Université de Nancy 1
Faculté des sciences

Centre de Recherche en Informatique de Nancy

METHODES D'UNIFICATION PAR SURREDUCTION

Thèse de doctorat de l'université de Nancy 1
spécialité informatique
présentée par

Pierre Réty

soutenue le 22 Mars 1988

devant la commission d'examen

Président	J. P. Jouannaud
Rapporteurs	J. P. Haton
	J. P. Jouannaud
Examinateurs	L. Fribourg
	H. Ganzinger
	G. Huet
	C. Kirchner
	P. Lescanne

Université de Nancy 1
Faculté des sciences

Centre de Recherche en Informatique de Nancy

METHODES D'UNIFICATION PAR SURREDUCTION

Thèse de doctorat de l'université de Nancy 1
spécialité informatique
présentée par

Pierre Réty

soutenue le 22 Mars 1988

devant la commission d'examen

Président	J. P. Jouannaud
Rapporteurs	J. P. Haton
	J. P. Jouannaud
Examinateurs	L. Fribourg
	H. Ganzinger
	G. Huet
	C. Kirchner
	P. Lescanne

Remerciements

Je remercie très sincèrement tous les membres du jury:

Laurent Fribourg qui a apporté au jury ses compétences sur la surréduction, et par ses remarques a donné un point de vue orienté vers les langages de programmation,

Harald Ganzinger pour l'intérêt qu'il a porté sur cette thèse en partie écrite en français,

Jean Paul Haton qui a accepté d'être rapporteur et a bien voulu apporter au jury ses compétences en intelligence artificielle,

Gérard Huet qui a bien voulu s'intéresser à ce travail, et qui a apporté au jury ses compétences mondialement reconnues,

Jean Pierre Jouannaud qui a accepté d'être rapporteur malgré sa charge importante à Orsay, et qui me fait l'honneur de présider ce jury,

Pierre Lescanne et Claude Kirchner qui ont chacun leur tour dirigé la réalisation de cette thèse. Leur compétence, leurs critiques amicales et constructives, ainsi que leur dynamisme m'ont été très précieux.

Cette thèse a été effectuée au sein de l'équipe EURECA du CRIN et je voudrais remercier tous les membres de l'équipe pour leur soutien. J'ai toujours pu trouver parmis eux aide et disponibilité, quelle que soit la nature du problème à résoudre. Je les remercie aussi pour l'ambiance de travail agréable et efficace qu'ils ont su créer.

Je remercie Jalel Mzali pour m'avoir initié au fonctionnement des machines du laboratoire, et Pierre Marchand pour son aide sur les langages réguliers.

Je remercie enfin tous ceux et celles qui, d'une manière ou d'une autre, m'ont aidé et soutenu tout au long de ce travail, et en particulier Christine pour son soutien moral discret mais précieux, et pour ses encouragements.

Table des matières

Introduction

Résumé

L'objet de cette thèse est l'étude et l'amélioration de la surréduction, en vue de son application à l'unification, c'est à dire à la résolution d'équations dans les théories équationnelles. Nous nous situerons toujours dans le cadre d'une théorie équationnelle spécifiée par un système de réécriture confluent et nœthérien. Dans ce cadre général, la surréduction donne une méthode d'unification complète, puisqu'elle énumère un ensemble générateur de toutes les solutions de l'équation à résoudre.

Mais comme beaucoup de méthodes générales et complètes, la surréduction est peu efficace et ne termine pas toujours. Le premier objectif de cette thèse est d'améliorer l'efficacité et la terminaison de la surréduction, en utilisant des optimisations de la surréduction qui restent complètes dans les systèmes de réécriture confluents et nœthériens.

- Si l'espace de recherche est infini, nous montrons que dans certains cas il peut être décrit finiment, et les solutions peuvent alors être décrites par un langage régulier.
- Nous définissons une nouvelle stratégie de surréduction: la surréduction normalisante basique de gauche à droite, et établissons sa complétude. Par l'intermédiaire d'une propriété de commutation de la surréduction, nous donnons un résultat sur la minimalité de l'ensemble des solutions générées par cette méthode.

Ces optimisations ont été implantées dans une version expérimentale du logiciel de réécriture REVE, en utilisant le fait que la surréduction peut être vue comme une complétion un peu particulière.

Le deuxième objectif est de donner une formulation équationnelle à la surréduction basique, c'est à dire une formulation basée sur des systèmes d'équations et dans laquelle n'apparait jamais la notion de substitution. Les opérations sont décrites par des règles d'inférence. Ce travail a été conduit en cherchant:

- à ce que les règles d'inférence puissent être, autant que possible, exécutées sans contrôle et en particulier de manière parallèle
- à intégrer dans le processus de nombreuses règles de simplification
- à ce que la méthode puisse être facilement étendue à la logique ordo-sortée et/ou à des systèmes de réécriture équationnelle.

Position du problème

Le problème qui nous intéresse est la résolution d'équations (unification) dans les théories équationnelles, c'est à dire modulo des équations appelées axiomes. Considérons par exemple

un symbole binaire + idempotent et possédant un élément neutre à droite, ce qui donne les axiomes suivants:

$$x + 0 = x$$
$$x + x = x$$

et essayons de résoudre l'équation $x + y \doteq x$. c'est à dire de trouver quelles valeurs doivent prendre les variables x et y pour que les deux membres de l'équation deviennent égaux, sachant que + est idempotent et possède un élément neutre à droite. Il suffit évidemment de prendre $y = 0$. On dit alors que la substitution $\theta_1 = \{y/0\}$ est une solution de l'equation $x + y \doteq x$ ou encore est un unificateur de $x + y$ et x. Les substitutions $\theta_2 = \{y/x\}$, $\theta_3 = \{y/0, x/0\}$ sont aussi des solutions. Mais puisque θ_1 est une solution et n'affecte pas la variable x cela signifie que x peut prendre n'importe quelle valeur. Donc toute substitution de la forme $\{y/0, x/t\}$ où t est un terme quelconque est une solution. θ_3 est justement de cette forme, et on dit que θ_1 est plus générale que θ_3. On ne cherchera pas à décrire en extension l'ensemble des solutions, car il est souvent infini, mais plutôt un ensemble générateur (on dit aussi complet) des solutions. Ici $\{\theta_1, \theta_2\}$ est un ensemble complet minimal.

Comment résoudre ce type de problème ? Il existe bien sur des algorithmes ad hoc pour telle ou telle théorie, par exemple pour la commutativité, pour l'associativité-commutativité, etc... Mais on souhaiterait avoir des méthodes d'unification plus générales. Claude Kirchner [44,42,45] a cherché à systématiser la construction d'algorithmes d'unification. Il a décomposé le processus en trois opérations: décomposition, fusion, mutation, où seule la mutation dépend de la théorie considérée. La difficulté principale consiste alors à trouver une opération de mutation qui convienne.

Une autre approche, que nous allons développer dans ce travail, consiste à utiliser les techniques de réécriture, c'est à dire la surréduction. On commence pour cela par orienter les axiomes pour obtenir des règles de réécriture. Orientons les axiomes de l'exemple précédent:

$$r_1 : x + 0 \to x$$
$$r_2 : x + x \to x$$

et considérons l'équation $x + y \doteq a$ où a est un symbole de constante. La surréduction (notée $\rightsquigarrow$) consiste à appliquer une règle de réécriture sur un des termes de l'équation en affectant ses variables si nécessaire. Par exemple:

$$x + y \doteq a \rightsquigarrow_{[y/0, r_2]} x \doteq a$$

Pour appliquer la règle r_2 sur $x + y$ il faut changer y en x (ou x en y) par la substitution $\theta_1 = y/x$, ce qui donne $x + x$, lequel se réduit par r_2 en x. Nous obtenons donc l'équation $x \doteq a$. On voit que la surréduction comprend deux étapes: instanciation des variables pour qu'une règle devienne applicable, puis application de la règle. Or l'équation résultat $x \doteq a$ a pour solution syntaxique (sans faire intervenir r_1 ni r_2) $\theta_2 = x/a$. Alors la composée des deux substitutions

$$\theta = \theta_2 \circ \theta_1 = x/a, y/a$$

est solution de $x + y \doteq a$. De la même façon, en appliquant la règle r_1

$$x + y \doteq a \rightsquigarrow_{[y/0, r_1]} x \doteq a$$

on obtient la solution $\theta' = x/a, y/0$.

La surréduction a été introduite par Slagle [70], étudiée par Lankford [52], Fay [17, 16], et Hullot [35,36], lequel a établi que l'ensemble des substitutions trouvées comme ci-dessus en parcourant toutes les dérivations de surréduction issues de l'équation à résoudre

est un ensemble complet de solutions, si le système de réécriture est confluent et nœthérien. Confluent signifie que les différentes réécritures issues d'un même terme doivent confluer vers un même terme, nœthérien (on dit aussi terminant) qu'il n'existe pas de dérivation de réécriture infinie. L'exemple précédent est un système confluent et nœthérien. Par contre si on avait orienté l'axiome $x + 0 = x$ en $x \to x + 0$, il n'aurait évidemment pas été nœthérien. La transformation d'un ensemble d'axiomes en un système de réécriture confluent et nœthérien peut être faite d'une manière systématique grâce à la procédure de complétion de Knuth-Bendix [50].

Nous nous sommes intéressés à la surréduction car le processus formé de la procédure de complétion suivie de la résolution d'équations par surréduction est une méthode systématique d'unification dans les théories équationnelles. Or l'unification équationnelle présente beaucoup d'intérêt puisqu'elle intervient dans de nombreux domaines. Citons en quelques-uns.

- Les langages de programmation logico-équationnels (comme par exemple SLOG [20] et EQLOG [26]): ce type de langage, qui se situe dans la lignée des langages de cinquième génération, est fondé sur la logique des clauses de Horn avec égalité, laquelle égalité est définie par des équations. Le procédé d'inférence n'est alors plus tout à fait la résolution de Prolog, mais une résolution dans laquelle il faudra unifier les paramètres des prédicats modulo des équations.
- En algèbre: citons l'exemple de Hullot [36,35], qui a trouvé grâce à la surréduction basique le premier algorithme complet et fini d'unification dans les quasi-groupes, et par la même a montré que l'unification dans cette théorie est décidable.
- Dans les domaines de l'intelligence artificielle: vision par ordinateur, compréhension du langage naturel, systèmes experts, ainsi que dans les bases de données.
- D'autres applications sont données dans [67].

Amélioration de l'efficacité et de la terminaison

La surréduction est peu efficace et ne termine pas toujours. Il y a redondance dans la recherche de solutions puisqu'il n'est pas rare que la même solution soit calculée plusieurs fois. Voici les améliorations que nous proposons dans cette thèse.

- La surréduction normalisante. Considérons le symbole $*$, possédant l'élément absorbant 0, lequel a 0 pour inverse:

$$r_1 : x * 0 \to 0$$
$$r_2 : i(0) \to 0$$

et résolvons par surréduction l'équation $e_0 : i(x) * (0 * y) \doteq 0$. Une possibilité est:

$$i(x) * (0 * y) \doteq 0 \rightsquigarrow_{[r_1, y/0]} (e_1 : i(x) * 0 \doteq 0) \quad \rightsquigarrow_{[r_1]} (e_2 : 0 \doteq 0)$$
$$\rightsquigarrow_{[r_2, x/0]} (e_3 : 0 * 0 \doteq 0) \rightsquigarrow_{[r_1]} (e_4 : 0 \doteq 0)$$

La branche qui conduit à e_2 donne la solution $\sigma_2 = y/0$, tandis que celle qui conduit à e_4 donne $\sigma_4 = y/0, x/0$. On voit que σ_2 est plus générale que σ_4, par conséquent la branche qui va de e_1 à e_4 est redondante. Or on s'aperçoit que l'étape qui va de e_1 à e_2 n'instancie pas les variables de e_1. Cette sorte d'étape sans instanciation est appelée étape de réécriture, et on dit que e_1 se réécrit en e_2. Par définition, un terme ou une équation est normalisé(e) ou en forme normale s'il (elle) est irréductible par réécriture. Ici e_1 n'est pas en forme normale.

Une idée pour éviter de générer e_4 consiste à remplacer e_1 par sa forme normale avant de poursuivre le processus de surréduction, autrement dit à considérer la relation formée d'une étape de surréduction suivie par la mise en forme normale de l'équation obtenue. On appelera cette relation surréduction normalisante, et on la notera $\leadsto$. Dans notre exemple e_1 serait remplacé directement par e_2 et du même coup la branche de e_1 à e_4 ne serait pas considérée. L'espace de recherche est réduit. Fay [17,16] a proposé une procédure réalisant la surréduction normalisante, et a prouvé sa complétude, i.e. le fait qu'elle fournisse un ensemble complet de solutions. Nous redonnons une preuve de complétude en faisant bien ressortir le concept de surréduction normalisante, et les raisons pour lesquelles cette méthode est complète.

- Suppression des branches subsumées. En cherchant à élaguer l'arbre de recherche, il est apparu que certaines branches étaient redondantes. Plus précisement, si e_0 est l'équation à résoudre et qu'il existe deux branches $e_0 \overset{*}{\leadsto}_{[\sigma]} e_k$ et $e_0 \overset{*}{\leadsto}_{[\sigma']} e'_n$, si $e'_n = \theta e_k$ et $\sigma' = \theta \circ \sigma$, alors il est inutile de calculer les étapes de surréduction issues de e'_n.

- Description des solutions par un langage régulier. Il est possible que les solutions de l'équation à résoudre ne puissent pas être décrites finiment, ce qui veut dire qu'il n'existe pas d'ensemble complet et fini de solutions. C'est le cas dans l'exemple suivant dû à Fages et Huet [14]: considérons les règles de réécriture:

$$\begin{aligned} r_1 &: \quad g(f(x,y)) \to g(y) \\ r_2 &: \quad f(0,x) \to x \end{aligned}$$

Les solutions de l'équation $g(x) \doteq g(a)$ sont
$\sigma_1 = \{x/a\}$
$\sigma_2 = \{x/f(x_1,a)\}$
$\sigma_3 = \{x/f(x_2,f(x_1,a))\}$
etc...
Lorsqu'on essaye de résoudre cette équation par surréduction, cela se traduit par la non-terminaison de la procédure, que la surréduction soit normalisante ou non:

$$g(y) \doteq g(a) \leadsto_{[r_1, y/f(x,y)]} g(y) \doteq g(a) \leadsto \ldots$$

L'arbre engendré par surréduction est rationnel. Nous avons montré d'une manière générale que lorsque l'arbre engendré par surréduction est infini mais rationnel, il suffit de le replier en un graphe, et un ensemble complet de solutions peut être décrit par un langage régulier. Nous obtenons donc dans ce cas un algorithme complet et fini d'unification.

- La surréduction normalisante basique de gauche à droite. Pour réduire l'espace de recherche, Hullot [35,36] a défini la surréduction basique et a prouvé qu'elle fournissait une méthode d'unification complète. Elle consiste à partager les termes en deux parties: les occurrences basiques et les occurrences protégées, et à ne surréduire qu'aux occurrences basiques. Herold [29] a amélioré cette méthode en définissant la surréduction basique de gauche à droite, dont il a prouvé la complétude. Dans le but de réduire encore l'espace de recherche, nous cherchons à combiner la surréduction basique de gauche à droite avec la surréduction normalisante en créant la surréduction normalisante basique de gauche à droite. La combinaison la plus simple ne fournit pas une procédure d'unification complète. Une combinaison plus compliquée a été nécessaire.

- Des résultats de minimalité. Dans le but d'étudier l'intérêt de la surréduction normalisante basique de gauche à droite, nous étudions la minimalité de l'ensemble des solutions trouvées par surréduction. Nous montrons qu'une certaine partie des solutions (les solution

de rang maximal) générées par surréduction basique de gauche à droite (Herold) est minimale, lorsque le système de réécriture considéré est régulier et sans paire critique. Nous établissons un résultat de commutation de la surréduction, grâce auquel nous montrons que l'ensemble des solutions de rang maximal générées par surréduction normalisante basique de gauche à droite est minimal, en supposant toutefois que le système de réécriture, en plus des hypothèses précédemment énoncées, est linéaire à droite. Cette optimisation nécessite donc une hypothèse supplémentaire pour obtenir le résultat de minimalité. Mais nous montrons par des exemples qu'elle peut améliorer la minimalité quand le système de réécriture n'est pas régulier ou possède des paires critiques. Ce critère de minimalité semble intéressant car c'est un moyen de mesurer la redondance des méthodes d'unification par surréduction.

Ces différentes relations ont été implantées dans une version expérimentale du logiciel de réécriture REVE [53]. Cette implantation a été faite en modifiant la procédure de complétion de Knuth-Bendix, d'après une idée de Dershowitz [12]. Nous donnons au chapitre 2 une description de l'implantation de la surréduction normalisante.

Formulation équationnelle de la surréduction basique

Cette aproche utilise une structure de donnée qui ne comporte que des systèmes d'équations. Les objets tels que substitution et ensemble d'occurrences basiques dont on avait besoin auparavent pour formuler la surréduction basique ont été supprimés. La surréduction et les simplifications s'expriment alors par des opérations simples sur cette structure de donnée mises sous forme de règles d'inférence. On montre qu'on peut transformer par ces règles l'équation à résoudre en un système d'équations ayant les mêmes solutions et en forme résolue, c'est à dire dont les solutions sont évidentes.

D'autre part, toutes les opérations auxquelles fait appel la surréduction ont été intégrées selon les mêmes principes. C'est pourquoi l'unification syntaxique ne fait pas appel à la notion de substitution, mais est exprimée par deux règles d'inférence, qui sont la décomposition et la fusion. Ainsi dans ce formalisme, l'unification syntaxique n'est pas définie de la façon habituelle, mais par un algorithme à la Martelli et Montanari [56].

La notion de simplification apparait clairement dans cette approche. La surréduction normalisante dont nous avons déjà parlé fait intervenir la réécriture. La réécriture conserve les solutions des équations qu'elle transforme, et c'est pourquoi nous dirons qu'elle est une règle de simplification. Elle sera formulée à l'aide de règles exprimant un algorithme de filtrage. D'autres simplifications sont intégrées, dont la décomposition et la collision des symboles décomposables, et le dépliage.

En vue d'une extension ultérieure de cette approche au cadre de la logique ordo-sortée ou dans les systèmes de réécriture équationnelle, cadres pour lesquels le calcul de la substitution surréductrice ne donne pas une substitution unique mais plusieurs, nous introduirons des disjonctions d'équations ou de systèmes d'équations.

Cette approche équationnelle donne une formulation simple et claire de la surréduction basique et de ses simplifications. Contrairement à la formulation traditionnelle de la surréduction, notre approche ne retourne pas un ensemble de substitutions mais un problème équationnel en forme résolue, c'est à dire un problème que l'on sait résoudre par d'autres méthodes. Si la résolution d'une équation est intégrée dans un processus plus vaste, spécifié lui aussi équationnellement, il est évidemment préférable de retourner un système en forme résolue équivalent à l'équation à résoudre plutôt que des substitutions. On espère que dans des cadres plus larges, comme par exemple dans les systèmes de réécriture équationnelle

ou en travaillant avec des diséquations, un système en forme résolue pourra exprimer à lui seul plusieurs solutions et qu'il ne sera jamais utile de calculer celles-ci explicitement. Cette formulation présentera alors un grand intérêt. Elle est exprimée à l'aide de règles d'inférence, et peut être vue comme une procédure indéterministe; elle permet par exemple d'exprimer la surréduction basique normalisante. La méthode de preuve devrait permettre l'adjonction aisée de nouvelles règles de simplification. Cette approche équationnelle peut être implantée très facilement et ce n'est pas le moindre de ses avantages, car bien des procédures dont l'expression est compliquée sont implantées sous une forme fortement modifiée, et il n'est plus sur que l'implantation exprime la même chose que la procédure. Elle peut de plus être aisément parallélisée puisque les règles d'inférence sont soumises à peu de contrôle.

Présentation des différents chapitres

Le chapitre 1 pose le cadre théorique nécessaire à la compréhension de la thèse. Il précise également les notations. Le chapitre 2 est composé de la version complète d'un article écrit en collaboration avec C. Kirchner, H. Kirchner, et P. Lescanne, et publié lors de la première conférence "on Rewriting Techniques and Application" [62]. Il présente la surréduction normalisante, son implantation dans REVE, et des améliorations. Le chapitre 3 est constitué d'un article publié lors de la deuxième conférence "on Rewriting Techniques and Applications" [61]. Il introduit la surréduction normalisante basique de gauche à droite, et établit sa complétude. Le chapitre 4 donne des résultats sur la minimalité des solutions générées par certaines méthodes de surréduction, ainsi que des exemples. Le chapitre 5 est composé d'un article écrit en collaboration avec W. Nutt et G. Smolka de l'université de Kaiserslautern, et soumis à publication. Il s'agit d'une première étape vers une approche équationnelle de la surréduction basique. Le chapitre 6 présente une formulation complètement équationnelle et précise la spécification de la réécriture.

Chapitre 1

Définitions générales

Nous présentons ici les bases théoriques nécessaires pour aborder les travaux présentés dans cette thèse. Les travaux présentés au chapitres 2, 3, 5, 6 se situent dans le cadre d'une théorie équationnelle non typée, spécifiée par un système de réécriture (non équationnelle) confluent et nœthérien. En vue de leur extension ultérieure au cadre d'une théorie équationnelle typée avec sous-types (logique ordo-sortée). spécifiée par un système de réécriture équationnelle, le chapitre 4 est présenté dans ce cadre étendu. Nous introduisons ici les bases de la logique ordo-sortée, puis la réécriture équationnelle ordo-sortée et la surréduction équationnelle ordo-sortée.

1.1 Signature et termes

Considérons trois ensembles de symboles, dénombrables et disjoints deux à deux;

- Les **symboles de type** (ξ, η).
- Les **symboles de fonctions** (f, g, h). A chaque symbole est associé un entier naturel appelé **arité**, ainsi qu'une ou plusieurs **déclarations de fonction** de la forme $f : \xi_1, \ldots, \xi_{arité(f)} \rightarrow \xi$. Les symboles d'arité zéro sont appelés **constantes**.
- Les **variables** (x, y, z). Chaque variable a un **type** τx qui est un symbole de type; on supposera que pour tout symbole de type ξ il existe une infinité de variables de type ξ.

On suppose que l'ensemble des symboles de type est partiellement ordonné par un ordre appelé **ordre de sous-type** et noté $<$.
Une **signature** $\mathcal{S}$ est formée par le produit cartésien de ces trois ensembles. Une signature est **finie** si l'ensemble des symboles de type est fini. Dans la suite une signature est fixée.

Un terme est une expression bien formée (c'est à dire qui respecte les arités et les types) sur les symboles de fonction et de variable. Plus précisément, un **terme de type** η est

1. soit une variable x telle que $\tau x \leq \eta$
2. ou a la forme $f(s_1, \ldots, s_n)$ et il existe une déclaration de fonction $f : \xi_1, \ldots, \xi_n \rightarrow \xi$ telle que $\xi \leq \eta$ et s_i est un terme de type ξ_i pour tout $1 \leq i \leq n$.

L'ensemble des variables de t est noté $V(t)$. Un terme qui ne contient pas de symbole de variable est dit **clos**. Les termes seront désignés par les lettres s, t, u. Avec cette définition, si t est de type η et $\eta < \xi$ alors t est aussi de type ξ.

Afin de manipuler les symboles ou les sous-termes d'un terme donné on aura besoin d'un moyen formel pour les désigner, à savoir la notion d'occurrence. Intuitivement, l'**occurrence** d'un symbole est une suite d'entiers qui désigne le chemin allant de la racine du terme au symbole considéré.
Soit t un terme et p une occurrence quelconque. Le symbole d'occurrence p dans t noté $t(p)$ est:

1. si t est une variable x, alors $t(p)$ est egal à x si $p = \epsilon$, et n'existe pas sinon,
2. si $t = f(s_1, \ldots, s_n)$, et on a $t(\epsilon) = f$, $t(i.q) = s_i(q)$ pour $1 \leq i \leq n$, et $t(q)$ n'est pas défini dans les autres cas.

On dit que p est une occurrence de t si et seulement si $t(p)$ est défini. On note $D(t)$ l'ensemble des occurrences de t et $O(t)$ l'ensemble des occurrences de non variable de t. Les occurrences sont partiellement ordonnées par l'ordre $<$ défini par $p < q \iff q = p.p'$ où p' est une occurrence.
De même on peut définir le sous-terme de t à l'occurrence p que l'on notera $t|p$. $s[p \leftarrow t]$ est l'expression obtenue en remplaçant dans s le sous-terme à l'occurrence p par t. Cette expression n'est pas forcément un terme car les contraintes de type peuvent être violées.

Une **équation** est un couple de termes, noté $s \doteq t$.

Une signature $\mathcal{S}$ est **régulière** si tout terme s admet un plus petit type τs, c'est à dire, s'il existe une unique fonction τ de l'ensemble des termes dans l'ensemble des symboles de type telle que s est un terme de type τs et $\tau s \leq \xi$ si s est un terme de type ξ.

Proposition 1.1 [72] *La régularité des signatures finies est décidable.*

La signature $\{a :\rightarrow A, a :\rightarrow B\}$ n'est pas régulière puisque la constante a n'a pas de plus petit type.
Dans la suite seules des signatures régulières seront considérées.

Une **spécification** S est la donnée d'une signature et d'un ensemble d'équations (appelés **axiomes**).

1.2 Substitutions

Une **substitution** est une application θ qui transforme un terme en un terme, telle que

1. pour toute constante a, on a $\theta(a) = a$
2. $\theta(f(s_1, \ldots, s_n)) = f(\theta(s_1), \ldots, \theta(s_n))$
3. si s est de type ξ alors $\theta(s)$ est de type ξ.

La condition (3.) signifie que l'application d'une substitution peut réduire le type, mais ne peut pas l'augmenter, soit formellement $\tau\theta(s) \leq \tau s$.

Les substitutions seront désignées par les lettres σ, θ, γ. La composée de deux substitutions est encore une substitution. La fonction identité sur les termes est appelée **substitution vide** et est notée Id. $D(\theta) = \{x \mid \theta(x) \neq x\}$ est appelé domaine de θ et $I(\theta) = \cup_{x \in D(\sigma)} V(\sigma(x))$ est l'ensemble des variables apportées par θ. Une substitution θ est **idempotente** si $\theta.\theta = \theta$, ce qui est équivalent à dire que $D(\theta)$ et $I(\theta)$ sont disjoints. Nous supposerons dans la suite que les substitutions considérées sont idempotentes. Soit W un ensemble de variables, on note $\sigma = \sigma' \ [W]$ si $\forall x \in W$, $\sigma(x) = \sigma'(x)$. La substitution σ est dite **close** si pour toute variable x, $\sigma(x)$ est un terme clos.

1.3 Egalité modulo des axiomes

Une **congruence** sur les termes est une relation d'équivalence $\sim$ stable par plongement, c'est à dire vérifiant: pour tout terme s, si $s[p \sim t$ et $s[p \leftarrow t]$ est un terme, alors $s \sim s[p \leftarrow t]$. La congruence $\sim$ est dite stable par instanciation si pour tous termes s, t et toute substitution σ, $s \sim t$ implique $\sigma(s) \sim \sigma(t)$. On appelle **égalité modulo E** ou **E-égalité** la plus petite congruence stable par instanciation telle que pour tout axiome $g = d$ de E et pour toute substitution σ, on ait $\sigma(g) \sim \sigma(d)$. La E-égalité sera notée $=_E$.

On dit que l'équation $t \doteq t'$ est un **théorème** si $t \sim t'$. On dit qu'elle est un **théorème inductif** si pour toute substitution close σ on a $\sigma(t) \sim \sigma(t')$.

1.4 Filtrage

Le problème du filtrage est particulièrement important puisqu'il est le moteur de la réécriture. Dans le but d'englober les différentes méthodes de réécriture équationelle dans un seul formalisme, nous introduisons le formalisme proposé par C. et H. Kirchner [44,48].

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 $\sigma(t) =_E t'$.
On appelle **méthode de filtrage modulo E** toute application $Filtre_E$ qui à deux termes t, t' associe un ensemble de substitutions tel que:

1. Toutes les substitutions de $Filtre_E(t, t')$ sont des E-filtres de t vers t'
2. le $\emptyset$-filtre de t vers t', s'il existe, appartient à $Filtre_E(t, t')$.

Remarque: Si $E = \emptyset$ on retrouve la notion habituelle de filtrage.

La notion de filtrage permet de définir un préordre sur les termes appelé **préordre de filtrage**, qui dépend de la méthode de filtrage $Filtre_E$ considérée:

$$t \leq_{Filtre_E} t' \text{ si et seulement si } Filtre_E(t, t') \text{ est non vide}$$

Ce préordre est compris entre le préordre de filtrage dans la théorie vide (quand $E = \emptyset$) noté $\leq_\emptyset$, dont on peut donner une caractérisation directe par:

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

et le préordre de E-filtrage (quand $Filtre_E$ retourne l'ensemble de tout les filtres modulo E), dont une caractérisation directe est:

$$t \leq_E t' \text{ si et seulement si il existe } \sigma \text{ telle que } \sigma(t) =_E t'.$$

Ce préordre s'étend aux substitutions: $\sigma \leq_{Filtre_E} \theta\,[V]$ où V est un ensemble de variables, si et seulement si il existe une substitution γ telle que pour toute variable $x \in V$, γ appartient à $Filtre_E(\sigma(x), \theta(x))$.

Le préordre $\leq_\emptyset$ est plus simplement noté $\leq$. Si $t \leq t'$ on dit que t est **plus général** que t' ou que t' est une **instance** de t. Cette définition s'étend aux substitutions.

1.5 Réécriture

Soit R un système de réécriture et E un ensemble d'axiomes. Posons $A = R \cup E$.
Une **règle de réécriture** $g \to d$ est une équation orientée vérifiant $V(d) \subseteq V(g)$. Un **système de réécriture** est une spécification dont les équations ont été orientées en règles de réécriture.

Définition 1.1 Une méthode de filtrage modulo E, $Filtre_E$, étant fixée, on dit que le terme t se réécrit en t', à l'occurrence p, par la règle $g \to d$ et le filtre σ, et on note

$$t \to^{RE}_{[p,g\to d,\sigma]} t'$$

si $\sigma \in Filtre_E(g, t|p)$ et $t' = t[p \leftarrow \sigma(d)]$. $\diamond$

$t \to^{RE}_{[p,g\to d,\sigma]} t'$ est appelée étape de RE-réécriture. On appelle relation de RE-réécriture la relation notée $\to^{RE}$ et définie par

$$t \to^{RE} t' \Longleftrightarrow \exists [p, g \to d, \sigma] \mid t \to^{RE}_{[p,g\to d,\sigma]} t'$$

La fermeture réflexive et transitive de $\to^{RE}$ sera notée $\overset{*}{\to}^{RE}$, la fermeture symétrique de $\overset{*}{\to}^{RE}$ sera notée $\overset{*}{\leftrightarrow}^{RE}$.

Une suite d'étapes de RE-réécriture est appelée **RE-dérivation**. On dit que le terme 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 réécrive en t'. On dit qu'une substitution σ est RE-normalisée si pour toute variable x, $\sigma(x)$ est un terme normalisé. Une forme normale (notée $t\!\downarrow$) de t est un terme RE-irréductible issu de t par réécriture.

Certaines approches de la réécriture équationnelle, comme celle décrite dans [51] simulent la réécriture induite par R dans les classes d'équivalence modulo E, en utilisant la réécriture notée $\to^{R/E}$ définie par $=_E \,.\, \to^R \,.\, =_E$. Cette réécriture ne peut pas être décrite par le formalisme précédent, ce qui n'est pas un hazard puisqu'on cherche à formaliser ici les diverses réécritures sur les termes et non pas sur les classes. On a l'inclusion

$$\to^{RE} \subseteq \to_{R/E}$$

Dans la réécriture non typée et non équationelle, les relations $=_R$ et $\overset{*}{\leftrightarrow}^R$ sont égales. En général, ce n'est pas vrai dans le cas de la réécriture équationnelle ou typée, où on a seulement $\overset{*}{\leftrightarrow}^{RE} \subset =_A$. En effet

1. La méthode de filtrage $Filtre_E$ ne peut évidemment pas être quelconque. Si $Filtre_E$ retourne un ensemble de substitutions trop petit, $=_A$ et $\overset{*}{\leftrightarrow}^{RE}$ ne seront pas égales.

2. Considérons la signature [72]

$$\mathcal{A} < \mathcal{B}$$

$$a, a' :\to \mathcal{A},\ \ b :\to \mathcal{B},\ \ f : \mathcal{A} \to \mathcal{A}$$

et le système de réécriture

$$R = \{a \to b,\ \ a' \to b\}$$

Par transitivité on a $a =_R a'$ donc d'après la stabilité par plongement $f(a) =_R f(a')$. La R-dérivation qui conduirait de $f(a)$ à $f(a')$ passe par $f(b)$ qui n'est pas un terme car mal typé. Donc $Non(f(a) \overset{*}{\leftrightarrow}^R f(a'))$.

Dans l'exemple ci-dessus, on peut créér par réécriture des expressions mal typées. Donnons une condition suffisante qui évite cela:

Définition 1.2 $\to^{RE}$ est dite **compatible** si et seulement si pour tout terme s et toute occurrence p de s, si $s|p \to^{RE} t$ alors $s[p \leftarrow t]$ est un terme. $\diamond$

Pour une relation de réécriture compatible, l'application d'une règle sur un terme n'est qu'un problème de filtrage, comme dans le cas non typé.

Définition 1.3 On dit que la relation $\to^{RE}$ est **E-noethérienne** ou **noethérienne modulo E** si $\to^{R/E}$ est noethérienne. $\diamond$

Définition 1.4 Posons $A = E \cup R$. Une paire de termes (t_1, t_2) est 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 $t_1 \overset{*}{\to}^{RE} t'_1$, $t_2 \overset{*}{\to}^{RE} t'_2$ et $t'_1 =_E t'_2$.

- $\to^{RE}$ est **Church-Rosser modulo E** si et seulement si pour tous termes t_1, t_2

$$t_1 =_A t_2 \text{ implique } t_1 \downarrow t_2$$

- $\to^{RE}$ est **confluente modulo E** si et seulement si pour tous termes t, t_1, t_2

$$t \overset{*}{\to}^{RE} t_1 \text{ et } t \overset{*}{\to}^{RE} t_2 \text{ implique } t_1 \downarrow t_2$$

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

$$t \overset{+}{\to}^{RE} t_1 \text{ et } t =_E t_2 \text{ implique il existe } t'_2 \text{ tel que } t_2 \to^{RE} t'_2 \text{ et } t_1 \downarrow t_2.$$

On obtient le résultat fondamental suivant:

Théorème 1.2 *Soit $A = R \cup E$. Si $\to^{RE}$ est compatible, noethérienne, confluente et cohérente modulo E, alors*

$$s =_A t \Longleftrightarrow s \downarrow t \Longleftrightarrow s \overset{*}{\leftrightarrow}^{RE} t$$

Preuve: 1) $s =_A t \Longrightarrow s \downarrow t$.
D'après la définition de $=_A$, il suffit de montrer que $\downarrow$ est une congruence satisfaisant toutes les instances des équations de A. La reflexivité et la symétrie de $\downarrow$ sont triviales.

Démontrons la transitivité par induction multiensemble sur la relation noethérienne $\rightarrow^{RE}$. Montrons que si le multiensemble à n éléments $\{s_1, \ldots, s_n\}$ vérifie $s_i \downarrow s_{i+1}$ pour tout $1 \leq i < n$, alors $s_1 \downarrow s_n$. Notons $\rightarrow^{RE}_{ext}$ l'extension multiensemble de $\rightarrow^{RE}$.
Pour tout $1 \leq i < n$, il existe t_i, t'_i tels que $s_i \stackrel{*}{\rightarrow}^{RE} t_i$, $s_{i+1} \stackrel{*}{\rightarrow}^{RE} t'_i$ et $t_i =_E t'_i$. Si pour tout $1 \leq i < n$, on a $s_i =_E s_{i+1}$, le théorème est prouvé. Sinon il existe au moins une dérivation de longueur non nulle $s_i \stackrel{+}{\rightarrow}^{RE} t_i$ (on supposera qu'elle est dans ce sens). Distinguons deux cas:
a) Si $s_i \stackrel{+}{\rightarrow}^{RE} t'_{i-1}$, on remplace dans le multiensemble $\{s_1, \ldots, s_n\}$ le terme s_i par les deux termes t'_{i-1}, t_i. Le multiensemble obtenu $\{s_1, \ldots, s_{i-1}, t'_{i-1}, t_i, s_{i+1}, \ldots, s_n\}$ est strictement plus petit au sens de $\rightarrow^{RE}_{ext}$. Par confluence modulo E, $t'_{i-1} \downarrow t_i$. Il est évident que $si - 1 \downarrow t'_{i-1}$ et $t_i \downarrow s_{i+1}$. L'hypothèse de récurrence pourra être appliquée sur ce multiensemble.
b) Sinon $s_i = t'_{i-1}$. on remplace dans le multiensemble $\{s_1, \ldots, s_n\}$ le terme s_i par le terme t_i. Le multiensemble obtenu $\{s_1, \ldots, s_{i-1}, t_i, s_{i+1}, \ldots, s_n\}$ est strictement plus petit au sens de $\rightarrow^{RE}_{ext}$. On a $t_{i-1} =_E s_i \stackrel{+}{\rightarrow}^{RE} t_i$. Par cohérence modulo E, $t_{i-1} \downarrow t_i$, d'où $s_{i-1} \downarrow t_i$. Il est par ailleurs évident que $t_i \downarrow s_{i+1}$. L'hypothèse de récurrence pourra être appliquée sur ce multiensemble.

