e_p \text{ -^->}[\sigma_1] e_1^p \text{ -^->}[\sigma_2] \ldots \text{ -^->}[\sigma_k] e_k^p$$

avec

* $e_q = e_k^p$
* e_p et e_q appartenant à Loop_Eq
* $p \neq q$
* $\forall$ i et j dans $1,\ldots,k$, $i \neq j \Rightarrow e_i^p \neq e_j^p$ (pas de boucle)$\}$

Un élément de $Inter\_loop_{pq}$ 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 Σ_i les ensembles associés à chaque équation origine d'une boucle définis récursivement comme suit.

* Σ_i contient $Prem_i$,

* Pour toute substitution β dans $Loop_i$ et α dans Σ_i, $\beta.\alpha$ appartient à Σ_i,

* Pour tout j dans $\mathbb{N}$ et toute substitution γ dans $Inter\_Loop_{ji}$ et α dans Σ_i, $\alpha.\gamma$ appartient à Σ_i.

Un élément de Σ_i correspond donc à un chemin possible de la racine de l'arbre de surdérivation jusqu'à un noeud étiqueté par e_i, après avoir éventuellement bouclé à des noeuds étiquetés par un e_j.

3.2.2. Définition récursive des A-solutions

Soit Σ l'ensemble des substitutions défini par :

* $Fin_0 \subseteq \Sigma$,

* Pour toute substitution α in Σ_i et λ dans Fin_i, $\lambda.\alpha$ appartient à Σ.

En d'autres termes un élément de Σ est issu, ou bien d'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 par un chemin débouchant sur un noeud de succès.

Théorème 16 : Σ est une 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 Σ 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, par un E-unificateur de e_n.

Réciproquement, si $\sigma = \tau.\alpha_n...\alpha_1$ est une substitution obtenue par la surdérivation

$$e_0 \text{ -}\hat{}*\text{->}[\alpha_1]\ e_1 \text{ -}\hat{}*\text{->}[\alpha_2]\ e_2 \ \ldots\ \text{-}\hat{}*\text{->}[\alpha_n]\ e_n = (t_n == t_n')$$

avec τ élément de $Unif_E(e_n)$ et $\alpha_1,\ldots,e_n$ toutes les équations de cette surdérivation d'où est issue une boucle (i.e. élément de Loop_Eq). On vérifie facilement que chacune des substitutions α_i appartient à l'un des ensembles $Prem_i$, Fin_i, $Loop_i$ ou $Inter\_loop_{ij}$. Donc σ appartient à Σ. □

Pour un arbre de surdérivation rationnel la description que nous venons de donner de Σ est finie, on peut donc modifier la procédure de surréduction de telle sorte qu'elle termine si l'arbre de surdérivation est rationnel. Il s'agit donc de détecter des boucles éventuelles dans l'arbre de surdérivation. Il suffit pour cela de garder l'histoire de la formation de l'arbre. On obtient dans ce cas un algorithme qui retourne une définition récursive finie des A-solutions de l'équation e_0.

3.2.3. Exemple de définition récursive des A-solutions

Si nous reprenons l'exemple donné au début, l'équation g(x) == g(o) n'est surréductible que par la règle (1) et par conséquent

* Pour tout i dans $\mathbb{N}$, $Fin_i = Fin_0 = \{(x \leftarrow 0)\}$

* L'ensemble Σ des A-solutions de l'équation trouvée ici sera donc récursivement défini par

- $\{(x \leftarrow o)\} \subseteq \Sigma$
- si $(x \leftarrow t) \in \Sigma$ alors $(x \leftarrow f(y,t)) \in \Sigma$.

3.3. 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 48 : Considérons le système de réécriture confluent

$$x + x \rightarrow x$$
$$(a + x) + (x + a) \rightarrow a + x$$

et l'équation $(x + y) + (y + x) = u$ où u est un terme quelconque.

On a parmi d'autres superdérivations

$(x + y) + (y + x) == u$ -^^->$[y \leftarrow x]$ $x == u$

$(x + y) + (y + x) == u$ -^^->$[x \leftarrow a]$ $a + y == u$ -^^->$[y \leftarrow a]$ $a == u$

La seconde superdérivation est une instance de la première en ce sens où $a == u$ est une instance de $x == u$ et $(y \leftarrow x) \leqq (x \leftarrow a)(y \leftarrow a)$. 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 $a == u$.
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 la condition que les chemins de l'arbre de surdérivation qui mènent à chacune de ces équations soient également instances l'un de l'autre.

Théorème 17 : Soient e_0, e_1 et e_2 trois équations telles que e_0 -*^->$[\sigma_1]$ e_1 et e_0 -*^->$[\sigma_2]$ e_2. Si il existe β telle que $\beta(e_1) =_A e_2$ et $\beta\sigma_1 =_A \sigma_2$ alors $ES(e_2,A).\sigma_2$ est A-inclus dans $ES(e_1,A).\sigma_1$

Preuve: Soit $\alpha \in ES(e_2,A)$, $\alpha(t_2) =_A \alpha(t_2')$ d'où

$$\alpha\beta(t_1) =_A \alpha(t_2) =_A \alpha(t_2') =_A \alpha\beta(t_1')$$

et par conséquent $\alpha\beta$ est une A-solution de e_1 donc

$$ES(e_2,A).\beta \subseteq ES(e_1,A)$$

ce qui compte tenu de la seconde hypothèse implique

$$ES(e_2,A).\sigma_2 =_A ES(e_2,A).\beta.\sigma_1 \subseteq ES(e_1,A).\sigma_1$$

□

Une conséquence immédiate de ce résultat est que l'on peut "couper" les branches de l'arbre de surdérivation instance d'une autre sans perdre la complétude.

3.3.1. Implantation

La surréduction basique s'implante facilement puisqu'il suffit de calculer à chaque étape l'ensemble des occurrences basiques. La détection des boucles et des branches subsumées nécessite de garder l'histoire des calculs précédents ce qui peut se réaliser efficacement avec une implantation de l'arbre de surréduction par addressage dispersé.

En ce qui concerne l'implantation de la surréduction elle même, la similitude entre la surréduction et le calcul de paires critiques peut être exploité en implantant la surréduction ou la superréduction par un algorithme de complétion de systèmes de réécriture, à la Knuth et Bendix. Cette idée décrite par N.Dershowitz [Dershowitz1984] est mise en forme dans [Kirchner1985]. Elle a été réalisée comme une extension du laboratoire de

réécriture $REVE_2$ pour la superréduction dans le cas E=ø par P.Rety et est décrite dans [Rety1985]. Un mécanisme de détection des boucles en tête de l'arbre de surdérivation donc de définition récursive de certains type de A-solutions, complète cette implantation.

CHAPITRE 7

UNIFICATION DANS DES MELANGES DE THEORIES

Nous abordons dans cette partie l'un des points les plus difficiles de la conception d'un algorithme d'unification : la preuve de sa terminaison. La correction partielle d'un algorithme d'unification construit avec les outils que nous avons exposé précédemment découle des résultats de correction de ces outils. La preuve de la correction totale est en général un problème difficile (la preuve de terminaison de l'algorithme d'unification associative commutative de Stickel est restée un problème ouvert près de 10 ans avant d'être résolue par F. Fages [Fages1984]) et excepté le travail récent de K. Yelick [Yelick1985] aucun outil n'était donné dans un cadre général pour travailler modulo des mélanges de théories, c'est à dire une théorie présentée par un ensemble d'axiomes A tel qu'il existe une partition de A en parties indépendantes c'est à dire agissant sur des symboles de fonctions différents.