D'après l'hypothèse de récurrence, il existe une paire de termes u_1, u_n issue de s_1, s_n par $\rightarrow^{RE}$, telle que $u_1 \downarrow u_n$. D'où $s_1 \downarrow s_n$.

$\downarrow$ est une congruence car si $s[p \downarrow t$ et $s[p \leftarrow t]$ est un terme, on a $s[p \stackrel{*}{\rightarrow}^{RE} u =_E u' \stackrel{*}{\leftarrow}^{RE} t$. D'après la propriété de compatibilité, $s \stackrel{*}{\rightarrow}^{RE} s[p \leftarrow u]$ et $s[p \leftarrow t] \stackrel{*}{\rightarrow}^{RE} s[p \leftarrow u']$, donc $s \downarrow s[p \leftarrow t]$. Il est évident que $\downarrow$ satisfait toutes les instances des axiomes de A.

2) $s \downarrow t \Longrightarrow s \stackrel{*}{\leftrightarrow}^{RE} t$. Evident.

3) $s \stackrel{*}{\leftrightarrow}^{RE} t" \Longrightarrow s =_A t$. Considérons la dérivation $s = s_1 \leftrightarrow^{RE} \ldots \leftrightarrow^{RE} s_n = t$, et prenons une étape intermédiaire quelconque $s_i \leftrightarrow^{RE} s_{i+1}$. Supposons que $s_i \rightarrow^{RE}_{[p,g \rightarrow d,\sigma]} s_{i+1}$. Alors $s_i|p =_E \sigma(g) =_R \sigma(d)$ c'est à dire $s_i|p =_A \sigma(d)$. Et puisque $s_{i+1} = s_i[p \leftarrow \sigma(d)]$ par propriété de congruence $s_i =_A s_{i+1}$ On démontre qu'il en est de même si $s_i \stackrel{RE}{\leftarrow} s_{i+1}$. Par transitivité, on conclut que $s =_A t$. □

Ainsi la donnée d'un algorithme de décidabilité de la E-égalité et d'un algorithme de filtrage modulo E permet de décider l'égalité modulo A.
Remarque: L'égalité $=_A$ est définie de manière syntaxique. Elle est équivalente à la validité dans tous les modèles si aucun type n'est vide.
Pour obtenir un système de réécriture confluent et cohérent modulo E dans le cadre de la logique ordo-sortée, voir [25].

1.6 Unification

L'unification est le problème de la résolution d'équations.

1.6.1 Définitions

Soit $\mathcal{S}$ une signature et E un ensemble d'axiomes. On appelle **E-unificateur** ou **unificateur modulo** E de deux termes t et t' toute substitution σ telle que $\sigma(t) =_E \sigma(t')$. On note par $EU_{(\mathcal{S},E)}(t,t')$, ou plus simplement par $EU_E(t,t')$ si ce n'est pas ambigu, l'ensemble des unificateurs de t et t'. Le plus souvent cet ensemble est infini, et on s'intéresse généralement à

un sous-ensemble générateur de $EU_E(t,t')$ appelé ensemble complet de E-unificateurs. Cette notion, fondamentale pour l'unification, a été définie pour la première fois par Plotkin [59]. Formellement, tout ensemble Σ de substitutions tel que:

1. $\Sigma \subseteq EU_E(t,t')$
2. $\forall \sigma \in EU_E(t,t'),\ \exists \theta \in \Sigma,\ \theta \leq_E \sigma$

est appelé **ensemble complet** de E-unificateurs de t et t'. Σ est de plus dit **minimal** si

$$\forall \sigma, \theta \in \Sigma,\ \sigma \leq_E \theta \Longrightarrow \sigma = \theta$$

Lemme 1.3 [Fages et Huet [15]]
S'il existe, l'ensemble complet minimal des E-unificateurs de t et t' est unique à un renommage de variables près.

Preuve: Soit B, C deux ensembles minimaux et complets de E-unificateurs de t et t'. Soit $\beta \in B$. Puisque C est complet, il existe $\gamma \in C$ telle que $\gamma \leq_E \beta$. Puisque B est complet il existe $\beta' \in B$ telle que $\beta' \leq_E \gamma$. Donc $\beta' \leq_E \beta$ et par minimalité de B il résulte $\beta = \beta'$. D'où $\gamma = \beta$ à un renommage de variables près, par conséquent B est inclus dans C à un renommage de variables près.
On démontre de même que $C \subseteq B$ à un renommage près. □

S'il existe, l'ensemble complet minimal des E-unificateurs de t et t' est noté $ECMU_E(t,t')$. Il est défini à un renommage de variables près. Par convention, il est pris égal à l'ensemble vide lorsque t et t' ne sont pas unifiables. S'il existe un ensemble complet fini de E-unificateurs de deux termes, alors il existe un ensemble minimal complet obtenu en supprimant les substitutions redondantes. Dans le cas contraire, il n'existe pas toujours d'ensemble complet et minimal de E-unificateurs, comme le montre un exemple de Fages et Huet [15], ou bien dans le cas de la théorie associative et idempotente [66], et dans les semigroupes idempotents [1].

En se basant sur l'existence et la cardinalité des ensembles minimaux complets d'unificateurs, on peut classifier les spécifications (Siekmann et Szabo [68]). Mais Bürckert, Herold, et Schmidt-Schauss [6] ont montré qu'il existait une théorie qui n'est pas de type zéro lorqu'on résoud des équations, mais qui devient de type zéro si on résoud des systèmes d'équations. Ils ont alors donné une classification qui prend en compte l'unification sur les systèmes d'équations. C'est cette classification que nous donnons ici. Nous supposons pour cela que les définitions précédentes sont trivialement étendues aux systèmes d'équations. Une spécification (S, E) est dite

- de **type unitaire** si tout système d'équations a un ensemble minimal complet de E-unificateurs contenant au plus un élément.
- de **type finitaire** si tout système d'équations a un ensemble minimal complet de E-unificateurs fini.
- de **type infinitaire** si tout système d'équations a un ensemble minimal complet de E-unificateurs, et il existe au moins un système d'équations dont l'ensemble minimal complet est infini.
- de **type zéro** s'il existe au moins un système d'équations qui n'admet pas d'ensemble minimal complet d'unificateurs.

Une procédure d'unification est une procédure qui prend en entrée une équation et fournit des E-unificateurs de cette équation. On dit que la procédure est complète si pour toute équation elle fournit en un temps fini un ensemble complet de E-unificateurs. On dit de plus qu'elle est minimale si cet ensemble est minimal.

Pour une spécification (S, E) on peut distinguer trois types de problèmes relatifs à l'unification:

1. Peut-on décider si deux termes sont E-unifiables ?
2. Quel est le type de l'unification ?
3. Si l'unification dans (S, E) n'est pas de type zéro, trouver une procédure (efficace si possible) d'unification qui énumère un ensemble minimal complet d'unificateurs.

Nous nous intéresserons dans cette thèse au troisième problème.

En vue de définir la surréduction dans les systèmes de réécriture équationnelle, nous introduisons le formalisme de C. Kirchner [44] qui englobe les différentes méthode de réécriture équationnelle.

Définition 1.5 Soit E un ensemble d'axiomes. On appelle **méthode d'unification modulo E** toute application $UNIF_E$ qui à deux termes t, t' associe un ensemble de substitutions $UNIF_E(t, t')$ tel que:

1. $UNIF_E(t, t') \subseteq EU_E(t, t')$
2. $V(t) \cap V(t') = \emptyset \Longrightarrow Filtre_E(t, t') \subseteq UNIF_E(t, t')$
3. pour tous termes g, t et pour toute substitution ρ telle que $D(\rho) \cap V(g) = \emptyset$ on a $\gamma \in UNIF_E(g, \rho(t)) \Longrightarrow \gamma.\rho \in UNIF_E(g, t)$.

$\diamond$

Remarque: La condition (2) assure la cohérence entre la méthode de filtrage et la méthode d'unification utilisées, la condition (3) assure que l'ensemble des unificateurs est suffisamment stable pour que l'on puisse retrouver les propriétés classiques.

Définition 1.6 L'application $Unif_E$ qui à deux termes associe un ensemble de substitutions est appelée générateur complet des unificateurs de la méthode $UNIF_E$ si pour tous termes t, t':

1. $Unif_E(t, t') \subseteq UNIF_E(t, t')$ (correction)
2. $\forall \sigma' \in UNIF_E(t, t'),\ \exists \sigma \in Unif_E(t, t'),\ \sigma \leq_{Filtre_E} \sigma'\ [V(t) \cup V(t')]$ (complétude)

On dit que $Unif_E$ est minimal si pour tous termes t, t':

$$\forall \sigma, \rho \in Unif_E(t, t'),\ (\sigma \leq_{Filtre_E} \rho \Longrightarrow \sigma = \rho)$$

1.6.2 Résultats existants

Rappelons les principaux résultats connus. Distinguons plusieurs cas en fonction de la spécification (S, E).

1. S ne contient qu'une sorte et $E = \emptyset$. On parle alors d'unification syntaxique et un $\emptyset$-unificateur est souvent appelé **unificateur syntaxique**. L'unification syntaxique est unitaire, c'est à dire que l'ensemble complet minimal des unificateurs est formé d'un seul élément appelé **unificateur principal**. Le premier algorithme d'unification syntaxique se trouve dans les travaux de Herbrand [27]. Plus tard, différents algorithmes ont été étudiés. Citons pour mémoire Robinson [63], ainsi que Martelli et Montanari [57].

Présentons sous la forme de règles d'inférence un algorithme d'unification syntaxique dérivé de celui de Martelli et Montanari. Il s'appuie sur des conjonctions d'équations (ou systèmes d'équations). Le symbole $\wedge$ est le symbole de conjonction. F est un nouveau symbole signifiant que le système n'est pas unifiable.

(Décomposition-$\doteq$) $\quad \dfrac{S \wedge f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)}{S \wedge s_1 \doteq t_1 \wedge \ldots \wedge s_n \doteq t_n}$

(Collision-$\doteq$) $\quad \dfrac{S \wedge f(\ldots) \doteq g(\ldots)}{F}$ si $f \neq g$

(Fusion-$\doteq$) $\quad \dfrac{S \wedge x \doteq t \wedge x \doteq t'}{S \wedge x \doteq t \wedge t \doteq t'}$ si t n'est pas une variable

Ces règles retournent ou bien F, ou bien un système de la forme $x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n$ où les variables $x_1, \ldots, x_n$ sont toutes différentes. Dans ce dernier cas il reste à vérifier que le système ne possède pas de cycle de variables (voir [33,57]).

2. S ne contient qu'une sorte et $E \neq \emptyset$. Les problèmes relatifs à l'unification ont été étudiés pour des théories particulières. La liste des résultats connus est donné dans [67]. Les liens entre des classes particulières de théories (régulière, collapse, finie, etc...) et le type d'une théorie ont été étudiés dans [7].

Le problème de la recherche d'algorithmes d'unification a été étudié d'une manière générale. En premier lieu dans le cadre des systèmes de réécriture confluents et nœthériens grâce à la relation de surréduction. Nous en parlerons de manière détaillée au paragraphe suivant. Une autre approche, élaborée par C. Kirchner [43,42], cherche à systématiser la construction d'algorithmes d'unification. Cette méthode utilise trois opérations: décomposition et fusion définies comme précédemment par les règles (Décomposition) et (Fusion) (à la différence près que la structure de base est la multiéquation au lieu de l'équation) qui ne dépendent pas de la théorie considérée, et d'une opération de mutation, qui elle, en dépend. La difficulté réside alors dans la recherche d'opérations de mutation. La surréduction peut dans certains cas être utilisée comme une opération de mutation. Une troisième approche a été présentée par Gallier et Snyder [24]. Il s'agit d'une procédure générale et complète de E unification décrite par des règles de transformation sur des systèmes d'équation. Elle est évidemment inefficace en général.

3. S contient plusieurs sortes. L'étude de l'unification dans les algèbres ordo-sortée est récente puisqu'elle remonte seulement à 1983 [11]. Dans ces algèbres, l'unification syntaxique n'est plus unitaire.

Proposition 1.4 [Schmidt-Schauss [65]]
Dans les signatures finies et régulières l'unification est finitaire.

C. Kirchner [46] propose un algorithme d'unification équationnel dans les algèbres ordo-sortées grâce a une extension de la méthode d'unification par décomposition, fusion, mutation.

1.7 Surréduction

La surréduction peut être abordée selon deux orientations suivant le but recherché. Si le but est de résoudre des équations dans des structures algébriques (unification), il s'agit d'une orientation "preuves de théorème'. On souhaite alors trouver des méthodes d'unification complètes dans des cas très généraux. La complétude est privilégiée par rapport à l'efficacité. Si par contre le but recherché est d'utiliser la surréduction pour interpréter des langages de programmation, l'efficacité devient primordiale. On se contente alors de la complétude sous des hypothèses restrictives. Dans cette thèse, nous avons choisi la première orientation. Nous utilisons la surréduction comme un moyen de résoudre des problèmes d'unification, dans des systèmes de réécriture confluents et nœthériens.

Nous introduisons d'abord les définitions et résultats de base. Puis nous aborderons diverses stratégies améliorant la surréduction, en les classifiant selon qu'il s'agisse de stratégies complètes dans les systèmes de réécriture confluents et nœthériens, ce qui correspond plutôt à l'orientation "preuve de théorème', ou bien qu'il s'agisse de stratégies complètes sous des hypothèses supplémentaires, mais plus efficaces, ce qui correspond à la deuxième orientation. Enfin, nous citerons quelques autres travaux. Etant donné l'orientation de nos recherches, nous détaillons surtout les travaux effectués sur les stratégies complètes dans les systèmes de réécriture confluents et nœthériens.

1.7.1 Les bases

Surréduire consiste à instancier pour faire apparaitre un sous-terme réductible, et à réécrire. La surréduction se différencie donc de la réécriture par l'ajout d'une phase d'instanciation.

La définition ci-dessous se situe dans le cadre très général d'une spécification ordo-sortée définie à l'aide d'un système de réécriture équationnel.

Définition 1.7 Une méthode d'unification modulo E, $Unif_E$, étant fixée, on dit que le terme t se surréduit en t', à l'occurrence p, par la règle $g \to d$ et avec l'unificateur σ, et on note

$$t \leadsto^{RE}_{[p,g\to d,\sigma]} t'$$

si $\sigma \in Unif_E(t|p, g)$ et $t' = \sigma(t[p \leftarrow d])$.
σ est appelée **substitution surréductrice**. $t \leadsto^{RE}_{[p,g\to d,\sigma]} t'$ est appelée étape de RE-surréduction. On appelle relation de RE-surréduction la relation notée $\leadsto^{RE}$ et définie par

$$t \leadsto^{RE} t' \Longleftrightarrow \exists\, (p, g \to d, \sigma) \mid t \leadsto^{RE}_{[p,g\to d,\sigma]} t'$$

La fermeture réflexive et transitive de $\leadsto^{RE}$ sera notée $\overset{*}{\leadsto}^{RE}$. Une suite d'étapes de RE-surréduction est appelée **RE-surdérivation**. $\diamond$

Remarque: La surréduction contient la réécriture et si $t \leadsto^{RE}_{[p,g\to d,\sigma]} t'$ alors $\sigma(t) \to^{RE}_{[p,g\to d]} t'$. Dans le cas non équationnel ($E = \emptyset$), on remplacera la condition $\sigma \in Unif_E(t|p, g)$ par $\sigma \in ECMU(t|p, g)$.

L'étude de la surréduction dans le cadre ordo-sorté est récente [72]. Dans les systèmes de réécriture équationnelle non sortés, la surréduction a d'abord été étudiée pour la R,E-réécriture [39], puis dans un formalisme (utilisé dans la définition ci-dessus) intégrant toutes les formes de réécriture équationnelle (C. Kirchner [44], H. Kirchner [48]).

Dans la suite de cette section, nous supposons que $E = \emptyset$ et que la spécification n'est pas sortée. Le système de réécriture R est fixé et implicite. Pour une meilleure lisibilité, nous redonnons sous ces hypothèses la définition de la surréduction.

Définition 1.8 On dit que le terme t se surréduit en t', à l'occurrence p, par la règle $g \to d$, et avec l'unificateur σ, et on note

$$t \leadsto_{[p,g\to d,\sigma]} t'$$

si σ est l'unificateur principal de $t|p$ et g, et $t' = \sigma(t[p \leftarrow d])$. $\diamond$

Exemple 1.1 Soit la règle de réécriture r : $f(0) \to 0$. Alors

$$h(f(x)) \leadsto_{[1,r,x/0]} h(0)$$

On retrouve ici la définition de Hullot [36,35]. Pour prouver que la surréduction fournit une procédure complète d'unification, Hullot a d'abord montré qu'il existait une correspondance entre réécriture et surréduction:

Proposition 1.5 [Hullot] *Soient t_0 un terme et ρ_0 une substitution normalisée. Posons $t'_0 = \rho_0(t_0)$. A toute dérivation issue de t'_0:*

$$t'_0 \to_{[p_0,g_0\to d_0]} t'_1 \to \ldots \to_{[p_{n-1},g_{n-1}\to d_{n-1}]} t'_n \quad (1)$$

on peut associer une surdérivation issue de t_0 et effectuée aux mêmes occurrences et par les mêmes règles:

$$t_0 \leadsto_{[p_0,g_0\to d_0,\sigma_0]} t_1 \leadsto \ldots \leadsto_{[p_{n-1},g_{n-1}\to d_{n-1},\sigma_{n-1}]} t_n \quad (2)$$

et pour tout $i \in \{1, \ldots, n\}$ une substitution ρ_i telle que

- $t'_i = \rho_i(t_i)$
- $\rho_0 = \rho_i.\sigma_{i-1} \ldots \sigma_0\ [V(t_0)]$

Nous avons défini la surréduction sur les termes. Il est facile de l'étendre aux équations en considérant que $\doteq$ est un nouveau symbole binaire de fonction, les équations apparaissant alors comme des termes. Si $t \doteq t' \overset{*}{\leadsto}_{[\sigma]} s \doteq s'$ où σ est la composée des substitutions surréductrices, et si s et s' sont syntaxiquement unifiables par μ, alors $\mu.\sigma(t \doteq t') \overset{*}{\to} \mu(s) \doteq \mu(s')$. Puisque $\mu(s) = \mu(s')$ il résulte $\mu.\sigma(t) =_R \mu.\sigma(t')$, donc $\mu.\sigma$ est une solution de $t \doteq t'$.

Réciproquement, soit θ une solution normalisée de l'équation $t \doteq t'$. Si le système de réécriture R est confluent et nœthérien, alors $\theta(t \doteq t') \overset{*}{\to} s \doteq s$. D'après la proposition précédente $t \doteq t' \overset{*}{\leadsto}_{[\sigma]} t_n \doteq t'_n$ où σ est la composée des substitutions surréductrices, et il existe μ tel que $\theta = \mu.\sigma\ [V(t \doteq t')]$.

La surréduction fournit donc une procédure d'unification correcte et complète dans les systèmes de réécriture confluents et nœthériens.

Théorème 1.6 [Hullot] *Soit R un système de réécriture confluent et nœthérien. L'ensemble des substitutions σ telles que*

1. *il existe une surdérivation*

$$t_0 \doteq t'_0 \leadsto_{[\sigma_1]} \leadsto \ldots \leadsto_{[\sigma_n]} t_n \doteq t'_n$$

telle que t_n et t'_n soient syntaxiquement unifiables par l'unificateur principal θ

2. *$\sigma = \theta.\sigma_n \ldots \sigma_1$ et σ est normalisée*

est un ensemble complet de R-unificateurs de t_0 et t'_0.

Exemple 1.2 Considérons deux opérations binaires $+$, $*$. On suppose que $+$ est idempotent, admet un élément neutre à droite 0, et que 0 est absorbant pour $*$.

$$R = \{ \begin{array}{lll} r_1: & y+y & \to y \\ r_2: & y+0 & \to y \\ r_3: & y*0 & \to 0 \end{array} \}$$

R est confluent et nœthérien. En résolvant $x + (0 * y) \doteq 0$ par surréduction, on obtient:

$$\begin{array}{lll} x+(0*y) \doteq 0 & \leadsto_{[r_1, x/0*y]} (0*y) \doteq 0 \leadsto_{[r_3, y/0]} 0 \doteq 0 & (1) \\ & \leadsto_{[r_3, y/0]} x + 0 \doteq 0 \quad \to_{[r_2]} \; x \doteq 0 & (2) \\ & \qquad\qquad\qquad\qquad \leadsto_{[r_1, x/0]} 0 \doteq 0 & (3) \end{array}$$

La branche (1) donne la solution $(x/0*0, y/0)$ qui n'est pas normalisée. Les branches (2) et (3) donnent toutes deux la solution $(x/0, y/0)$. Le singleton $\{(x/0, y/0)\}$ est donc un ensemble complet de R-unificateurs de $x + (0 * y)$ et 0.

Les premiers travaux sur la surréduction (narrowing en anglais) se trouvent dans les articles de Slagle [70], Lankford [52] et Fay [17,16]. Dans ces articles, l'étape de narrowing était la composée d'une étape de surréduction (au sens de la definition 1.8) suivie par la mise en forme normale du résultat. Nous préférons lui donner le nom d'étape de surréduction normalisante, Mais contrairement à Fay, on ne supposera pas que le terme à surréduire est en forme normale.

Définition 1.9 Une étape de surréduction normalisante est définie par:

$$t \leadsto\!\!\!\!\to_{[p,g\to d,\sigma]} t' \iff t \leadsto_{[p,g\to d,\sigma]} t_1 \text{ et } t' = t_1\downarrow$$

$\diamond$

Théorème 1.7 [Fay] *Le théorème 1.6 est encore vrai pour la surréduction normalisante.*

Pour désigner la surréduction normalisante, on trouve parfois l'expression "narrowing with eager reduction", comme dans [38].

1.7.2 Stratégies complètes

Soit σ une solution de $t \doteq t'$, la proposition 1.5 montre qu'à chaque dérivation (1) allant de $\sigma(t \doteq t')$ à sa forme normale, on peut associer une surdérivation (2) qui, d'après le théorème 1.6, fournit une solution plus générale ou égale à σ. Comme il existe généralement plusieurs dérivations allant de $\sigma(t \doteq t')$ à sa forme normale, il y a redondance. L'idée est alors de limiter leur nombre en introduisant une stratégie ST de normalisation de $\sigma(t \doteq t')$, à laquelle on associera une stratégie ST' de surréduction de $t \doteq t'$. Pour que cette méthode reste complète, il faut, comme dans la proposition 1.5, qu'à toute dérivation (1) issue de $\sigma(t \doteq t')$ suivant la stratégie ST, on puisse associer une surdérivation (2) issue de $t \doteq t'$ suivant la stratégie ST'. Etudions les différents cas possibles.

1. ST est la stratégie de bas-en-haut (innermost). Il est bien évident que (1) est de bas-en-haut n'implique pas que (2) soit de bas-en-haut car (2) est moins instanciée que (1). Par contre, après une étape de réécriture de bas-en-haut $t'_i \to_{[p_i, g_i \to d_i, \sigma_i]} t'_{i+1}$, la partie de t'_{i+1} provenant de σ_i, c'est à dire située sous d_i, est normalisée. Elle ne pourra donc pas être réécrite. Alors dans (2), la partie de t_{i+1} située sous d_i n'est pas surréduite. Par conséquent, la solution σ peut être trouvée sans surréduire cette partie. Hullot a formalisé cette idée en définissant la surréduction basique, a prouvé sa complétude et a donné une condition suffisante de terminaison.

Définition 1.10 [Hullot] Soit une surdérivation

$$t_0 \leadsto_{[p_1, g_1 \to d_1, \sigma_1]} \leadsto \ldots \leadsto_{[p_n, g_n \to d_n, \sigma_n]} t_n$$

et $U_0, \ldots, U_n$ des ensembles d'occurrence de non variable de $t_0, \ldots, t_n$ respectivement. Cette surdérivation est dite **basique** si $U_0 = O(t_0)$ et pour tout i, $p_i \in U_i$ et

$$U_{i+1} = (U_i - \{v \in U_i,\ v \geq p_i\}) \cup \{p_i.v,\ v \in O(d_i)\}$$

Si $U_0 \neq O(t_0)$, cette surdérivation est dite basée sur U_0. $\diamond$

Par abus de langage, on utilisera le mot basique pour désigner une surdérivation basée sur un certain ensemble U_0 non spécifié. Pour chaque $i \in \{0, \ldots, n\}$, les éléments de U_i sont appelés **occurrences basiques**, tandis que les éléments de $O(t_i) - U_i$ sont appelés **occurrences protégées.**

Exemple 1.3

$$R = \{ \begin{array}{lll} r_1: & f(0,x) & \to x \\ r_2: & g(0) & \to 0 \end{array} \}$$

Le symbole % désigne les occurrences protégées. La surdérivation suivante est basique:

$$f(y, g(z)) \leadsto_{[2, r_2, z/0]} f(y, 0) \leadsto_{[\epsilon, r_1, y/0]} 0$$

Par contre, celle-là ne l'est pas:

$$f(y, g(z)) \leadsto_{[\epsilon, r_1, y/0]} \%g(z) \leadsto_{[\epsilon, r_2, z/0]} 0$$

Remarquons que toute réécriture de bas-en-haut est basique (la réciproque est fausse) et que la réécriture basique n'est pas confluente.

Théorème 1.8 [Hullot] *Le théorème 1.6 est encore vrai quand on se restreint à des surdérivations basiques (i.e. basées sur $O(t_0 \doteq t'_0)$).*

Proposition 1.9 [Hullot] *Si toute surdérivation basique issue d'un membre droit de règle de réécriture termine, alors toute surdérivation basique termine.*

En particulier, si les membres droits de toutes les règles de réécriture sont des variables, la surréduction basique termine.

2. *ST* est la stratégie de gauche à droite (leftmost). Mais dans la proposition 1.5, (1) est de gauche à droite n'implique pas que (2) le soit. Par contre, après une étape de réécriture de gauche à droite $t'_i \rightarrow_{[p_i,g_i \rightarrow d_i,\sigma_i]} t'_{i+1}$, la partie de t'_{i+1} située à gauche de p_i est normalisée. Elle ne pourra donc pas être réécrite. Alors dans (2), la partie de t_{i+1} à gauche de p_i n'est pas surréduite. Par conséquent, la solution σ peut être trouvée sans surréduire cette partie. Herold [29] a formalisé cette idée et l'a combinée avec la surréduction basique, pour obtenir la surréduction basique de gauche à droite.

Définition 1.11 Soient v, v' deux occurrences. On dit que v est à gauche de v', et on note $v \lhd v'$ s'il existe des occurrences p, w, w' et deux entiers j, j' tels que $j < j'$, $v = p.j.w$, $v' = p.j'.w'$. $\diamond$

Définition 1.12 [Herold] Soit une surdérivation

$$t_0 \leadsto_{[p_1,g_1 \rightarrow d_1,\sigma_1]} \leadsto \ldots \leadsto_{[p_n,g_n \rightarrow d_n,\sigma_n]} t_n$$

et $U_0, \ldots, U_n$ des ensembles d'occurrence de non variable de $t_0, \ldots, t_n$ respectivement. Cette surdérivation est dite **basique de gauche à droite** si $U_0 = O(t_0)$ et pour tout i, $p_i \in U_i$ et

$$U_{i+1} = (U_i - \{v \in U_i,\ v \geq p_i \text{ ou } v \lhd p_i\}) \cup \{p_i.v,\ v \in O(d_i)\}$$

Si $U_0 \neq O(t_0)$, cette surdérivation est dite basée de gauche à droite sur U_0. $\diamond$

Par abus de langage, on utilisera le mot basique de gauche à droite pour désigner une surdérivation basée de gauche à droite sur un certain ensemble U_0 non spécifié. Pour chaque $i \in \{0, \ldots, n\}$, les éléments de U_i sont appelés **occurrences basiques**, tandis que les éléments de $O(t_i) - U_i$ sont appelés **occurrences protégées**.

Exemple 1.4

$$R = \{r :\ g(0) \rightarrow 0\}$$

Le symbole % désigne les occurrences protégées. La surdérivation suivante est basique de gauche à droite:

$$f(g(x), g(y)) \leadsto_{[1,r,x/0]} f(0, g(y)) \leadsto_{[2,r,y/0]} f(0,0)$$

Par contre, celle-ci ne l'est pas:

$$f(g(x), g(y)) \leadsto_{[2,r,y/0]} f(\%g(x), 0) \leadsto_{[1,r,x/0]} f(0,0)$$

On peut remarquer que toute réécriture "leftmost-innermost" est basique de gauche à droite.

Théorème 1.10 [Herold] *Le théorème 1.6 est encore vrai quand on se restreint à des surdérivations basiques de gauche à droite (i.e. basées de gauche à droite sur $O(t_0 \doteq t'_0)$).*

On pourrait évidemment définir de la même manière la surréduction basique de droite à gauche. Bosco et al. [4, page 284] donnent une procédure indéterministe qui contient toutes les stratégies possibles dans le sens de la largeur. La surréduction basique de gauche à droite est donc un cas particulier de leur procédure.

3. *ST* est la stratégie de haut-en-bas (outermost). Il n'est pas possible d'opérer comme précédemment, car aucune partie d'un terme obtenu par une étape de réécriture de haut-en-bas n'est assurée être normalisée. La stratégie "outermost" de Martelli,Moiso,Rossi [54,55] n'est pas une stratégie de surréduction qui, dans la proposition 1.5, correspondrait à une stratégie de réécriture "outermost", mais est une formulation différente de la surréduction. En particulier, la surréduction n'est appliquée qu'en tête des membres des équations, mais le calcul de la substitution surréductrice ne provient pas d'une unification syntaxique, mais d'une unification modulo le système de réécriture R. Par rapport à l'approche traditionnelle (définition 1.8), cela peut changer l'ordre d'application des règles de réécriture, comme le montre l'exemple qui suit.

Exemple 1.5

$$R = \{\ \begin{array}{lrl} r_1 : & f(g(x')) & \rightarrow f'(x') \\ r_2 : & h(0) & \rightarrow g(0)\ \} \end{array}$$

L'équation $f(h(x)) \doteq f'(y)$ se surréduit en:

$$f(h(x)) \doteq f'(y) \leadsto_{[r_2,x/0]} f(g(0)) \doteq f'(y) \rightarrow_{[r_1]} f'(0) \doteq f'(y)$$

On trouve donc la solution $x/0,\ y/0$. Par la méthode de Martelli et al., on obtient

$$\begin{array}{ll} f(h(x)) \doteq f'(y) & \longrightarrow_{[r_1]} g(x') \doteq h(x) \ \wedge\ f'(x') \doteq f'(y) \\ & \longrightarrow_{[r_2]} h(x) \doteq h(0) \ \wedge\ g(x') \doteq g(0) \ \wedge\ f'(x') \doteq f'(y) \\ & \longrightarrow x \doteq 0 \ \wedge\ y \doteq 0 \end{array}$$

En outre, cette méthode contient d'une manière intrinsèque la détection des collisions (clashes), que nous allons introduire maintenant.

Il n'est pas rare que la procédure d'unification par surréduction cherche indéfiniment les solutions d'une équation qui n'en a manifestement pas.

Exemple 1.6

$$r :\ f(g(x)) \rightarrow f(x)$$

Il est clair que l'équation $h(f(y)) \doteq g(z)$ n'a pas de solution. En la résolvant par surréduction, l'espace de recherche est infini:

$$h(f(y)) \doteq g(z) \leadsto_{[r,y/g(y)]} h(f(y)) \doteq g(z) \leadsto \ldots$$

Dans cet exemple l'équation à résoudre n'a pas de solution car les symboles de tête de ses membres sont des symboles décomposables.

Définition 1.13 Soit R un système de réécriture. Le symbole de fonction (ou de constante) f est dit **décomposable** dans R s'il n'est le symbole de tête d'aucun membre gauche de règle de R. $\diamond$

Proposition 1.11 *Toute équation de la forme $f(\ldots) \doteq g(\ldots)$ où f, g sont des symboles décomposables et $f \neq g$, est sans solution. On dit qu'il s'agit d'un cas de* **collision**.

Si par contre $f = g$, on peut alors simplifier l'équation.

Proposition 1.12 *Toute équation e de la forme $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ où f est un symbole décomposable a les mêmes solutions que le système S d'équations $s_1 \doteq t_1, \ldots, s_n \doteq t_n$. La transformation de e en S est appelée* **décomposition sur les symboles décomposables**.

La décomposition sur les symboles décomposables, combinée avec la détection des collisions, font terminer la surréduction plus souvent, et réduisent l'espace de recherche. Ces idées ne sont pas nouvelles puisqu'on les trouve décrites par Fribourg dans [19] à l'aide de constructeurs. Malheureusement, elles ne s'appliquent plus dès qu'un des membres de l'équation a en tête un symbole non décomposable.

Exemple 1.7

$$r : f(g(x)) \to f(x)$$

L'équation $f(y) \doteq g(z)$ n'a pas de solution car toute surdérivation qui en est issu est de la forme $f(y) \doteq g(z) \overset{\cdot}{\rightsquigarrow} f(\ldots) \doteq g(\ldots)$, donc conduit à une équation non unifiable syntaxiquement. En la résolvant par surréduction, l'espace de recherche est infini:

$$f(y) \doteq g(z) \overset{\cdot}{\rightsquigarrow}_{[r, y/g(y)]} f(y) \doteq g(z) \overset{\cdot}{\rightsquigarrow} \ldots$$

L'approche de Sivakumar et Dershowitz [69] permet d'exploiter tous les symboles, y compris les symboles non décomposables. Elle est fondée sur le fait que pour trouver une solution d'une équation donnée, il faut la transformer par surréduction en une équation dont les membres ont les mêmes symboles de tête. Grâce à un graphe de portée d'opérateur, seules sont calculées les étapes de surréduction susceptibles de conduire à une solution. Cette méthode appliquée à l'exemple précédent aurait généré un espace de recherche fini.

1.7.3 Stratégies complètes sous des hypothèses supplémentaires

Introduisons d'abords quelques définitions. Supposons que les symboles de fonction soient divisés en constructeurs et en opérations définies. Un terme est dit "innermost" (Fribourg [22]) si son symbole de tête est une opération définie et si ses sous-termes stricts ne contiennent que des constructeurs et des variables. Un système de réécriture est dit "lhs-innermost" si les membres gauches des règles sont des termes "innermost". Dans un tel système, un même symbole ne peut pas être à la fois en tête d'un membre gauche d'une règle, et dans un sous-terme strict d'un membre gauche. On en déduit que le système

$$s(p(x)) \to x$$

$$p(s(x)) \to x$$

n'est pas "innermost".

Padawitz [58] a défini une surréduction linéaire "leftmost-outermost", et sous une hypothèse très restrictive (UNI page 252) montre qu'elle fournit une procédure d'unification

complète. You [74] propose le "outer narrowing", et montre sa complétude dans les systèmes de réécriture "lhs-innermost", linéaires à gauche, et sans paire critique. Il montre aussi que cette stratégie fournit des ensembles minimaux de filtres pour des termes ayant des variables disjointes. Une autre stratégie est donnée par le "lazy narrowing" de Reddy [60].

Fribourg [22] a défini la surréduction "innermost". Cette stratégie consiste à ne surréduire les termes qu'à une seule occurrence "innermost", dont le choix est indéterministe. Par rapport à la surréduction ordinaire, où toutes les occurrences des termes doivent être surréduites, cette stratégie apporte certainement un gain important d'efficacité. Mais sa complétude est assurée en supposant que le système de réécriture réécriture est "lhs-innermost" et que tout terme "innermost" clos est réductible. Dans [21], Fribourg donne une formulation de la surréduction "leftmost innermost" avec simplification par réécriture, et à l'aide de clauses de Horn.

La surréduction est utilisée dans de nombreux langages de programmation, comme dans EQLOG [26], SLOG [22], FUNLOG [73], QUTE [64]. Dans LEAF [3], la surréduction est directement intégrée dans le procédé d'inférence sous forme de "flat resolution". La procédure d'unification de Bosco et al. [4], dont nous avons déjà parlé, pourrait être intégrée directement dans un processus de résolution puisqu'elle est exprimée sous la forme de "flat SLD-resolution".

1.7.4 Autres travaux

Quand on résoud une équation $t \doteq t'$, on cherche en fait les substitutions σ telles que $\sigma(t \doteq t')$ soit un théorème. Mais il est fréquent de ne s'intéresser qu'à la résolution inductive d'équations: on dit qu'on résoud $t \doteq t'$ inductivement si on cherche les substitutions σ telles que $\sigma(t \doteq t')$ soit un théorème inductif. Fribourg [18] a montré que la surréduction pouvait apporter une réponse efficace à ce problème, à condition de connaitre des théorèmes inductifs, et d'intercaler entre les étapes de surréduction des étapes de réécriture par ces théorèmes considérés alors comme des règles de réécriture.