Pour de telles théories nous allons montrer que la connaissance d'un algorithme de complète décomposition d'un unificande modulo chacun des éléments de la partition de A, permet de donner un algorithme de complète décomposition dans la théorie A. Cela généralise donc les résultats de Fages sur la terminaison de l'unification AC [loc.cit] et constitue une approche unifiée différente de celle proposée par K. Yelick [loc.cit.].

1. Combinaison d'algorithmes de complète décomposition

1.1. Structuration des systèmes de multiéquations

Nous précisons tout d'abord sur quel type de théorie nous allons travailler.

Définition 82 : Si E est un sous ensemble de A, on note **Symb(E)** ou F_E l'ensemble des symboles de fonction qui ont une occurrence dans les axiomes de E.

$$F_E = \{f \mid f \in F \text{ et } f \in Symb(t_1) \cup Symb(t_2) \text{ pour } t_1 = t_2 \in E\}$$

○

De manière à simplifier les preuves nous imposons une forme particulière aux équations :

Définition 83 : On dit qu'un système de multiéquations est **simple** s'il est fusionné et si ses éléments e sont de la forme suivante, e = (V(e)) (i.e réduit à des variables) ou e = (V(e) = {t}) avec t non variable ou e = $(t_1 == t_2)$ où t_1 et t_2 sont des termes non variables. ○

Il est facile de montrer que tout système de multiéquations est équivalent à un système de multiéquations simple.

Exemple 49 : Le système { $(x == y == z == t_1 == t_2)$ } est équivalent au système simple { $(x == y == z == t_1)$, $(t_1 == t_2)$ }

Afin de résoudre le problème posé par l'égalité multiple de variables comme x == y == z dans l'exemple ci-dessus, nous allons définir une forme canonique des systèmes dans laquelle toutes les variables d'une même multiéquation sont remplacées par un représentant déterminé.

Définition 84 : Pour un système de multiéquations fusionné S, soit R_S la relation définie sur l'ensemble X des variables par

$x\ R_S\ y$ <=> il existe une multiéquation e de S telle que x, y ∈ e

On notera $x^{\sim}$ un représentant de la classe de x et $S^{\sim}$ un système obtenu en remplacant dans tous les termes non variables de S chaque variable par le représentant canonique de sa classe modulo R_S. $S^{\sim}$ sera dit **non-ambigu** et noté, pour des représentants fixés, Rempl-Var(S). ○

Par application du lemme sur le remplacement, S et $S^{\sim}$ sont A-équivalents.

Exemple 50 : Si on considère le système

S = { $(x == y == z == a + z)$, $(x_1 == (x + y) * a)$ }

on a alors en choisissant x comme représentant de la classe {x, y, z}

$S^{\sim}$ = { $(x == y == z == a + x)$, $(x_1 == (x + x) * a)$ }

Il faut noter qu'en géneral il n'y a pas unicité du représentant de la classe et donc que $S^{\sim}$ n'est pas défini de façon unique.

Nous partitionnons maintenant un système en sous-systèmes homogènes :

Définition 85 : Soit P = $\{E_1, E_2, \ldots, E_p\}$ une partition d'un ensemble

d'axiomes A, telle que pour tous k et j éléments de [1..p] avec $k \neq j$ on ait $Symb(E_k) \cap Symb(E_j) = \emptyset$. On dit que P est une **partition séparée** de A.

On associe à la partition P la partition suivante de F :

$$P_F = \{F_{E_1}, F_{E_2}, \ldots, F_{E_p}, F_\emptyset\}$$

où $F_\emptyset$ désigne les symboles de F qui n'apparaissent dans aucun axiome de A :

$$F_\emptyset = F - \bigcup_{i=1}^{i=p} F_{E_i}.$$

Pour simplifier les notations nous noterons

$$P_F = \{F_1, F_2, \ldots, F_p, F_0\}$$

○

Exemple 51 : Si on prend

F = {+, *, ^, !, ., eg, a, b, c, f, g},

A = {ac(*), ac(+), T(^,eg), c(.), c(!)} et

E_1 = {ac(+)}

E_2 = {ac(*)}

E_3 = {T(^,eg), c(.), c(!)}

$P = \{E_1, E_2, E_3\}$ est une partition séparée de A et

F_1 = {+}, F_2 = {*}, F_3 = {^, eg, ., !}.

Définition 86 : Pour tout système S indécomposable et fusionné, on note **cd(S)** l'ensemble des multiéquations complètement décomposées de S et **am(S)** le complémentaire de cd(S) dans S, c'est à dire l'ensemble des multiéquations de S qui sont indécomposables mais non complètement décomposées (i.e. les équations à muter)

Si P est une partition séparée de l'ensemble d'axiomes A, pour tout élément E de P, on note $am_E(S)$ (respectivement $cd_E(S)$) l'ensemble des équations de am(S) (respectivement de cd(S)) dont tous les symboles de fonction sont

dans F_E :

$$am_E(S) = \{e \mid e \in am(S) \text{ et } \forall\, t \in e,\ Symb(t) \subseteq F_E\}$$

On notera $S_E = cd_E(S) \cup am_E(S)$ ○

Exemple 52 : Pour S = {x == a!z, x.c == y.(a!z), x*y == a*z} et pour la partition de A donnée dans l'exemple précédent on a :

cd_{E_3} = {x == a!z}, am_{E_3} = {x.c == y.(a!z)}

$cd_{E_2} = \emptyset$, am_{E_2} = {x*y == a*z}

Définition 87 : Soit P une partition séparée de l'ensemble d'axiomes A. On dit que le système S est **P-séparé** si toutes les équations e de S sont telles qu'il existe un élément F_i de P_F tel que $Symb(e) \subseteq F_i$.

○

Lemme 34 : Si P est une partition séparée de A, tout système S est généralisable en un système S' qui est P-séparé.

Preuve: Il suffit de généraliser tous les termes intervenant dans le système. □

Exemple 53 : (Suite des précédents)

L'équation e = (f(x*y,a.(c eg x)) == f(a*(z+u),b) n'est pas P-séparée. On la généralise dans le système séparé

S = {$f(x_1,x_2) == f(x_3,x_4)$,

$x_1 == x * y$,

$x_2 == x_5 . (x_6$ eg x),

$x_3 == a * x_7$,

x_4 == b,

$x_5 == a$,

$x_6 == c$

$x_7 == z + u\}$

Remarque : On peut généraliser moins abruptement en définissant la P-séparation de manière moins stricte en prenant dans la définition ci-dessus $Symb(e) \subseteq F_E \cup F_\emptyset$. En fait, les symboles de $F_\emptyset$ ont un status particulier car on peut les faire intervenir dans certains sous-systèmes dès que l'on sait donner un algorithme fini d'unification de ces sous-systèmes. C'est le cas pour la partition E_3 de l'exemple car nous avons vu dans les chapitres précédents un critère simple de terminaison du processus de décomposition-fusion-mutation pour la théorie E_3.

Notation : Dans toute la suite, afin de clarifier les preuves, nous n'utiliserons pas la remarque précédente et par conséquent nous associerons à tout système S fusionné et P-séparé, la partition $\{S_v, S_\emptyset, S_{E_1}, S_{E_2}, \ldots, S_{E_p}\}$ où S_v désigne l'ensemble des multiéquations e de S qui sont de la forme e = (V(e)) (i.e. réduite à des variables) et S_{E_i} les équations de S dont les symboles sont dans E_i. S_{E_i} sera encore noté S_i.

1.2. Une stratégie de complète décomposition dans les mélanges de théories

Nous allons montrer que si on connait un algorithme de complète décomposition pour chacune des théories E_k, c'est à dire un algorithme qui, prennant en entrée un unificande quelconque, retourne un unificande complètement décomposé, alors il existe un algorithme de complète décomposition pour A.

Commençons par donner la procédure générale de complète décomposition pour les mélanges de théories. Soit P une partition séparée de l'ensemble d'axiomes A telle qu'il existe un algorithme de complète décomposition pour chaque élément de P.

Nous noterons $COMP\text{-}DEC_k$ la procédure de complète décomposition dans la théorie E_k. Son profil est le suivant :

$COMP\text{-}DEC_k$(S : système d'équations construites sur F_k)

RETOURNE(unificande complètement décomposé)

SIGNALE(échec)

```
COMP-DEC(S : système P-séparé simple et non ambigu)
                          RETOURNE(unificande complètement décomposé)
                          SIGNALE(échec)

   Soit W un ensemble de variables tel que Var(S) ⊆ W.
   Choisir un sous-système S_k de S qui ne soit pas complètement décomposé.

   SI il n'en existe pas

   ALORS RETOURNER(S) (S est complètement décomposé)

   SINON [R_i]_{i∈I} = COMP-DEC_k(S_k)

            RESIGNALER(échec)   (propagation du signal d'échec)

         soit R_i' = (S-S_k) ∪ R_i

         soit R_i" = Rempl-Var(Fus(R_i'))

         RETOURNER ([COMP-DEC(R_i'')]_{i∈I})
   FSI
FIN COMP-DEC
```

Nous allons montrer que si chacune des procédures $COMP\text{-}DEC_k$ termine alors COMP-DEC termine. Il retournera alors une disjonction de systèmes complètement décomposés ou bien signalera échec.

1.3. Terminaison de l'algorithme COMP-DEC

On suppose dans tout ce paragraphe que la théorie A considérée est non potente et que P est une partition séparée de A. Tous les systèmes considérés sont supposés séparés et mis sous forme simple.

L'ordre considéré sur les n-uplets d'entiers est l'ordre lexicographique.

Nous introduisons maintenant les mesures de complexité dont la combinaison va permettre de prouver la terminaison.

Tout d'abord une mesure du nombre de variables qui occurent sous plusieures familles de symboles.

Définition 88 : Pour un système P-séparé S, on note $\chi_S(x)$ le nombre de familles de symboles différentes sous lesquelles la variable x apparait, $v_i(S)$ le nombre de variables de Var(S) qui occurrent sous exactement i familles de symboles différentes : $v_i(S) = |\{ x \text{ tel que } \chi_S(x) = i\}|$

On pose $imax(S) = \max_{v_i(S)\neq 0} i$.

v est le imax(S)-uplet défini par

$$v(S) = (v_{imax(S)}(S), v_{imax(S)-1}(S),\ldots, v_1(S))$$

Si U est une disjonction de systèmes, on prend pour l'ordre lexicographique:

$$v(U) = \max_{S\in U} v(S)$$

○

Exemple 54 : (Suite des précédents)

Pour le système

$S = \{ x_1 * x_2 == x_3 * x_4,$

$x_1 == x \text{ eg } x_4$

$x \,.\, y == x_4 \,.\, (x_1 \,!\, z)\}$

on a $v_1 = 5$, $v_2 = 2$, imax = 2 et v(S) = (2,5).

Des équations particulières vont apparaitre dans les systèmes, nous les appelons immutables car elles ne seront jamais modifiées lors des transformations successives de S.

Définition 89 : Une équation e d'un système S fusionné est dite **immutable** si et seulement si elle est de la forme e = (V(e) == {t}) et si V(e) ∩ Var(Term(S-{e})) = ø. Une équation non immutable est dite **mutable**. L'ensemble des équations immutables de S est noté **Immu(S)**. Les équations mutables (respectivement immutables) de cd(S) seront notées cdi(S) (respectivement cdm(S)). Le cardinal de $cdm(S_{E_k})$ sera noté κ(S,k). On pose pour tout système S

$$\kappa(S) = \sum_{k=1}^{k=p} \kappa(S,k)$$

et pour toute disjonction de systèmes U

$$\kappa(U) = \max_{S\in U} \kappa(S)$$

○

Enfin nous aurons besoin de mesurer le degré de résolution d'un système ou d'une disjonction de systèmes :

Définition 90 : Pour tout système S on note δ(S) le nombre de sous-systèmes S_{E_k} non complètement décomposés. Pour une disjonction de système U on pose

$$\delta(U) = \max_{S\in U} \delta(S).$$

○

Enfin nous combinons ces trois mesures de la manière suivante :

Définition 91 : Pour tout système S on note $\zeta(S) = (\nu(S), \kappa(S), \delta(S))$. ○

La preuve de terminaison va se dérouler selon le schéma suivant : la résolution d'un système non complètement décomposé $S_k = S_{E_k}$ va soit faire décroitre le nombre de variables partagées par plusieures familles, soit provoquer des fusions dans S_k, ce qui diminuera éventuellement le nombre d'équations complètement décomposées et mutables d'autres sous-systèmes, soit augmentera le degré de résolution du système ou de la disjonction de systèmes.

Nous donnons tout d'abord quelques remarques techniques.

Remarque : Nous utiliserons dans la suite la propriété suivante : pour toute sous-théorie E_k de A et pour tout système S homogène sur E_k, si $x == t$ $(t \notin X)$ appartient à S alors $x == z$ avec $z \in X$ n'est pas élément de $Eq^W(S)$ (qui rappelons le est fusionné). Cela est simplement dû au fait que la théorie A est supposée non potente.

Lemme 35 : Si un système S a des A-solutions et est P-séparé alors Fus(S) est également P-séparé. Par ailleurs la fusion conserve ν.

Preuve: Soit S est un système séparé. Si on fusionne des équations $x == t$ et $x == t'$, soit t et t' sont homogènes sur la même classe de symboles F_E pour E dans P, auquel cas $t == t'$ est encore P-séparée, soit t et t' sont homogènes sur des classes de symboles différentes. Auquel cas puisque la théorie n'est pas potente $t ==$ t' n'a pas de A-solution, ce qui contredit l'hypothèse.

Par ailleurs la fusion ne peut pas modifier ν puisqu'elle n'agit pas sur la structure interne des termes. □

Nous prouvons un premier résultat qui montre que si on résoud un sous-système $S_k = S_{E_k}$ de S qui partage des variables avec d'autres sous-systèmes de S, sous des symboles de ces systèmes, alors $\zeta(S)$ diminue strictement.