La complétude de l'unification par surréduction a été étudiée dans les systèmes de réécriture non nœthériens [76,75,31]. La surréduction a été étudiée et montrée complète [23, p. 37] dans les systèmes obtenus par la procédure de complétion sans échec "Unfailed Knuth-Bendix" [32,2], ces systèmes pouvant comporter des équations non orientées. Elle a été étendue au cas de la réécriture conditionnelle, lorsque la précondition est une conjonction d'équations [37,40,41].

Chapitre 2

La surréduction normalisante

Ce chapitre traite de la surréduction normalisante, relation formée d'une étape de surréduction suivie de la mise en forme normale du résultat. Il est constitué d'un article écrit en collaboration avec C. Kirchner, H. Kirchner et P. Lescanne, et publié sous forme abrégée [62] lors de la première "Conference on Rewriting Techniques and Applications".

Le logiciel REVE [53] est un laboratoire de réécriture, qui implante, entre autres fonctionnalités, la procédure de complétion de Knuth-Bendix. Il permet donc de fabriquer des systèmes de réécriture confluents et nœthériens. Nous savons que dans de tels systèmes, la relation de surréduction est une méthode d'unification complète. Nous avons voulu intégrer la surréduction dans REVE comme méthode d'unification, en utilisant une idée de Dershowitz [12], qui montrait que la surréduction pouvait être vue comme une complétion particulière. Nous avons développé cette idée, montré de manière précise comment transformer la procédure de complétion en procédure de surréduction, et implanté ainsi la surréduction dans REVE. Cette idée est aussi développée dans [13], dans un contexte de langage de programmation.

La complétion de REVE normalise systématiquement les paires critiques. Cette particularité n'a pas été modifiée, ce qui implante en réalité non pas la surréduction, mais la surréduction normalisante qui est plus efficace. Fay [16] montra sa complétude, i.e. la fait qu'elle donne une méthode d'unification complète. Nous donnons une autre preuve de complétude, qui présente l'avantage de bien faire ressortir les concepts de surréduction et de surréduction normalisante. D'autre part, les raisons pour lesquelles la surréduction normalisante est complète y apparaissent de manière plus explicite.

Dans le but de réduire l'espace de recherche, nous avons montré qu'il était inutile de parcourir les branches qui sont des instances d'autres branches. D'autre part, lorsque l'arbre des surréductions normalisantes issu de l'équation à résoudre est infini, ce qui n'est pas rare, la méthode reste complète mais ne permettra pas en pratique de trouver à coup sur un ensemble complet de solutions. Fages [14] a montré qu'on pouvait décrire l'arbre finiment quand la divergence était provoquée par des boucles du type $e_0 \stackrel{*}{\rightsquigarrow} e_0$, où e_0 est l'équation à résoudre. Nous avons généralisé cette idée en montrant que lorsque l'arbre est infini, mais que l'ensemble de ses nœuds est fini, un ensemble complet de solutions peut être décrit par un langage régulier, en considérant que les symboles du langage sont des substitutions et que la concaténation est la composition (en sens inverse) des substitutions. Ce langage sera en fait défini par son automate. Il est ainsi possible de décrire toutes les solutions d'une équation même s'il n'existe pas d'ensemble générateur fini.

La résolution linéaire de Prolog et la surréduction présentent des similarités. Mais, si on

compare la résolution avec la surréduction normalisante, on voit que cette dernière effectue en plus une étape de normalisation qui simplifie l'expression obtenue. Les langages de programmation logique qui combinent ces deux stratégies (résolution et simplification) sont de bons candidats pour succéder à Prolog.

L'article est structuré comme suit. Le paragraphe 1 introduit le sujet et expose un exemple d'éxécution de notre implémentation dans REVE. Le paragraphe 2 contient la preuve de complétude de la surréduction normalisante, une procédure appelée NARROWER qui ressemble beaucoup à la procédure de complétion de Knuth-Bendix, et la preuve que NARROWER réalise la surréduction normalisante. La propriété qui affirme que les branches qui sont instances d'autres branches sont inutiles est donnée et prouvée au paragraphe 3. Les quelques différences entre NARROWER et la procédure de complétion sont exposées au paragraphe 4. Le paragraphe 5 explique comment décrire l'ensemble des solutions par un langage régulier. Et pour terminer, le paragraphe 6 fait le lien entre la programmation logique et la surréduction normalisante. On trouvera en appendice l'exemple classique du coloriage de carte, résolu par NARROWER.

Dans ce papier, la surréduction s'appelle U-réduction (U pour unification) et se note $-\hat{}\rightarrow$, tandis que la surréduction normalisante s'appelle narrowing et se note $-\hat{}\hat{}\rightarrow$.

An algorithm for unification based on narrowing.(*)

Pierre RETY
Claude KIRCHNER
Hélène KIRCHNER
Pierre LESCANNE

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

1. INTRODUCTION

In this paper, an algorithm based on narrowing for solving equations in equational theories is proposed. The originality of the method is that we use the narrowing relation with normalization. Some results on narrowing are given and proved. The correctness and the completeness of the algorithm are also proved and an implementation called NARROWER is described. It is made by extending the software REVE. Some experiments are also presented. The algorithm and its implementation have the nice property that when the set of solutions is infinite, it can often describe the solutions as the words of a regular language, using a finite automata. We also try to show the connection of such an algorithm with the design of a logic programming language, the purpose of which is to solve equations.

We would like to explain our ideas on an example. As usual, we suppose that we provide a convergent set of rules that some people would call an "abstract data type" and others an "universe". We give an example in the classical frame of the integers. A convergent set of rules, later referred to as R, for describing properties of INTEGER is proposed in Figure 1.

(*) This research was supported by the GRECO of programmation.

```
% relations between constructors 1..2
s(p(x)) → x
p(s(x)) → x

% definitions of "+"
(0 + x) → x
(s(x) + y) → s((x + y))
(p(x) + y) → p((x + y))

% proprieties of "+"
(x + 0) → x
(x + s(y)) → s((x + y))
(x + p(y)) → p((x + y))
((x + y) + z) → (x + (y + z))

% definitions of "-"
-(0) → 0
-(s(x)) → p(-(x))
-(p(x)) → s(-(x))

% proprieties of "-"
-(-(x)) → x
(-(x) + x) → 0
x + -(x) → 0
(x + (-(x) + z)) → z
(-(x) + (x + z)) → z
-((x + y)) → (-(y) + -(x))

% definitions of "*"
(0 * x) → 0
(s(x) * y) → (y + (x * y))
(p(x) * y) → (-(y) + (x * y))

% proprieties of "*"
(x * 0) → 0
(x * s(y)) → ((x * y) + x)
(x * p(y)) → ((x * y) + -(x))
```

Figure 1. A possible specification of INTEGER.

In this framework, we may want to solve classical equations as we learn in high school, such as linear equations or quadratic equations. This works well with NARROWER. Suppose we would like to know the solution of the equation

$$2x - 1 = 1$$

that we code here into

$$x + x + p(0) = s(0).$$

We just type the command **narrow** followed by the equation and the system starts by adding a rule of the form

$$x === x \rightarrow \textbf{true}$$

whose superposition with a term means that a solution is discovered. The new set of rules will be called R1. NARROWER also creates a pair of terms of the form

$$\langle t === t', \textbf{solution}(x_1,\ldots,x_n)\rangle$$

where the right-hand side is for storing parts of the substitutions. Then it is ready to compute the solution of the equations. In some cases $t === t'$ unifies with $x === x$, which means that t can be made equal to t' by a substitution τ. $\tau(x_1,\ldots,x_n)$ is a solution of the equation. Otherwise the left-hand side $t === t'$ is compared with the left-hand side g of a rule $g \rightarrow d$ in R. More precisely a unifier of g with a subterm of t is looked for. If such a unifier σ exists, say at occurrence u, then NARROWER creates a new pair of the form

$$\langle \sigma(t === t')[u \leftarrow \sigma(d)],\ \textbf{solution}(\sigma(x_1),\ldots,\sigma(x_n))\rangle.$$

Usually, this last pair can be reduced by R, thus it could be more useful or more efficient to work only with the normal forms. Thus NARROWER appends a pair $\langle s === s', \textbf{solution}(t_1,\ldots,t_n)\rangle$ where $s === s'$, $t_1,\ldots,t_n$ are the normal forms of the previous terms. The process that goes from $t === t'$ to $s === s'$ is called **narrowing**. In the previous example, the system first reduces

$$\langle x + x + p(0) === s(0), \textbf{solution}(x)\rangle$$

to

$$\langle p(x + x) === s(0), \textbf{solution}(x)\rangle.$$

It then superposes the pair with $0 + x \rightarrow x$ which gives

$$\langle p(0) \equiv\equiv s(0),\ \textbf{solution}(0)\rangle$$

that leads to no solution. It also superposes, among others, with $s(x) + y \rightarrow s(x + y)$ which gives

$$\langle p(s(x + s(x)) \equiv\equiv s(0),\ \textbf{solution}(s(x))\rangle$$

before reduction and

$$\langle s(x + x) \equiv\equiv s(0),\ \textbf{solution}(s(x))\rangle$$

after normalization. This pair superposes with $0 + x \rightarrow x$ and produces

$$\langle s(0) \equiv\equiv s(0),\ \textbf{solution}(s(0))\rangle$$

which superposes with $x \equiv\equiv x \rightarrow$ **true** yielding

$$\langle \textbf{true},\ \textbf{solution}(s(0))\rangle$$

showing that x / s(0) is the solution. In Appendix, we will look at other example.

In Section 2, we introduce the basic concepts and notations, including the narrowing process and the NARROWER procedure. In Section 3, we describe how this procedure can be optimized and we give more details on our implementation in Section 4. In some cases the solutions i.e., the substitutions generated by the system are infinitely many, but they usually share the same structure and can be described as the words of a regular language. One of the originalities of the procedure proposed here is that it discovers regularities and suggests an finite automata to describe infinite sets of solutions. This feature is explained in Section 5. In Section 6, we use examples to show what can be expected from such a procedure from the logic programming point of view.

2. NARROWING

In this section we introduce the concept of narrowing and show that it provides a complete method for solving equation in a theory described by a confluent and noetherian term rewriting system. The following notations and properties are valid for the whole paper. They are consistent with [10,13].

Let A be an equational theory, and R a convergent (that is confluent and noetherian) term rewriting system equivalent to A. F is the set of symbols of A, X a set of variables, and T(F, X) the set of terms on F and X. For each $t \in T(F,X)$, V(t) is the set of variables that occurs in t. $\rightarrow$ is the rewriting relation derived from R and $t\downarrow R$ denotes the normal form of the term t using R. $t[u \leftarrow t']$ is the term obtained from t by changing the subterm of t at the occurrence u by t'.

Substitutions σ are defined as endomorphisms on T(F,X) that extend mappings from X to T(F,X) with a finite domain $D(\sigma)$. A substitution σ is denoted by $\{(x_1/t_1),\ldots,(x_n/t_n)\}$. $\sigma|W$ is the restriction of the substitution σ to the subset W of X and $I(\sigma)$ is the union of all $V(\sigma(x))$ for any x in $D(\sigma)$. We note $\sigma=\sigma'\ [W]$ iff $\sigma|W = \sigma'|W$.

We write $<$ the subsumption quasi-ordering on T(F,X) defined by: $t<t'$ iff $t' = \sigma(t)$ for a substitution σ (called a match from t to t'). Composition of substitutions σ and ρ is denoted by $\sigma.\rho$.

Given an equational theory A, two terms t and t' are said to be A-unifiable [17,9] iff there exists a substitution σ such that $\sigma(t) =_A \sigma(t')$. σ is also called an A-solution of the equation $t=t'$. Given a subset V of X, we define $\sigma \leq_A \sigma'[V]$ iff $\sigma' =_A \sigma''.\sigma\ [V]$ for a substitution σ''. If V=X, V is omitted. Σ is a complete set of A-unifiers of t and t' away from W containing the set V of the variables of t and t' iff:

- for all $\sigma \in \Sigma$, $D(\sigma) \subseteq V$ and $I(\sigma) \cap W = \emptyset$ (The goal of this technical restriction is only to avoid conflict between variables)
- for all $\sigma \in \Sigma$, $\sigma(t) =_A \sigma(t')$

- for all unifiers σ', there exists an $\sigma \in \Sigma$ such that $\sigma \leq_A \sigma'[V]$.

In addition Σ is said to be minimal if it satisfies the further condition:

for all σ and $\sigma' \in \Sigma$, $\sigma \leq_A \sigma'$ implies $\sigma = \sigma'$.

An A-unification algorithm is complete if it generates a complete set of A-unifiers. Note that this set may not be finite.

2.1. The narrowing: a method for solving equations in equational theories

The interest of the narrowing is to reduce the problem of unification in an equational theory A to the well-known problem of term unification, provided a convergent term rewriting system equivalent to A exists.

2.1.1. Definitions

Informally, the narrowing of a term consists of three steps. First, a substitution is used to create a subterm $\sigma(t)$ that matches a left hand side of a rule in R. Second, this rule is applied to $\sigma(t)$ in order to reduce it to s. The third step computes the R-normal form of s. Let us remark that if t is a term, W a finite set of variables containing V(t) and $g \rightarrow d$ a rule of R, it is always possible to rename the variables of g such that the intersection of V(g) and W is empty. W' is the union of V(g) and W.

Definition 1:[4] Let t and t' be two terms, u a non variable occurrence of t, $g \rightarrow d$ a rule of R and σ a substitution, we write

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

iff

- The subterm of t at the occurrence u (written $t|_u$) and g are unifiable, and σ is the most general unifier of these two terms away from W'.
- $t' = \sigma(t[u \leftarrow d])\downarrow R$

The substitution σ is called a **narrowing substitution**, the process that transforms t into t' is called **narrowing**, and a **narrowing derivation** (written -^*^-> or -^^->*) is any sequence of narrowings issued from a term. The transformation of t into $\sigma(t[u \leftarrow d])$ is sometimes called **paramodulation** (written -^->). We propose the name **U-reduction** (U for unification).

Remark: It is obvious that t -^^->[σ] t' implies $\sigma(t) \rightarrow^+ t'$.

2.1.2. Connection between narrowing and rewriting

This lemma shows the connection between some substitutions, the rewriting relation, and the U-reduction.

LEMMA: Let s be a normalized term, δ a substitution such that

$\delta|_{V(s)}$ is normalized

$\delta(s) \rightarrow[u, g \rightarrow d, \sigma] \ t_1$.

Then there exists a term s_1 and two substitutions θ and δ_1 such that:

s -^->$[u, g \rightarrow d, \theta]$ s_1

$\delta_1(s_1) = t_1$

$\delta_1|_{V(s_1)}$ is normalized

$\delta = \delta_1 . \theta \ [D(\delta) \cup V(s)]$

In addition, if $s_1 \rightarrow^* s_2$, then $t_1 \rightarrow^* t_2$ and we have $\delta_1(s_2) = t_2$. ○

Remark: The first part of the lemma says that if s is not reducible, but $\delta(s)$ is reducible at the occurrence u using $g \rightarrow d$, there exists a most general substitution θ such that $\theta(s)$ is reducible at the same occurrence by the same rule. This generates a U-reduction and produces a term s_1. If s_1 is reducible then the rules that reduce s_1 can be lifted by δ_1, for reducing t_1. So, we have the schema:

$$\begin{array}{ccccc} \delta(s) & \rightarrow[u, g \rightarrow d, \sigma] & t_1 & \rightarrow^* & t_2 \\ \uparrow & & \uparrow 1 & & \uparrow 2 \\ | \ \delta & & | \ \delta_1 & & | \ \delta_1 \\ | & & | & & | \\ s & \text{-^->}[u, g \rightarrow d, \theta] & s_1 & \rightarrow^* & s_2 \end{array}$$

One can see that the lemma is still valid when s is not a normalized term.

Proof: The first part is from J.M.Hullot[11]. The second part comes from the stability by instanciation of the rewriting relation. □

The next theorem makes up relation between rewriting and narrowing. A similar result was proved by J.M.Hullot[11] with the U-reduction. The difference with Hullot's result is the following. Since narrowing is used and then normalization are performed at each step, the correspondence

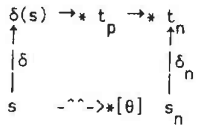
occurs only at the end.

THEOREM 1: Let s be a term, δ a substitution such that $\delta|V(s)$ is normalized.

With any derivation issuing from $\delta(s)$: $\delta(s) \rightarrow^* t_p$, we can associate a narrowing derivation issuing from s: $s \text{ -}\hat{}\hat{}\text{->}^*[\theta]\ s_n$ such that there exists a substitution δ_n and a term t_n that satisfy:

a) $\delta_n(s_n) = t_n$
b) $t_p \rightarrow^* t_n$
c) $\delta_n|V(s_n)$ is normalized
d) $\delta = \delta_n.\theta\ [D(\delta) \cup V(s)]$ O

Remark: We have the following schema:



Proof: By noetherian induction on $\rightarrow$. We denote $\delta(s)$ by t.

If t is not reducible, the theorem is obvious.

Otherwise t is reduced by a rule, say $g \rightarrow d$

$$t \rightarrow [u, g \rightarrow d, \sigma]\ t_1 \text{ and } t_1 \rightarrow^* t_p.$$

Thus, from the previous lemma, there exists a term s_1 and two substitutions θ_1 and δ_1 such that:

$s \text{ -}\hat{}\text{->}[u, g \rightarrow d, \theta_1]\ s_1$

$\delta_1(s_1) = t_1$

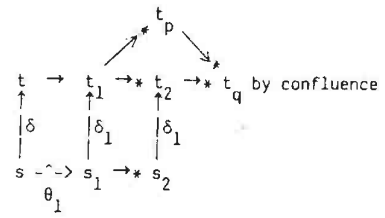
$\delta_1|V(s_1)$ is normalized

$\delta = \delta_1.\theta_1\ [D(\delta) \cup V(s)]$

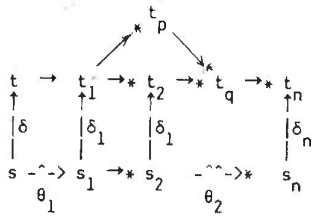
In addition, if $s_1 \rightarrow^* s_2$ then $t_1 \rightarrow^* t_2$ and we have $\delta_1(s_2) = t_2$.

If we choose $s_2 = s_1{\downarrow}R$, we have a narrowing reduction $s \text{ -}\hat{}\hat{} > s_2$.

Therefore the next diagram follows:



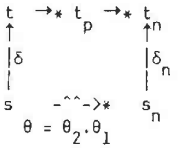
Since $t \rightarrow^+ t_2$, we can use the induction hypothesis from t_2. Thus: With the derivation $t_2 \rightarrow^* t_q$, we can associate a narrowing derivation issuing from s_2:



such that there exists a substitution δ_n and a term t_n that satisfy:

a') $\delta_n(s_n) = t_n$
b') $t_q \rightarrow^* t_n$
c') $\delta_n|V(s_n)$ is normalized
d') $\delta_1 = (\delta_n.\theta_2)\ [D(\delta_1) \cup V(s_2)]$

Therefore the next diagram follows:



Therefore a), b), c) are satisfied.

Let us prove d): $\delta = (\delta_n.\theta)\ [D(\delta) \cup V(s)]$

we recall that $\delta = \delta_1.\theta_1\ [D(\delta) \cup V(s)]$

Let x be a variable in $D(\delta) \cup V(s)$, The substitution θ_1 is defined up to a variable renaming, we can choose it such that

for any variable $y \in D(\delta) \cup V(s)$, $V(\theta_1(y)) \subseteq D(\delta_1)$.
Therefore $\delta(x) = \delta_1.\theta_1(x) = \delta_n.\theta_2.\theta_1(x)$
Hence $\delta(x) = \delta_n.\theta_2.\theta_1(x) = \delta_n.\theta(x)$. □

2.1.3. A method for equations solving

We now make clear the connection between the narrowing process and the solutions of equations in an equational theory described by a convergent term rewriting system R. Let $t_0 = t'_0$ be the equation to be solved. The method consists of building from $(t_0 = t'_0)$ all the possible narrowing derivations and to collect the corresponding narrowing substitutions, until we obtain equations $(t_n = t'_n)$ such that t_n and t'_n are unifiable. The unification problem in the equational theory is then reduced to the narrowing together with the standard unification of terms.

To establish the correctness and the completeness of this method is the purpose of the following theorem. Correctness means that the method provides effectively solutions of the given equation $(t_0=t'_0)$. Completeness means that for any solution θ, the method provides a solution σ such that σ is less than or equal to θ for the subsumption quasi-ordering $\leq_A$ on substitutions. Similar results on narrowing were established by J.M.Hullot [12,11], but only for U-reduction, while ours are for narrowing in general.

The normalization step that distinguishes narrowing from U-reduction improves both the efficiency of the method and the termination of the process. For instance, consider the confluent and noetherian rewriting system defined by the rules:

$g(f(x,y)) \rightarrow g(y)$, $h(g(y)) \rightarrow b$

where b is a constant symbol.

With the U-reduction the term h(g(y)) reduces in itself, thus we have

$h(g(y)) \text{ -^-> } h(g(y)) \text{ -^-> } \ldots \text{ -^-> } h(g(y))$ which does not terminate,

while using narrowing, h(g(y)) rewrites into b, and the process terminates.

In order to iterate the narrowing process on the two terms in parallel, we introduce a special operator === which does not belong to the operator set F, and the process starts with the term $t_0===t'_0$. It is obvious that if $t_0===t'_0 \text{ -^^-> } t$ then t can be written as $t_i===t'_i$.

THEOREM 2: Completeness theorem:

Let t_0, t'_0 be two terms and B be the set of substitutions σ that satisfies the following conditions:

- There exists a narrowing derivation issued from $t_0===t'_0$

$$t_0===t'_0 \text{ -^^-}[\sigma_1]\text{-> } t_1===t_1' \text{ -^^-}[\sigma_2]\text{-> } \ldots \text{ -^^-}[\sigma_n]\text{-> } t_n===t'_n$$

such that t_n and t'_n are unifiable by the most general unifier β

- $\sigma = \beta.\sigma_n \ldots \sigma_1$

Then **B is a complete set of R-unifiers of t_0 and t'_0.**

The theorem is still valid if we add the condition: $\sigma_n \ldots \sigma_1 | V(t_0===t'_0)$ is normalized. ○

Remark: This result was proved by C.Kirchner[15] by a simpler method, but without a normalization condition on the substitution composition $(\sigma_n \ldots \sigma_1)|V(t_0===t'_0)$.

Proof: a) correctness: Let us show that σ is a R-unifier of t_0 and t'_0.

For all $i \in \{1,..,n\}$, we have: $\sigma_i(t_{i-1}===t'_{i-1}) \rightarrow^* t_i===t'_i$

Thus $\sigma(t_0===t'_0) = \beta.\sigma_n \ldots \sigma_1(t_0===t'_0)$

$\rightarrow^* \beta.\sigma_n \ldots \sigma_2(t_1===t_1') \rightarrow^* \ldots \rightarrow^* \beta.\sigma_n(t_{n-1}===t'_{n-1})$

$\rightarrow^* \beta(t_n===t'_n)$

We get $\sigma(t_0) =_R \beta(t_n)$

$\sigma(t'_0) =_R \beta(t'_n)$.

And since by hypothesis β unifies t_n and t'_n, we have $\sigma(t_0) =_R \sigma(t'_0)$.

b) completeness: follows from Theorem 1.

Let α be a R-unifier of t_0 and t'_0, let us show that a substitution more general than α (modulo R) can be obtained by a narrowing derivation.

Let us define δ and s by: $\delta = \alpha\downarrow_R$

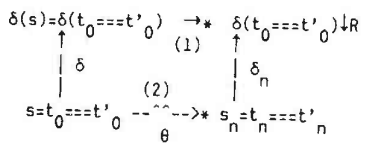
$s = t_0===t'_0$

δ is a R-unifier of t_0 and t'_0.

From Theorem 1, it results:

With the derivation (1) issuing from $\delta(t_0===t'_0)$ and going to the term $\delta(t_0===t'_0)\downarrow_R$, we can associate a narrowing derivation (2)

issuing from s:



such that there exists a substitution δ_n that satisfies:

a) $\delta_n(t_n === t'_n) = \delta(t_0 === t'_0)\downarrow R$

b) $\delta_n|V(s_n)$ is normalized

c) $\delta = \delta_n.\theta\ [D(\delta) \cup V(s)]$

From a) we deduce:

$\delta_n(t_n) = \delta(t_0)\downarrow R$

$\delta_n(t'_n) = \delta(t'_0)\downarrow R$

and since δ R-unifies t_0 and t'_0, we have $\delta_n(t_n) = \delta_n(t'_n)$.

δ_n unifies t_n and t'_n: there exists a most general unifier β, and a substitution γ that satisfy:

$$\delta_n = \gamma.\beta$$

From c) we deduce:

$$\delta = \delta_n.\theta = \gamma.\beta.\theta\ [D(\delta) \cup V(s)]$$

Finally $\beta.\theta \leq_R \delta =_R \alpha\ [V(s)]$.

Since δ is normalized, we get that $\theta|V(s)$ is normalized. □

Theoretically the process can be seen as building a tree whose nodes are labeled by equations and whose edges are labeled by substitutions. The equations are those successively generated by the narrowing process and the substitutions are the narrowing substitutions used at each step to get the corresponding equation. A solution can be seen as a path of the tree leading to a node labeled by an equation whose members are unifiable. Such a node is called successful. This tree, hereafter called **narrowing tree**, can be infinite.

2.2. A narrowing procedure

We propose here a narrowing procedure. Clearly two operations enable us to perform the method: normalization and overlapping (also called superposition). Let us first remind the notion of overlapping and the related concept of critical pairs.

Definition 2: Let (l,r) and (g,d) be two directed pairs of terms such that

- V(l) and V(g) are disjoint
- g overlaps l at the occurrence u with the substitution σ iff σ is the most general unifier away from V(l) ∪ V(g) of the term g and the subterm of l at the occurrence u.

The pair of terms $(\sigma(l[u \leftarrow d]), \sigma(r))$ is called a critical pair of the pair (g,d) on the pair (l, r) at the occurrence u. In what follows the pair (g,d) will often be built from a rule $g \rightarrow d$.

In the narrowing process, the rules of R are overlapped on the terms t===t' and normalizations are performed using R. On the other hand we are interested in recording the values substituted to the variables of t and t'. For this purpose, we consider directed pairs (u===u', **solution**$(\sigma(x_1),\ldots,\sigma(x_n))$) where === and **solution** are new function symbols, $x_1,\ldots,x_n$ are the variables of t and t', and σ is the composition of the narrowing substitutions. Such a pair can be interpreted as follows: in the context where the values of $x_1,\ldots x_n$ are respectively $\sigma(x_1),\ldots,\sigma(x_n)$, the initial equation is equivalent to u===u', which means that both have the same set of solutions. The normalized left-hand side of a critical pair obtained by overlapping a rule of R on such a pair can be interpreted as the result of one step of narrowing, while its right-hand side is the new values of the variables $x_1,\ldots,x_n$. The narrowing process provides a solution as soon as t and t' are unifiable. For this purpose, we add to R the rule x===x → **true**. Notice that the new term rewriting system R ∪ {x===x → **true**} denoted R1 is always convergent. Overlapping this rule on a pair produces the critical pair (**true**, **solution**$(\theta(x_1),\ldots,\theta(x_n))$ and θ can be interpreted as a solution of the initial equation.

2.2.1. The procedure

Let t_0 and t'_0 be the terms to unify, $W = V(t_0) \cup V(t'_0) = \{x_1,\ldots,x_n\}$. The parameters of the procedure are a convergent term rewriting system $R1 = R \cup \{x===x \rightarrow \textbf{true}\}$, a set DP of directed pairs $(t===t'$, $\textbf{solution}(\sigma(x_1),\ldots,\sigma(x_n))$ and a set P of critical pairs obtained by overlapping rules of R1 into directed pairs of DP. Each element of DP is obtained by normalizing a critical pair in P. Its left part can be understood as the label of a node in the narrowing tree, while its right part can be seen as the path which leads to this node. Figure 2 gives a description of the NARROWER procedure.

Initialization

$R1 = R \cup \{ x===x \rightarrow \textbf{true} \}$

$DP = \{(t_0===t'_0, \textbf{solution}(x_1,\ldots,x_n)\}$

$P = \emptyset$

NARROWER(P, R1, DP)
  if P is not empty
    then choose a critical pair (p, q) in P
        $g = p{\downarrow}R$, $d = q{\downarrow}R$
        NARROWER(P - {(p, q)}, R1, DP $\cup$ {(g, d)})
    else if all the critical pairs between the left-hand sides in DP and R1
          have been computed
          then STOP
          else choose a directed pair (l, r) of DP
              with which critical pairs have not been computed
              P = CRITICAL_PAIRS((l, r), R1)
              NARROWER(P, R1, DP)
          endif
    endif
end

Figure 2. NARROWER

CRITICAL_PAIRS((l, r), R1) computes the critical pairs of all the rules of R1 on l. We always suppose that the strategy for choosing the pairs of DP in order to compute critical pairs satisfies the following fairness hypothesis.

Fairness hypothesis:

Let H be the union (possibly infinite) of the values of the parameter DP at each recursive call of NARROWER. For any directed pair (l, r) in H, there exists eventually a recursive call such that (l, r) is chosen so that CRITICAL_PAIRS((l, r), R1) is called.

This hypothesis is necessary to insure the completeness of the procedure. Now notice two facts. NARROWER never adds rules in R1 and the directed pairs (l, r) of DP could not be considered as rewrite rules because V(l) does not always contain V(r).

2.2.2. Proof of the NARROWER procedure

The proof of the NARROWER procedure consists of two steps: correctness and completeness. NARROWER is correct in the sense that with any pair in H, we can associate a narrowing derivation issued from $t_0===t'_0$. NARROWER is complete: that is, any term issued from $t_0===t'_0$ by a narrowing derivation is produced by the procedure in H.

Definitions and notations:

DP_i and P_i denote respectively the values of the arguments DP and P at the i-th recursive call of NARROWER, and thus $H = \cup DP_i$. We write dp_0 the initial directed pair in DP:

$dp_0 = (t_0===t'_0, \textbf{solution}(x_1,\ldots,x_n))$, and $e_0 = t_0===t'_0$.

With these notations, $DP_1 = \{dp_0\}$.

We write $\textbf{solution}(\sigma)$ the term $\textbf{solution}(\sigma(x_1),..,\sigma(x_n))$. We call solution of the procedure any substitution σ such that the pair $(\textbf{true}, \textbf{solution}(\sigma))$ is in H.

The narrowings are always performed with the term rewriting system R.

Let us define the noetherian relation $<$ on the directed pairs of H by:

$dp < dp'$ iff

- $dp \in DP_i$, $dp' \in DP_j$, with $i < j$

- dp' is obtained by overlapping a rule $g \rightarrow d$ of R1 on dp.

Notice that $<$ is noetherian and that NARROWER only adds pairs to DP, but neither modifies nor removes ones. Thus $i < j$ implies DP_i included into DP_j.

Correctness lemma: Let dp_n be a pair in H different from dp_0, there exists a narrowing derivation issued from e_0

$$e_0 \text{ -^^->}[\sigma_1]\ e_1 \text{ -^^->}[\sigma_2] \ldots. \text{ -^^->}[\sigma_{n-1}]\ e_{n-1}$$

such that

dp_n is $(e_n = t_n === t'_n,\ \mathbf{solution}(\sigma'_n))$
with e_{n-1} -^^->$[\sigma_n]$ e_n

or

dp_n is $(\mathbf{true},\ \mathbf{solution}(\sigma'_n))$
and t_{n-1} and t'_{n-1} are unifiable by the most general unifier σ_n

with $\sigma'_n = (\sigma_n.\sigma_{n-1}\ldots\sigma_1)\downarrow R$ ○

Proof: Let us prove by noetherian induction that dp_n is into one of the two previous form. Since dp_n is different of dp_0, it is the result of an overlap of a rule in R1 on a pair dp_{n-1} in H. By induction hypothesis applied on $dp_{n-1} < dp_n$:

- either dp_{n-1} is $(\mathbf{true},\ \mathbf{solution}(\sigma'_{n-1}))$, which is not possible because no rule in R1 can superpose on the constant **true**.

- or dp_{n-1} is $(t_{n-1} === t'_{n-1},\ \mathbf{solution}(\sigma'_{n-1}))$, and

$$e_0 \text{ -^^->}[\sigma_1]\ e_1 \text{ -^^-> } \ldots \text{ -^^->}[\sigma_{n-1}]\ e_{n-1}$$

therefore dp_n is issued from an overlap of a rule $g \rightarrow d$ in R1 on e_{n-1}, at the occurrence u, with the substitution σ_n. σ_n is a unifier of g and $e_{n-1}|_u$. This overlap creates the critical pair

$$(p, q) = (\sigma_n(e_{n-1}[u \leftarrow d]),\ \mathbf{solution}(\sigma_n.\sigma'_{n-1})).$$

dp_n is obtained from (p, q) by normalization:

$$dp_n = (\sigma_n(e_{n-1}[u \leftarrow d]),\ \mathbf{solution}(\sigma_n.\sigma'_{n-1}))\downarrow R.$$

Then $e_n = (\sigma_n(e_{n-1}[u \leftarrow d]))\downarrow R$ and then e_{n-1} -^^->$[u, g \rightarrow d, \sigma_n]$ e_n

$\sigma'_n = (\sigma_n.\sigma'_{n-1})\downarrow R$.

Since $g \rightarrow d \in R1$, we consider two cases:

- case 1: if $g \rightarrow d \in R$, we can write e_{n-1} -^^->$[R, \sigma_n]$ e_n. Since the top symbol of e_{n-1} is ===, u is not the empty occurrence, and e_n is of the form $t_n === t'_n$.

- case 2: otherwise $g \rightarrow d$ is the rule $x === x \rightarrow \mathbf{true}$, u is the empty occurrence, and σ_n is a unifier of e_n with $x === x$. Then σ_n is a unifier of t_{n-1} and t'_{n-1} and dp_n is of the form $(\mathbf{true},\ \mathbf{solution}(\sigma'_n))$. □

Completeness lemma:

If e_0 -^^->$[\sigma_1]$ e_1 -^^->$[\sigma_2]$ ... -^^->$[\sigma_n]$ $e_n = t_n === t'_n$ is a narrowing derivation issuing from $e_0 = t_0 === t'_0$, there exists a pair dp_n: $(t_n === t'_n,\ \mathbf{solution}(\sigma'_n))$ in H, with $\sigma'_n = (\sigma_n \ldots \sigma_1)\downarrow R$.

In addition, if t_n and t'_n are unifiable by the most general unifier θ, there exists a pair $(\mathbf{true},\ \mathbf{solution}((\theta.\sigma'_n)\downarrow R))$ in H. ○

Proof: a) The proof is by induction on the length n of the narrowing derivation. Let us suppose the result true for any length less than n:

There exists a pair $dp_{n-1} = (e_{n-1},\ \mathbf{solution}(\sigma'_{n-1}))$ in H. By fairness hypothesis, there is a recursive call j such that dp_{n-1} is chosen and CRITICAL_PAIRS(dp_{n-1},R1) is called.

We have e_{n-1} -^^->$[u, g \rightarrow d, \sigma_n]$ e_n, which means $e_{n-1}|_u$ and g are unifiable by the most general unifier σ_n, therefore the critical pair $(\sigma_n(e_{n-1}[u \leftarrow d]),\ \mathbf{solution}(\sigma_n.\sigma'_{n-1}))$ is created. Since Narrower normalizes all the pairs of P, there exists a call $k > j$ such that DP_k contains $dp_n = (\sigma_n(e_{n-1}[u \leftarrow d])\downarrow R,\ \mathbf{solution}((\sigma_n.\sigma'_{n-1})\downarrow R))$
$= (e_n,\ \mathbf{solution}(\sigma'_n))$ denoting $\sigma'_n = (\sigma_n.\sigma'_{n-1})\downarrow R$.

b) If t_n and t'_n are unifiable by the most general unifier θ, we have $t_n === t'_n$ -^^->$[\varepsilon, x === x \rightarrow \mathbf{true}, \theta]$ **true**. The previous proof can be applied on this narrowing reduction, and we obtains: there exists a pair $(\mathbf{true},\ \mathbf{solution}((\theta.\sigma'_n)\downarrow R))$ in H. □

From these two results we deduce that NARROWER correctly simulates the narrowing process. Hence the following theorem can be stated:

VALIDITY THEOREM: The set of solutions of the procedure NARROWER applied on t_0 and t'_0 is a complete set of R-unifiers of t_0 and t'_0. ○

Proof: a) Let θ be a solution of NARROWER. From the correctness lemma, there exists a narrowing derivation issued from e_0:

$$e_0=t_0===t'_0 \text{ -^^->}[\sigma_1]\ e_1 \text{ -^^-> } \dots \text{ -^^->}[\sigma_n]\ e_n=t_n===t'_n$$

such that t_n and t'_n are unifiable by the most general unifier σ and $\theta = (\sigma.\sigma_n \dots \sigma_1)\downarrow R$.

From Theorem 2, $\sigma.\sigma_n \dots \sigma_0$ is a R-unifier of t_0 and t'_0, then θ is a R-unifier of t_0 and t'_0.

b) Let α be a R-unifier of t_0 and t'_0. From Theorem 2 , there exists a narrowing derivation

$$e_0 \text{ -^^->}[\sigma_1]\ e_1 \text{ -^^-> } \dots \text{ -^^->}[\sigma_n]\ e_n=t_n===t'_n$$

such that t_n and t'_n are unifiable by the most general unifier σ and $\theta = \sigma.\sigma_n \dots \sigma_1 \leq R\ \alpha$.

Thus $\theta\downarrow R \leq R\ \alpha$.

From the completeness lemma, $\theta\downarrow R$ is a solution of NARROWER. □

Remark: If we modify NARROWER by changing the normalization of the critical pairs using R into a normalization using R1, the validity theorem is still valid. That means that the narrowing relation using R1 gives a complete set of R-unifiers of t_0 and t'_0.

More precisely $\{ \sigma \mid t_0===t'_0 \text{ -^^->}*[R1,\sigma]\ \textbf{true}\}$ is a complete set of R-unifiers of t_0 and t'_0.

Proof: We denote $S = \{ \sigma \mid t_0===t'_0 \text{ -^^->}*[R1,\sigma]\ \textbf{true}\}$. S is a set of R-unifiers of t_0 and t'_0 from Theorem 2. Let us prove that it is complete.

Let θ be a R-unifier of t_0 and t'_0. From Theorem 2, there exists a narrowing derivation:

$$t_0===t'_0 \text{ -^^->}[\sigma_0]\ t_1===t_1' \text{ -^^-> } \dots \text{ -^^->}[\sigma_{n-1}]\ t_n===t'_n$$

such that t_n and t'_n are unifiable by the most general unifier σ_n and $\sigma_n \dots \sigma_0 \leq R\ \theta$.

If for all i in {1,..,n}, $t_i===t'_i$ is not reducible by the rule x===x → **true**, $\sigma_n \dots \sigma_0$ is in S.

Otherwise there exists $j \in \{1,..,n\}$ such that $t_j===t'_j \rightarrow[x===x\rightarrow\textbf{true}]\ \textbf{true}$. Then $t_j = t'_j$ and $\sigma_{j-1} \dots \sigma_0 \in S$. Therefore $\sigma_{j-1} \dots \sigma_0 \leq \sigma_n \dots \sigma_0 \leq R\ \theta$. □

3. OPTIMIZATIONS

The aim of this section is to find optimizations of the NARROWER procedure. This means to decide that in some cases, the subtree issued from a given node in the narrowing tree is redundant and can be dropped without loosing the completeness of the returned set of unifiers.

Such optimizations have been studied in automatic theorem proving and logic programming. One of them is subsumption. This plays the same role as the simplification of a left-hand side of a rule in term rewriting system. A rule g→d is said simplifiable by g'→d' if a subterm of g is an instance of g'. The question then arises to know whether such simplifications can be performed in the narrowing process in order to restrict the narrowing tree. But care must be taken to avoid loosing the completeness. Let us give an example of such a situation.

Example: Let us consider the confluent and noetherian rewriting system as follows:

x + x → x

(a + x) + (x + a) → a + x

x * 0 → 0

x * x → 0

The narrowing process may be used to solve the equation (x + y) + (y + x) = z. Among other possible solutions, the process will generate the following narrowing derivations:

(x + y) + (y + x) = z -^^->[x/a] a + y = z -^^->[y/a] a = z

-^^->[y/x] x = z

with the corresponding solutions:

(1) (x=a, y=a, z=a)

(2) (y=x, z=x)

NARROWER creates the two pairs:

(a === z, **solution**(a,a,z))

(x === z, **solution**(x,x,z))

The first pair is an instance of the second one, therefore it is "useless". But care must be taken that also the set of solutions is subsumed, otherwise solutions can be lost this way. For example, let us solve the equation

$(x + y) * y = z$. Among other possible solutions, the process will generate the following narrowing derivations:

$(x + y) * y = z$ -^^->[x/y] $0 = z$ -^^->[z/0] $0 = 0$

-^^->[y/0] $0 = z$ -^^->[z/0] $0 = 0$

with the corresponding solutions:

(1) (x=y, z=0)

(2) (y=0, z=0)

If we remove the directed pair that corresponds to the first narrowing derivation, we avoid the solution (1). Therefore we loose the completeness of the process.

Restricting the kind of allowed simplifications, we propose here the following optimization: if NARROWER generates two directed pairs such that one is an instance of the other, the first one is useless. This is equivalent to say that the subtree issued from the first node does not lead to new solutions in the narrowing tree. More formally:

SUBSUMPTION THEOREM: Let dp_i and dp_j be two pairs such that $dp_i = (e_i, \textbf{solution}(\delta_i))$, $dp_j = (e_j, \textbf{solution}(\delta_j))$, $e_i = \theta(e_j)$ and $\delta_i = \theta.\delta_j$. Then dp_i can be dropped without loosing the completeness of the set of unifiers. ○

Proof: Let $<$ be the relation on the directed pairs defined by: $dp < dp'$ iff dp' is obtained by overlapping a rule $g \rightarrow d$ on dp. We denote $<+$ the transitive closing of $<$.

Let us first consider the simple case where $dp_i = (\textbf{true}, \textbf{solution}(\delta_i))$. Then dp_j is $(\textbf{true}, \textbf{solution}(\delta_j))$ and $\delta_i = \theta.\delta_j$ means that δ_i is a solution less general than δ_j.

In the other cases e_i and e_j are equations, three cases are to be considered: $dp_i <+ dp_j$. $dp_j <+ dp_i$ and dp_i and dp_j not comparable with $<+$.

- $dp_i <+ dp_j$. That means that dp_i is first generated, then all the critical pairs of dp_i with rules of R are computed and dp_j is obtained either from one of these critical pairs or from one of their successors. Thus deleting dp_i from DP has no effect on the following computations in this case.

- $dp_j <+ dp_i$. This case is incompatible with the hypothesis that dp_i is an instance of dp_j.

  Since $dp_j <+ dp_i$ and from the validity theorem, there exists a narrowing derivation

  e_0 -^^->[δ_j] e_j -^^->[β] $e_i = \theta(e_j)$ and then $\beta(e_j) \rightarrow+ e_i = \theta(e_j)$.
  On the other hand, $\delta_i = \beta.\delta_j = \theta.\delta_j$ [$V(e_0)$], therefore $\beta(e_j) = \theta(e_j)$, and $\theta(e_j) \rightarrow+ \theta(e_j)$, which yields a contradiction with the finite termination of R.

- dp_i and dp_j are not comparable. In this case, the result is a consequence of the following lemma.

Lemma : Let be two narrowing derivations issued from e_0.

e_0 -^^->*[α] e_i -^^->*[α'] $e_n = (t_n === t'_n)$

gives the solution $\alpha''.\alpha'.\alpha$

(where α'' is a unifier of t_n and t'_n), and

e_0 -^^->*[β] e_j

is such that $e_i = \theta(e_j)$ and $\alpha =_R \theta.\beta$

Then the second narrowing derivation can be completed into

e_0 -^^->*[β] e_j -^^->*[β'] $e_p = (t_p === t'_p)$

which gives a more general solution. More precisely: $\beta''.\beta'.\beta <_R \alpha''.\alpha'.\alpha$

(where β'' is a unifier of t_p and t'_p). ○

Proof: As $\alpha''.\alpha'$ is a R-unifier of the equation $e_i = (t_i === t'_i)$, $\alpha''.\alpha'.\theta$ is a R-unifier of $e_j = (t_j === t'_j)$. By the completeness of the narrowing process, there exists a narrowing derivation

e_j -^^->*[β'] $e_p = (t_p === t'_p)$

such that $\beta''.\beta' \leq_R \alpha''.\alpha'.\theta$ [$V(e_j)$]

(where β'' is a unifier of t_p and t'_p).

We then deduce that $\beta''.\beta'.\beta \leq_R \alpha''.\alpha'.\theta.\beta =_R \alpha''.\alpha'.\alpha$ [$V(e_0)$]. □

Let us now give an example where the hypothesis of the theorem are satisfied:

Example: Let R be the convergent rewriting system

$R = \{g(f(x,x),0) \rightarrow g(x,0),\ g(g(f(x,x),y),z) \rightarrow g(g(x,y),z)\}$

and $(g(f(x_1,x_2),0) === g(g(f(x_1,x_2),x_3),x_4))$ the equation to be solved.

By narrowing two equations are found:

$g(x_1,0) \equiv\equiv g(g(x_1,0),x_4)$

when using the narrowing substitution $(x_2/x_1,x_3/0)$ and the first rule at occurrence 1 in the right-hand side of the equation and

$g(x_1,0) \equiv\equiv g(g(x_1,x_3),x_4)$

when using the narrowing substitution (x_2/x_1) and the first rule at the top of the left-hand side of the equation. NARROWER generates the corresponding pairs:

$(g(x_1,0) \equiv\equiv g(g(x_1,0),x_4))$, **solution**$(x_1,x_1,0,x_4))$

$(g(x_1,0) \equiv\equiv g(g(x_1,x_3),x_4))$, **solution**$(x_1,x_1,x_3,x_4))$.

With $\theta = (x_3/0)$, the hypothesis of the previous theorem are satisfied and the first pair is useless.

4. IMPLEMENTATION

The aim of this section is to show how the completion procedure has been adapted to perform narrowing and implemented as an extension of REVE [16].

Remind first that the Knuth-Bendix completion procedure attempts to transform a set of equations that defines an equational theory into an equivalent term rewriting system which is confluent, noetherian and interreduced. For this purpose, the completion procedure computes critical pairs between rules, normalizes them and directs them using a noetherian ordering on terms, in order to produce new rules. Whenever a new rule is introduced, it simplifies all the other existing rules. Thus completion and narrowing are both based on overlapping and normalization. Nevertheless let us point out some differences.

- From the proof of correctness of NARROWER, we have seen that the overlapping is always computed in the left-hand side of the pair $(t,$ **solution**$(t_1,..,t_n))$. However these directed pairs could not be made into a rule because the condition on variables is violated.

- Assume that the pair of terms $(t \equiv\equiv t'$, **solution**$(\sigma(x_1),\ldots,\sigma(x_n)))$ is implemented as a "rule" $t \equiv\equiv t' \rightarrow$ **solution**$(\sigma(x_1),\ldots,\sigma(x_n))$. Such a

"rule" would never be used to reduce a term, and could not be merged into the rewriting system like in the completion procedure.

- The normalization of a critical pair is only performed using rules of R. The directed pairs are not used for simplification. However, as said in the previous section, subsumption tests are performed.

- Only rules of R U $\{x \equiv\equiv x \rightarrow$ **true**$\}$ are overlapped on directed pairs to produce critical pairs.

Among other functionalities, the system REVE provides an implementation of the Knuth-Bendix completion procedure. The modularity of the system enabled us to extend it by adding a new command **narrow** which solves equations by narrowing. The implementation has been derived from the Knuth-Bendix algorithm using the described modifications.

5. DESCRIPTION OF THE SET OF SOLUTIONS BY A REGULAR LANGUAGE

In this section we discuss the case of infinite narrowing derivations where the same equation occurs many times. We first give an example of this situation due to F.Fages [3]. Let us consider the following confluent and noetherian term rewriting system R:

$g(f(x, y)) \rightarrow g(y)$

and the following equation:

$g(x) \equiv\equiv g(0)$

If we superpose the left-hand side of the equation with the rule, we obtain the equation $g(f(u, z)) \equiv\equiv g(0)$ which normalizes to $g(z) \equiv\equiv g(0)$. Up to a renaming of the variables it is the same equation as the given one. The narrowing derivation:

$(g(x) \equiv\equiv g(0)) \text{-}\hat{}\hat{}\text{->}[x \rightarrow f(u,x)]\ (g(x) \equiv\equiv g(0)) \text{-}\hat{}\hat{}\text{->}[x \rightarrow f(u,x)] \ldots$

$\ldots\ (g(x) \equiv\equiv g(0)) \text{-}\hat{}\hat{}\text{->}[x \rightarrow f(u,x)]\ (g(x) \equiv\equiv g(0)) \text{-}\hat{}\hat{}\text{->}[x \rightarrow f(u,x)] \ldots$

is an example of a serious difficulty with the narrowing process: it does not terminate. The purpose of this section is to give a finite description of the solutions, which allows the termination of NARROWER. To support the

intuition, let us consider the narrowing tree corresponding to such a situation. There exists at least one node labeled by an equation e such that e -^^->+ e. Such a tree can be represented as a directed cyclic graph. A finite description of a complete set of unifiers of the given equation will be obtained by looking for all the possible paths in the narrowing tree leading to a successful node.

5.1. Characterization of the regular language

Let us consider the narrowing tree as a valued graph: the nodes are labeled by the equations obtained by narrowing, the edges are labeled by the narrowing substitutions. The successful nodes are the nodes labeled by **true**. We remark that the successors of a node e and that the labels of the edges that go to them depend only on the equation e, and not on the predecessors of e. By the way, the narrowing tree is then not any tree.

If the set of the nodes of the tree is finite (denoted by $E = \{e_0,..,e_n\}$), from the previous remark we can reduce the tree into a finite labeled directed graph the nodes of which are $e_0,..,e_n$. Conversely, a finite graph can be unwind into a tree whose set of nodes is finite.

In what follows we consider a narrowing tree whose set of nodes is finite, and that is denoted $E = \{e_0,...,e_n\}$. In order to transform the tree into a finite graph, it suffices to chain on the already found nodes when we build the tree. Let us denote G this finite graph.

To this graph we can associate a finite automata A, the vocabulary of which is the set of labels of edges of G (there are substitutions), and the states are the nodes of G. More formally $A = (E, e_0, . , E')$
where E is the set of states

e_0 is the initial state

. is the transition relation of the automata defined by

$$e_i.\sigma = e_j \text{ iff } e_i \text{ -^^->}[\sigma] e_j$$

The solutions of the equation e_0 are the composition of the substitutions obtained by traversing the paths of G from e_0 to **true**. Therefore by defining the set E' of the successful states of the automata by E' = {**true**}, the solutions of e_0 are given by the words of the regular language L recognized

by A, if we interpret the word $\sigma.\theta$ by the composition of substitutions $\theta.\sigma$.

Let us show that A is a deterministic automata.

Let us suppose that $e_i.\sigma = e_p$ then that e_i -^^->[σ] e_p
$e_i.\sigma = e_q$ e_i -^^->[α] e_q

Therefore $\sigma(e_i) \rightarrow^+ e_p$, $\sigma(e_i) \rightarrow^+ e_q$. Since e_p and e_q are normalized and by confluence of R, we have $e_p = e_q$.

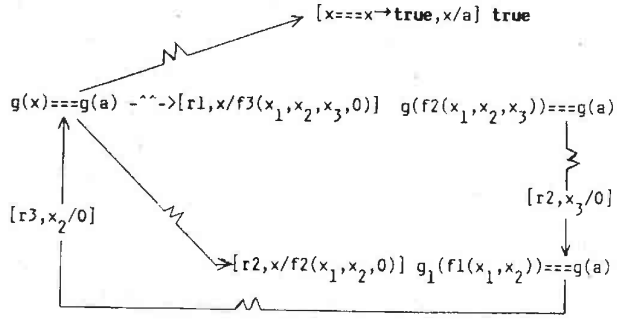
It is possible that the narrowing tree has nodes that do not lead to a successful node, and then are useless. This means that A is not the minimum automata that recognizes L. From A we can compute the minimum automata A_{min}.
From A_{min}, using the algorithm that proves the Kleene theorem, we will obtain a rational expression that defines completely L.

We have implanted a process that detects loops in our implementation of NARROWER in REVE.

4.2. Example

Let R = { r1: $g(f3(x_1,x_2,x_3,0)) \rightarrow g(f2(x_1,x_2,x_3))$
r2: $g(f2(x_1,x_2,0)) \rightarrow g_1(f1(x_1,x_2))$
r3: $g_1(f1(x_1,0)) \rightarrow g(x_1)$ }

R is confluent, noetherian, and interreduced. Let us solve by narrowing the equation: g(x) = g(a). The narrowing tree can be reduced into the following graph:



The automata associated to this graph is A = (E,e_0,.,{**true**}) with

E = { e_0: g(x)===g(a),
e_1: g(f2(x_1,x_2,x_3))===g(a),
e_2: g_1(f1(x_1,x_2))===g(a),
e_3: **true** }

The automata vocabulary is:

V = { σ_1: (x/f3(x_1,x_2,x_3,0)),
σ_2: (x_3/0),
σ_3: (x/f2(x_1,x_2,0)),
σ_4: (x_2/0),
σ_5: (x/a) }

The transition relation of the automata is:

.	σ_1	σ_2	σ_3	σ_4	σ_5
e_0	e_1		e_2		e_3
e_1		e_2			
e_2				e_0	
e_3					

This automata is minimal and recognizes the regular language that can be characterized by the rational expression: $(\{\sigma_1.\sigma_2, \sigma_3\}.\sigma_4)^*.\sigma_5$.
This expression gives a finite description of an infinite set of substitutions.

6. NARROWING AND LOGIC PROGRAMMING

The main mechanism of the Prolog language is the linear resolution principle which has some similarities with the U-reduction presented here. Dershowitz was the first to point out to us [1,2] this similarity. Goguen and Meseguer [7] in their language EQLOG, Fribourg in his language SLOG[5,6] presented a close idea. They all claim that the future logic programming languages will be built upon a principle that joints U-reduction from Prolog and the normalization from Functional Programming. Unfortunately "the mechanism that joins the predicate logic and equational logic features, namely narrowing, has not been implemented in a programming language context" (loc. cit.) NARROWER is an attempt to propose such tools in order to make experiments.

The main difference between the Prolog strategy and the NARROWER strategy is that Prolog only does resolution, i.e., U-reduction only at the top, when NARROWER does U-reduction followed by a step of reductions. If we assimilate the U-reduction to the resolution, we see that NARROWER performs a crucial action that simplifies the form of the expression it manipulates. Usually the U-reduction is rather expansive since it is based on unification and the reductions are much cheaper since they are based on matching. Therefore the power of NARROWER lies in mixing programming logic features to infer new facts from old ones and functional programming features to simplify the form of the new facts in order to keep them always in a more manageable form. This combination of the two strategies makes logic programming languages based on narrowing good candidates to succeed Prolog.

As usual in logic programming the universe of the program is described by a set of facts that state its basic properties. This set is also called specification of the abstract data type of the program and this specification is given in an algebraic style. In the current implementation, these properties are only equalities. They have to be presented as a noetherian and confluent set of rules. Such properties are usually provided from any presentation using the tools available in REVE. In the future it would be possible to also assert statements in the first order predicate calculus, based on Hsiang's works [8]. These statements will be equalities in the boolean ring, in other words equivalences between predicates, unlike Prolog

that deals only with implications. This will require mechanisms for handling associative and commutative equations that are now present in REVEUR 3 [14]. Such mechanisms exist and we plan to extend the current attempt to this framework. Another restriction lies in the absence of sorts or types in our system. They will also be introduced in the future.

Here we would like to present our ideas on two examples, the "integers" (Figure 1) and the " map coloration" (Figure 3 in Appendix). In the introduction we have seen how to solve linear equations, here we propose to solve the quadratic equation

$$x^2 + 3x + 2 = 0$$

This is transformed into

$$(s(s((x * x) + x + x + x)) === 0, \textbf{solution}(x))$$

NARROWER finds the solution p(0) into two steps. A superposition for example with $p(x) + y \rightarrow p(x + y)$ yields

$$(s(s((p(x) * p(x)) + p(x + p(x)) + p(x)))) === 0, \textbf{solution}(x))$$

that reduces to

$$(-(x) + ((x * x) + (x + x)) === 0, \textbf{solution}(p(x)))$$

and a new superposition with $x + 0 \rightarrow x$ gives the solution p(0). NARROWER finds also the solution p(p(0)), but does not terminate because it does not know that the equation has only two solutions and so it runs forever, looking for other solutions. Obviously a strategy that works depth-first and stops after finding the first solution would terminate in this case.

References

1. N. Dershowitz, "Computing With Term Rewriting Systems," Procedings of An NSF Workshop On The Rewrite Rule Laboratory, April 1984.

2. N. Dershowitz, "Equations as programming language," Fourth Jerusalem Conference on Information Technology, pp. 114-123, Jerusalem (Israel), May 1984.

3. 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.

4. M. Fay, "First-Order Unification in an Equational Theory," Proceedings of the 4th Workshop on Automated Deduction, pp. 161-167, Austin, Texas, 1979.

5. L. Fribourg, "Oriented Equational Clauses as a Programming Language," J. Logic Programming, vol. 1:2, pp. 165-177, 1984.

6. L. Fribourg, "Handling function definitions through innermost superposition and rewriting," in Proc. 1st Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 202, pp. 325-344, Springer Verlag, Dijon (France), 1985.

7. J. Goguen and J. Meseguer, "Equality, Types, Modules and Generics for Logic Programming," SRI Internal Publication, Menlo Park, 1984.

8. J. Hsiang, "Refutational Theorem Proving Using Term Rewriting Systems," Ph.D Thesis, University of Illinois, Urbana, 1981.

9. G. Huet, "Résolution d'Equations dans des Langages d'Ordre 1,2, ... ω," Thèse d'Etat, Université de Paris VII, 1976.

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

11. J.M. Hullot, "Compilation de Formes Canoniques dans les Théories Equationnelles," Thèse de 3ème Cycle, Université de Paris Sud, 1980.

12. 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.

13. 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.

14. C. Kirchner and H. Kirchner, "Implementation of a General Completion Procedure Parameterized by Built-in Theories And Strategies," Rapport Crin 84-R-85, 1984.

15. C. Kirchner, "Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles," Thèse de doctorat d'Etat, Université de Nancy I, 1985.

16. P. Lescanne, "Computer Experiments with the REVE Term Rewriting System Generator," in 10th ACM Conf. on Principles of Programming Languages, pp. 99-108, Austin Texas, January 1983.

17. G. Plotkin, "Building-In Equational Theories," Machine Intelligence, vol. 7, pp. 73-90, 1972.

APPENDIX The map coloration

The map coloration is a classical problem in logic programming. Figure 3 shows the universe as it was presented to NARROWER and Figure 4 is a picture of the map. The problem that was asked to NARROWER was to find the color of a map where 1 is "b" and 2 is "y". NARROWER found the 8 possible solutions as shown in Figure 5.

```
% Boolean operations

x & ff == ff
ff & x == ff
tt & x == x
x & tt == x
(x & ( y & z)) == ((x & y) & z)

% Colors

next(b, y) == tt
next(b, g) == tt
next(b, r) == tt

next(g, y) == tt
next(g, b) == tt
next(g, r) == tt

next(y, b) == tt
next(y, g) == tt
next(y, r) == tt

next(r, y) == tt
next(r, g) == tt
next(r, b) == tt

next(x, x) == ff

% The map

map(x1, x2, x3, x4, x5) == next(x1, x2) & next(x1, x3) &
next(x1, x5) & next(x2, x3) & next(x2, x4) & next(x2, x5) & next(x3, x4)
```

Figure 3. Map Coloration

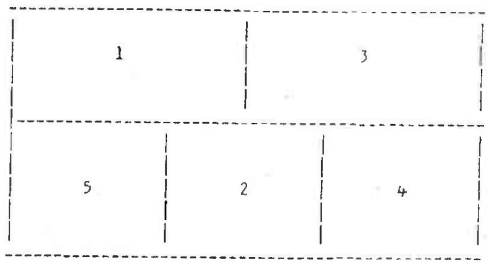


Figure 4. A map

The equation map(b, y, x_3, x_4, x_5) === tt has the set of r-unifiers as follows:

1.
x_3 / g
x_4 / b
x_5 / g

2.
x_3 / g
x_4 / b
x_5 / r

3.
x_3 / g
x_4 / r
x_5 / g

4.
x_3 / g
x_4 / r
x_5 / r

5.
x_3 / r
x_4 / g
x_5 / g

6.
x_3 / r
x_4 / g
x_5 / r

7.
x_3 / r
x_4 / b
x_5 / g

8.
x_3 / r
x_4 / b
x_5 / r

Figure 5. The answer of NARROWER to the map problem.

Chapitre 3

La surréduction normalisante basique de gauche à droite

Ce chapitre présente une nouvelle relation de surréduction: la surréduction normalisante basique de gauche à droite. Il s'agit d'une combinaison non triviale de la surréduction normalisante et de la surréduction basique de gauche à droite; la combinaison triviale donnant une méthode d'unification non complète. Il est composée d'une version légèrement modifiée d'un article [61] publié lors de la deuxième conférence "Rewriting Techniques and Applications".

La surréduction normalisante a été étudiée au chapitre précédent. La surréduction basique a été définie et étudiée par Hullot [35,36], puis prolongée en surréduction basique de gauche à droite par Herold [29]. Il s'agit d'une restriction de la surréduction, puisque les règles de réécriture ne sont appliquées qu'à certaines occurrences dites basiques, les autres occurrences étant dites protégées. Ces relations sont définies en détail au chapitre 1.

La nouvelle relation que nous proposons ici provient de l'idée suivante. On montre facilement qu'une équation peut être résolue par surréduction basique (de gauche à droite) si toutes ses occurrences protégées (non basiques) donnent des sous-termes en forme normale. Si donc on normalise cette équation tout en conservant cette propriété, on pourra encore la résoudre. Cette condition n'est pas satisfaite par les occurrences basiques telles que les a définies Hullot, nous avons donc créé un nouveau calcul des occurrences basiques, appelé faiblement basique. Nous prouvons que cette nouvelle relation, appelée surréduction normalisante basique de gauche à droite, fournit une méthode complète d'unification. Elle été implantée en modifiant la procédure NARROWER du chapitre précédent.

Dans cet article, nous n'avons pas voulu nous restreindre à la surréduction normalisante, c'est pourquoi nous avons considéré une surréduction tout a fait générale, c'est à dire composée d'une étape de surréduction suivie d'une dérivation quelconque de réécriture (qui ne conduit pas forcément à la forme normale). Dans cet article, cette relation générale s'appelle narrowing est se note $-\hat{}\hat{}\rightarrow$. La surréduction s'appelle S-narrowing (S comme simple) et se note $-\hat{}\rightarrow$, la surréduction normalisante s'appelle N-narrowing et se note $-\hat{}\hat{}\hat{}\rightarrow$. Signalons qu'un récapitulatif des relations utilisées se trouve en appendice à la fin de l'article.

L'intérêt de cette nouvelle méthode de surréduction est évalué à la fin du chapitre 4, grâce à une étude sur la minimalité des solutions qu'elle génère.

Improving basic narrowing techniques[2]

Pierre Réty

Centre de Recherche en Informatique de Nancy
BP 239
54506 Vandoeuvre Les Nancy Cedex
France
E-mail: mcvax!inria!crin!rety

Abstract

In this paper, we propose a new and complete method based on narrowing for solving equations in equational theories. It is a combination of basic narrowing and normalizing narrowing, which is not obvious, because their naive combination is not a complete method. It provides an algorithm that has been implemented as an extension of the REVE software.

1 Introduction

Narrowing is a general method to solve equations in equational theories, that was introduced by Slagle [19], and studied by Fay [2, 1] and Hullot [8, 9]. It needs a convergent set of rewrite rules equivalent to the considered equational theory, and returns a complete set of solutions (also called unifiers), i.e. a basis of the set of all the solutions. Implementations are described in [10, 17]. But this method has drawbacks: it is inefficient and often does not terminate. Narrowing has some similarities with linear resolution principle of the Prolog language, and is used in logic programming language like Eqlog [3].

Let us describe what narrowing is. Assume that we have a convergent set of rewrite rules. The narrowing of a term t consists of two passes. The first one instantiates t so that it becomes reducible by a rule. The second one reduces it by this rule. The resulting term t_1 may be reducible into t_2 without any further instantiation. If so, t_2 is reduced until one gets a irreducible term (said in normal form) t_n. The relation that transforms t into t_1 was called narrowing in [9, 10], and we call the transformation of t into t_n normalizing narrowing. If one considers all the narrowing derivations issued from t (the narrowing tree), the intermediate terms t_1, t_2 are nodes from which edges are issued, while they do not appear in the tree using normalizing narrowing. So, the narrowing tree contains the normalizing narrowing tree.

In order to compute solutions of an equation $t = t'$ modulo a term rewriting system, one computes the narrowing derivations (normalizing or not) issued from $t = t'$, $=$ being considered as a binary function symbol, and check at each node whether the corresponding equation has a syntactic solution. If the term rewriting system is confluent and noetherian, all the solutions are found by building the whole narrowing tree (which can be infinite). In order to get a smaller tree, it is obviously better to use the normalizing

[2] This research was supported by the GRECO of programmation.

narrowing relation. Another idea [9] consists of using only narrowing at some occurrences called basic. This method is called basic narrowing, and gives another tree included into the narrowing tree.

Our idea is to mix the two previous relations, in order to further reduce the tree. The simplest way (we will say naive) is to consider their intersection. Unfortunately, the set of solutions that it provides is not always complete. Therefore, we had to build another relation in a non trivial way that preserves the completeness of the solution set. It use at normalization time a new computation of the basic occurrences based on the residual notion [6], that we will call weakly basic. It has been implemented as an extension of the REVE software.

In section 2, we introduce the basic concepts and definitions, and recall the existing results. In section 3, we consider the naive combination of normalizing and basic narrowings, and using an example, show that this method does not generate a complete set of solutions. Therefore, we propose in section 4 a new combination of the two relations and prove that it provides all the solutions in section 5. An implementation is presented in section 6. The various narrowing relations used in this paper are summarized in the appendix.

2 Definitions and existing results

In this section we introduce the concept of narrowing and recall that it provides a complete method for solving equation in a theory described by a confluent and noetherian term rewriting system. The following notations and properties are valid for the whole paper. They are consistent with [7, 11].

Let F be a set of symbols, X be a set of variables. A **term** is a partial application from N_+^* (the free monoid on N_+ whose elements are called **occurrences**) into $F \cup X$ that respects the symbol arities. $T(F,X)$ is the set of terms build on F and X. For each $t \in T(F,X)$, $D(t)$ is the set of occurrences of t, $O(t)$ is the set of non variable occurrences and $V(t)$ is the set of variables that occurs in t. t is said **linear** if each variable of t occurs once in t.

An **equation** $s = t$ is a pair of terms, a **rewrite rule** $s \to t$ is a directed pair of terms satisfying $V(t) \subseteq V(s)$. $t[u \leftarrow t']$ is the term obtained from t by changing the subterm of t at the occurrence u by t'. An **equational theory** A is a set of equations and one writes $=_A$ the smallest congruence induced by A. A **term rewriting system** R is a set of rewrite rules and $\to$ is the rewriting relation derived from R and $\to^*$ its transitive closure. A sequence of rewriting steps is called a **derivation**. A term t is said normalized if it is not reducible by $\to$, and the term t' is a normal form of t if $t \to^* t'$ and t' is normalized, t' is also denoted $t\downarrow$. R is **confluent** if for any term t, $t \to^* t_1$ and $t \to^* t_2$ implies there exists a term t' such that $t_1 \to^* t'$ and $t_2 \to^* t'$. R is **noetherian** if the relation $\to$ is noetherian. R is **interreduced** if for any rule $g \to d$ in R, d is normalized, and g is normalized with respect to $R - \{g \to d\}$. One says that R is **convergent** if it is confluent and noetherian, and **canonical** if it is also interreduced. R is **regular** if for all rule $g \to d$ in R, $V(d) = V(g)$. $=_R$ is the relation defined by $=_R = (\to \cup \leftarrow)^*$ where $\leftarrow$ is the rewriting relation obtained by reversing the rules of R.

Substitutions σ are defined as endomorphisms on $T(F,X)$ that extend mappings from X to $T(F,X)$ with a finite domain $D(\sigma)$. A substitution σ is denoted by $\{(x_1 / t_1), \ldots, (x_n / t_n)\}$.

We write $\leq$ the subsumption quasi-ordering on $T(F,X)$ defined by: $t \leq t'$ iff $t' = \sigma(t)$ for a substitution σ (called a match from t to t'). Composition of substitutions σ and ρ is denoted by $\sigma \cdot \rho$, then $(\sigma \cdot \rho)(t) = \sigma(\rho(t))$.

Given an equational theory A, two terms t and t' are said to be **A-unifiable** [15, 5] iff there exists a substitution σ such that $\sigma(t) =_A \sigma(t')$. σ is also called an **A-solution** of the equation $t = t'$. Given a subset V of X, we define $\sigma \leq_A \sigma'[V]$ iff $\sigma' =_A \sigma'' \cdot \sigma[V]$ for some substitution σ'' (the notation $[V]$ means that the formula is valid for any variable in V). If $V = X$, V is omitted. Γ is a **complete set of A-unifiers** of t and t' away from W containing the set V of the variables of t and t' iff:

- for all $\sigma \in \Gamma$, $D(\sigma) \subseteq V$ and $I(\sigma) \cap W = \emptyset$ (The goal of this technical restriction is only to avoid conflict between variables)
- for all $\sigma \in \Gamma$, $\sigma(t) =_A \sigma(t')$
- for all unifiers σ', there exists $\sigma \in \Gamma$ such that $\sigma \leq_A \sigma'[V]$.

In addition Γ is said to be **minimal** if it satisfies the further condition: for all σ and $\sigma' \in \Gamma$, $\sigma \leq_A \sigma'$ implies $\sigma = \sigma'$.

An A-unification algorithm is **complete** if it generates a complete set of A-unifiers. Note that this set may be infinite.

We now give a very general definition of narrowing by introducing any fixed mapping $\neg$ such that $\neg \subseteq \to^*$. One will say that a given derivation $s \to^* s'$ is **compatible** with $\neg$ iff $s \neg s'$.

Definition: We say that t is narrowable to t' at the occurrence u, using the rule $g \to d$ and with the substitution σ iff

- $t|u$ and g are unifiable by the most general unifier σ
- $t_1 = \sigma(t)[u \leftarrow \sigma(d)]$
- $t_1 \neg t'$

We call this relation **narrowing** and denote it $t \overset{}{\leadsto}_{[u, g \to d, \sigma]} t'$. A sequence of narrowing steps is called a **narrowing derivation**.

This definition is generic because by choosing the mapping $\neg$ one obtains different narrowing relations, in particular the two followings:

- If $\neg$ is the identity then $t_1 = t'$. We have the relation called narrowing by Hullot [9], and that we will call **simple narrowing** or **S-narrowing** and we write $t \overset{}{\leadsto}_{[u, g \to d, \sigma]} t'$. A sequence of S-narrowing steps is called a **S-narrowing derivation**.
- If $\neg$ is the normalization mapping then t' is in normal form. We have the relation called narrowing by Fay [2]. We propose to call it **normalizing narrowing** or **N-narrowing** and we denote it by $t \overset{}{\leadsto}_{[u, g \to d, \sigma]} t'$. A sequence of N-narrowing steps is called a **N-narrowing derivation**.

With these notations we have:

$$\left[t \overset{}{\leadsto}_{[u, g \to d, \sigma]} t' \right] \Leftrightarrow \left[t \overset{}{\leadsto}_{[u, g \to d, \sigma]} t_1 \text{ and } t' = t_1 \downarrow \right]$$

$$\to \subseteq \overset{}{\leadsto} \qquad \overset{}{\leadsto} \subseteq \overset{}{\leadsto}^*$$

If σ is a match from g to t/u, the step $t \overset{}{\leadsto} t_1$ is in fact a rewriting step.

In the following we suppose the mapping $\neg$ fixed, so that the narrowing relation $\overset{}{\leadsto}$ is fixed.

The narrowing relation provides a method to compute a complete set of unifiers of two terms modulo a convergent term rewriting system. The method consists in building all the possible narrowing derivations issued from $t_0 = t_0'$ and to collect the corresponding narrowing substitutions, until we obtain equations

$t_n = t'_n$ such that t_n and t'_n are unifiable. The unification problem in the equational theory is then reduced to the narrowing together with the standard unification of terms.

In order to iterate the narrowing process on the two terms, = is considered as a new operator of the equational theory, and the process starts with the term $t_0 = t'_0$. It is obvious that if $t_0 = t'_0 \overset{*}{\leadsto} t$ then t is of the form $t_i = t'_i$.

The following result has been proved by Hullot [8] for the S−narrowing, by C. and H. Kirchner [12, 13] and Réty(et al) [18] for any narrowing relation.

Theorem: Let R be a convergent term rewriting system, t_0 and t'_0 be two terms. The set of substitutions σ such that

- there exists a narrowing derivation issued from $t_0 = t'_0$ $t_0 = t'_0 \overset{}{\leadsto}_{[\sigma_1]} t_1 = t'_1 \overset{}{\leadsto} \ldots \overset{}{\leadsto}_{[\sigma_n]} t_n = t'_n$ such that t_n and t'_n are unifiable by the most general unifier β and that $\beta \cdot \sigma_n \ldots \sigma_1$ is normalized on $V(t_0 = t'_0)$
- $\sigma = \beta \cdot \sigma_n \ldots \sigma_1$

is a complete set of R−unifiers of t_0 and t'_0.