Lemme 36 : Soit S_k un sous-système de S non complètement décomposé tel qu'il existe n dans [1..p] $(n \neq k)$ avec $Var(Term(S_k)) \cap Var(Term(S_n)) \neq \emptyset$. Alors en reprenant les notations prises dans COMP-DEC, si S_k est le système choisit dans la première étape de COMP-DEC, on a pour tout i de I, $\zeta(R_i") < \zeta(S)$.

Preuve: Nous allons montrer que ζ diminue, même si Rempl-Var peut faire temporairement croitre ν.

Les notations utilisées sont les mêmes que dans COMP-DEC. Soit $i \in I$.

Soit x une variable occurrant dans un terme de S_k et un terme de S_n.

Si x apparait dans R_i sous la forme $x == t$ $(t \notin X)$ où les variables de t sont des nouvelles variables (i.e. en dehors de W), x n'apparait donc plus sous aucun symbole de F_k et par conséquent ν a diminué de S_k à $R_i"$, la preuve est terminée dans ce cas.

Si x apparait dans R_i sous la forme $x == nv$ $(nv \in X-W)$, le représentant de la classe de x pourra apparaitre à nouveau sous un symbole de F_k. Dans ce cas ν restera constant de S à $R_i"$.

Et alors de deux choses l'une,

* ou bien il existe une variable y apparaissant sous un terme de S_k et un terme de S_j pour $j \in I$ $(j \neq k)$ telle que son image dans R_i soit de la forme $x == t$ $(t \notin X)$, auquel cas ν a diminué ainsi que nous l'indiquions ci-dessus,

* ou bien ce n'est pas le cas et toutes les variables apparaissant dans les termes de S_k donnent une équation de la forme $x == nv$ $(nv \in X)$ dans R_i. Dans ce cas aucune nouvelle équation mutable n'est apparue dans S_k, $\kappa(S,k)$ n'a pas augmenté. Dans cette éventualité il reste deux possibilités :

  - soit la résolution de S_k n'a pas fait fusionner d'équations dans $S\text{-}S_k$ et dans ce cas le système a gagné un degré de résolution $\delta(S) > \delta(R_i")$

  - ou bien certaines équations ont fusionné, faisant peut être croitre δ mais diminuant κ puisqu'il y a alors strictement moins d'équations complètement décomposées mutables dans les systèmes qui ont fusionné et qu'il n'y a pas, comme nous l'avons vu, de nouvelle équation mutable dans S_k.

Dans tous les cas $\forall\, i \in I$, $\zeta(S) > \zeta(R_i")$. □

Exemple 55 : (Suite des précédents).

Si on résoud la première équation de S, on obtient entre autre système

$S = \{x_1 == nv_1,$ qui fusionne en $\{x_1 == x_4 == nv_1 == x \text{ eg } x_4$
$x_2 == nv_2,$ $x_2 == x_3 == nv_2,$
$x_3 == nv_2,$ $x^2.\ y == x_4\ .\ (x_1\ !\ z)\}$
$x_4 == nv_1,$
$x_1 == x \text{ eg } x_4$
$x^2.\ y == x_4\ .\ (x_1\ !\ z)\}$

et par Rempl-Var, on obtient :

$R" = \{ x_1 == x_4 == nv_1 == x \text{ eg } x_1,$
$x_2 == x_3 == nv_2,$
$x^2.\ y == x_1\ .\ (x_1\ !\ z)\}$

qui est tel que $\nu(R") = (0,4)$

Lemme 37 : Dans COMP-DEC, si le sous-système choisi S_k est non complètement décomposé et ne partage pas de variables avec le reste du système (i.e. tel que $Var(Term(S_k)) \cap Var(Term(S\text{-}S_k)) = \emptyset$), alors $\kappa(S) \geq \kappa(R_i")$.

Preuve: Les équations issues de la complète décomposition de S_k et qui composent les systèmes R_i, ne peuvent êtres que de l'un des trois types suivants:

(1) $x == t$ $(t \notin X)$ et $x \in Var(Term(S_k))$,

(2) $x == t$ $(t \notin X)$ et $x \in V(S_k)$,

(3) $x == nv$ $(nv \in X\text{-}W)$

* Les équations du type (1) sont immutables car par hypothèse, $x \notin Var(Term(S\text{-}S_k))$. Il faut noter que nous n'en sommes sûr que parceque, à cause du renomage Rempl-Var, aucune variable de la classe de x ne peut apparaitre dans $Var(Term(S\text{-}S_k))$.

* Les variables "x" des équations de type (2) proviennent par définition des équations de cd(S), il y a donc moins (ou autant) d'équations de ce type dans R_i que dans S_k.

* Les équations de type (3) n'interviennent pas ici car elles n'interviennent pas dans la définition de κ.

La fusion de R_i' provoque les modifications suivantes :

Pour les équations de type (1), si elles fusionnent, cela peut être :

- soit avec l'équation qui définit la classe de x i.e. du type V(e) avec $x \in e$. Par hypothèse sur Rempl-Var, cette nouvelle équation obtenue par fusion est toujours immutable,
- soit avec une équation $x == t'$ de S_j avec $k \neq j$. Il y a alors à l'évidence une collision de symbole puisque A est supposée non potente et R_i" n'a pas de A-solution.

Pour les équations de type (2), on a les mêmes possibilités :

- fusion avec une équation de la forme e = (V(e)), l'équation, si elle était mutable le restera, de même si elle est immutable,
- fusion avec une équation $x == t'$ de S_j avec $k \neq j$. Il y a collision de symbole.

Les équations de type (3) peuvent provoquer des fusions dans les autres systèmes S_j ($j \neq k$) elles diminuent alors le nombre d'équations mutables complètement décomposées de ces systèmes puisqu'on a alors la configuration suivante :

$x_1 == nv$ qui fusionne en $x_1 == x_2 == nv == t_1 == t_2$
$x_2 == nv$ et que l'on réécrit par convention en
$x_1 == t_1$ $x_1 == x_2 == nv == t_1$
$x_2 == t_2$ $t_1 == t_2$

(**) Soit il y a collision de symbole si $x_1 == t_1$ et $x_2 == t_2$ ne sont pas dans le même sous-système S_j, soit il y a une équation complètement décomposée et mutable de moins.

Dans tous les cas, le nombre d'équations complètement décomposées et mutables n'a pas augmenté i.e. $\kappa(R_i', k) \leq \kappa(S_k)$.

Reste à étudier l'impact de Rempl-Var sur κ dans le passage de R_i' à R_i".