Basic S−narrowing was defined and studied by Hullot. It consists of protecting the occurrences resulting from the substitution. This notion was extended by Herold [4] to left−to−right basic S−narrowing. It consists of protecting moreover the occurrences that occurs strictly left of the applied occurrence.

Definition [Hullot, Herold]: Given the S−narrowing derivation

$$t_0 \overset{}{\leadsto}_{[u_0, g_0 \to d_0, \sigma_0]} t_1 \overset{}{\leadsto} \ldots \overset{}{\leadsto}_{[u_{n-1}, g_{n-1} \to d_{n-1}, \sigma_{n-1}]} t_n \qquad (1)$$

and $U_0, \ldots, U_n$ sets of non variable occurrences of $t_0, \ldots, t_n$ respectively. One says that this S−narrowing derivation is **based on** U_0 iff for all i

$$u_i \in U_i$$

$$U_{i+1} = \left[U_i - \{v \in U_i \,/\, u_i \leqq v\}\right] \cup \{u_i \cdot v \,/\, v \in O(d_i)\}$$

We write $U_{i+1} = B(U_i)$, or $U_{i+1} = B(t_{i+1}, (1))$ where (1) specifies which derivation is considered, or more simply $U_{i+1} = B(t_{i+1})$ if it is not ambiguous. The occurrences that belong to $U_0, \ldots, U_n$ are said basic.

If it is not ambiguous we will say more simply that this S−narrowing derivation is **basic**, or that this is a **basic S−narrowing derivation**.

This S−narrowing derivation is **left−to−right** based on U_0 iff for all i

$$U_{i+1} = \left[U_i - \{v \in U_i \,/\, u_i \leqq v \text{ or } (v = w \cdot j' \cdot w'', u_i = w \cdot j \cdot w'; \text{and} j' < j)\}\right] \cup \{u_i \cdot v \,/\, v \in O(d_i)\}$$

We write $U_{i+1} = LB(U_i)$.

If it is not ambiguous we will say more simply that this S−narrowing derivation is **left−to−right basic**, or that this is a **left−to−right basic S−narrowing derivation**.

Remark: B is monotonic i.e. $U \subseteq U'$ implies $B(U) \subseteq B(U')$, and preserves the closure by prefix i.e. U is closed by prefix implies $B(U)$ is closed by prefix. LB satisfies these properties and moreover preserves the right closure.

The basic method consists in building all the S−narrowing derivations issued from $t_0 = t'_0$ and left−to−right based on $O(t_0 = t'_0)$.

Theorem [Herold]: The previous theorem is still valid when we restrict to S−narrowing derivations left−to−right based on $U_0 = O(t_0 = t'_0)$.

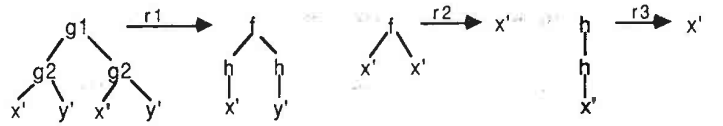
One of the interests of the basic S−narrowing is its termination property.

Termination property [Hullot]: If all the basic S−narrowing derivations issued from a right hand side of a rewrite rule terminate, then all basic S−narrowing derivation issued from any term terminates.

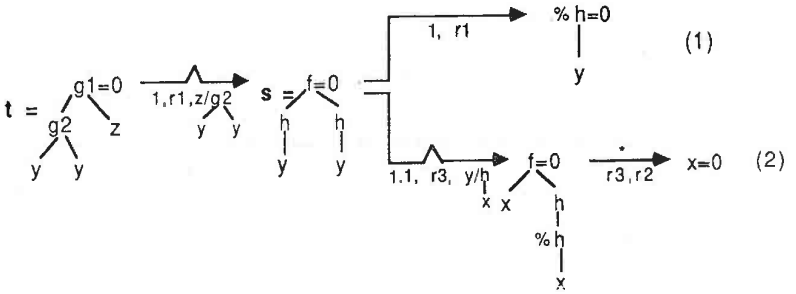
3 The naive basic narrowing

Actually, the "basic" concept is only used for the S−narrowing, and we are loading to extend it to all the narrowing relations (including the N−narrowing). Let us recall that a narrowing step is formed by a step of S−narrowing followed by steps of rewriting. The first idea that comes to mind is to define the basic occurrence sets along the rewriting steps as Hullot did, i.e. by using the mapping B. But this method misses some solutions.

Example 1: Consider the canonical term rewriting system R.



Let $t = g_1(g_2(y, y), z)$ and suppose one wants to solve modulo R the equation $t = 0$. The "%" symbol means that the corresponding occurrence is not basic. $t = 0$ is considered as a term whose top symbol is =. The tree of basic S−narrowing is formed by the branches (1) and (2):



The branch (2) gives the substitution $\sigma = (y / h(0), z / g_2(h(0), h(0)))$ which is the unique solution.

The leaf of the branch (1) can not be narrowed at a basic occurrence, and since $h(y)$ and 0 are not unifiable this branch does not give a solution.

If one uses the naive basic N−narrowing, the term s disappears from the tree, and with it the branch (2). Therefore the solution will not be found by this method.

Nevertheless in order to find σ, our idea is to compute on a larger set of basic occurrences during the rewriting steps. This computing will be said weakly basic. For it the term $h(y) = 0$ can be narrowed into $x = 0$ by the rule r_3 using the substitution $(y / h(x))$ which gives the solution σ.

Another difficulty is that the rewriting steps do not respect the basic occurrences.

Example 2: Let R be the canonical term rewriting system that contains the associativity rule: $R = \{f(x, f(y,z)) \to f(f(x,y),z)\}$. Let us apply basic N−narrowing on the term $f(f(y;x'),x')$. One possibility is:

$$f(f(y;x'),x') \leadsto_{[\varepsilon,\sigma]} t: = f(f(\%f(y;f(y,z)),y),z)$$
$$\text{with } \sigma = (x'/f(y,z), x/f(y;f(y,z)))$$

The occurrence pointed out by % is the unique occurrence of t on which the rule can be applied. Since this occurrence is not basic it is not possible to normalize t by a basic derivation.
In the following, we will define a property on basic occurrence sets that will guarantee that basic normalization is possible.

4 The basic narrowing

The aim of the following definition is to characterize the sets of basic occurrences that allow to find a solution. We will say that a set of occurrences U is sufficiently large on a term t if all the subterms that correspond to occurrences outside of U are normalized.

Definition: Let t be a term, U a set of occurrences of t, we say that U is **sufficiently large** on t iff:

$$(u \in D(t) \text{ and } u \notin U) \Rightarrow t/u \text{ is in normal form.}$$

Lemma 1: Let t_0 be a term, U_0 a set of occurrences of t_0 sufficiently large on t_0. Then all the derivations issuing from t_0 and following a leftmost innermost strategy $t_0 \to t_1 \to \ldots \to t_n$ are left−to−right based on U_0. If we denote by $U_0, \ldots, U_n$ the sets of basic occurrences then for all $0 \leq i \leq n$, U_i is sufficiently large on t_i.

Proof: By induction on the size of the derivation.

If $n = 0$ the lemma obviously holds. If the property is true for i, U_i is sufficiently large on t_i, then the step $t_i \to_{[u_i, g_i \to d_i, \sigma_i]} t_{i+1}$ satisfies $u_i \in U_i$. Since the strategy is innermost, the match σ_i is normalized. Since the strategy is leftmost, the part occurring strictly left to u_i is normalized. Therefore the non basic occurrences of t_{i+1} are normalized. ◆

Corollary 1: If U_0 is sufficiently large on t_0, there exists a derivation left−to−right based on U_0, leading to the normal form of t_0 and such that for any term t_i in this derivation, the set U_i of the basic occurrences of t_i is sufficiently large on t_i.

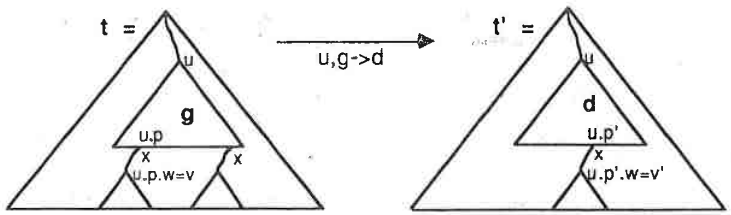
But, there exist basic derivations that do not preserve the sufficient largeness property of the occurrence sets. For instance, consider the rewriting system of the example 1, the term $t = f(h(h(x)), h(h(x)))$, and the occurrence set $U = \{\varepsilon, 1, 11, 2, 21\}$. U is sufficiently large on t, $t \to_{[\varepsilon, r_2]} t' = h(h(x))$ and $B(t') = \emptyset$. Since t' is not normalized, $B(t')$ is not sufficiently large on t'.

We must define a new notion of basic derivation, that always preserves the sufficient largeness property. For that, we introduce the antecedent notion that is (nearly) the dual of the residual notion introduced by

Church for the λ−calculus and by Huet and Levy [6] for left−linear term rewriting systems. It characterizes the fact that along a rewriting step, a subterm can be preserved.

Definition: Let $t \to_{[u, g \to d, \sigma]} t'$ be a step of rewriting and $v' \in D(t')$. We say that the occurrence v of t is an **antecedent** of v' iff

$v = v'$ and are not comparable to u
or
there exits an occurrence p' of a variable x in d such that
$v' = u.p'.w$



We extend this definition to a derivation by transitive closure of the rewriting relation. We say that v' is a **residual** of v iff v is an antecedent of v'.

Remarks: With the notations of the previous definition we have:

- $t'/v' = t/v$
- v' may have no antecedent if $v' = u.p'$ with $p' \in O(d)$ or if $v' < u$,
- v' may have several antecedents if g is not linear.

Definition: Given the derivation

$$t_0 \to_{[u_0, g_0 \to d_0]} t_1 \to \ldots \to_{[u_{n-1}, g_{n-1} \to d_{n-1}]} t_n$$

and $U_0, \ldots, U_n$ sets of non variable occurrences of $t_0, \ldots, t_n$ respectively. We say that this derivation is **weakly based on** U_0 iff for all i

- $u_i \in U_i$
- $U_{i+1} = \left[U_i - \{v \in U_i / u_i \leq v\}\right] \cup \{u_i.v / v \in O(d_i)\}$
  $\cup \{v \in O(t_{i+1}) / v = u_i.w, w \notin O(d_i)$ and all antecedents of v in t_i are in $U_i\}$

We write $U_{i+1} = WB(U_i)$, $U_{i+1} = WB(t_{i+1}, (1))$ or more simply $U_{i+1} = WB(t_{i+1})$ if it is not ambiguous. One will say U_{i+1} is the base of t_{i+1}. The occurrences that belong to $U_0, \ldots, U_n$ are said basic. If it is not ambiguous we will say more simply that this derivation is **weakly basic**, or that it is a **weakly basic derivation**.

Remark: WB is increasing i.e. $U \subseteq U'$ implies $WB(U) \subseteq WB(U')$, and preserves the closure by prefix i.e U is closed by prefix implies $WB(U)$ is closed by prefix.

This definition differs from Hullot's one by addition of the last line, i.e. the occurrences under d_i may belong to U_{i+1}. Therefore $B(U_i) \subseteq WB(U_i)$.

In practice the set U_0 is supposed to be closed by prefix, and since the weakly basic reduction preserves this property then all the U_i are closed by prefix. Therefore we can get a new and simpler definition of *WB* by writing:

- $U_{i+1} = \{v \in O(t_{i+1}) \ / \text{ all the antecedents of } v \text{ in } t_i \text{ are in } U_i\}$

In the following we will use this definition.

The interest of the weakly basic derivations is pointed out in the following lemma, that emphasizes the fact that the notions of weakly basic derivation and sufficient large occurrence set are very linked.

Lemma 2: Let $t_0 \rightarrow^* t_n$ be a derivation, and U_0 be a set of occurrences of t_0 sufficiently large on t_0. Then $t_0 \rightarrow^* t_n$ is weakly based on U_0 and the set U_n of basic occurrences of t_n is sufficiently large on t_n.

Proof: By induction on the length n of the derivation.
If $n = 0$ the lemma obviously holds. Assume $t_0 \rightarrow^* t_{n-1}$ is weakly basic on U_0 and the basic occurrence set U_{n-1} of t_{n-1} is sufficiently large on t_{n-1}. Thus the reduction occurrence of $t_{n-1} \rightarrow t_n$ must be in U_{n-1}. Let U_n be the basic occurrence set of t_n and $v_n \in D(t_n)$ such that $v_n \notin U_n$. From the definition, there exists at least an antecedent v_{n-1} of v_n in t_{n-1} that does not belong to U_{n-1}. Therefore $t_n/v_n = t_{n-1}/v_{n-1}$ which is normalized by hypothesis. ◆

We can now define the basic narrowing with sufficient largeness as a step of left−to−right basic S−narrowing such that the sufficient largeness property is preserved, followed by a derivation compatible with $\neg$. The previous lemma ensures that this derivation is weakly basic, and that the method will be complete.

Definition: Let t_0 be a term, U_0 an occurrence set of t_0, the step of narrowing $t_0 \overset{\wedge}{\rightsquigarrow} t_n$ (which is equivalent to $t_0 \overset{\wedge}{\rightsquigarrow} t_1 \neg t_n$ i.e. $t_0 \overset{\wedge}{\rightsquigarrow} t_1 \rightarrow \ldots \rightarrow t_n$) is said **left−to−right based on U_0 with sufficient largeness** or **left−to−right SL−based on U_0** iff there are occurrence sets $U_1, \ldots, U_n$ such that:

a) $t_0 \overset{\wedge}{\rightsquigarrow} t_1$ is left−to−right based on U_0 and $U_1 = LB(U_0)$,
b) U_1 is sufficiently large on t_1,
c) For all $i \in \{1, \ldots, n-1\}$, $U_{i+1} = WB(U_i)$.

We extend this definition to a narrowing derivation and we will say that a narrowing derivation is left−to−right SL−based on U_0. If the set U_0 is not specified we will say more simply that this narrowing derivation is **left−to−right SL−basic**, or that it is a **left−to−right SL−basic narrowing derivation**.

This definition prunes the narrowing tree because all the nodes that do not satisfy the sufficient largeness property are cut. As the narrowing definition, this definition is generic and can be instanciated in two particular cases: the **left−to−right SL−basic S−narrowing** when $\neg$ is the identity, and the **left−to−right SL−basic N−narrowing** when $\neg$ is the normalization mapping.
The left−to−right SL−basic S−narrowing and the left−to−right basic S−narrowing (from Herold) are not the same relations because of the sufficient largeness property imposed by the point b) in the previous definition. The left−to−right SL−basic S−narrowing relation is included in the left−to−right basic S−narrowing relation.

5 The proof of completeness

We use here a short and original proof method that was introduced by C.Kirchner [12] for equational narrowing.

We must first introduce technical definitions.
Definitions: Let

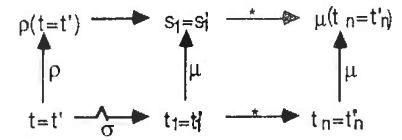
- $SU(t = t', R)$ be the set of the R−unifiers of t and t' (*SU* as set of unifiers),
- $BU(t = t', R, U)$ be the set of substitutions found by the considered narrowing relation $\overset{\wedge}{\rightsquigarrow}$ on the equation $t = t'$ and left−to−right SL−based on U, using the term rewriting system R (*BU* as basic unifiers),
- $SU^{-}N(t = t', R, U) = \{\sigma \ / \ \sigma$ is a normalized R−unifier of t and t' and U is sufficiently large on $\sigma(t = t')\}$.

Lemma 3: Let R be a convergent term rewriting system, t and t' two terms, and U a subset of $O(t = t')$. Then $SU^{-}N(t = t', R, U) \subseteq BU(t = t', R, U)$.

Proof: Let ρ be any element in $SU^{-}N(t = t', R, U)$, and let us prove that $\rho \in BU(t = t', R, U)$.
For that, we prove by noetherian induction on $\rightarrow$ that for a given term in the form $\rho(t = t')$ and a given subset U of $O(t = t')$ such that $\rho \in SU^{-}N(t = t', R, U)$, we have $\rho \in BU(t = t', R, U)$.
If $\rho(t = t')$ is normalized then ρ is a syntactic unifier of t and t', then it is an element of $BU(t = t', R, U)$. Otherwise, $\rho(t = t')$ is reducible, and since U is sufficiently large on $\rho(t = t')$, it is reducible into its normal form by a derivation left−to−right based on U (corollary 1). Consider the first step of this derivation: $\rho(t = t') \rightarrow s_1 = s_1'$. From the correspondance lemma between rewriting and S−narrowing (Hullot [8]), there exists a term $t_1 = t_1'$ and a substitution μ such that:

$$t = t' \overset{\wedge}{\rightsquigarrow}_{[\sigma]} t_1 = t_1', \quad \mu(t_1 = t_1') = s_1 = s_1', \quad \rho = \mu \circ \sigma$$

which is summarized by the following diagram:



From corollary 1, the set U_1 of basic occurrences of $s_1 = s_1'$ is sufficiently large on $s_1 = s_1'$, which is $\mu(t_1 = t_1')$. Let $t_n = t_n'$ be the term defined by $t_1 = t_1' \neg t_n = t_n'$. We have $t_1 = t_1' \rightarrow^* t_n = t_n'$ and consider the instantiated derivation $\mu(t_1 = t_1') \rightarrow^* \mu(t_n = t_n')$. From lemma 2, it is weakly based on U_1 and the set U_n of basic occurrences of $\mu(t_n = t_n')$ is sufficiently large on $\mu(t_n = t_n')$.
μ is normalized, and is a R−unifier of t_1 and t_1' then of t_n and t_n'. U_n is sufficiently large on $\mu(t_n = t_n')$, then $\mu \in SU^{-}N(t_n = t_n', R, U_n)$.
$\mu(t_n = t_n')$ is a strict son of $\rho(t = t')$ for the rewriting relation, so by induction hypothesis we deduce that $\mu \in BU(t_n = t_n', R, U_n)$. Since $\rho = \mu \circ \sigma$, then $\rho \in BU(t = t', R, U)$. ◆

Notation: We write $A \subseteq_R B$ for A, B sets of substitutions iff for any $\sigma \in A$ there exists $\theta \in B$ such that $\sigma =_R \theta$.

Corollary: $BU(t = t', R, O(t = t')) =_R SU(t = t', R)$.

Proof: $SU\text{-}N(t = t'; R, O(t = t')) \subseteq BU(t = t'; R, O(t = t'))$ from the previous lemma
$\subseteq SU(t = t'; R)$ by correctness of the narrowing
$\subseteq_R SU\text{-}N(t = t'; R, O(t = t'))$ by definition. ◆

Any left−to−right SL−basic narrowing relation provides a complete method for unifying in a convergent term rewriting system.

Theorem: Let R be a convergent term rewriting system, t_0 and t_0' be two terms. The set of substitutions σ such that

- there exists a narrowing derivation issued from $t_0 = t_0'$ and left−to−right SL−based on $O(t_0 = t_0')$: $t_0 = t_0' \hat{\rightarrow}_{[\sigma_1]} t_1 = t_1' \hat{\rightarrow} \ldots \hat{\rightarrow}_{[\sigma_n]} t_n = t_n'$ such that t_n and t_n' are unifiable by the most general unifier β and that $\beta \cdot \sigma_n \ldots \sigma_1$ is normalized on $V(t_0 = t_0')$
- $\sigma = \beta \cdot \sigma_n \ldots \sigma_1$

is a complete set of R−unifiers of t_0 and t_0'.

6 Implementation

We have implemented SL−basic N−narrowing within an experimental version of the rewriting software REVE [14] as a modification of the procedure NARROWER [17]. In order to mark the basic occurrences, we have bound a boolean to each occurrence of term, which is called occurrence indicator. Now, our implementation does not check the sufficient largeness property and the rewriting steps are not exactly weakly basic because an occurrence is considered to be basic if the most left antecedent is basic, which is less optimized than our definition.

The weakly basic rewriting steps can be done as follows. Let $t \rightarrow_{[u, g \rightarrow d, \sigma]} t'$ be a weakly basic rewriting step; the occurrence indicator of an occurrence v from t' is the boolean product of the occurrence indicators of its antecedents in t. Therefore the occurrence indicators of the part of t' resulting from σ can be computed whitin the matching process. Remark that this computation is very simple when g is linear.

Appendix: denomination of the various narrowings

A reduction is a sequence of rewriting steps, a normalization is a reduction that leads to the normal form.

narrowing relation	definition	denoted in the literature by:
simple narrowing or S−narrowing (denoted by $-\hat{\rightarrow}$)	more general instantiation and reduction by one rule	narrowing [9]
narrowing (denoted by $-\hat{\rightarrow}$)	step of S−narrowing followed by a given reduction	
normalizing narrowing or N−narrowing (denoted by $-\hat{\rightarrow}$)	step of S−narrowing followed by a normalization	narrowing [2, 1], narrowing with eager reduction [16, 10]
basic S−narrowing	S−narrowing with respect to occurrences obtained by a basic computation	basic narrowing [9]
basic narrowing	step of basic S−narrowing followed by a given and weakly basic reduction	
basic N−narrowing	step of basic S−narrowing followed by a weakly basic normalization	
SL−basic S−narrowing	step of S−narrowing such that the leaded term satisfies the sufficient largeness property	
SL−basic narrowing	step of SL−basic S−narrowing followed by a given and weakly basic reduction	
SL−basic N−narrowing	step of SL−basic S−narrowing followed by a weakly basic normalization	

References

[1] M. Fay: "First−Order Unification in An Equational Theory", *Master Thesis, U. of California At Santa Cruz* , Tech. Report 78−5−002, May 1978.

[2] M. Fay: "First−Order Unification in an Equational Theory", *Proceedings of the 4th Workshop on Automated Deduction* , pp. 161−167, Austin, Texas, 1979.

[3] J. Goguen and J. Meseguer: "EQLOG: Equality, types, and generic Modules for logic programming", *in Functional and Logic Programming* , D. DeGroot and G. Lindstrom, editors, Springer−Verlag, 1985.

[4] A. Herold: "Narrowing techniques applied to idempotent unification", *SEKI report SR−86−16* , August 1986.

[5] G. Huet: "Résolution d'équations dans les langages d'ordre 1,2,...ω", *Thèse d'état* , Université de Paris VII, 1976.

[6] G. Huet and JJ. Levy: "Call by need Computations in Non−ambiguous Linear Term Rewriting Systems", *INRIA Laboria, report 359* , 1979.

[7] G. Huet and D. Oppen: "Equations and Rewrite Rules: A Survey", *in Formal Language Theory: Perspectives and Open Problems* , ed. Book R, Academic Press, 1980.

[8] J.M. Hullot: "Compilation de Formes Canoniques dans les Théories Equationnelles", *Thèse de 3ème Cycle* , Université de Paris Sud, 1980.

[9] J.M. Hullot: "Canonical Forms And Unification", *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.

[10] A. Josephson and N. Dershowitz: "Efficient Implementation of Narrowing: the RITE way", *in Proceedings of the International Conference of Logic Programming* , Salk Lake City, 1986.

[11] J−P. Jouannaud, C. Kirchner, 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, 1983.

[12] C. Kirchner: "Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles", *Thèse de doctorat d'Etat* , Université de Nancy I, 1985.

[13] H. Kirchner: "Preuves par complétion dans les variétés d'algèbres", *Thèse de doctorat d'Etat* , Université de Nancy I, 1985.

[14] P. Lescanne: "Computer experiments with the REVE term rewriting system generator", *in 10th ACM Conf. on Principles of Programming Languages* , pp. 99−108, Austin Texas, January 1983.

[15] G. Plotkin: "Building−in Equational Theories", *in Machine Intelligence* , vol. 7, pp. 73−90, 1972.

[16] U.S. Reddy: "On the relationship between logic and functional languages", *in Logic Programming: Relations, Functions, and Equations* , D. DeGroot and G. Lindstrom, eds. Prentice Hall, Englewood Cliffs, NJ, 1985.

[17] P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne: "NARROWER: A new Algorithm for Unification and its application to Logic Programming", *Proc. 1rst Conf. on Rewriting Techniques and Applications* , Lecture Notes in Computer Science,vol. 202, pp. 141−157, Springer Verlag, Dijon France, 1985.

[18] P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne: "NARROWER: An algorithm for unification based on narrowing", *CRIN report 86−R−131* , 1986.

[19] J.R. Slagle: "Automated theorem−proving for theories with simplifiers, commutativity and associativity", *J. of ACM* , vol 21, pp. 622−642, 1974.

Chapitre 4

Etude de la minimalité de l'ensemble des solutions générées par surréduction

Le but de ce chapitre est de comparer les relations de surréduction en étudiant la minimalité de l'ensemble des solutions générées par surréduction. Il fournit un résultat pour la surréduction basique de gauche à droite en supposant que le système de réécriture est régulier et sans paires critiques, et fait apparaitre l'intérêt de la suppression des branches instances (chapitre 2) et de l'introduction de la réécriture faiblement basique (chapitre 3).

Pour établir ces résultats, des étapes préliminaires sont nécessaires, et en particulier il va falloir établir une propriété de commutation de la surréduction. Commuter une surdérivation de deux étapes consiste à changer l'ordre dans lequel on utilise les règles de réécriture. La commutation a déjà été étudiée par Huet et Lévy [34] pour la réécriture, et par Herold [29] pour la surréduction dans le cas où les occurrences ne sont pas comparables.

En vue d'une extension ultérieure des travaux du chapitre précédent au cadre de la logique ordo-sortée et et/ou à la réécriture équationnelle, nous prouvons directement nos résultats dans un cadre ordo-sorté, puis dans un cadre ordo-sorté avec des systèmes de réécriture équationnelle. Dans ce chapitre R est un système de réécriture et E un système d'équations, tout deux fixés. Les définitions de la RE-réécriture et de la RE-surréduction sont données au chapitre 1.

Détaillons le plan du chapitre. La section 1 donne quelques définitions. La section 2 est une étude sur la commutation de la réécriture. La section 3 présente des lemmes techniques, qui serviront à établir la propriété de commutation de la surréduction, présentée en section 4. La section 5 donne des exemples de commutation. La section 6 contient une étude sur la commutation de la surréduction équationnelle. La section 7 présente un résultat de comparaison de la surréduction basique de gauche à droite avec la $\rightsquigarrow$-surréduction basique de gauche à droite (introduction de la réécriture faiblement basique). Enfin la section 8 présente une étude sur la minimalité des ensembles de solutions.

4.1 Antécédent et résidu

La surréduction, et donc en particulier la réécriture, ne modifie pas complètement les termes sur lesquels elle s'applique. En effet au cours d'une étape $t \rightsquigarrow_{[q,g\to d,\sigma]} t'$, les sous-

termes d'occurrence non comparable à q ou bien ceux situés en dessous de l'application de g dans t, disons $t|v$, ne sont pas détruits mais simplement instanciés par σ. Ils pourront donc apparaitre dans t', par exemple à l'occurrence v', et peuvent être dupliqués plusieurs fois. On dira alors que v' est un résidu de v ou encore que v est un antécédent de v'.
Définissons formellement ces notions d'abord pour la surréduction non équationelle, et ensuite dans le cas général. Nous donnons une définition explicite de l'antécédent, et nous définissons le résidu comme sa réciproque, car dans la suite c'est surtout l'antécédent qui nous sera utile.

Cette définition est donnée dans le cas de la surréduction. Elle reformule d'une manière plus directe, mais équivalente, la définition donnée au chapitre précédent. La réécriture étant un cas particulier de surréduction, elle fournit aussi une définition pour la réécriture.

Définition 4.1 Soit $t \leadsto_{[q,g\to d,\sigma]} t'$ une étape de surréduction non équationnelle, et v' une occurrence de t'. On dit que l'occurrence v est un **antécédent** de v' dans t à travers cette étape si:

1. $v \in D(t)$
2. v' n'est pas comparable à q et $v = v'$
   ou
   il existe une occurrence de variable p' (notons x la variable en question) de d telle que:

$$\begin{aligned} v' &= q.p'.w \\ v &= q.p.w \text{ où } p \text{ est une occurrence de } x \text{ dans } g. \end{aligned}$$

On dit que v' est un **résidu** de v si v est un antécédent de v'. ◇

Remarques: Avec les notations précédentes on a:

- $t'|v' = \sigma(t|v)$ et s'il s'agit de réécriture on a même $t'|v' = t|v$.
- Si g n'est pas linéaire, v' peut avoir plusieurs antécédents. Si d n'est pas linéaire, v peut avoir plusieurs résidus.
- Si $v' < q$ ou s'il existe $p' \in O(d)$ tel que $v' = q.p'$, alors v' n'a pas d'antécédent. De même si $v < q$ ou s'il existe $p \in O(g)$ tel que $v = q.p$, alors v n'a pas de résidu, ce qui diffère de la définition de Huet et Lévy [34, p 5] où il peut y avoir résidu quand $v < q$.

Etendons nos concepts à plusieurs étapes de surréduction.

Définition 4.2 Soit

$$t_0 \leadsto_{[p_0,g_0\to d_0,\sigma_0]} t_1 \leadsto \ldots \leadsto t_n$$

une dérivation de surréductions, v_0 une occurrence de t_0, v_n une occurrence de t_n. On dit que v_0 est un **antécédent** de v_n à travers cette dérivation s'il existe des occurrences $v_1 \in D(t_1), \ldots, v_{n-1} \in D(t_{n-1})$ telles que pour tout $0 \leq i < n$, v_i est un antécédent de v_{i+1} à travers l'étape $t_i \to t_{i+1}$. On dit que v_n est un **résidu** de v_0 si v_0 est un antécédent de v_n.
◇

Pour définir les notions d'antécédent et de résidu à travers des étapes de E-égalité, on va transformer le système d'équations E en un système de réécriture, puis on appliquera les définitions précédentes.

Définition 4.3 Soit E un ensemble d'équations. On appelle **système de réécriture associé à** E, noté $\mathbf{E_{\mathcal{R}}}$, le système de réécriture non nœthérien formé des équations de E orientées dans les deux sens. ◇

Dans le cas non-sorté on a $t =_E t' \Longleftrightarrow t \overset{*}{\to}^{E_{\mathcal{R}}} t'$, mais ce n'est pas forcément vrai dans le cas ordo-sorté. Nous nous plaçons dans le cadre ordo-sorté en supposant néanmoins que cette propriété est satisfaite.

Hypothèse générale 4.1 *Dans tout ce chapitre, E est tel que pour tous termes t et t':*

$$t =_E t' \Longleftrightarrow t \overset{*}{\to}^{E_{\mathcal{R}}} t'$$

Il est maintenant facile de définir l'antécédent et le résidu à travers une E-égalité.

Définition 4.4 Soit $t =_E t'$ une E-égalité, v une occurrence de t, v' une occurrence de t'. On dit que v est un **antécédent** de v' à travers cette E-égalité si v est un antécédent de v' dans la dérivation $t \overset{*}{\to}^{E_{\mathcal{R}}} t'$. On dit que v' est un **résidu** de v si v est un antécédent de v'.
◇

La RE-surréduction peut être vue comme la composée de trois étapes: instanciation, E-égalité, réécriture non équationnelle. Il est donc maintenant facile d'étendre les notions d'antécédent et de résidu à la RE-surréduction.

Définition 4.5 Soit $t \leadsto^{RE}_{[q,g\to d,\sigma]} t'$ une étape de RE-surréduction, v' une occurrence de t', v une occurrence de t. On a $\sigma(t) =_E \sigma(t[q \leftarrow g]) \to t'$.
On dit que v est un **antécédent** de v' s'il existe une occurrence w de $\sigma(t[q \leftarrow g])$ telle que w soit un antécédent de v' et v soit un antécédent de w. On dit que v' est un **résidu** de v si v est un antécédent de v'.
Cette définition peut s'étendre à plusieurs étapes de RE-surréduction de la même manière que dans la définition 4.2. ◇

4.2 Commutation de la réécriture

Huet et Lévy [34] ont établi un résultat de commutation sur la réécriture standart dans le cas des systèmes de réécriture sans paires critiques et linéaires à gauche (voir aussi les travaux de Boudol [5]). L'utilisation d'une notion de résidu un peu plus restrictive que la leur nous permet d'étendre leurs résultats au cas où il y a des paires critiques. La commutation de Huet et Lévy est une commutation "en marche avant", elle consiste à faire glisser une étape $s \to t$ à travers une ou plusieurs étapes issues de t, et elle utilise la notion de résidu. Pour comparer les relations de surréduction, nous avons besoin de commuter "en marche arrière', c'est à dire de faire glisser $s \to t$ à travers une ou plusieurs étapes aboutissant à s, et pour cela nous utilisons la notion d'antécédent. Ainsi, l'hypothèse qu'il nous faut n'est pas la linéarité à gauche du système de réécriture, mais la linéarité à droite. Enfin nous étendrons ces résultats à la réécriture équationnelle. Pour cela, la méthode de filtrage $Filtre_E$ ne pourra pas être quelconque, mais devra satisfaire une hypothèse de stabilité.

La commutation en deux coups ne nécessite aucune hypothèse particulière.

Lemme 4.1 (commutation de la réécriture non équationnelle)
Soient s, t, u trois termes. Soit

$$s \to_{[p,g\to d]} t \to_{[q,l\to r]} u \quad (1)$$

deux étapes de réécriture telles que q admet des antécédents dans s, notés $p_0, \ldots, p_{m-1}$ (liste exhaustive).
Alors (1) peut être commuté en:

$$s \rightarrow_{[p_0, l \rightarrow r]} t'_1 \rightarrow \ldots \rightarrow_{[p_{m-1}, l \rightarrow r]} t'_m \rightarrow_{[p, g \rightarrow d]} u' \quad (2)$$

tel que $u \xrightarrow{*}_{[q_1, \ldots, q_n; l \rightarrow r]} u'$ où $q, q_1, \ldots, q_n$ sont les résidus de $p_0, \ldots, p_{m-1}$ dans t.

Preuve: Evidente. □

Remarque: Si d est linéaire ou u est normalisé ou p et q ne sont pas comparables, alors $u' = u$. Autrement dit la commutation ne change pas le dernier terme de la dérivation.

Nous cherchons maintenant à établir une propriété de commutation sur la réécriture équationnelle. Caractérisons le fait qu'en remontant une dérivation, tout les antécédents d'une occurrence donnée ont eux même des antécédents, c'est à dire qu'aucun antécédent n'est perdu au cours de la dérivation.

Définition 4.6 Soit $t_0 \rightarrow \ldots \rightarrow t_n$ une dérivation et v une occurrence de t_n. On dit que v admet tous ses antécédents dans t_0 si pour toute étape $t_i \rightarrow t_{i+1}$ de la dérivation et pour tout antécédent v_{i+1} de v dans t_{i+1}, v_{i+1} admets des antécédents dans t_i. ◇

Lemme 4.2 (commutation à travers plusieurs étapes)
Supposons que R soit linéaire à droite. Soient

$$t_0 \rightarrow_{[q_0, g_0 \rightarrow d_0]} \cdots \rightarrow_{[q_{n-1}, g_{n-1} \rightarrow d_{n-1}]} t_n \quad (1)$$

$$t_n \rightarrow_{[v, l \rightarrow r]} t_{n+1} \quad (2)$$

deux dérivations telles que v admette à travers (1) tous ses antécédents dans t_0, notés $p_0, \ldots, p_{m-1}$ (liste exhaustive).
Alors (2) peut être commuté à travers (1) en

$$t_0 \xrightarrow{*}_{[p_0, \ldots, p_{m-1}; l \rightarrow r]} t'_1 \rightarrow_{[q_0, g_0 \rightarrow d_0]} t'_2 \rightarrow \ldots \rightarrow_{[q_{n-1}, g_{n-1} \rightarrow d_{n-1}]} t'_{n+1}$$

et on a $t'_{n+1} = t_{n+1}$.

Preuve: Puisque R est linéaire à droite, le résultat précédent permet de commuter une dérivation de deux étapes sans changer les termes de ses extrémités. Il suffit donc d'appliquer ce résultat autant de fois qu'il convient. □

Notation: $q, v_1, \ldots, v_n$ étant des occurrences, $r_1, \ldots, r_n$ étant des règles de réécriture, on note la dérivation $\rightarrow_{[q.v_1, r_1, \sigma_1]} \rightarrow \ldots \rightarrow_{[q.v_n, r_n, \sigma_n]}$ sous forme abrégée par

$$\xrightarrow{*}_{[q.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]}$$

On note la dérivation $\rightarrow_{[p_0.v_1, r_1, \sigma_1]} \rightarrow \ldots \rightarrow_{[p_k.v_1, r_1, \sigma_1]}$ sous forme abrégée par

$$\xrightarrow{*}_{[(p_0, \ldots, p_k).v_1, r_1, \sigma_1]}$$

Lemme 4.3 Si les étapes $s \xrightarrow{*} t$ (1) se font par des règles linéaires à droite, et s'il existe une dérivation de la forme $t \xrightarrow{*}_{[q.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]} u$ (2) telle que l'occurrence q de t admet à travers (1) tous ses antécédents $(p_0, \ldots, p_k)$ dans s,
alors (2) peut être commutée à travers (1), et on obtient:

$$s \xrightarrow{*}_{[p_0.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]} \cdots \xrightarrow{*}_{[p_k.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]} t' \xrightarrow{*} t$$

Preuve: a) Puisque q admet tous ces antécédents dans s, alors pour $1 \leq i \leq n$, $q.v_i$ aussi, et ce sont $p_0.v_i, \ldots, p_k.v_i$. D'après le lemme précédent et par récurrence il est alors facile de montrer que (2) commute à travers (1), et on obtient:

$$s \xrightarrow{*}_{[(p_0, \ldots, p_k).v_1, r_1, \sigma_1]} \cdots \xrightarrow{*}_{[(p_0, \ldots, p_k).v_n, r_n, \sigma_n]} t' \xrightarrow{*} t$$

b) $p_0, \ldots, p_k$ étant les antécédents d'une même occurrence, ils sont incomparables deux à deux. Par commutation on peut regrouper les occurrences qui commencent par p_0, puis celles qui commencent par p_1, etc..., ce qui donne:

$$s \xrightarrow{*}_{[p_0.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]} \xrightarrow{*} \cdots \xrightarrow{*}_{[p_k.(v_1, \ldots, v_n); r_1, \ldots, r_n; \sigma_1, \ldots, \sigma_n]} t' \xrightarrow{*} t$$

□

Pour que la réécriture équationnelle commute, la méthode de filtrage $Filtre_E$ ne saurait être quelconque.

Définition 4.7 La méthode de filtrage $Filtre_E$ est dite **stable** si la donnée de

$$t_1 \xrightarrow{*}{}^{E_R} t_n = \sigma(g)$$

$$t'_1 \xrightarrow{*}{}^{E_R} t'_n = \sigma'(g)$$

tel que les étapes de E-égalité $t_1 \xrightarrow{*}{}^{E_R} t_n$ et $t'_1 \xrightarrow{*}{}^{E_R} t'_n$ se font aux mêmes occurrences et par les mêmes règles,
implique

$$\sigma \in Filtre_E(g, t_1) \Longleftrightarrow \sigma' \in Filtre_E(g, t'_1)$$

◇

Dans le cas de la R,E-réécriture, $Filtre_E(t, t')$ est l'ensemble des E-filtres de t vers t', donc $Filtre_E$ est stable.

Théorème 4.4 *(commutation de la réécriture équationnelle)*
Supposons que $Filtre_E$ est stable. Soient s, t, u trois termes et

$$s \rightarrow^{RE}_{[p, g \rightarrow d]} t \rightarrow^{RE}_{[q, l \rightarrow r]} u \quad (1)$$

deux étapes de réécriture équationnelle telles que:

- *l'étape $s \rightarrow^{RE} t$ n'utilise que des règles de $E_R \cup R$ linéaires à droite,*
- *$q \in O(t)$ admet tout ses antécédents dans s, notés $p_0, \ldots, p_{m-1}$ (liste exhaustive).*

Alors (1) peut être commuté en

$$s \rightarrow^{RE}_{[p_0, l \rightarrow r]} t'_1 \rightarrow^{RE} \ldots \rightarrow^{RE}_{[p_{m-1}, l \rightarrow r]} t'_m \rightarrow^{RE}_{[p, g \rightarrow d]} u \quad (2)$$

Preuve: Les deux étapes de réécriture sont équivalentes à

$$s \xrightarrow[[p'_0,\ldots,p'_i]]{*_{E_\mathcal{R}}} s_1 \xrightarrow[[p,g\to d,\sigma]]{}_R t \xrightarrow[[q'_0,\ldots,q'_j;r_0,\ldots,r_j,\theta_0,\ldots,\theta_j]]{*_{E_\mathcal{R}}} t_1 \xrightarrow[[q,l\to r,\theta]]{}_R u$$

et par définition de la réécriture équationnelle les occurrences $q'_0,\ldots,q'_j$ sont plus grandes que q. Posons donc $q'_0 = q.v_0,\ldots,q'_j = q.v_j$. Puisque q admet tous ses antécédents $p_0,\ldots,p_{m-1}$ dans s, d'après le lemme précédent il résulte:

$$s \xrightarrow[[p_0.(v_0,\ldots,v_j,\epsilon);r_0,\ldots,r_j,l\to r;\theta_0,\ldots,\theta_j,\theta]]{*} \cdots \xrightarrow{*}_{[p_{m-1}.(v_0,\ldots,v_j,\epsilon);r_0,\ldots,r_j,l\to r;\theta_0,\ldots,\theta_j,\theta]} t'$$

$$t' \xrightarrow[[p'_0,\ldots,p'_i]]{*_{E_\mathcal{R}}} \xrightarrow[[p,g\to d,\sigma']]{}_R u$$

c'est à dire

$$s \xrightarrow[[p_0,l\to r,\theta]]{}_{RE} \to^{RE} \cdots \to^{RE}_{[p_{m-1},l\to r,\theta]} t' \xrightarrow[[p'_0,\ldots,p'_i]]{*_{E_\mathcal{R}}} t'_1 \xrightarrow[[p,g\to d,\sigma']]{}_R u$$

On avait $s|p \xrightarrow{*_{E_\mathcal{R}}} s_1|p = \sigma(g)$, et maintenant $t'|p \xrightarrow{*_{E_\mathcal{R}}} t'_1|p = \sigma'(g)$ aux mêmes occurrences et par les mêmes équations. Par stabilité de $Filtre_E$, il vient que $\sigma' \in Filtre_E(g, t'_1|p)$. Par conséquent $t' \xrightarrow[[p,g\to d,\sigma']]{}_{RE} u$, ce qui prouve le théorème. □

4.3 Lien entre l'unification des termes et l'unification des substitutions

Pour généraliser les résultats précédents à la surréduction, il faut d'abord établir des liens entre l'unification sur les termes et l'unification sur les substitutions.

Définissons l'unification de deux substitutions.

Définition 4.8 On appelle **unificateur** de deux substitutions σ et θ toute substitution μ telle que $\mu.\sigma = \mu.\theta$. On note $EU(\sigma,\theta)$ l'ensemble des unificateurs de σ et θ. On définit l'ensemble minimal complet d'unificateurs de σ et θ (noté $ECMU(\sigma,\theta)$) de la même manière que pour les termes, et on prouve son unicité en étendant le lemme 1.3. ◇

Remarque: On a alors

$$\mu \in ECMU(\sigma,\theta) \Longrightarrow D(\mu) \subseteq (D(\sigma) \cup D(\theta))$$

Notations: OP étant une opération sur les termes (respectivement sur les substitutions), on l'étend de manière naturelle à des ensembles $T_1,\ldots,T_n$ de termes (respectivement de substitutions) par:

$$OP(T_1,\ldots,T_n) = \{OP(t_1,\ldots,t_n), t_1 \in T_1,\ldots,t_n \in T_n\}$$

Par ailleurs si Σ est un ensemble de substitutions et t un terme on définit:

$$\Sigma(t) = \{\sigma(t), \sigma \in \Sigma\}$$

De plus l'élément et son singleton seront confondus, par exemple on écrira θ à la place de $\{\theta\}$.
En pratique nous utiliseront ce formalisme sur la composée des substitutions, sur EU, $ECMU$ et UF, Uf définis plus loin.

Lemme 4.5 Soit Σ un ensemble de substitution, Σ_{min} l'ensemble complet minimal de Σ, et θ une substitution. L'ensemble complet minimal de $\Sigma.\theta$ est $\Sigma_{min}.\theta$.

Preuve: C'est l'ensemble des substitutions minimales de $\Sigma.\theta$. Si $\sigma = \beta.\theta, \beta \in \Sigma$ est minimale, alors β est minimale dans Σ. Réciproquement si $\beta \in \Sigma_{min}$, alors $\beta.\theta$ est minimale dans $\Sigma.\theta$. □

Les lemmes 4.6 et 4.8 ont été établis par Herold [30, p 42] pour l'unification non sortée. Nous les prouvons dans le cadre ordo-sorté en détaillant bien les conditions techniques sur les variables.

Lemme 4.6 Soient σ et θ deux substitutions idempotentes telles que $I(\sigma) \cap I(\theta) = \emptyset$. S'il existe deux substitutions γ et μ telles que $\gamma.\sigma = \mu.\theta$, alors σ et θ sont unifiables par une substitution α vérifiant $\alpha.\sigma = \gamma.\sigma$ et $\alpha.\theta = \mu.\theta$.

Preuve: Définissons α comme le recollement de γ et μ. Soit x une variable, discutons quatre cas.

1. Si $x \notin D(\sigma)$ et $x \notin D(\theta)$, posons $\alpha(x) = \gamma(x) = \mu(x)$.
2. Si $x \in D(\sigma)$ et $x \notin D(\theta)$, alors $\gamma.\sigma(x) = \mu(x)$. Puisque σ est idempotente on a $x \notin V(\sigma(x))$. Il est donc possible de poser $\alpha(x) = \mu(x)$ et $\forall y \in V(\sigma(x)), \alpha(y) = \gamma(y)$.
3. Si $x \notin D(\sigma)$ et $x \in D(\theta)$. Même raisonnement que ci-dessus en échangeant les lettres.
4. Si $x \in D(\sigma) \cap D(\theta)$, on a $\gamma.\sigma(x) = \mu.\theta(x)$. Puisque $I(\sigma) \cap I(\theta) = \emptyset$, on a $V(\sigma(x)) \cap V(\theta(x)) = \emptyset$. Il est donc possible de poser $\forall y \in V(\sigma(x)), \alpha(y) = \gamma(y)$ et $\forall y \in V(\theta(x)), \alpha(y) = \mu(y)$.

α a été défini tel que $\alpha.\sigma = \gamma.\sigma$ et $\alpha.\theta = \mu.\theta$, donc d'après l'hypothèse $\alpha.\sigma = \alpha.\theta$. □

L'unification ordo-sortée n'étant pas unitaire, nous sommes obligés d'introduire des notations ensemblistes.

Définition 4.9 On appelle ensemble des **unifiés** de deux substitutions σ et θ l'ensemble $UF(\sigma,\theta) = \{\alpha.\sigma, \alpha \in EU(\sigma,\theta)\}$. On appelle ensemble complet des unifiés de σ et θ l'ensemble $Uf(\sigma,\theta) = \{\alpha.\sigma, \alpha \in ECMU(\sigma,\theta)\}$. ◇

Remarques:

1. D'après le lemme 4.5, $Uf(\sigma,\theta)$ est exactement l'ensemble minimal complet de $UF(\sigma,\theta)$.
2. UF est commutatif et par unicité Uf aussi.
3. Quand on dit "soit $\sigma \in Uf(\theta,\mu)$", on suppose implicitement que θ et μ sont unifiables.

Lemme 4.7 Lorsque les substitutions considérées ont des images disjointes alors UF est associatif, et donc Uf aussi.

Preuve: Soit $\mu \in UF(\sigma_1, UF(\sigma_2, \sigma_3))$. Il existe des substitutions θ, γ telles que:

$$\theta.\sigma_2 = \theta.\sigma_3 \in UF(\sigma_2, \sigma_3)$$

$$\mu = \gamma.\sigma_1 = \gamma.\theta.\sigma_2 = \gamma.\theta.\sigma_3$$

Puisque $I(\sigma_1) \cap I(\sigma_2) = \emptyset$ et d'après le lemme 4.6 il existe θ' telle que

$$\mu = \theta'.\sigma_1 = \theta'.\sigma_2 \in UF(\sigma_1, \sigma_2)$$

D'où $\mu = (\theta'.\sigma_1) = (\gamma.\theta).\sigma_3$. Donc $\mu \in UF(UF(\sigma_1, \sigma_2), \sigma_3)$.
L'autre inclusion se prouve de la même manière. □

Notations: Sous les hypothèses du lemme précédent nous utiliserons les notations aplaties: $UF(\sigma_1, \ldots, \sigma_n)$ et $Uf(\sigma_1, \ldots, \sigma_n)$.

Le lemme suivant, relie l'unification sur les termes et l'unification sur les substitutions. L'inclusion $Uf(ECMU(s,t),\theta) \subseteq ECMU(\theta(s),\theta(t)).\theta$ est trivialement vraie si on utilise EU et UF, mais nous montrons qu'elle est encore vraie pour les ensembles complets minimaux. L'inclusion réciproque $Uf(ECMU(s,t),\theta) \supseteq ECMU(\theta(s),\theta(t)).\theta$ peut se comprendre ainsi: si $\theta(s)$ et $\theta(t)$ sont unifiables par l'unificateur minimal μ, alors $\mu.\theta$ est un unificateur de s et t. Il existe donc $\sigma \in ECMU(s,t)$, et une substitution γ tel que $\gamma.\sigma = \mu.\theta$. Alors σ et θ sont unifiables par $\gamma \cup \mu$ qui se trouve être un unificateur minimal.

Lemme 4.8 Soient s, t deux termes et θ une substitution. On a:

$$Uf(ECMU(s,t),\theta) = ECMU(\theta(s),\theta(t)).\theta$$

Preuve: a) Montrons que $Uf(ECMU(s,t),\theta) \subseteq ECMU(\theta(s),\theta(t)).\theta$. Soit $\sigma \in ECMU(s,t)$, soit $\beta \in UF(\sigma,\theta)$. β s'écrit $\beta = \mu.\sigma = \mu.\theta$, avec $\mu \in EU(\sigma,\theta)$. On a

$$\mu.\theta(s) = \mu.\sigma(s) = \mu.\sigma(t) = \mu.\theta(t)$$

Par conséquent $\mu \in EU(\theta(s),\theta(t))$, et alors $\beta \in EU(\theta(s),\theta(t)).\theta$.

b) Montrons que $Uf(ECMU(s,t),\theta) \supseteq ECMU(\theta(s),\theta(t)).\theta$. Soit $\beta \in EU(\theta(s),\theta(t)).\theta$, elle s'écrit $\beta = \mu.\theta$ avec $\mu \in EU(\theta(s),\theta(t))$. Donc $\beta = \mu.\theta \in EU(s,t)$. Par complétude de $ECMU$ il existe $\sigma \in ECMU(s,t)$ telle que $\sigma \leq \mu.\theta$. Il existe donc une substitution γ telle que $\gamma.\sigma = \mu.\theta$. Puisque $ECMU$ est définie à un renommage de variables près, on peut supposer que σ satisfait $I(\sigma) \cap I(\theta) = \emptyset$. D'après le lemme 4.6, σ et θ sont unifiables par une substitution α telle que $\mu.\theta = \alpha.\theta$. Par définition $\alpha.\theta \in UF(\sigma,\theta)$, donc $\beta = \mu.\theta = \alpha.\theta \in UF(\sigma,\theta) \subseteq UF(ECMU(s,t),\theta)$.

c) D'aprés le lemme 1.3 les ensembles minimaux de $UF(ECMU(s,t),\theta)$ et de $EU(\theta(s),\theta(t)).\theta$ sont égaux, et grâce au lemme 4.5 on obtient $Uf(ECMU(s,t),\theta) = ECMU(\theta(s),\theta(t)).\theta$.
□

4.4 Commutation de la surréduction

Quand on commute deux étapes de surréduction, la difficulté essentielle est de montrer que la composée des substitutions surréductrices est inchangée. Pour y parvenir l'idée est la suivante: des étapes de surréduction données on déduit une propriété sur des termes (tel sous-terme est unifiable par telle substitution minimale), puis par le lemme 4.8 on en déduit une propriété sur les substitutions surréductrices. Ensuite on applique de nouveau le lemme 4.8 mais dans l'autre sens, dont on déduit une propriété sur des termes, qui fournit des étapes de surréductions dans l'ordre inverse de celles de départ.

Lemme 4.9 Soit s un terme et des occurrences $p_0, \ldots, p_{m-1} \in O(s)$ non comparables deux à deux. Soit une règle de réécriture $l \to r$. S'il existe une substitution

$$\sigma \in Uf(ECMU(s|p_{m-1}, l), \ldots, ECMU(s|p_0, l))$$

alors il existe une dérivation de la forme

$$s \leadsto_{[p_0, l \to r, \theta'_0]} t'_1 \leadsto \ldots \leadsto_{[p_{m-1}, l \to r, \theta'_{m-1}]} t'_m$$

telle que $\sigma = \theta'_{m-1} \ldots \theta'_0$.

Preuve: Par récurrence sur le nombre i d'antécédents.
Pour $i = 1$ la propriété est vraie par définition de la surréduction.

Supposons la propriété vraie pour $i \in \{1, \ldots, m-1\}$. Soit

$$\sigma \in Uf(ECMU(s|p_i, l), \ldots, ECMU(s|p_0, l))$$

Par associativité,

$$\sigma \in Uf(ECMU(s|p_i, l), Uf(ECMU(s|p_{i-1}, l), \ldots, ECMU(s|p_0, l)))$$

Par hypothèse de récurrence il existe une dérivation

$$s \leadsto_{[p_0, l \to r, \theta'_0]} \leadsto \ldots \leadsto_{[p_{i-1}, l \to r, \theta'_{i-1}]} t'_i \quad (1)$$

telle que $\sigma \in Uf(ECMU(s|p_i, l), \theta'_{i-1} \ldots \theta'_0)$. Posons $\theta = \theta'_{i-1} \ldots \theta'_0$. En appliquant le lemme 4.8 sur les termes $s|p_i, l$ et la substitution θ, on obtient $\sigma \in ECMU(\theta(s|p_i), \theta(l)).\theta$. On peut supposer que les variables de la règle $l \to r$ sont renommées en de nouvelles variables non utilisées dans la dérivation (1), d'où $\theta(l) = l$. Il existe $\theta'_i \in ECMU(\theta(s|p_i), l)$ telle que $\sigma = \theta'_i.\theta$. Puisque $p_0, \ldots, p_i$ ne sont pas comparables on a $t'_i|p_i = \theta(s|p_i)$, donc $\theta'_i \in ECMU(t'_i|p_i, l)$. Comme de plus $p_i \in O(s)$, t'_i se surréduit par l'étape $t'_i \leadsto_{[p_i, l \to r, \theta'_i]} t'_{i+1}$. □

Remarque: Si $Uf(\theta_1, \sigma) = \ldots = Uf(\theta_n, \sigma)$ alors pour tout $i \in \{1, \ldots, n\}$,

$$Uf(\theta_1, \ldots, \theta_n, \sigma) = Uf(\theta_i, \sigma)$$

Lemme 4.10 Soient

$$s \leadsto_{[p, g \to d, \sigma]} t \leadsto_{[q, l \to r, \theta]} u$$

deux étapes de surréduction issues de s. Si $p_0, \ldots, p_{m-1} \in O(s)$ sont des antécédent de q, alors il existe une dérivation

$$s \leadsto_{[p_0, l \to r, \theta'_0]} t'_1 \leadsto \ldots \leadsto_{[p_{m-1}, l \to r, \theta'_{m-1}]} t'_m$$

et une substitution $\sigma' \in ECMU(\theta'_{m-1} \ldots \theta'_0(s|p), g)$ telles que

$$\sigma'.\theta'_{m-1} \ldots \theta'_0 = \theta.\sigma$$

Preuve: a) Soit $i \in \{0, \ldots, m-1\}$, d'après le lemme 4.8 appliqué sur les termes $s|p_i, l$ et la substitution σ, on a

$$Uf(ECMU(s|p_i, l), \sigma) = ECMU(\sigma(s|p_i), \sigma(l)).\sigma$$

Or d'après les hypothèses, $t|q = \sigma(s|p_i)$ et $l = \sigma(l)$, d'où

$$Uf(ECMU(s|p_i, l), \sigma) = ECMU(t|q, l).\sigma$$

Ceci est vrai pour tout les i, et d'après la remarque précédente:

$$Uf(ECMU(s|p_0, l), \ldots, ECMU(s|p_{m-1}, l), \sigma) = ECMU(t|q, l).\sigma$$

b) Par hypothèse $\theta \in ECMU(t|q, l)$. L'égalité précédente appliquée de droite à gauche implique

$$\theta.\sigma \in Uf(ECMU(s|p_0, l), \ldots, ECMU(s|p_{m-1}, l), \sigma)$$

Donc il existe $\theta'_0 \in ECMU(s|p_0, l), \ldots \theta'_{m-1} \in ECMU(s|p_{m-1}, l)$ telles que

$$\theta.\sigma \in Uf(\theta'_0, \ldots, \theta'_{m-1}, \sigma)$$

Par associativité de Uf on peut écrite

$$\theta.\sigma \in Uf(Uf(\theta'_0, \ldots, \theta'_{m-1}), \sigma)$$

Il existe donc $\gamma \in Uf(\theta'_0, \ldots, \theta'_{m-1})$ telle que $\theta.\sigma \in Uf(\gamma, \sigma)$. D'après le lemme 4.8 appliqué sur les termes $s|p, g$ et sur la substitution γ il vient

$$Uf(ECMU(s|p, g), \gamma) = ECMU(\gamma(s|p), \gamma(g)).\gamma$$

Par hypothèse $\sigma \in ECMU(s|p, g)$, et puisque Uf est commutatif on a

$$\theta.\sigma \in ECMU(\gamma(s|p), \gamma(g)).\gamma$$

Puisque $ECMU$ est minimal $\gamma(g) = g$. Il existe donc $\sigma' \in ECMU(\gamma(s|p), g)$ telle que $\theta.\sigma = \sigma'.\gamma$. D'après le lemme 4.9 il existe une dérivation

$$s \twoheadrightarrow_{[p_0, l \to r, \theta'_0]} t'_1 \twoheadrightarrow \ldots \twoheadrightarrow_{[p_{m-1}, l \to r, \theta'_{m-1}]} t'_m$$

telle que $\gamma = \theta'_{m-1} \ldots \theta'_0$. □

La propriété de commutation dans le cas où les occurrences ne sont pas comparables se déduit très facilement du lemme précédent. Elle a été établie par Herold [29] dans le cadre non sorté.

Proposition 4.11 *Soient*

$$s \twoheadrightarrow_{[p, g \to d, \sigma]} t \twoheadrightarrow_{[q, l \to r, \theta]} u \quad (1)$$

deux étapes de surréduction issues de s telles que:

1. $p, q \in O(s)$
2. *p et q ne sont pas comparables.*

Alors (1) peut être commuté en:

$$s \twoheadrightarrow_{[q, l \to r, \theta']} t' \twoheadrightarrow_{[p, g \to d, \sigma']} u \quad (2)$$

tel que $\sigma'.\theta' = \theta.\sigma$.

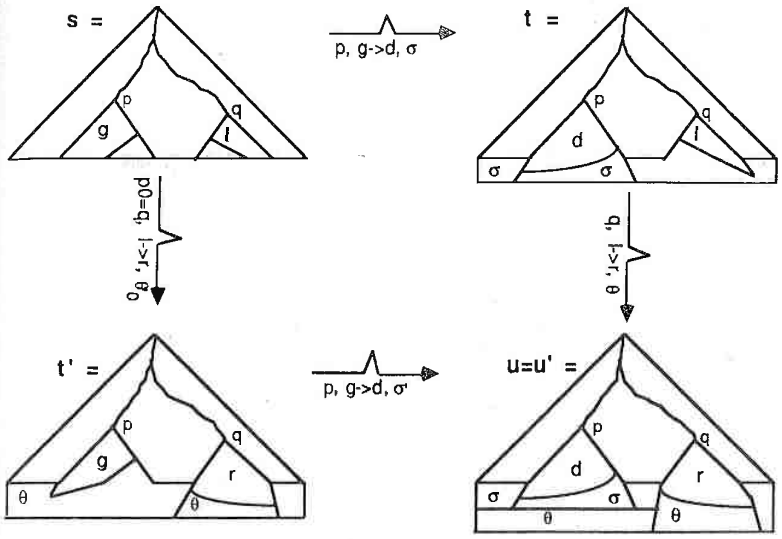


Figure 4.1: Commutation de la surréduction: cas où les occurrences ne sont pas comparables

Cette proposition est illustrée par la figure 4.1.

Preuve: L'occurrence q de s est un antécédent de l'occurrence q de t. D'après le lemme 4.10 il existe une surréduction $s \twoheadrightarrow_{[q, l \to r, \theta']} t'$ et $\sigma' \in ECMU(\theta'(s|p), g)$ telles que $\sigma'.\theta' = \theta.\sigma$. Puisque $t'|p = \theta'(s|p)$ on a $t' \twoheadrightarrow_{[p, g \to d, \sigma']} u'$. En considérant les réécritures associées $\theta.\sigma(s) \to_{[p, g \to d]} \to_{[q, l \to r]} u$ et $\sigma'.\theta'(s) \to_{[q, l \to r]} \to_{[p, g \to d]} u'$, il est évident que $u = u'$. □

Contrairement au cas précédent, quand les occurrences p et q ne sont pas comparables le lemme 4.10 ne permet pas de conclure, car les termes $t'_m|p$ et $\theta'_{m-1} \ldots \theta'_0(s|p)$ ne sont pas égaux, mais sont liés par $\theta'_{m-1} \ldots \theta'_0(s) \to_{[p_0, \ldots p_{m-1}, l \to r]} t'_m$. Il faut donc montrer que cette dérivation laisse inchangés les unificateurs minimaux avec g. C'est le but des deux lemmes suivants.

Notation: On note par V l'ensemble des variables de la signature et par $V - \{x\}$ l'ensemble V dont on a retiré x.

Lemme 4.12 Soient s, g deux termes sans variables communes. Supposons qu'il existe une variable $x \in V(g)$, des occurrences $q_0, \ldots, q_{m-1} \in O(s)$ non comparables entre elles et une occurrence w tel que pour tout $i \in \{0, \ldots, m-1\}$ on ait $q_i = v_i.w$ et $g|_{v_i} = x$. Soit z une nouvelle variable, posons $t = s[q_0 \leftarrow z] \ldots [q_{m-1} \leftarrow z]$. Alors $\sigma \in ECMU(s, g)$ si et seulement si il existe $\gamma_1 \in ECMU(t.g)$, $\gamma_2 \in ECMU(\gamma_1(s|q_0), \ldots, \gamma_1(s|q_{m-1}))$ tel que $\sigma = \gamma_2.\gamma_1\ [V - \{x\}]$.

Preuve: a) Soit $\sigma \in EU(s,g)$. Décomposons σ en

$$\begin{array}{rll} \gamma_1(y) = & \sigma(y) & \text{si } y \neq x \\ = & \sigma(x)[w \leftarrow z] & \text{si } y = x \end{array}$$

et $\gamma_2 = (z/\sigma(x)|w)$. Par construction $\sigma = \gamma_2.\gamma_1$.

$$\gamma_1(t) = \gamma_1(s[q_0 \leftarrow z]\dots[q_{m-1} \leftarrow z]) = \gamma_1(s)[q_0 \leftarrow z]\dots[q_{m-1} \leftarrow z]$$

et puisque $\gamma_1(s) = \sigma(s) = \sigma(g)$,

$$\gamma_1(t) = \sigma(g)[q_0 \leftarrow z]\dots[q_{m-1} \leftarrow z] = \gamma_1(g)$$

Donc $\gamma_1 \in EU(t,g)$. Soit $i,j \in \{0,\dots,m-1\}$,

$$\gamma_2.\gamma_1(s|q_i) = \sigma(s|q_i) = \sigma(x|w) = \sigma(s|q_j) = \gamma_2.\gamma_1(s|q_j)$$

Donc $\gamma_2 \in EU(\gamma_1(s|q_0),\dots,\gamma_1(s|q_{m-1}))$.

b) Soit $\gamma_1 \in EU(t,g)$, $\gamma_2 \in EU(\gamma_1(s|q_0),\dots,\gamma_1(s|q_{m-1}))$. Posons

$$\begin{array}{rll} \sigma(y) = & \gamma_2.\gamma_1(y) & \text{si } y \neq x \\ = & \gamma_2.\gamma_1(x)[w \leftarrow \gamma_2.\gamma_1(s|q_0)] & \text{si } y = x \end{array}$$

On voit que $\sigma = \gamma_2.\gamma_1 \ [V - \{x\}]$.
Puisque les apparitions de x dans g sont aux occurrences $v_0,\dots,v_{m-1}$ et que $q_0 = v_0.w,\dots,q_{m-1} = v_{m-1}.w$ on a

$$\sigma(g) = \gamma_2.\gamma_1(g)[q_0 \leftarrow \gamma_2.\gamma_1(s|q_0)]\dots[q_{m-1} \leftarrow \gamma_2.\gamma_1(s|q_0)]$$

Par définition de γ_1

$$\sigma(g) = \gamma_2.\gamma_1(t)[q_0 \leftarrow \gamma_2.\gamma_1(s|q_0)]\dots[q_{m-1} \leftarrow \gamma_2.\gamma_1(s|q_0)]$$

Excepté pour les sous-termes aux occurrences $q_0,\dots,q_{m-1}$, les termes s et t sont égaux, d'où:

$$\sigma(g) = \gamma_2.\gamma_1(s)[q_0 \leftarrow \gamma_2.\gamma_1(s|q_0)]\dots[q_{m-1} \leftarrow \gamma_2.\gamma_1(s|q_0)]$$

Par définition de γ_2:

$$\sigma(g) = \gamma_2.\gamma_1(s)[q_0 \leftarrow \gamma_2.\gamma_1(s|q_0)]\dots[q_{m-1} \leftarrow \gamma_2.\gamma_1(s|q_{m-1})] = \gamma_2.\gamma_1(s)$$

Puisque $x \in V(g)$ et que les variables de s et g sont disjointes alors $\gamma_2.\gamma_1 = \sigma \ [V(s)]$.
Par conséquent $\sigma(g) = \sigma(s)$, c'est à dire $\sigma \in EU(s,g)$.

c) Le résultat s'obtient grâce à la propriété d'unicité des ensembles minimaux (lemme 1.3). □

Définition 4.10 Une règle de réécriture $l \to r$ est dite **régulière** si $V(l) = V(r)$. ◇

Lemme 4.13 Soient s' un terme, $g \to d$ et $l \to r$ deux règles. Si

1. g est linéaire ou bien $l \to r$ est une règle régulière,
2. $s' \xrightarrow{*}_{[p_0,\dots,p_{m-1},l\to r]} t'_m$ et il existe une variable $x \in V(g)$ et des occurrences p, w tel que pour tout $i \in \{0,\dots,m-1\}$ on a $p_i = p.v_i.w$ et $g|v_i = x$,

alors

$$ECMU(s'|p,g) \subseteq ECMU(t'_m|p,g) \ [V - \{x\}]$$

Preuve: s' et t'_m s'écrivent:

$$s' = t[p_0 \leftarrow \sigma_0(l)]\dots[p_{m-1} \leftarrow \sigma_{m-1}(l)], \ \ t'_m = t[p_0 \leftarrow \sigma_0(r)]\dots[p_{m-1} \leftarrow \sigma_{m-1}(r)]$$

Puisque $g \to d$ est une règle on peut supposer que s' et g n'ont pas de variable commune, donc t'_m et g non plus. Soit $\theta \in ECMU(s'|p,g)$, d'après le lemme précédent appliqué sur $s'|p,g$ et $t|p$ il existe $\gamma_1 \in ECMU(t|p,g)$, $\gamma_2 \in ECMU(\gamma_1(\sigma_0(l)),\dots,\gamma_1(\sigma_{m-1}(l)))$ tel que $\theta = \gamma_2.\gamma_1 \ [V - \{x\}]$.
Si g est linéaire, $ECMU(\gamma_1(\sigma_0(l))) = ECMU(\gamma_1(\sigma_0(r))) = \{Id\}$. u étant un terme quelconque, on a

$$\sigma \in ECMU(\gamma_1(\sigma_0(u)),\dots,\gamma_1(\sigma_{m-1}(u))) \Longleftrightarrow \forall y \in V(u), \sigma \in ECMU(\gamma_1(\sigma_0(y)),\dots,\gamma_1(\sigma_{m-1}(y)))$$

Donc si $V(r) = V(l)$ alors

$$ECMU(\gamma_1(\sigma_0(l)),\dots,\gamma_1(\sigma_{m-1}(l))) = ECMU(\gamma_1(\sigma_0(r)),\dots,\gamma_1(\sigma_{m-1}(r)))$$

Finalement $\gamma_2 \in ECMU(\gamma_1(\sigma_0(r)),\dots,\gamma_1(\sigma_{m-1}(r)))$. D'après le lemme précédent appliqué aux termes $t'_m|p,g$ et $t|p$, il existe $\theta' \in ECMU(t'_m|p,g)$ telle que $\theta' = \gamma_2.\gamma_1 \ [V - \{x\}]$. D'où $\theta' = \theta \ [V - \{x\}]$. □

Donnons une première version de la propriété générale de commutation. Des exemples sont donnés au paragraphe suivant.

Proposition 4.14 *Soient*

$$s \xrightarrow[[p,g\to d,\sigma]]{} t \xrightarrow[[q,l\to r,\theta]]{} u \quad (1)$$

deux étapes de surréduction issues de s telles que

1. *A travers l'étape de réécriture $\sigma(t) \to_{[q,g\to d]} t$, q admets des antécédents dans $\sigma(s)$, dont la liste exhaustive est $p_0,\dots,p_{m-1}$, et de plus $p_0,\dots,p_{m-1} \in O(s)$,*
2. *$g \to d$ est linéaire à gauche ou $l \to r$ est régulière.*

Alors (1) peut être commuté en

$$s \xrightarrow[[p_0,l\to r,\theta'_0]]{} t'_1 \xrightarrow{} \dots \xrightarrow[[p_{m-1},l\to r,\theta'_{m-1}]]{} t'_m \xrightarrow[[p,g\to d,\sigma']]{} u' \quad (2)$$

tel que

- $\sigma'.\theta'_{m-1}\dots\theta'_0 = \theta.\sigma \ [V - V(d)]$
- $u \xrightarrow{*}_{[q_1,\dots,q_n,l\to r]} u'$ *où $q, q_1,\dots,q_n$ sont les résidus de $p_0,\dots,p_{m-1}$ dans t.*

Preuve: Si p et q ne sont pas comparables, voir la proposition 4.11. Sinon il existe une variable $x \in V(g)$ telle que $q = p.v'.w$ et $p_0 = p.v_0.w,\dots,p_{m-1} = p.v_{m-1}.w$ où v' est une occurrence de x dans d et $v_0,\dots,v_{m-1}$ sont les occurrences de x dans g.

D'après le lemme 4.10 il existe une dérivation

$$s \leadsto_{[p_0, l \to r, \theta'_0]} t'_1 \leadsto \ldots \leadsto_{[p_{m-1}, l \to r, \theta'_{m-1}]} t'_m$$

et une substitution $\sigma'' \in Unif(\theta'_{m-1} \ldots \theta'_0(s|p), g)$ telles que

$$\sigma''.\theta'_{m-1} \ldots \theta'_0 = \theta.\sigma$$

Ainsi $\theta'_{m-1} \ldots \theta'_0(s) \xrightarrow{*}_{[p_0, \ldots, p_{m-1}, l \to r]} t'_m$. D'après le lemme précédent, il existe $\sigma' \in Unif(t'_m|p, g)$ telle que $\sigma' = \sigma'' \ [-\{x\}]$. Puisque $x \in V(d)$ on peut écrire

$$\sigma'.\theta'_{m-1} \ldots \theta'_0 = \theta.\sigma \ [-V(d)]$$

D'autre part on peut donc surréduire t'_m par l'étape $t'_m \leadsto_{[p, g \to d, \sigma']} u'$. En considérant les dérivations associées $\theta.\sigma(s) \to_{[p, g \to d]} \to_{[q, l \to r]} u$ et $\sigma'.\theta'_{m-1} \ldots \theta'_0(s) \to_{[p_0, l \to r]} \ldots \to_{[p_{m-1}, l \to r]} \to_{[p, g \to d]} u'$, il est clair que $u \xrightarrow{*}_{[q_1, \ldots, q_n, l \to r]} u'$. □

Remarques:

- Si d est linéaire, ou u est normalisé, ou p et q ne sont pas comparables, alors $u' = u$.
- Lorsque p et q ne sont pas comparables il n'est pas nécessaire que la condition 3. soit satisfaite.

La condition 1. est technique, cherchons une condition suffisante qui le soit moins. Supposons qu'il existe un antécédent p_i de $\sigma(s)$ tel que $p_i \notin O(s)$. Par définition $\sigma(s|p_i) = t|q$. Par conséquent

$$\theta.\sigma(s|p_i) = \theta(t|q) \to_{[l \to r]} u|q$$

Il existe une variable x d'occurrence w dans s telle que $w \leq p_i$. Donc $\theta.\sigma(x)$ n'est pas normalisé, ce qui prouve que $\theta.\sigma$ n'est pas normalisé sur $V(s)$. Par contraposée, $\theta.\sigma$ normalisé sur $V(s)$ implique tous les antécédents de q sont des occurrences de $O(s)$. On peut donc donner une version moins technique de la propriété de commutation.