Les classes de variables pour la relation R ont été modifiées par la fusion précédente. Mais il est clair que cela ne peut pas tranformer une équation immutable en mutable, κ est conservé : $\kappa(R_i", k) = \kappa(R_i', k)$. □

Lemme 38 : Sous les hypothèses du lemme précédent on a $(\kappa(S), \delta(S)) > (\kappa(R_i"), \delta(R_i"))$

Preuve: En effet soit $\kappa(S) = \kappa(R_i")$, ce qui signifie compte tenu de la remarque (**) faite dans la preuve ci-dessus qu'il n'y a pas eu de fusion dans d'autre sous-systèmes, et par conséquent $(R_i'')_{E_k}$ étant complètement décomposé, δ a diminué de 1. Soit $\kappa(S) > \kappa(R_i")$ et la conclusion est claire. □

Théorème 18 : Si A est une théorie non potente, $P = \{E_1, \ldots, E_p\}$ une partition séparée de A telle qu'un algorithme fini de complète décomposition dans E_k soit connu pour tout $k \in [1..p]$ alors COMP-DEC est un algorithme fini de complète décomposition dans A.

Preuve: * La correction de l'algorithme résulte de la correction de chacune des transformation qu'il évoque.

* La terminaison résulte des résultats précédents car la mesure $(v(S), \kappa(S), \delta(S))$ décroit strictement à chaque appel de COMP-DEC. □

Nous venons donc d'établir que sans autre hypothèse sur les théories E_k que l'existence d'un algorithme fini de complète décomposition et la non potence, la terminaison de l'algorithme de complète décomposition dans la théorie A est assurée.

Pour donner un algorithme de A-unification il faudra alors ajouter à l'étape de complète décomposition, la résolution de la disjonction de systèmes de multiéquations complètement décomposées. On pourra, si la théorie A est stricte et fortement complète, utiliser les techniques décrites dans le chapitre 2. Dans d'autre applications, on aura intérêt à conserver les systèmes complètement décomposés sans chercher à construire ses A-solutions finies. Cela revient alors à travailler sur dans les termes infinis puisqu'un système complètement décomposé est une représentation d'un tel terme [Courcelle1984].

Ce résultat est donc sensiblement différent de celui de K. Yelick [Yelick1985] qui montre que pour toute théorie A, P-séparée régulière et non potente, la combinaison des algorithmes de E_k-unification telle qu'elle la définie termine.

* D'une part parceque l'algorithme qu'elle donne fonctionne "à la Robinson" et repose donc sur des principes différents de ceux développés dans ce travail comme nous l'avons déjà souligné.

* D'autre part parceque, en utilisant les outils que nous avons développé ici, nous avons besoin d'hypothèses plus fortes (théories strictes et fortement complète) pour obtenir une conclusion analogue à celle de K. Yelick.

* Mais aussi, comme nous le disions plus haut, parceque nous avons besoin de moins d'hypothèses pour obtenir un unificande complètement décomposé.

2. Applications

Le théorème précédent est un outil fondamental pour la conception d'algorithmes d'unification dans des mélanges de théories. Nous en donnons maintenant quelques exemples.

Exemple 56 : Les théories associative-commutative

Nous avons donné au chapitre 4 un algorithme de complète décomposition pour un seul symbole AC. Le théorème précédent permet donc de construire un algorithme de complète décomposition modulo un nombre fini d'axiomes de commutativité-associativité, et comme les théories AC sont strictes et fortement complètes nous obtenons un nouvel algorithme fini de AC-unification pour un nombre quelconques de symboles AC, en présence ou non de symboles sans propriétés.

Les avantages de ce nouvel algorithme sont les suivants :

* il est clairement parallélisable puisque chaque élément de la disjonction de systèmes courante peut être résolue indépendamment des autres,

* il est plus efficace que celui de Stickel comme le montre les expérimentations réalisées avec l'implantation faite en CLU dans le système REVEUR3. En particulier les ensembles complets de AC-solutions comportent en général moins d'éléments que ceux retournés par l'algorithme de Stickel implanté par K. Yelick. Cela tient à deux choses :

  - d'une part à la méthode de passage de l'équation initiale à un unificande U tel que p(U) = 0 (c.f. chapitre 4),

  - d'autre part au fait que la résolution des systèmes d'équations diophantiennes réduit l'espace des AC-solutions à gérer.

* enfin, il est facilement intégrable à une bibliothèque d'algorithmes d'unification standards.

Exemple 57 : Si on considère la théorie donnée dans les exemples de la section précédente : A = {ac(*), ac(+), T(^,eg), c(.), c(!)}, nous avons plusieures façons de construire un algorithme d'unification pour A. En effet nous avons le choix de la partition P de A.

* si on choisit la partition

E_1 = {ac(+)}

E_2 = {ac(*)}

E_3 = {T(^,eg), c(.), c(!)}

le processus de décomposition-fusion-mutation détermine un algorithme fini d'unification pour la théorie E_3 puisque, comme nous l'avons vu, la terminaison du processus est assurée par un critère simple sur la taille des termes. Connaissant un algorithme d'unification fini pour E_1 et E_2 par ailleurs, COMP-DEC fournit donc un système complètement décomposé que SUBST résoud car la théorie A est à stricte puisque qu'elle est permutative.

* Mais une autre possibilité consiste à choisir pour partition

E_1 = {ac(+)}

E_2 = {ac(*)}

E_3 = {T(^,eg)}

E_4 = {c(.)}

E_5 = {c(!)}

auquel cas, puisque nous connaissons un algorithme d'unification pour chacune des théories de la partition, COMP-DEC détermine également dans ce cas un algorithme fini de complète décomposition. La différence avec le précédent réside dans le fait que considèrant chaque théorie séparément, le nombre de généralisation nécessaire va être plus important et par conséquent le second algorithme sera moins efficace que le premier.

Cette dernière remarque est tout à fait générale. La décomposition par les symboles de $F_\emptyset$ étant la mutation de la théorie vide, on peut toujours considérer la partition la plus fine possible dans laquelle les symboles de $F_\emptyset$ seront considéres à part. Mais on perd alors en efficacité puisque on est amener à généraliser davantage que si on insère le processus de décomposition dans le plus de théories possible de la partition.

Terminons cette partie par une remarque concernant la restriction de l'approche précédente à des théories non potentes.

Cette restriction est due au fait que l'on souhaite avoir une partition séparée de A. Elle ne peut donc être levée facilement. Mais on peut noter que les axiomes de la forme t = x sont justement ceux qui, orienté en $t \rightarrow x$, satisfont trivialement les hypothèses nécessaires à la terminaison de la RE-surréduction basique pourvu que le couple (R,E) soit E-confluent et E-noetherien. Pour une théorie A potente, si A' est la plus grande sous théorie de A qui soit non potente, en posant A" = A-A' on cherchera à orienter A' est un système de réécriture R puis on complètera R modulo A' de manière à obtenir un système de réécriture R' qui soit E-confluent et E-noetherien. Sur R' on essaiera alors d'appliquer les outils donnés par la surréduction et la superréduction. C'est la démarche qu'il faudra suivre afin d'obtenir un algorithme d'unification pour les axiomes du "si alors sinon" présenté dans le chapitre sur les théories syntaxiques.

CONCLUSION

Résoudre une équation dans une théorie équationnelle A est un problème indécidable en général. Il faut par conséquent se donner les moyens d'aborder ce problème avec des méthodes propres à chaque théorie tout en cherchant à conserver le maximum de généralité.

Ce travail sur la recherche de méthodes et d'outils pour l'étude systématique d'un problème d'unification dans une théorie équationnelle doit donc être considéré rétrospectivement en se posant les questions suivantes :

- que permet-il de faire?
- quelles sont ses limites?
- quelles perpectives ouvre-t-il?