Théorème 4.15 *Soient*

$$s \leadsto_{[p, g \to d, \sigma]} t \leadsto_{[q, l \to r, \theta]} u \quad (1)$$

deux étapes de surréduction issues de s telles que

1. *q admets des antécédents dans s, notés $p_0, \ldots, p_{m-1}$ (liste exhaustive),*
2. *$\theta.\sigma$ est normalisé sur les variables de s,*
3. *$g \to d$ est linéaire à gauche ou $l \to r$ est régulière.*

Alors (1) peut être commuté en

$$s \leadsto_{[p_0, l \to r, \theta'_0]} t'_1 \leadsto \ldots \leadsto_{[p_{m-1}, l \to r, \theta'_{m-1}]} t'_m \leadsto_{[p, g \to d, \sigma']} u' \quad (2)$$

tel que

- $\sigma'.\theta'_{m-1} \ldots \theta'_0 = \theta.\sigma \ [V(s)]$
- $u \xrightarrow{*}_{[q_1, \ldots, q_n, l \to r]} u'$ *où $q, q_1, \ldots, q_n$ sont les résidus de $p_0, \ldots, p_{m-1}$ dans t.*

Ce théorème est illustré par la figure 4.1 lorsque les occurrences p et q ne sont pas comparables, et par la figure 4.2 lorsqu'elles le sont.

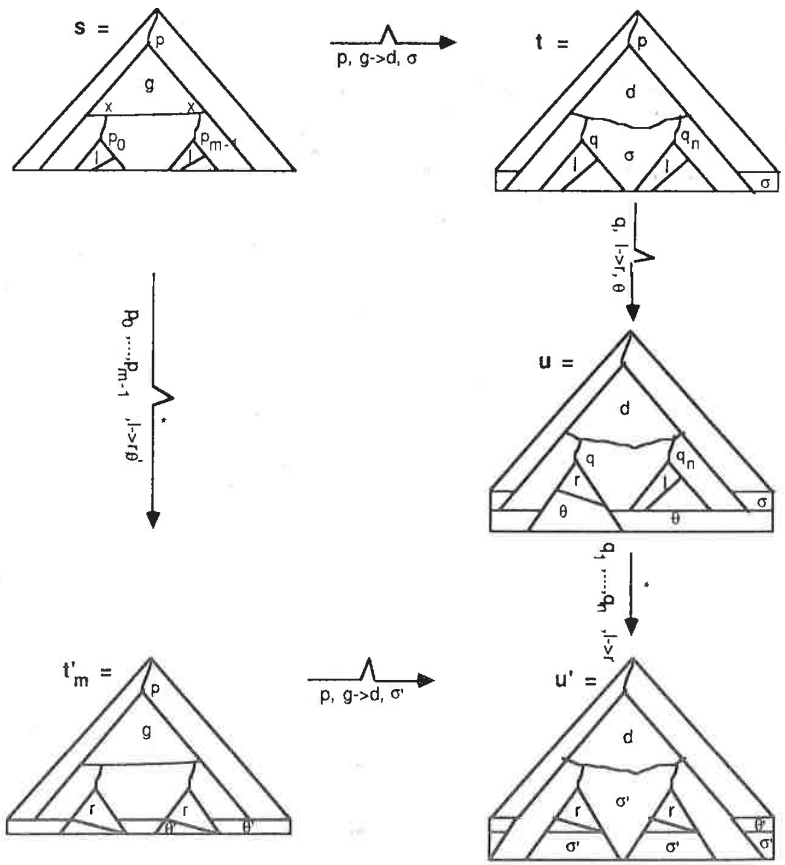
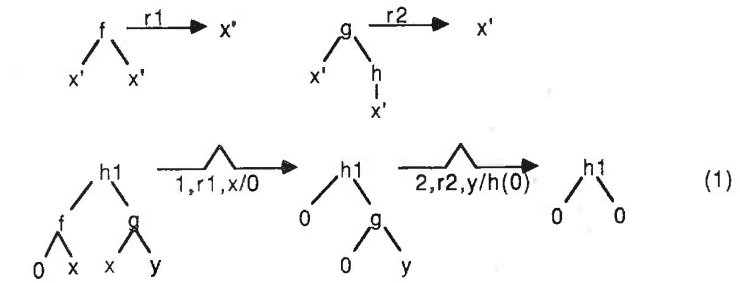


Figure 4.2 : Commutation de la surréduction: cas où les occurrences sont comparables

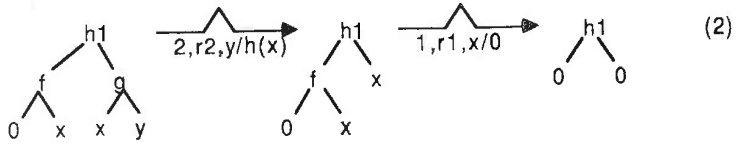
4.5 Exemples

Exemple 4.1 Considérons deux règles de réécriture r_1, r_2 et deux étapes de surréduction (1):



(1)

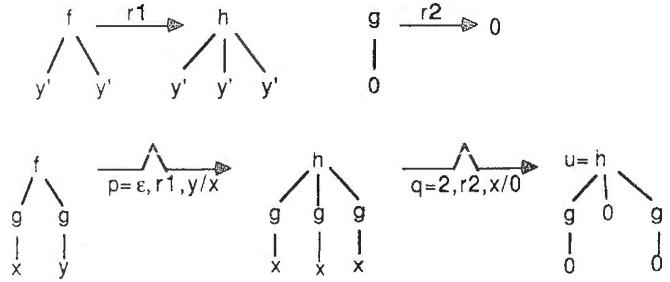
Dans (1), les occurrences d'application des règles ne sont pas comparables, on peut donc commuter (1) en (2):



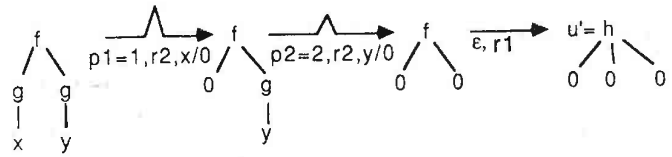
(2)

Remarquons que l'application de r_2 ne crée pas les mêmes substitutions surréductrices dans (1) et dans (2). Par contre la composée des substitutions surréductrices est la même dans (1) et dans (2).

Exemple 4.2 Considérons deux règles de réécriture r_1, r_2 et deux étapes de surréduction:

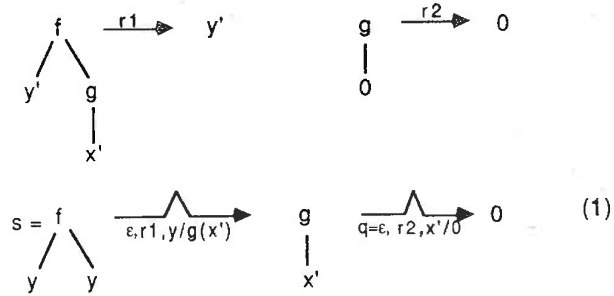


L'occurrence q admet deux antécédents $p_1 = 1$ et $p_2 = 2$. Donc les deux étapes de surréduction commutent en trois étapes:



Les termes finaux u et u' sont différents car p_1 et p_2 admettent pour résidu q, mais aussi $q_1 = 1$ et $q_2 = 3$. Donc u et u' sont liés par $u \rightarrow_{[q_1,r_2]} \rightarrow_{[q_2,r_2]} u'$.

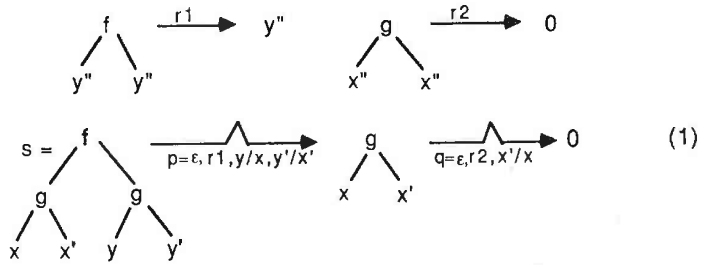
Exemple 4.3 Considérons deux règles de réécriture r_1, r_2 et deux étapes de surréduction (1):



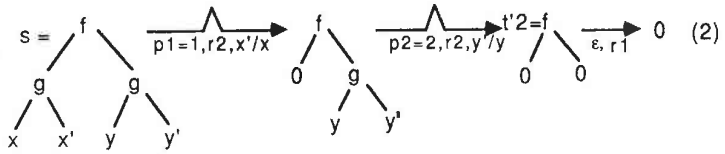
(1)

L'antécédent de $q = \epsilon$ est $p_1 = 1$, qui est une occurrence de variable de s. Puisque les variables ne peuvent pas être surréduites, il est impossible de commuter (1). Remarquons qu'alors la composée des substitutions surréductrices dans (1), qui est égale à $x'/0,\ y/g(0)$, n'est pas normalisée.

Exemple 4.4 Considérons deux règles de réécriture r_1, r_2 et deux étapes de surréduction (1):



Les antécédents de $q = \epsilon$ sont $p_1 = 1$ et $p_2 = 2$. La composée des substitutions surréductrices de (1) est $\theta_1 = y/x, x'/x, y'/x$. En commutant (1) on obtient la dérivation (2) ci-dessous dont la composée des substitutions surréductrices $\theta_2 = x'/x, y'/y$ n'est pas égale à θ_1, mais est plus générale. Cela provient du fait que ni r_1 n'est linéaire à gauche, ni r_2 n'est régulière.



4.6 Commutation de la surréduction équationnelle

Reprenons et généralisons les lemmes qui ont été établis pour la commutation de la réécriture équationnelle.

Lemme 4.16 (commutation à travers plusieurs étapes)
Supposons que le système de réécriture courant R satisfasse les deux conditions suivantes:

- R est linéaire à droite,
- R est linéaire à gauche ou R est régulier.

Soient

$$t_0 \overset{}{\rightsquigarrow}_{[q_0, g_0 \to d_0, \sigma_0]} \cdots \overset{}{\rightsquigarrow}_{[q_{n-1}, g_{n-1} \to d_{n-1}, \sigma_{n-1}]} t_n \quad (1)$$

$$t_n \overset{}{\rightsquigarrow}_{[v, l \to r, \theta]} t_{n+1} \quad (2)$$

deux dérivations de surréductions, et $v \in O(t_n)$. Si dans la dérivation $\sigma_{n-1} \ldots \sigma_0(t_0) \overset{*}{\to} t_n$, v admets tous ses antécédents dans $\sigma_{n-1} \ldots \sigma_0(t_0)$, notés $p_0, \ldots, p_{m-1}$, et que $p_0, \ldots, p_{m-1} \in O(t_0)$, alors (2) peut être commuté à travers (1) en

$$t_0 \overset{*}{\rightsquigarrow}_{[p_0, \ldots, p_{m-1}, l \to r, \theta']} t'_1 \rightsquigarrow_{[q_0, g_0 \to d_0, \sigma'_0]} t'_2 \rightsquigarrow \cdots \rightsquigarrow_{[q_{n-1}, g_{n-1} \to d_{n-1}, \sigma'_{n-1}]} t'_{n+1}$$

tel que

- $\sigma'_{n-1} \ldots \sigma'_0 . \theta' = \theta . \sigma_{n-1} \ldots \sigma_0 \ [V(t_0)]$
- $t'_{n+1} = t_{n+1}$

Preuve: Comme pour le lemme 4.2. Ajoutons que la composée des substitutions surréductrices est inchangée lors de la commutation de deux étapes, sauf sur les variables des règles. Il suffit donc de supposer que les règles ont des variables différentes de celles des termes de la dérivation, pour affirmer que

$$\sigma'_{n-1} \ldots \sigma'_0 . \theta' = \theta . \sigma_{n-1} \ldots \sigma_0 \ [V(t_0)]$$

□

Le lemme précédent n'est plus vrai sans l'hypothèse de linéarité à droite des règles de réécriture. En effet, sans cette hypothèse, on ne peut pas commuter deux étapes à l'intérieur d'une surdérivation, puisqu'alors le terme u résultant de ces deux étapes serait changé en un terme u' tel que $u \overset{*}{\to} u'$ (voir théorème 4.15). Par exemple si on commute les deux premières étapes de la surdérivation

$$s \rightsquigarrow t \rightsquigarrow u \overset{*}{\rightsquigarrow}_{[0_1]} u_n \quad (1)$$

on obtient $s \rightsquigarrow . \rightsquigarrow u'$ et on ne peut pas surréduire u' en u_n. Si le système de réécriture est confluent, il est possible de surréduire u', mais on ne peut pas toujours aboutir à u_n mais à un terme plus général et avec une substitution plus générale que θ_1.

Dans les deux exemples qui suivent, on montre comment faire glisser l'étape de surréduction relativement à la règle r_3 de la dernière position (surdérivation (1)) à la première position (surdérivation (3)). Les systèmes de réécriture considérés sont confluents, nœthérien, interrréduits, et ont tout deux une paire critique obtenu par superposition de r_3 sur r_2.

Exemple 4.5 Dans cet exemple (figure 4.3), les surdérivations (1), (2), (3) aboutissent au même terme, mais dans (3) la composée des substitutions surréductrices est plus générale que dans (1) et (2). Ceci est dû à la non régularité de r_3.

Exemple 4.6 Dans cet exemple (figure 4.4), toutes les règles de réécriture sont régulières. Ici, (3) abouti à un terme plus général que (1) et (2), et dans (3) la composée des substitutions surréductrices est plus générale que dans (1) et (2). Ceci est dû au fait que r_4 n'est pas issue de l'orientation de la paire critique qui résulte de la superposition de r_3 sur r_2, mais est plus générale (au sens du préordre de filtrage).

Nous conjecturons néanmoins que la commutation est possible si le système de réécriture n'est pas linéaire à droite et n'a pas de paires critiques.

Remarque: Dans le lemme précédent, nous aurions pu supposer, comme dans le théorème 4.15, la condition suffisante que la composée des substitutions surréductrices est en forme normale relativement aux règles utilisées dans la dérivation. Nous ne l'avons pas fait car ce lemme sera appliqué sur des dérivations qui utilisent les règles de E_R, qui est formé d'équations orientées dans les deux sens. Il y a donc peu de chance que la composée des substitutions surréductrices soit en forme normale par rapport à E_R.

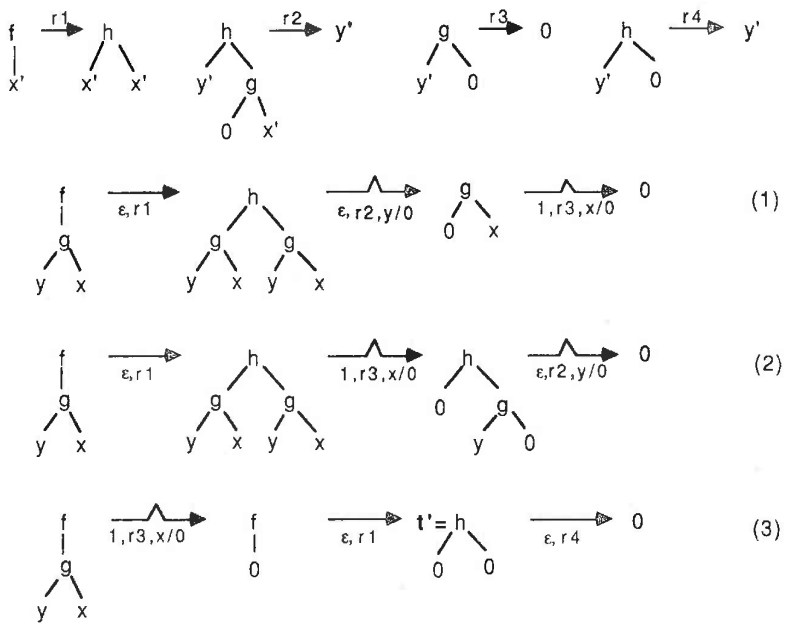


Figure 4.3 :

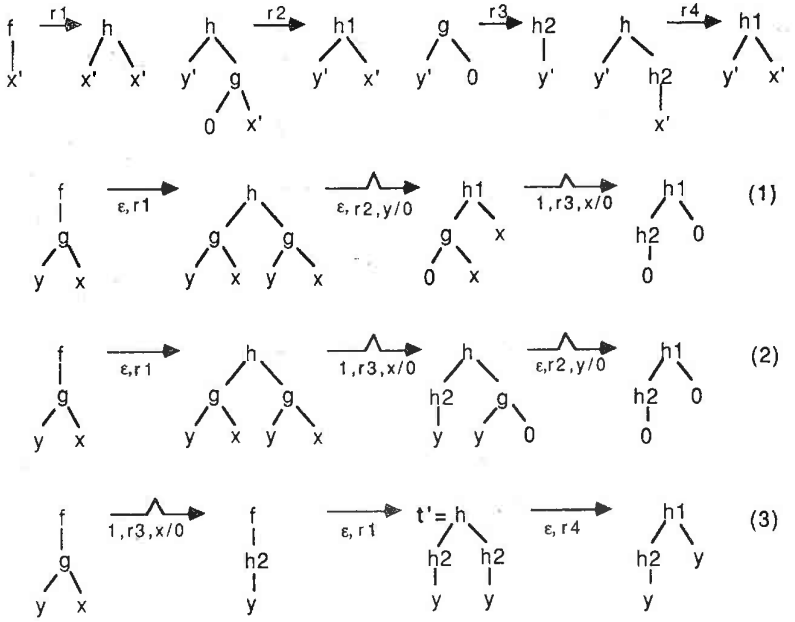


Figure 4.4 :

Lemme 4.17 On suppose que les règles de réécriture qui interviennent sont toutes linéaires à gauche ou bien toutes régulières. Si de plus les étapes $s \overset{*}{\rightsquigarrow}_{[\theta]} t$ (1) se font par des règles linéaires à droites, et s'il existe une dérivation de la forme

$$t \overset{*}{\rightsquigarrow}_{[q.(v_1,\ldots,v_n);r_1,\ldots,r_n;\sigma]} u \quad (2)$$

telle que l'occurrence q de t admet tous ses antécédents $(p_0,\ldots,pk)$ dans $\theta(s)$ à travers la dérivation $\theta(s) \overset{*}{\rightarrow} t$ et que $p_0,..,pk \in O(s)$, alors (2) peut être commutée à travers (1), et on obtient:

$$s \overset{*}{\rightsquigarrow}_{[p_0.(v_1,\ldots,v_n);r_1,\ldots,r_n;\theta'_0]} \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow}_{[p_k.(v_1,\ldots,v_n);r_1,\ldots,r_n;\theta'_k]} t' \overset{*}{\rightsquigarrow}_{[\theta']} t$$

avec $\theta'.\theta'_k \ldots \theta'_0 = \sigma.\theta \ [V(s)]$.

Preuve: a) Puisque q admet tous ces antécédents dans s, alors pour $1 \leq i \leq n$, $q.v_i$ aussi, et ce sont $p_0.v_i,\ldots,p_k.v_i$. D'après le lemme précédent et par récurrence il est alors facile de montrer que (2) commute à travers (1), et on obtient:

$$s \overset{*}{\rightsquigarrow}_{[(p_0,\ldots,p_k).v_1,\sigma_1]} \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow}_{[(p_0,\ldots,p_k).v_n,r_n]} t' \overset{*}{\rightsquigarrow} t$$

b) $p_0,\ldots,p_k$ étant les antécédents d'une même occurrence, ils sont incomparables deux à deux. Par commutation on peut regrouper les occurrences qui commencent par p_0, puis celles qui commencent par p_1, etc..., ce qui donne:

$$s \overset{*}{\rightsquigarrow}_{[p_0.(v_1,\ldots,v_n);r_1,\ldots,r_n;\theta'_0]} \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow}_{[p_k.(v_1,\ldots,v_n);r_1,\ldots,r_n;\theta'_k]} t' \overset{*}{\rightsquigarrow}_{[\theta']} t$$

Toutes les commutations effectuées laissent la composée des substitutions surréductrices inchangée sur les variables de s. □

Définition 4.11 La méthode d'unification $Unif_E$ est dite **R-stable** si pour toute règle $g \to d$ de R et pour toutes substitutions $\sigma, \sigma_1, \sigma_2$ telles que $\sigma = \sigma_2.\sigma_1$ on a

$$t \rightsquigarrow^{RE}_{[p,g\to d,\sigma]} t' \iff t|p \overset{*}{\rightsquigarrow}_{[\sigma_1]} \rightsquigarrow_{[\epsilon,g\to d,\sigma_2]} t'$$

◇

Théorème 4.18 *(commutation de la surréduction équationnelle)*
Supposons que $Unif_E$ est R-stable. Soient s,t,u trois termes et

$$s \rightsquigarrow^{RE}_{[p,g\to d,\sigma]} t \rightsquigarrow^{RE}_{[q,l\to r,\theta]} u \quad (1)$$

deux étapes de réécriture équationnelle telles que:

- *Les règles de $E_{\mathcal{R}} \cup R$ utilisées dans (1) sont soit toutes linéaires à gauche, soit toutes régulières.*
- *l'étape $s \rightsquigarrow^{RE} t$ n'utilise que des règles de $E_{\mathcal{R}} \cup R$ linéaires à droites.*
- *l'occurrence q admet à travers la dérivation $\sigma(s) \to t$ tous ses antécédents dans s, notés $p_0,\ldots,p_{m-1}$ (liste exhaustive), et $p_0,\ldots,p_{m-1} \in O(s)$.*

Alors (1) peut être commuté en

$$s \rightsquigarrow^{RE}_{[p_0,l\to r,\theta'_0]} t'_1 \rightsquigarrow^{RE} \ldots \rightsquigarrow^{RE}_{[p_{m-1},l\to r,\theta'_{m-1}]} t'_m \rightsquigarrow^{RE}_{[p,g\to d,\sigma']} u \quad (2)$$

tel que $\sigma'.\theta'_{m-1}\ldots\theta'_0 = \theta.\sigma \ [V(s)]$.

Preuve: Puisque $Unif_E$ est R-stable, (1) peut s'écrire:

$$s \overset{*}{\rightsquigarrow}^{E_{\mathcal{R}}}_{[p'_0.v',\ldots,p'_i;\sigma_1]} s_1 \rightsquigarrow^{R}_{[p,g\to d,\sigma_2]} t \overset{*}{\rightsquigarrow}^{E_{\mathcal{R}}}_{[q'_0,\ldots,q'_j;r_0,\ldots,r_j;\theta_1]} t_1 \rightsquigarrow^{R}_{[q,l\to r,\theta_2]} u$$

avec $\sigma_2.\sigma_1 = \sigma$ et $\theta_2.\theta_1 = \theta$. Par définition de la surréduction équationnelle les occurrences $q'_0,\ldots,q'_j$ sont plus grandes que q. Posons donc $q'_0 = q.v_0,\ldots,q'_j = q.v_j$. Puisque q admet tous ses antécédents $p_0,\ldots,p_{m-1}$ dans s, d'après le lemme précédent il résulte:

$$s \overset{*}{\rightsquigarrow}_{[p_0.(v_0,\ldots,v_j,\epsilon);r_0,\ldots,r_j,l\to r,\theta'_0]} \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow}_{[p_{m-1}.(v_0,\ldots,v_j,\epsilon);r_0,\ldots,r_j,l\to r,\theta'_{m-1}]} t'$$

$$t' \overset{*}{\rightsquigarrow}^{E_{\mathcal{R}}}_{[p'_0,\ldots,p'_i;\sigma'_1]} \rightsquigarrow^{R}_{[p,g\to d,\sigma'_2]} u$$

avec $\sigma'_2.\sigma'_1.\theta'_{m-1}\ldots\theta'_0 = \theta_2.\theta_1.\sigma_2.\sigma_1 = \theta.\sigma \ [V(s)]$. D'où par stabilité de $Unif_E$:

$$s \rightsquigarrow^{RE}_{[p_0,l\to r,\theta'_0]} \rightsquigarrow^{RE} \ldots \rightsquigarrow^{RE}_{[p_{m-1},l\to r,\theta'_{m-1}]} t' \rightsquigarrow^{RE}_{[p,g\to d,\sigma'_2.\sigma'_1]} u$$

ce qui prouve le théorème. □

4.7 Comparaison des relations de surréduction

Dans ce paragraphe, nous nous plaçons toujours dans le cadre ordo-sorté mais nous travaillons sur la surréduction non équationnelle. Nous avions défini au chapitre précédent la notion de réécriture faiblement basique. Définissons maintenant la surréduction faiblement basique.

Définition 4.12 Soit $t \rightsquigarrow_{[p,g\to d,\theta]} t'$ une étape de surréduction et $v' \in O(t')$. On dit que v' provient de l'instanciation de t si en considérant l'étape de réécriture $\theta(t) \rightarrow_{[p,g\to d]} t'$, l'occurrence v' admet des antécédents dans $\theta(t)$ dont au moins un n'appartient pas à $O(t)$. ◇

Remarque: Si l'étape de surréduction est une réécriture, alors aucune occurrence de t' provient de l'instanciation.

Définition 4.13 Soit une surdérivation

$$t_0 \rightsquigarrow_{[p_0,g_0\to d_0,\sigma_0]} \rightsquigarrow \ldots \rightsquigarrow_{[p_{n-1},g_{n-1}\to d_{n-1},\sigma_{n-1}]} t_n \quad D_1$$

et $U_0,\ldots,U_n$ des ensembles d'occurrences de non variable de $t_0,\ldots,t_n$ respectivement. On dit que D_1 est **faiblement basée** sur U_0 si pour tout $i \in \{0,\ldots,n-1\}$, $p_i \in U_i$ et U_{i+1} contient les occurrences de t_{i+1} qui ne proviennent pas de l'instanciation et dont tous les antécédents, s'il en existe, sont dans U_i. On dit que D_1 est **faiblement basique** s'il existe $U_0 \subseteq O(t_0)$ tel que D_1 soit faiblement basée sur U_0.
Si ce n'est pas ambigu, on définit la notation $FB(t_i, D_1) = U_i$. ◇

Remarque: Les occurrences provenant de l'instanciation sont à la fois protégées dans une étape basique et dans une étape faiblement basique.

Définition 4.14 Soit D une surdérivation et b une étape d'occurrence q de D. On dit que b est **terminale**

- si elle est la première étape de D
- ou si elle est précédée d'une étape b' d'occurrence p telle que
  - q n'admet pas d'antécédent à travers b'
  - ou q est strictement à droite de p.

On dit que D est terminale si toutes ses étapes sont terminales. $\diamond$

Lemme 4.19 Toute surdérivation à la fois terminale et faiblement basée sur U est basée de gauche à droite sur U.

Preuve: Par récurrence sur la longueur i de la surdérivation.

Si $i = 1$ la propriété est vraie.

Supposons la propriété pour i et considérons une surdérivation de longueur $i+1$. Intéressons nous à ses deux dernières étapes:

$$s \rightsquigarrow_{[p,g\to d,\sigma]} t \rightsquigarrow_{[q,l\to r,\theta]} u$$

Par hypothèse de récurrence $p \in B(s)$ ($B(s)$ est l'ensemble des occurrences basiques de s). la surdérivation étant supposée faiblement basique, q n'est pas une occurrence provenant de l'instanciation. Ceci étant, discutons les différentes positions de q par rapport à p.

- Si $q < p$ alors $q \in B(t)$ car l'ensemble des occurrences basiques est clos par préfixe.

- Si q provient du membre droit de règle d, alors $q \in B(t)$.

- Si q est situé sous le membre droit d; puisque q ne provient pas de l'instanciation, il admet des antécédents dans s, ce qui est impossible puisque la surdérivation est terminale.

- Si q est à strictement à gauche de p, puisque la surdérivation est terminale q n'a pas d'antécédent dans s, donc provient de l'instanciation, ce qui est impossible.

- Si q est strictement à droite de p, alors $q \in B(t)$ car $B(t)$ est clos à droite. $\square$

Définition 4.15 Soit deux surdérivations $s \overset{*}{\rightsquigarrow}_{[\gamma]} u\ D_1$ et $s \overset{*}{\rightsquigarrow}_{[\gamma']} u\ D_2$. On dit que D_1 peut être transformé **légalement** en D_2 ou encore que la transformation de D_1 en D_2 est **légale** si

- $\gamma' = \gamma\ [V(s)]$
- D_1 est faiblement basée sur $U \subseteq O(s)$ implique D_2 est faiblement basée sur U et $FB(u, D_2) = FB(u, D_1)$.

$\diamond$

Lemme 4.20 La commutation de la surréduction définie par le théorème 4.15 est une transformation légale.

Preuve: Facile. $\square$

Dans la suite, nous supposons que toutes les règles du système de réécriture courant sont linéaires à gauche ou régulières, et linéaires à droite. Nous supposons aussi que toute surdérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta]} t_n$ est telle que θ soit normalisée sur les variables de t_0. Ainsi, le théorème de commutation (théorème 4.15) pourra être appliqué sur des morceaux de surdérivation de longueur deux, sans changer les termes aux extrémités.

Lemme 4.21 Soit une surdérivation $s \rightsquigarrow_{[p,g\to d,\sigma]} t \overset{*}{\rightsquigarrow} t_n\ D_1$ telle que la partie $t \overset{*}{\rightsquigarrow} t_n$ soit formée par des étapes d'occurrences non comparables deux à deux. Alors D_1 peut être légalement transformée en une surdérivation terminale de la forme $s \overset{*}{\rightsquigarrow} s' \rightsquigarrow_{[p,g\to d,\sigma]} t' \overset{*}{\rightsquigarrow} t_n\ D_2$, où la partie $s \overset{*}{\rightsquigarrow} s'$ est formée par des étapes d'ocurrences non comparables deux à deux.

Preuve: Les occurrences de la partie $t \overset{*}{\rightsquigarrow} t_n$ étant incomparables deux à deux, ses étapes peuvent être commutées et mises dans n'importe quel ordre. Elles peuvent être ordonnée sur leurs occurrences d'application dans le sens de la gauche vers la droite, ce qui donne une surdérivation terminale. Après cela, si l'étape qui suit $s \rightsquigarrow_{[p,g\to d,\sigma]} t$ n'est pas terminale, elle admet des antécédents, et d'après le théorème de commutation peut être placée avant $s \rightsquigarrow_{[p,g\to d,\sigma]} t$. Ainsi il est possible d'obtenir une surdérivation de la forme

$$s \overset{*}{\rightsquigarrow} s' \rightsquigarrow_{[p,g\to d]} t' \overset{*}{\rightsquigarrow} t_n$$

dont les étapes qui succèdent à $s \rightsquigarrow t$ sont terminales et dont la partie $s \overset{*}{\rightsquigarrow} s'$ est formée d'étapes d'occurrences non comparables. $s \overset{*}{\rightsquigarrow} s'$ peut être commutée en une surdérivation terminale. L'étape $s' \rightsquigarrow_{[p,g\to d]} t'$ est terminale car elle est précédée d'une étape qui provient d'une commutation à travers elle. Toutes les transformations effectuées sont légales. $\square$

Lemme 4.22 Soit $t_0 \overset{*}{\rightsquigarrow} t_n\ D_1$ une surdérivation terminale, et soit une surdérivation issue de t_n: $t_n \overset{*}{\rightsquigarrow} t_k\ D_2$ formée d'étapes d'occurrences non comparables (en général $D_1 + D_2$ n'est pas terminale). Alors $D_1 + D_2$ peut être légalement transformée en une surdérivation terminale.

Preuve: Par récurrence sur la longueur de D_1.
Si elle est nulle, on peut commuter D_2 en une surdérivation terminale.

Supposons la propriété pour une longueur de $n-1$. Considérons la dernière étape de D_1 suivie de D_2:

$$t_{n-1} \rightsquigarrow_{[p_{n-1},g_{n-1}\to d_{n-1}]} t_n \overset{*}{\rightsquigarrow} t_k$$

D'après le lemme 4.21 elle peut être tranformée légalement en une surdérivation terminale de la forme

$$t_{n-1} \overset{*}{\rightsquigarrow} s' \rightsquigarrow_{[p_{n-1},g_{n-1}\to d_{n-1}]} t' \overset{*}{\rightsquigarrow} t_k$$

où $t_{n-1} \overset{*}{\rightsquigarrow} s'$ est formée d'étapes d'occurrences non comparables. D'après l'hypothèse de récurrence appliquée sur les surdérivations $t_0 \overset{*}{\rightsquigarrow} t_{n-1}$ D_4 et $t_{n-1} \overset{*}{\rightsquigarrow} s'$ D_5, $D_4 + D_5$ peut être légalement transformée en une surdérivation D_5 terminale. Soit

$$D_3 = D_6 + (s' \rightsquigarrow_{[p_{n-1}, g_{n-1} \to d_{n-1}]} t' \overset{*}{\rightsquigarrow} t_k)$$

Dans D_3 l'étape $s' \rightsquigarrow t'$ est terminale car l'étape qui la précède est soit $t_{n-2} \rightsquigarrow t_{n-1}$ qui est son prédécesseur dans D_1, ou est une étape provenant de la commutation. D_3 provient de $D_1 + D_2$ par une transformation légale car toutes les transformations effectuées furent légales. □

Lemme 4.23 Toute surdérivation peut être légalement transformée en une surdérivation terminale.

Preuve: Par récurrence sur la longueur de la surdérivation. Si la longueur est égale à 1, la propriété est triviale.

Soit $t_0 \overset{*}{\rightsquigarrow} t_{n-1} \rightsquigarrow t_n$ une surdérivation. Par hypothèse de récurrence $t_0 \overset{*}{\rightsquigarrow} t_{n-1}$ peut être légalement transformée en une surdérivation terminale D_3. D'après le lemme précédent appliqué à D_3 et $t_{n-1} \rightsquigarrow t_n$ on obtient une surdérivation terminale par une transformation légale. □

Corollaire 4.24 *Toute surdérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta]} t_n$ D_1 faiblement basée sur $U_0 \subseteq O(t_0)$ peut être transformée en une surdérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta']} t_n$ D_2 basée de gauche à droite sur U_0 telle que $\theta' = \theta$ $[V(t_0)]$ et $FB(t_n, D_2) = FB(t_n, D_1)$.*

Preuve: Le résultat résulte des lemmes 4.19 et 4.23. □

On en déduit le théorème suivant qui permet de comparer les $\rightsquigarrow$-dérivations basiques de gauche à droite avec les $\rightsquigarrow$-dérivations basiques de gauche à droite. Rappelons que $\rightarrow$ désigne la surréduction, $\rightsquigarrow$ est définie par $\rightsquigarrow = \rightarrow_!\overset{*}{\rightarrow}$ où $\overset{*}{\rightarrow}$ est une dérivation de réécriture quelconque mais fixée (par exemple par telle ou telle stratégie), et $\rightsquigarrow$ est la surréduction normalisante: $t \rightsquigarrow t' \iff t \rightarrow_! t_1$, $t' = t_1 \downarrow$. Le théorème suivant est vrai pour n'importe quel choix de $\rightsquigarrow$, donc en particulier pour la surréduction normalisante.

Théorème 4.25 *Supposons que toutes les règles du système de réécriture courant soient linéaires à gauche ou régulières, et linéaires à droite. Soit une $\rightsquigarrow$-dérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta]} t_n$ basée de gauche à droite sur $U_0 \subseteq O(t_0)$ telle que θ soit normalisée sur $[V(t_0)]$. Alors il existe une $\rightsquigarrow$-dérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta']} t_n$ basée de gauche à droite sur U_0 telle que $\theta' = \theta$ $[V(t_0)]$.*

Preuve: Une étape de la $\rightsquigarrow$-dérivation $t_i \rightsquigarrow t_{i+1}$ est formée d'une étape de $\rightarrow$ basique de gauche à droite (donc basique) suivie de réécritures faiblement basiques, c'est à dire $t_i \rightsquigarrow t'_i \overset{*}{\rightarrow} t_{i+1}$. Soit U_i, U'_i, U_{i+1} les ensembles d'occurrences basiques des termes t_i, t'_i, t_{i+1} respectivement. Si on considère $t_i \rightarrow t_{i+1}$ comme une étape faiblement basique, alors U'_i puis U_{i+1} sont agrandis. Donc $t_{i+1} \overset{*}{\rightsquigarrow} t_n$ est encore basée sur le nouvel U_{i+1}. En réitérant ce raisonnement, et en considérant les étapes de réécriture comme des étapes de $\rightarrow$-surréduction, on obtient une surdérivation $t_0 \overset{*}{\rightsquigarrow}_{[\theta]} t_n$ faiblement basée sur U_0. Il suffit maintenant d'appliquer le corollaire précédent. □

Ainsi, l'ensemble des solutions trouvées par la $\rightsquigarrow$-surréduction basique est inclus dans l'ensemble des solutions trouvées par la $\rightarrow$-basique. Donc en utilisant la $\rightsquigarrow$-surréduction basique, l'ensemble des solutions, sous les hypothèses du théorème, est au moins aussi minimal que l'ensemble des solutions qui aurait été obtenu en utilisant la $\rightarrow$-surréduction basique.