C'est en cherchant à répondre à ces trois questions que nous reprennons maintenant les principaux résultats obtenus.

1. Des approches complémentaires

Notre étude de la résolution d'un problème d'unification nous a conduit à en structurer et hiérarchiser l'approche.

- **Structurer**, car pour résoudre un problème d'unification il faudra :

1) <u>transformer</u> ce problème en un problème "simple" c'est-à-dire en une disjonction de systèmes complètement décomposés. Pour cela il faut déterminer l'opération de mutation associée à la théorie, puis prouver que le processus de décomposition-fusion-mutation termine.

- Aucune hypothèse particulière n'est requise à ce niveau si l'on aborde le problème directement.

- Une approche semi-automatique est proposée si la théorie n'est pas potente : si un ensemble d'axiomes est résolvant, alors l'opération de mutation se déduit de la forme des axiomes. De plus nous avons donné une procédure de complétion unificationnelle qui permet de compléter un ensemble d'axiomes en un ensemble d'axiomes résolvant. Cette approche est particulièrement bien adaptée aux théories permutatives.

- On peut prendre une approche modulaire : nous avons montré qu'en séparant la théorie A en sous-théories disjointes $[A_i]_{i\in I}$ alors, sous la seule condition que la théorie A n'est pas potente, la connaissance d'un algorithme de complète décomposition dans chacune des sous théories A_i détermine un algorithme fini de complète décomposition dans la théorie A.

2) <u>Résoudre</u> un problème "simple", c'est à dire savoir déterminer pour tout système d'équations complètement décomposées, un ensemble complet de solutions dans la théorie équationnelle considérée. Nous avons montré que si la théorie A est stricte et fortement complète (ce qui est le cas pour les théories permutatives) alors cela peut être réalisé par un algorithme simple (SUBST).

Pour les théories strictes et fortement complètes cette structuration

offre plusieurs avantages :

* d'une part elle permet de dégager une structure unifiée d'algorithmes d'unification, ce qui permet outre la simplification des concepts, la conception et l'extension aisée et à moindre coût de bibliothèques de tels algorithmes,

* d'autre part elle permet la construction d'algorithmes efficaces, car on évite les instanciations "à la Robinson" par les parties de solutions déjà découvertes, et le regroupement des contraintes dans les systèmes diminue l'espace de recherche des solutions potentielles,

* enfin, elle met en évidence le parallélisme, puisque chaque élément d'une disjonction de systèmes peut-être simplifié et résolu indépendamment des autres.

- **Hiérarchiser**, en utilisant la surréduction. Si l'on peut partitionner l'ensemble des axiomes A en E et R, tels que R soit un système de réécriture RE-confluent et E-noetherien alors nous avons montré que la connaissance d'un algorithme d'unification pour la théorie E détermine une procédure d'unification par RE-surréduction dans la théorie A = E U R. Malheureusement cette procédure ne termine pas systématiquement, aussi avons nous donné une condition suffisante de terminaison en utilisant la RE-surréduction basique. Nous avons également montré comment factoriser l'arbre de surdérivations et le simplifier, ce qui permet encore d'améliorer la procédure de surréduction.

Par conséquent, à condition de pouvoir satisfaire les hypothèses relatives à la terminaison de la RE-surréduction basique, ou de disposer d'une preuve de terminaison de la RE-surréduction appropriée à la

théorie, nous savons déterminer des algorithmes d'unification pour des théories qui sont potentes ou/et non régulières.

Au cours de ce travail nous avons donné un certain nombre d'exemples d'application de cette approche globale. Nous allons maintenant discuter de son efficacité en montrant comment elle permet d'étudier deux problèmes ouverts d'unification : la résolution d'équations modulo la commutativité des crochets de Lie et la résolution d'équations dans les groupes abéliens.

2. Exemples d'approches et de limitations

2.1. Etude de l'unification modulo la commutativité des crochets de Lie

Soit L(+) l'axiome (x+y)+z = z+(y+x) traditionnellement noté [[x,y],z] = [z,[y,x]].

L'ensemble des symboles décomposables est donc F-{+}. Cette théorie étant permutative elle est stricte et fortement complète et par conséquent la seule difficulté est de trouver une opération de mutation telle que le processus de décomposition-fusion-mutation termine.

La théorie étant non potente, on va chercher à montrer qu'elle est syntaxique et par conséquent nous appliquons la procédure de complétion unificationnelle à l'axiome L(+). Nous laissons le lecteur se convaincre (à la main pour l'instant) que l'ensemble des axiomes résultants est L(+) et (x+x')+(y+y')=(x'+x)+(y'+y). La théorie engendrée par L(+) est donc syntaxique et une opération de mutation possible est la suivante :

$$u+v == u'+v' \Leftarrow \begin{cases} u == v \\ v == v' \end{cases} \quad \begin{cases} u == x+y \\ v == u' \\ v' == y+x \end{cases} \quad \begin{cases} u == v' \\ v == y+x \\ u' == x+y \end{cases} \quad \begin{cases} u == x+x' \\ v == y+y' \\ u' == x'+x \\ v' == y'+y \end{cases}$$

Cette transformation fait décroitre la taille des équations sauf si, par fusion, on retrouve des termes de même taille qu'initialement. C'est le cas par exemple pour le second système si u = v'. Dans ce cas, il faut donc savoir résoudre directement l'équation

$$x+y == y+x.$$

Or une étude simple montre que cette équation a pour solutions

$\sigma_1 = \{ x \leftarrow z, y \leftarrow z \}$

$\sigma_2 = \{ x \leftarrow z+z \}$

$\sigma_3 = \{ x \leftarrow z_1+(z+z) \}$

$\sigma_4 = \{ x \leftarrow z_1+(z_2+(z+z)) \}$

...

qui sont toutes incomparables. Par conséquent L(+) est une théorie infinitaire.

On voit comment les outils introduits dans cette thèse permettent dans ce cas de "mettre le doigt" sur les équations qui posent problème.

2.2. Unification dans les groupes abéliens

Si F = {+, i, 0} alors un algorithme d'unification est donné par [Lankford1984]. Le problème n'est pas résolu si F comporte d'autres symboles et c'est dans ce cas que nous allons chercher à appliquer nos outils.

Rappelons les axiomes définissant les groupes abéliens :

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

$$(x + y) == (y + x)$$

$$(x + 0) == x$$

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

Cette théorie n'étant pas régulière, elle n'est pas stricte et on ne peut donc pas lui appliquer les outils de résolution directe des systèmes d'équations complètement décomposées. Il est donc naturel de chercher à utiliser ici une approche hiérarchique, donc à déterminer un système de réécriture RE-confluent et E-noetherien équivalent à l'ensemble des axiomes de départ. Pratiquement au moins deux tels systèmes de réécriture sont connus aujourd'hui, comme on a pu le voir plus en détail dans l'annexe. Le premier initialement trouvé par Peterson et Stickel [Peterson1981] en utilisant la R,E-réécriture est le suivant :

L'ensemble E des axiomes :
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
L'ensemble R des règles :
1. i(0) -> 0
2. (x + 0) -> x
qui a pour extension :
3. (y + (x + 0)) -> (y + x)
4. i(i(x)) -> x
5. (x + i(x)) -> 0
qui a pour extension :
6. (y + (x + i(x))) -> y
7. i((x + y)) -> (i(y) + i(x))

On sait donc faire de la R,E-surréduction avec cet ensemble de règles, mais

la procédure de surréduction y compris basique ne terminera pas, à cause par exemple de la règle 7 mais aussi de l'extension 3.
Si l'on considère l'autre ensemble de règles décrit dans l'annexe et découvert par le système REVEUR3, on a :

L'ensemble E des axiomes :
((x + y) + z) == (x + (y + z))
(x + y) == (y + x)
L'ensemble R des règles :
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))
6. (x + i(x)) -> 0
qui a pour extension :
7. ((x + i(x)) + z) -> z

Ici seule la règle 5 ne permet pas d'assurer la terminaison de la surréduction basique. On va donc la conserver en tant qu'axiome. Il faut alors compléter l'ensemble des axiomes précédents modulo ce nouvel ensemble d'axiomes. Mais la superposition de la règle 4 dans l'axiome i(x+y) = i(x)+i(y) va engendrer une paire critique non confluente de la forme (x+i(y), i(i(x)+y) qui si elle était orientée en règle ne satisferait plus la condition requise pour la terminaison de la surréduction basique. On va donc également garder i(i(x)) = x comme axiome. Il faut alors être capable de faire de l'unification modulo la théorie (minus U AC), ce qui parait un objectif raisonable compte tenu de la connaissance que l'on a de ces deux théories. Il faut noter que la théorie (minus U AC) n'est pas stricte mais que des méthodes analogues à celles développée pour l'étude de la théorie minus ont de fortes chances de permettrent de conclure. Après implantation de cet algorithme d'unification dans REVEUR3, il faudra compléter la règle restante en un système de réécriture RE-confluent. Un

algorithme d'unification dans les groupes abéliens découlera alors de la procédure de surréduction basique.

On voit que la conception d'un algorithme d'unification peut mettre en oeuvre des techniques et des logiciels dont nous avons essayé de montrer dans cette thèse qu'ils étaient complémentaires.

3. Problèmes ouverts, perspectives de recherches et d'applications

Nous revenons pour terminer sur les principaux problèmes ouverts et les perspectives de recherches qu'ouvre cette thèse.

- En ce qui concerne l'étape de complète décomposition d'un unificande en un unificande complètement décomposé, citons
  - les preuves de terminaison de la procédure de décomposition-fusion-mutation. Il serait en particulier intéressant de généraliser la méthode de preuve de terminaison donnée par [Arnborg1985] dans le cas de la distributivité gauche.
  - l'automatisation de la preuve de terminaison de la procédure de décomposition-fusion-mutation, soit par un procédé direct, soit par le codage sous forme de règles de réécriture des opérations de décomposition, fusion et mutation qui rappelons le, ont pour objectif de normaliser un problème d'unification.
  - l'affaiblissement des conditions suffisantes permettant de vérifier qu'un ensemble d'axiomes est résolvant.
  - éliminer l'hypothèse de non potence que nous avons imposée dans l'étude des mélanges de théories.

- Les résultats sur la résolution des systèmes d'équations complètement décomposées sont ceux qui nécessitent les hypothèses les plus fortes

dans ce que nous avons fait. En particulier demeurent ouverts les points suivants :

- les théories strictes sont elles permutatives?
- comment étendre les résultats à des théories non strictes en généralisant par exemple les techniques utilisées pour la théorie minus?

- Les outils que nous avons développés et qui structurent clairement les algorithmes d'unification pour les théories strictes et fortement complètes, vont permettre l'étude de leurs complexité, en particulier dans le cas d'éxécution en parallèle.

- Compte tenu de son importance en unification et en programmation logique, il faut chercher à réduire la taille de l'arbre de surdérivation. En particulier en suivant les méthodes de réduction du nombre des paires critiques utiles dans l'algorithme de complétion, développées par W. Kuchlin [Kuchlin1985].

- Appliquer les méthodes développées dans ce travail pour étudier directement l'unification typée.

- Des théories jouant un rôle important sont souvent infinitaires : l'associativité, minus, la commutativité des crochets de Lie par exemple. Deux directions de recherches complémentaires devront être explorées :
  - Pour une théorie donnée, caractériser les équations qui n'ont pas d'ensemble complet fini de A-unificateurs. Par exemple pour les crochets de Lie, l'équation x+y == y+x est-elle la seule qui soit infinitaire?

- Décrire des ensembles complets infinis de A-unificateurs à l'aide de méta-unificateurs comme dans [Kirchner1982] afin de les utiliser pour faire par exemple de la méta-complétion [Kirchner1985].

BIBLIOGRAPHIE

Arnborg1985.
S. Arnborg and E. Tiden, "Unification problems with one-sided distributivity," Proceeding of the RTA international conference, Dijon (France), 1985.

Benoit1984.
C. Benoit, "Axiomatisation des tests et systèmes complets de preuve," These de l'université de Paris 7, Paris, 1984.

Birkhoff1935.
G. Birkhoff, "On the Structure of Abstract Algebras," Proc. Cambridge Phil. Soc. 31, pp. 433-454, 1935.

Bloom1983.
S. Bloom and R. Tindell, "Varieties of IF THEN ELSE," S.I.A.M. J. on Computing, vol. 12, no. 4, pp. 677-707, 1983.

Clifford1967.
A.H. Clifford and G.B. Preston, The algebraic theory of semigroups, volume I and II, American Mathematical Society, 1961 and 1967.

Corbin1983.
J. Corbin and M. Bidoit, "Pour une réhabilitation de l'algorithme d'unification de Robinson," IFIP 1983, 1983.

Courcelle1984.
B. Courcelle, "Equivalences and transformations of regular systems, applications to recursive progam shemes and grammars," 8430, Universite de Bordeaux 1, Bordeaux, 1984.

Dershowitz1984.
N. Dershowitz, "Computing With Term Rewriting Systems," Procedings of An NSF Workshop On The Rewrite Rule Laboratory, April 1984.

Fages1983.
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.

Fages1983.
F. Fages, "Formes canoniques dans les algèbres booléennes," Thèse de 3eme cycle. Université de Paris 6, 1983.

Fages1984.
F. Fages, "Le systeme KB : manuel de reference : presentation et bibliographie, mise en oeuvre," R.G. 10.84, Greco de Programmation, Bordeaux, 1984.

Fages1984.
F. Fages, "Associative-Commutative Unification," Proceedings 7th international Conference on Automated Deduction, Napa Valley (California, USA), 1984.

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.

Gallaire1981.
H. Gallaire, "Impacts of logic on data bases," VLDB, 1981.

Gobel1983.
R. Gobel, "A completion procedure for globally finite term rewriting systems," Memo SEKI-83-12. Universitat Kaiserslaustern, 1983.

Guessarian1977.
I. Guessarian, "Les Tests et Leur Caract'Erisation Syntaxique," R.A.I.R.O. Informatique Th'Eorique 11, pp. 133-156, 1977.

Herbrand1930.
J. Herbrand, "Recherches sur la theorie de la demonstration," These, U. de Paris, In: Ecrits Logiques de Jacques Herbrand PUF Paris 1968, 1930.

Huet1976.
G. Huet, "Résolution d'Equations dans des Langages d'Ordre 1,2, ... ω," Thèse d'Etat, Université de Paris VII, 1976.

Huet1978.
G. Huet, "An Algorithm to Generate the Basis of Solutions to Homogenous Linear Diophantine Equations," Information Processing Letters, vol. 7, no. 3, pp. 144-147, 1978.

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.

Hullot1980.
J.M. Hullot, "Compilation de Formes Canoniques dans les Théories Equationnelles," Thèse de 3ème Cycle, Université de Paris Sud, 1980.

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.

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, Les Arcs (Savoie, France), 1980.

Jouannaud1982.
J.P. Jouannaud, Cours de DEA, Nancy, 1982.

Jouannaud1983.
J. P. Jouannaud, C. Kirchner, and H. Kirchner, "Incremental Construction 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.

Jouannaud1983.
J.P. Jouannaud, "Church-Rosser computations with equational term rewriting systems," Proc. 4th Conference on Automata, Algebra and Programming, L'Aquila, 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.

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.

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, Napa Valley (California, USA), 1984.

Kirchner1985.
C. Kirchner and H. Kirchner, "Implementation of a general completion procedure parameterized by built-in theories and strategies," Proceedings of the EUROCAL `85 conference, 1985.

Kirchner1985.
H. Kirchner, "Preuves par complétion dans les variétés d'algèbres," 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.

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.

Lankford1984.
D. Lankford, G. Butler, and B. Brady, "Abelian group unification algorithms for elementary terms," in Automated theorem proving: After 25 years, ed. W.W. Bledsoe and W. Loveland, vol. 29, AMS, 1984.

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.

Livesey1976.
M. Livesey and J. Siekmann, "Unification of A+C-Terms (Bags) And A+C+I-Terms (Sets)," Interner Bericht Nr. 3/76. Institut Fur Informatik I, Universitat Karlsruhe, 1976.

Makanin1977.
G.S. Makanin, "The problem of solvability of equations in a free semigroup," akad. nauk. sssr, p. 233, 1977.

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.

Mzali1985.
J. Mzali, "Filtrage associatif, commutatif ou idempotent," Proceedings of the conference "Materiels et logiciels pour la 5ieme generation", pp. 243-258, Paris (France), 1985.

Paterson1978.
M.S. Paterson and M.N. Wegman, "Linear Unification," J. of Computer And Systems Sciences, vol. 16, pp. 158-167, 1978.

Paul1984.
Etienne Paul, "A new interpretation of the resolution principle," Proceedings 7th international Conference on Automated Deduction, Napa Valley (California, USA), 1984.

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.

Plotkin1972.
G. Plotkin, "Building-In Equational Theories," Machine Intelligence, vol. 7, pp. 73-90, 1972.

Raoult1982.
J.C. Raoult, Cours de DEA, Orsay, 1982.

Raulefs1978.
P. Raulefs and J. Siekmann, "Unification of Idempotent Functions," Report, Institut Fur Informatik 1, Univ. Karlsruhe, 1978.

Redei1963.
L. Redei, Theorie des endlich erzeugbaren kommutativen halbsgruppen, p. Hamburger mathematische einzelschriften, Physica-Verlag, Würzburg, 1963.

Rety1985.
P. Rety, C. Kirchner, H. Kirchner, and P. Lescanne, "Narrower: a new algorithm for unification and its application to Logic Programming," Proceedings of the RTA conference. To appear., 1985.

Robinson1965.
J.A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," J. of ACM, vol. 12, pp. 32-41, 1965.

Robinson1971.
J.A. Robinson, "Computational Logic: the Unification Computation," Machine Intelligence 6, Eds B. Meltzer And D.Michie American Elsevier, New-York, 1971.

Sabbatel1985.
G. Berger Sabbatel, W. Dang, and J.C. Ianeselli, "Stratégie de recherche et unification pour la machine opale," Proc. of the conference "Matériels et logiciels pour la 5ieme génération", pp. 189-199, Paris (France), 1985.

Sato1984.
M. Sato and T. Sakurai, Qute: A functional language based on unification, Departement of information science, University of Tokyo, 1984.

Siekmann1979.
J. Siekmann, "Unification of Commutative Terms," Proceedings of Conference on Symbolic and Algebraic Manipulation, 1979.

Siekmann1984.
J. Siekmann and P. Szabo, "Universal Unification," Proceedings 7th international Conference on Automated Deduction, pp. 1-42, Napa Valley (California, USA), 1984.

Slagle1974.
J.R. Slagle, "Automated Theorem-Proving for Theories With Simplifiers, Commutativity And Associativity," J. of ACM, vol. 21, pp. 622-642, 1974.

Stickel1981.
M. Stickel, "A Complete Unification Algorithm for Associative-Commutative Functions," J. of ACM, vol. 28, no. 3, pp. 423-434, 1981.

Stickel1975.
M.E. Stickel, "A Complete Unification Algorithm for Associative-Commutative Functions," 4Th International Joint Conference on Artificial Intelligence, Tbilisi, 1975.

Szabo1982.
P. Szabo, "Unificationtheorie erster Ordnung," Doktorarbeit, Universitat Karlsruhe, 1982.

Vogel1978.
E. Vogel, "Morphismenunifikation," Diplomarbeit, Universitat Karlsruhe, 1978.

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.

Winkler1985.
F. Winkler, "Reducing the complexity of the Knuth and Bendix completion algorithm: A "unification" of different approaches"," Proceedings of Eurocal Conference LNCS, Linz (Austria), 1985.

Yelick1985.
K. Yelick, "Combining unification algorithms for confined equational theories," Proceedings of the RTA international conference, Dijon (France), 1985.

NOM DE L'ETUDIANT : KIRCHNER Claude

NATURE DE LA THESE : DOCTORAT ES SCIENCES

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 14 Juin 1985 n° 1102

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

RESUME

L'objectif de cette thèse est de présenter des outils pour l'étude et la conception d'algorithmes d'unification dans les théories équationnelles puis de les mettre en oeuvre sur des exemples significatifs.

Ces outils sans être universels constituent, à notre connaissance, la première tentative d'étude systématique de moyens utilisés pour concevoir et réaliser des algorithmes d'unification dans des théories équationnelles. L'approche unifiée qu'ils justifient, permet de faire de l'unification modulo des mélanges de théories.

Nous montrons comment ils permettent en particulier d'étudier avec succès des théories constituées d'axiomes permutatifs tels la commutativité, la transitivité ou encore l'associativité commutativité ou la distributivité droite ou gauche, mais aussi des théories non permutatives telle la théorie minus.

Par ailleurs nous montrons comment étendre la surréduction à toutes les réécritures équationnelles connues à ce jour et ce en définissant de façon abstraite la réécriture équationnelle sur les termes ainsi que la surréduction associée.

Enfin une partie de ce travail est constituée de la description du laboratoire de réécriture équationnelle REVEUR3, dans lequel le travail que nous décrivons ici joue un rôle central.