Soit $D = t_0 \overset{*}{\rightsquigarrow}_{[\theta]} t_n$ une $\rightsquigarrow$-dérivation basique. On suppose que θ est normalisée sur $V(t_0)$. d'après le théorème précédent, on peut lui faire correspondre une $\rightarrow$-dérivation basique D'. Nous savons d'après les preuves que D' est obtenu à partir de D en appliquant un certain nombre de fois le théorème de commutation (théorème 4.15). Notons par $\|D\|$ le nombre d'étapes de D. Puisque $\rightsquigarrow \subseteq \overset{*}{\rightarrow}$ on peut voir D comme une $\rightarrow$-dérivation, et définir son nombre d'étapes, noté $|D|$. Les règles de réécriture étant supposées linéaires à droite, l'application du théorème de commutation ne diminue pas le nombre d'étapes. Donc $\|D\| \leq |D| \leq |D'|$. Autrement dit D' a plus ou autant d'étapes que D. Par passage à la limite, on en déduit que la $\rightsquigarrow$-surréduction basique, et en particulier la surréduction basique normalisante, termine au moins aussi souvent que la surréduction basique de Hullot. Elle termine même plus souvent, comme le montre l'exemple du chapitre 2 au paragraphe 2.1.3.

4.8 Etude de la minimalité de l'ensemble des solutions générées par surréduction

Nous voulons ici faire apparaitre l'intérêt de certaines optimisations de la surréduction. Le critère d'appréciation utilisé est la minimalité du multiensemble des solutions générées par la méthode, c'est à dire la minimalité de l'ensemble des solutions avec la requête qu'une même solution ne doit pas être générée deux fois. Nous donnons un résultat pour la surréduction basique de gauche à droite en supposant que le système de réécriture courant est régulier et sans paires critiques. Dans les autres cas, nous montrons sur des exemples que la suppression des branches instances (chapitre 2), ainsi que l'introduction de la réécriture faiblement basique (chapitre 3), peuvent diminuer la redondance du multiensemble des solutions.

Nous avons choisi d'étudier la minimalité plutôt que la taille de l'espace de recherche car ce dernier critère est difficile à évaluer. Les exemples que nous donnons montrent que ces deux critères sont liés.

4.8.1 Cas des systèmes de réécriture réguliers et sans paires critiques

Sous cette hypothèse, nous allons montrer qu'une partie du multiensemble des solutions est minimal (théorème 4.28).

Rappelons que les notions de résidu et de largeur suffisante ont été définies au chapitre 3.

Lemme 4.26 Le système de réécriture est supposé régulier et sans paires critiques. Soit t un terme et $U \subseteq O(t)$ un ensemble d'occurrences clos à droite et par préfixe. Soit $t \rightarrow t'$ une étape de réécriture basée sur U de gauche à droite, et U' l'ensemble des occurrences basiques de t'. Si U n'est pas suffisamment large sur t, alors U' n'est pas suffisamment large sur t'.

Preuve: Supposons $t \rightarrow_{[p,g \to d,\sigma]} t'$. Puisque U n'est pas suffisamment large, il existe $q \in$

$O(t) - U$ tel que $t \rightarrow_{[q,l\rightarrow r,\theta]} t''$. Puisque U est clos à droite et par préfixe, seulement deux cas sont possibles:

1. q est strictement à gauche de p. Alors $q \in O(t') - U'$, donc U' n'est pas suffisamment large.
2. $q < p$. Puisque le système de réécriture est régulier et sans paires critiques, q admet au moins un résidu q' dans t'. Or $q' \notin U'$ et $t'|q' = t|q$ n'est pas normalisé. Par conséquent U' n'est pas suffisamment large sur t'.

□

Proposition 4.27 *Le système de réécriture est supposé régulier et sans paires critiques. Alors pour tout terme t, il existe au plus une dérivation de réécriture basique de gauche à droite allant de t à sa forme normale.*

Preuve: Par l'absurde. Supposons qu'il y en ait deux. Elles sont de la forme:

$$\begin{array}{rl} t \xrightarrow{*} s & \rightarrow_{[p_1,g_1\rightarrow d_1]} s_1 \xrightarrow{*} t\downarrow \\ & \rightarrow_{[p_2,g_2\rightarrow d_2]} s_2 \xrightarrow{*} t\downarrow \quad (2) \end{array}$$

avec $(g_1 \rightarrow d_1) \neq (g_2 \rightarrow d_2)$ ou $p_1 \neq p_2$. Puisqu'il n'y a pas de paires critiques, on a dans les deux cas $p_1 \neq p_2$. Etudions les cas possibles:

1. p_1 et p_2 ne sont pas comparables. Sans réduire la généralité de la preuve on peut supposer que p_1 est à gauche de p_2. Alors dans s_2, l'occurrence p_1 n'est pas basique.
2. Sinon on peut supposer que $p_1 < p_2$. Puisque le système de réécriture est régulier et sans paires critiques, l'occurrence p_1 admet un résidu dans s_2 qui n'est pas basique.

Ainsi, l'ensemble des occurrences basiques de s_2 n'est pas suffisamment large. Nous savons par ailleurs qu'il est clos à droite et par préfixe. D'après le lemme précédent l'ensemble des occurrences basiques de $t\downarrow$ dans la dérivation (2) n'est pas suffisamment large, ce qui est impossible. □

Nous n'obtenons pas la minimalité de tout le multiensemble des solutions trouvées par surréduction basique de gauche à droite, mais seulement d'une partie: les solutions de rang maximal. Une solution σ de $t \doteq t'$ est de rang maximal si la seule équation triviale issue par des étapes de réécriture de $\sigma(t \doteq t')$ est sa forme normale. En d'autres termes. cela signifie qu'il est nécessaire d'aller jusqu'à la forme normale pour prouver que σ est une solution de $t \doteq t'$.

Définition 4.16 Soit $t \doteq t'$ une équation et σ une solution de $t \doteq t'$ modulo le système de réécriture R (i.e. $\sigma(t) =_R \sigma(t')$). Toute dérivation de la forme $\sigma(t \doteq t') \xrightarrow{*} s \doteq s$ est appelée preuve de la solution σ de $t \doteq t'$. ◇

Remarque: s n'est pas nécessairement en forme normale.

Définition 4.17 Soit σ une substitution et $t \doteq t'$ une équation. On dit que σ est une solution de $t \doteq t'$ **de rang maximal** si

- $\sigma(t) =_R \sigma(t')$
- Dans toute preuve $\sigma(t \doteq t') \xrightarrow{*} s \doteq s$ de la solution σ de $t \doteq t'$, l'équation $s \doteq s$ est normalisée

◇

Nous considérons une notion de minimalité qui ne prend pas en compte les règles du système de réécriture R.

Définition 4.18 Un multiensemble S de substitutions est dit **minimal** si

- S ne contient pas deux fois la même solution
- et pour tous $\sigma, \theta \in S$, on a $(\sigma \neq \theta \implies \sigma \not\leq \theta \ \wedge \ \theta \not\leq \sigma)$

◇

Théorème 4.28 *Le système de réécriture R est supposé régulier et sans paires critiques. Pour toute équation e, le multiensemble des solutions de e normalisées, de rang maximal, et trouvées par surréduction basique de gauche à droite, est minimal.*

Preuve: Soient θ, θ' deux solutions normalisées de e de rang maximal trouvées par surréduction basique de gauche à droite, et supposons que $\theta \leq \theta'$. Il existe deux surdérivations basiques de gauche à droite:

$$e \rightsquigarrow_{[p_1,g_1\rightarrow d_1,\sigma_1]} e_1 \rightsquigarrow \ldots \rightsquigarrow_{[p_n,g_n\rightarrow d_n,\sigma_n]} e_n$$

$$e \rightsquigarrow_{[q_1,l_1\rightarrow r_1,\sigma'_1]} e'_1 \rightsquigarrow \ldots \rightsquigarrow_{[q_k,l_k\rightarrow r_k,\sigma'_k]} e'_k$$

et deux unificateurs syntaxiques minimaux μ, μ' de e_n et e'_k respectivement, tels que $\theta = \mu.\sigma_n \ldots \sigma_1$ et $\theta' = \mu'.\sigma'_k \ldots \sigma'_1$. Montrons que ces deux surdérivations sont les mêmes.

De ce qui précède, on déduit

$$\theta(e) \rightarrow_{[p_1,g_1\rightarrow d_1]} \rightarrow \ldots \rightarrow_{[p_n,g_n\rightarrow d_n]} \mu(e_n)$$

$$\theta'(e) \rightarrow_{[q_1,l_1\rightarrow r_1]} \rightarrow \ldots \rightarrow_{[q_k,l_k\rightarrow r_k]} \mu'(e'_k)$$

Puisque $\theta \leq \theta'$, posons $\theta' = \gamma.\theta$. Alors

$$\theta'(e) = \gamma.\theta(e) \rightarrow_{[p_1,g_1\rightarrow d_1]} \rightarrow \ldots \rightarrow_{[p_n,g_n\rightarrow d_n]} \gamma.\mu(e_n)$$

Puisque μ est un unificateur syntaxique de e_n, on obtient deux preuves de la solution θ' de e. Puisque θ' est de rang maximal, $\mu'(e'_k)$ et $\gamma.\mu(e_n)$ sont normalisés. D'après la proposition précédente, ces deux dérivations sont identiques, d'où $k = n$ et pour tout $i \in \{1, \ldots, n\}$, $p_i = q_i$, $g_i = l_i$, $d_i = r_i$. □

Remarque: Si l'un des membres de e est clos et normalisé, alors toute solution de e est de rang maximal. En effet soit $e = (t \doteq t')$ et supposons que t' soit clos et normalisé. Pour toute solution σ de e, le terme $\sigma(t') = t'$ est normalisé. Toute dérivation issue de e s'écrit alors $\sigma(e) \xrightarrow{*} s \doteq t'$, donc ne peut être une preuve que si $s \doteq t'$ est normalisée.

Remarque: Soit t un terme et t' un terme normalisé. Si les variables de t et de t' sont disjointes, les filtres de t vers t' sont des unificateurs qui n'instancient pas t'. D'après la remarque précédente, la surréduction basique de gauche à droite fournit donc une procédure de filtrage de t vers t' minimale, i.e. sans redondance, dans les systèmes de réécriture réguliers et sans paires critiques. You obtient un résultat analogue pour son "outer narrowing" [74], mais sa méthode n'est complète que sous des hypothèses restrictives.

4.8.2 Autres cas

Nous montrons sur des exemples que certaines modifications proposées dans cette thèse, à savoir l'élimination des branches instances d'autres branches (voir chapitre 2), et l'introduction de la réécriture faiblement basique (voir chapitre 3), peuvent améliorer la minimalité du multiensemble des solutions lorsque le système de réécriture a des paires critiques ou n'est pas régulier.

Montrons d'abord que la suppression des branches instances peut améliorer la minimalité.

Exemple 4.7 Soit le système de réécriture

$$R = \{ \begin{array}{lll} r_1: & f(x,0,0,z) & \to g(x,z) \\ r_2: & f(x,0,y,0) & \to g(x,y) \\ r_3: & g(0,z) & \to h(z) \ \} \end{array}$$

R est régulier, mais possède une paire critique obtenue en superposant en tête r_1 et r_2. Qu'on utilise la surréduction ou la surréduction basique de gauche à droite pour résoudre l'équation $f(u,x,0,z) \doteq h(y)$, l'arbre de recherche est le même, et est:

$$\begin{array}{ll} f(u,x,0,z) \doteq h(y) & \rightsquigarrow_{[r_1,\sigma_1=(z/0)]} e_1 = (g(u,z) \doteq h(y)) \rightsquigarrow_{[r_3,u/0]} h(z) \doteq h(y) \\ & \rightsquigarrow_{[r_2,\sigma_2=(z/0,z/0)]} e_2 = (g(u,0) \doteq h(y)) \rightsquigarrow_{[r_3,u/0]} h(0) \doteq h(y) \end{array}$$

La première branche donne la solution $\theta_1 = (x/0, u/0, y/z)$, tandis que la seconde donne $\theta_2 = (x/0, z/0, u/0, y/0)$. L'ensemble des solutions trouvées $\{\theta_1, \theta_2\}$ n'est pas minimal car $\theta_1 \leq \theta_2$. Posons $\theta = (z/0)$. On voit que $e_2 = \theta(e_1)$ et $\sigma_2 = \theta.\sigma_1$. D'après un résultat du chapitre 2, on peut s'abstenir de calculer le successeur de e_2, et ainsi seule la solution θ_1 est calculée, ce qui fournit un ensemble de solutions minimal.

Mais on ne peut pas toujours obtenir la minimalité de cette manière.

Exemple 4.8 Soit

$$R = \{ \begin{array}{lll} r_1: & x + x & \to x \\ r_2: & x + 0 & \to x \ \} \end{array}$$

R possède une paire critique. En résolvant $0 + x \doteq 0$ on trouve deux fois la même solution:

$$\begin{array}{ll} 0 + x \doteq 0 & \rightsquigarrow_{[r_1,x/0]} 0 \doteq 0 \\ & \rightsquigarrow_{[r_2,x/0]} 0 \doteq 0 \end{array}$$

Etudions maintenant l'influence de la réécriture faiblement basique (chapitre 3) placée entre les étapes de surréduction basique de gauche à droite, c'est à dire considérons la $\rightsquigarrow$-surréduction basique de gauche à droite. Rappelons qu'une étape de $\rightsquigarrow$-surréduction basique de gauche à droite est composée d'une étape de surréduction basique de gauche à droite, suivie d'étapes de réécriture faiblement basique. Avec une l'hypothèse supplémentaire, le résultat de comparaison du chapitre 4 (théorème 4.25) permet d'affirmer que l'introduction de la réécriture faiblement basique ne diminue pas la minimalité, d'où la proposition:

Théorème 4.29 *Le système de réécriture R est supposé régulier, linéaire à droite, et sans paires critiques. Alors pour toute équation e, le multiensemble des solutions de e, normalisées, de rang maximal, et obtenues par la $\rightsquigarrow$-surréduction basique de gauche à droite est minimal.*

Preuve: Par l'absurde. supposons qu'il existe deux $\rightsquigarrow$-dérivations basiques de gauche à droite donnant les solutions normalisées θ_1 et θ_2, et supposons que $\theta_1 \leq \theta_2$. D'après le théorème 4.25 il existe deux $\rightsquigarrow$-dérivations basique de gauche à droite donnant les solutions θ_1 et θ_2, ce qui est impossible à cause du théorème précédent. □

Ce théorème est plus restrictif que le précédent car le système de réécriture est supposé linéaire à droite. Mais l'introduction de la réécriture faiblement basique présente certains avantages.

Elle peut améliorer la minimalité lorsque le système de réécriture a des paires critiques.

Exemple 4.9 Soit le système de réécriture canonique

$$R = \{ \begin{array}{lll} r_1: & f(y,y) & \to y \\ r_2: & f(y,0) & \to y \\ r_3: & g(h(y),x) & \to f(y,x) \ \} \end{array}$$

En résolvant l'équation $g(z,0) \doteq 0$ par surréduction basique de gauche à droite, on obtient:

$$\begin{array}{ll} g(z,0) \doteq 0 \rightsquigarrow_{[r_3,z/h(y)]} f(y,0) \doteq 0 & \to_{[r_2]} y \doteq 0 \\ & \rightsquigarrow_{[r_1,y/0]} 0 \doteq 0 \end{array}$$

La solution $\theta = (z/h(0),\ y/0)$ est trouvée deux fois. En utilisant la $\rightsquigarrow$-surréduction basique de gauche à droite (les équations sont normalisées après chaque surréduction), la deuxième branche n'est pas calculée, et θ n'est donc générée qu'une seule fois.

Exemple 4.10 Considérons deux opérations binaires $+$, $*$. On suppose que $+$ est idempotent, admet un élément neutre à droite 0, et que 0 est absorbant pour $*$.

$$R = \{ \begin{array}{lll} r_1: & y + y & \to y \\ r_2: & y + 0 & \to y \\ r_3: & y * 0 & \to 0 \ \} \end{array}$$

% désignera les occurrences protégées (non basique). En résolvant $x + (0 * y) \doteq 0$ par surréduction basique de gauche à droite on obtient:

$$\begin{array}{ll} x + (0 * y) \doteq 0 & \rightsquigarrow_{[r_1,x/0*y]} \%(0 * y) \doteq 0 \\ & \rightsquigarrow_{[r_3,y/0]} x + 0 \doteq 0 \ \to_{[r_2]} x \doteq 0 \\ & \rightsquigarrow_{[r_1,x/0]} 0 \doteq 0 \end{array}$$

La solution $\theta = (x/0, y/0)$ est trouvée deux fois. En utilisant la $\rightsquigarrow$-surréduction basique de gauche à droite, la dernière branche n'est pas générée et θ n'est trouvée qu'une seule fois.

L'introduction de la réécriture ne permet pas toujours d'obtenir la minimalité. En effet, elle ne modifierait en rien l'exemple 4.8, dans lequel la solution est générée deux fois.

La réécriture peut améliorer la minimalité lorsque le système de réécriture n'est pas régulier.

Exemple 4.11

$$R = \{ \quad \begin{array}{lrl} r_1 : & f(x,0) & \rightarrow 0 \\ r_2 : & g(0) & \rightarrow 0 \\ r_3 : & h(h'(x),y) & \rightarrow f(g(x),y) \quad \} \end{array}$$

En résolvant l'équation $h(z,0) \doteq 0$ par surréduction basique de gauche à droite on obtient:

$$\begin{array}{llll} h(z,0) \doteq 0 \rightsquigarrow_{[r_3, z/h'(x)]} f(g(x),0) \doteq 0 & \rightarrow_{[r_1]} 0 \doteq 0 & & \\ & \rightsquigarrow_{[r_2, x/0]} f(0,0) \doteq 0 & \rightarrow_{[r_1]} & 0 \doteq 0 \end{array}$$

La première branche fournit la solution $\sigma_1 = (z/h'(x))$, tandis que la deuxième fournit $\sigma_2 = (z/h'(0))$. Ces deux solutions sont comparables. En utilisant la $\leadsto$-surréduction basique de gauche à droite, la deuxième branche n'est pas générée et seule σ_1 est calculée.

Enfin, la réécriture peut améliorer la minimalité quand il existe des solutions qui ne sont pas de rang maximal.

Exemple 4.12

$$R = \{ \quad \begin{array}{lll} r_1 : & f(0) & \rightarrow 0 \\ r_2 : & g(0) & \rightarrow 0 \quad \} \end{array}$$

En résolvant l'équation $f(g(x)) \doteq f(x)$, la seule surdérivation basique de gauche à droite menant à des solutions est:

$$f(g(x)) \doteq f(x) \rightsquigarrow_{[r_2, x/0]} e_1 = \ (f(0) \doteq f(0)) \rightarrow 0 \doteq f(0) \rightarrow e_2 = \ (0 \doteq 0)$$

Les équations e_1 et e_2 fournissent toutes les deux la solution $x/0$. En utilisant la $\leadsto$-surréduction basique de gauche à droite on obtient:

$$f(g(x)) \doteq f(x) \leadsto_{[x/0]} 0 \doteq 0$$

La solution $x/0$ n'est générée qu'une fois, car l'équation e_1 n'apparait plus explicitement.

En conclusion, on peut dire que les chapitres 2 et 3 de cette thèse, par l'élimination des branches instances et l'introduction de la réécriture faiblement basique, apportent une amélioration sensible et intéressante dans l'élimination des redondances du multiensemble des solutions.

Chapitre 5

Une approche transformationnelle de la surréduction basique simplifiante

Ce chapitre présente un article fait en collaboration avec W. Nutt et G. Smolka de l'université de Kaiserslautern, et qui sera prochainement publié dans le "Journal of Symbolic Computation". Bien que ma contribution à ce travail soit essentiellement limitée à la recherche d'une formulation de la réécriture faiblement basique, cet article est donné ici dans sa totalité, afin de faire apparaitre cette collaboration. Il s'agit d'une formulation de la surréduction basique à l'aide de règles d'inférence, et qui intègre des règles de simplification. C'est une première étape vers une formulation équationnelle de la surréduction basique; une formulation complètement équationnelle sera donnée au chapitre suivant. L'idée de base est la suivante. Dans la surréduction basique, la partie apportée par l'unificateur est protégée, donc aucune règle de réécriture ne lui est appliqué. Il n'est donc pas nécessaire d'appliquer l'unificateur, mais il suffit de le mémoriser à part. On utilisera donc la structure de donnée $C.E$ où C est la substitution à appliquer sur E et E la partie à surréduire.

Un deuxième aspect de ce travail est l'introduction de règles de simplification. Une règle de simplification est une règle d'inférence qui transforme une paire $C.E$ en une paire $C'.E'$ qui possède les mêmes solutions. Nous avons voulu intégrer quelques règles connues: décomposition des symboles décomposables, dépliage, et réécriture. La formulation de la réécriture que nous présentons ici est le fruit de la combinaison des travaux de G. Smolka et al. [71] et des travaux présentés au chapitre 3. G. Smolka avait présenté la surréduction basique formulée sur des systèmes d'équation, et introduit la réécriture comme règle de simplification, mais son calcul des occurrences basiques lors des étapes de réécriture était peu optimisé. Au chapitre 3 nous avons présenté la surréduction basique (de gauche à droite) avec réécriture dans un formalisme traditionnel. La combinaison de nos travaux a abouti à l'intégration de la réécriture dans un formalisme de règles d'inférence.

Regardons plus en détail quelques particularités de ce travail.

Quand on infère $C.E \longrightarrow C'.E'$ par surréduction il faut calculer la substitution surréductrice, donc résoudre un problème d'unification syntaxique. On peut remettre ce calcul à plus tard en mettant dans C' non pas la substitution surréductrice, mais le problème d'unification syntaxique sous la forme d'une équation à résoudre. Ainsi ce n'est pas C' qui s'applique sur E, mais la solution de C'. Cette formulation a l'avantage de réduire le controle, donc de permettre une meilleure parallélisation.

Exemple 5.1 Soit $R = \{f(0) \to 0\}$. Pour résoudre modulo R l'équation $f(x) \doteq 0$ on part de la paire $\emptyset.f(x) \doteq 0$ et on infère

$$\emptyset.f(x) \doteq 0 \longrightarrow f(x) \doteq f(0).0 \doteq 0$$

La paire $f(x) \doteq f(0).0 \doteq 0$ peut être interprétée par: si x est choisi tel que $f(x) \doteq f(0)$ alors $0 \doteq 0$.

L'approche de Martelli, Moiso, Rossi [55] présente la même idée, mais décrit une surréduction qui n'est pas basique.

Pour intégrer la décomposition et la collision des symboles décomposables, il faut introduire dans la partie droite des paires non pas une seule équation mais un système d'équations. En effet, soit R le système de réécriture courant, et considérons les symboles de fonction qui n'apparaissent jamais en tête d'un membre gauche de règle. Si f est un tel symbole, on dit qu'il est décomposable dans R puisqu'alors l'équation $f(s_1, \ldots, s_n) \doteq_R f(t_1, \ldots, t_n)$ a les mêmes solutions que le système $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$. Si f, g sont de tels symboles, avec $f \neq g$, on dit que f et g sont disjoints dans R puisqu'alors l'équation $f(\ldots) \doteq_R g(\ldots)$ n'a pas de solutions.

Exemple 5.2 Soit $R = \{g(f(x)) \to g(x)\}$ et résolvons l'équation $h(g(x_1)) \doteq_R f(y)$. En utilisant la surréduction, la procédure ne termine pas car

$$h(g(x_1)) \doteq f(y) \overset{}{\leadsto}_{[1, x_1/f(x_2)]} h(g(x_2)) \doteq f(y) \overset{*}{\leadsto} \ldots$$

Pourtant, d'après les arguments ci-dessus nous savons que cette équation n'a pas de solution modulo R.

Pour prendre en compte ces arguments nous introduisons une règle de décomposition et une règle de collision. Mais pour la décomposition il faudra travailler sur des systèmes d'équations.

Lors des étapes de réécriture, le calcul des occurrences basiques présenté dans ce chapitre est un peu plus optimisé que celui du chapitre 3 (réécriture faiblement basique).

Exemple 5.3 Soit $R = \{g(f(y'), x') \to g(y', f(x'))\}$ et considérons l'équation $g(\%f(x), x) \doteq t$, où % est un marqueur qui désigne les occurrences protégées. Voyons ce qui se passe quand on réécrit le membre gauche de cette équation.
Avec la réécriture faiblement basique:

$$g(\%f(x), x) \doteq t \rightarrow g(x, f(x)) \doteq t$$

Toutes les occurrences deviennent basiques après réécriture.
Avec le formalisme de ce chapitre:

$$\begin{array}{rll} & y \doteq f(x).g(y, x) \doteq t & \\ \longrightarrow & y \doteq f(x).g(x, f(x)) \doteq t & \text{par (R)} \\ \longrightarrow & y \doteq f(x).z \doteq f(x) \& g(x, z) \doteq t & \text{par (Unf)} \\ \longrightarrow C.E = & (y \doteq f(x) \& z = f(x).g(x, z) \doteq t) & \text{par (SBi)} \end{array}$$

En appliquant C sur E, on obtient l'équation $g(x, \%f(x)) \doteq t$, dont le sous-terme $f(x)$ est maintenant protégé. On voit que contrairement à l'approche du chapitre 3, les sous-termes apportés par le membre droit de la règle de réécriture peuvent être protégés. Mais il faut tenter de nombreux dépliages par (unf) et faire de nombreuses comparaisons pour appliquer (SB1).

Ajoutons quelques explications sur certaines règles présentées dans ce chapitre.

- (A) contient le principe même de la surréduction basique
- (B) correspond au fait qu'il faut tenter de résoudre syntaxiquement toutes les équations de l'arbre des surréductions, puisque la surréduction transforme un problème d'unification modulo en un problème d'unification syntaxique.
- (Uni) correspond au calcul de la substitution surréductrice, ce qui nécessite le calcul d'un unificateur principal.
- La règle d'échec numéro 1 correspond au fait que la substitution surréductrice n'existe pas car la partie S de $S.P$ n'a pas de solution syntaxique.
- La règle d'échec numéro 2 signifie que la partie protégée doit être en forme normale, c'est à dire que la propriété de largeur suffisante (sufficient largeness property) du chapitre 3 doit être satisfaite.
- La règle d'échec numéro 4 est la détection des collisions (clashes).
- La combinaison des règles (R), (Unf), et (SB1) permet d'obtenir la réécriture faiblement basique du chapitre 3.

Basic Narrowing Revisited

Werner Nutt
Universität Kaiserslautern

Pierre Réty
Centre de Recherche en Informatique de Nancy

Gert Smolka
Universität Kaiserslautern

Abstract

In this paper we study basic narrowing as a method for solving equations in the initial algebra specified by a ground confluent and terminating term rewriting system. Since we are interested in equation solving, we don't study basic narrowing as a reduction relation on terms but consider immediately it's reformulation as an equation solving rule. This reformulation leads to a technically simpler presentation and reveals that the essence of basic narrowing can be captured without recourse to term unification.

We present an equation solving calculus that features three classes of rules. Resolution rules, whose application is don't care nondeterministic, are the basic rules and suffice for a complete solution procedure. Failure rules detect inconsistent parts of the search space. Simplification rules, whose application is don't care nondeterministic, enhance the power of the failure rules and reduce the number of necessary don't know steps.

Three of the presented simplification rules are new. The rewriting rule allows for don't care nondeterministic rewriting and thus yields a marriage of basic and normalizing narrowing. The safe blocking rule is specific to basic narrowing and is particulary useful in conjunction with the rewriting rule. Finally, the unfolding rule allows for a variety of search strategies that reduce the number of don't know alternatives that need to be explored.

Keywords: Equation Solving, Universal Unification, Narrowing, Basic Narrowing, Normalizing Narrowing, Rewriting.

Acknowledgments: Nutt and Smolka are funded by the Bundesminister für Forschung und Technologie under grant ITR8501A.

Addresses of Authors: Werner Nutt, Gert Smolka, FB Informatik, Universität Kaiserslautern, 6750 Kaiserslautern, West Germany. Pierre Réty, CRIN, Campus Scientifique, BP 239, 54506 Vandœuvre Les Nancy Cedex, France. UUCP: nutt@uklirb, rety@crin, smolka@uklirb.

1 Introduction

Narrowing first appeared in the context of resolution based theorem proving as an adaption of the paramodulation rule [Robinson/Wos 69] to canonical term rewriting systems [Slagle 74, Lankford 75]. Fay [78] realized that narrowing can be employed as a universal unification procedure that solves equations in the theory defined by a canonical rewriting system. Hullot [80] continued Fay's [78] work and devised a new narrowing strategy called basic narrowing. Kirchner [85] extended narrowing to rewriting modulo equations. Kaplan [84] and Hußmann [85] investigated narrowing for conditional term rewriting systems. The recent interest in logic programming with equations [Dershowitz/Plaisted 85, Goguen/Meseguer 86] has generated much work on universal unification (often called E-unification) [Gallier/Snyder 87, Hölldobler 87, Martelli et al. 86] and narrowing [Bosco et al. 87, Fribourg 85, Josephson/Dershowitz 86, Réty et al. 85, You/Subrahmanyam 86] in particular.

Technically, narrowing combines term unification and rewriting. To perform a narrowing step on a term t means to replace t by $\theta(t[\pi \leftarrow v])$, where t/π is a nonvariable subterm of t, $u \to v$ is a variable disjoint copy of a rule, and θ is the most general unifier of the subterm t/π and the left hand side u of the rule. The thus obtained narrowing relation extends the rewriting relation since every rewriting step is also a narrowing step.

Fay's [78] unification procedure employs a normalizing narrowing strategy, where a proper narrowing step is only performed if no rewriting step is possible. In other words, after every proper narrowing step the obtained term is rewritten to normal form. While the application of a rewriting step is don't care nondeterministic (that is, it doesn't matter which rewriting step is applied next), the application of a narrowing step is don't know nondeterministic (that is, it matters which narrowing step is applied next). The advantage of normalizing narrowing over pure narrowing is that it yields a unification procedure with a smaller search space.

Huliot's [80] basic narrowing strategy obtains a search space reduction by restricting narrowing steps to subterms that were not introduced by instantiation. The drawback of this stragtegy is that the application of a narrowing step that is actually a rewriting step is no longer don't care nondeterministic. Recently, the authors [Réty 87, Smolka/Nutt 87] devised special rewriting rules that are compatible with the basic narrowing strategy and whose application is still don't care nondeterministic. This present paper combines and simplifies our results.

We study basic narrowing and its optimizations as a method for solving equations in the initial algebra specified by a ground confluent and terminating term rewriting system. Since we are interested in equation solving, we don't study basic narrowing as a reduction relation on terms but consider immediately it's reformulation as an equation solving rule. This reformulation leads to a technically simpler presentation and reveals that the essence of basic narrowing can be captured without recourse to term unification.

There are several advantages gained from weakening the usual confluence requirement to ground confluence. Applications in algebraic specification and logic programming usually employ initial algebra semantics, which means that ground confluence rather than full confluence is the natural requirement. A typical example is the specification of the integers shown in Figure 1. This specification is a terminating and ground confluent rewriting system, which is not confluent since, for instance, $x * y$ and $((x * y) + y) + (-y)$ are two distinct normal forms of $p(s(x)) * y$. An automatic completion of this system seems to be difficult if not impossible. Réty et al. [85] give a confluent extension of this system by adding thirteen

(1) $p(s(x)) \to x$
(2) $s(p(x)) \to x$

(3) $0 + y \to y$
(4) $s(x) + y \to s(x + y)$
(5) $p(x) + y \to p(x + y)$

(6) $-0 \to 0$
(7) $-s(x) \to p(-x)$
(8) $-p(x) \to s(-x)$
(9) $0 * y \to 0$
(10) $s(x) * y \to (x * y) + y$
(11) $p(x) * y \to (x * y) + (-y)$

Figure 1.1. A specification of the integers as a ground confluent and terminating rewriting system.

inductive consequences. This more than doubles the original rules and thus increases the search space of a narrowing based unification procedure. To be able to weaken the usual confluence requirement to ground confluence, completeness must be defined with respect to solutions, which map variables into irreducible ground terms, rather than unifiers, which map variables to terms possibly containing variables.

Our equation solving calculus employs three classes of rules: *resolution rules* whose application is don't know nondeterministic, *simplification rules* whose application is don't care nondeterministic, and *failure rules* allowing to prune inconsistent parts of a search tree. The resolution rules are the basic rules and suffice for a complete solution procedure. The purpose of the simplification rules is to reduce the search space. In some cases, the use of simplification rules can cut down an infinite search space to a finite one.

Three of the presented simplification rules are new. The rewriting rule allows for don't care nondeterministic rewriting and thus yields a marriage of basic and normalizing narrowing that enjoys the advantages of both approaches. The safe blocking rule is specific to basic narrowing and is particulary useful in conjunction with the rewriting rule. Finally, the unfolding rule allows for a variety of search strategies that reduce the number of don't know alternatives that need to be explored.

Our equation solving calculus is the basis for a class of solution procedures, where the don't know application of a resolution step is followed by the don't care application of finitely many simplification steps. The completeness of these procedures is shown with a new proof technique yielding a scheme that is easily applied to additional or alternative rules. As an application of our proof scheme, we show the completeness of an innermost constructor strategy similar to the one proposed by Fribourg [85].

The paper is organized as follows. In Section 2 we fix our notation for equations and rewriting systems. In Section 3 we present two resolution rules that yield a complete but very inefficient solution procedure. In Section 4, which is the heart of the paper, we extend the equation solving calculus with failure and simplification rules, thus obtaining a far more efficient solution procedure. In Section 5 we show the completeness of a solution procedure that uses inductive consequences for rewriting and prove the completeness of an innermost constructor strategy.

For most applications the use of many-sorted or even order-sorted (many-sorted with subsorts) equational logic is essential. Nevertheless, in this paper we consider only unsorted logic since it suffices to demonstrate our ideas. The generalization of our results to the many-sorted case without subsorts is straightforward. The generalization to the order-sorted case is also not difficult if sort-decreasing rewriting systems [Smolka et al. 87] are employed.

2 Equations and Rewriting Systems

In this section we review the necessary notations for equations and rewriting systems. The reader not familiar with the theory of term rewriting systems may consult [Huet 80, Huet/Oppen 80].

We assume that a set of **function symbols** (ranged over by f, g, and h) and an infinite set of **variables** (ranged over by x, y, z) are given. Every function symbol comes with an **arity**, which is a nonnegative integer.

Terms (ranged over by s, t, u, and v) and **occurrences** of terms (ranged over by π) are defined as usual. We use s/π to denote the **subterm of s at occurrence π** and $s[\pi \leftarrow t]$ to denote the term obtainable from s by replacing the subterm at occurrence π with t. An **equation** $s \doteq t$ is an ordered pair consisting of two terms s and t. The letter P will always range over equations. An **equation system** is a bag $P_1 \,\&\, \cdots \,\&\, P_n$ of equations; we use $\emptyset$ to denote the **empty equation system**. The letter E will always range over equation systems. An equation is called **trivial** if it has the form $s \doteq s$; an equation system is called **trivial** if each of its equations is trivial.

A **syntactical object** is either a term, an equation, or an equation system. A syntactical object is called **ground** if it does not contain variables. We use $\mathcal{V}(O)$ to denote the set of variables occurring in a syntactical object O.

A **signature** is a set of function symbols. The letter Σ will always range over signatures. A syntactical object is called a Σ-object if every function symbol occurring in it is in Σ.

Let Σ be a signature. A **Σ-substitution** is a function from Σ-terms into Σ-terms such that $\theta f(s_1, \ldots, s_n) = f(\theta s_1, \ldots, \theta s_n)$ and $\mathcal{D}\theta := \{x \mid \theta x \neq x\}$ is finite. In abuse of notation, $\mathcal{D}\theta$ is called the **domain of** θ and $\mathcal{C}\theta := \{\theta x \mid x \in \mathcal{D}\theta\}$ is called the **codomain of** θ. Furthermore, $\mathcal{I}\theta := \mathcal{V}(\mathcal{C}\theta)$ is called the set of **variables introduced by** θ. The letters θ, ψ, and ϕ will always range over substitutions. The composition of Σ-substitutions is again a Σ-substitution. Σ-substitutions are extended to syntactical Σ-objects as usual. A substitution θ is ground if θx is a ground term for all $x \in \mathcal{D}\theta$. A substitution θ is **idempotent** if $\theta\theta = \theta$. Note that θ is idempotent if and only if $\mathcal{D}\theta$ and $\mathcal{I}\theta$ are disjoint.

The **equational representation** $[\theta]$ of a substitution θ is the equation system

$$x_1 \doteq \theta x_1 \,\&\, \cdots \,\&\, x_n \doteq \theta x_n$$

where $\{x_1, \ldots, x_n\} = \mathcal{D}\theta$. Two substitutions are equal if and only if their equational representations are equal. Conversely, every Σ-equation system $x_1 \doteq s_1 \,\&\, \cdots \,\&\, x_n \doteq s_n$ such that $x_1, \ldots, x_n$ are distinct variables is the equational representation of some Σ-substitution, which we denote with $\langle x_1 \doteq s_1 \,\&\, \cdots \,\&\, x_n \doteq s_n \rangle$. Note that $\theta = \langle [\theta] \rangle$ for every substitution θ.

Let θ be a substitution and V be a set of variables. The **restriction** $\theta|_V$ of θ to V is defined by: $\theta|_V(x) := \theta x$ if $x \in V$, otherwise $\theta|_V(x) := x$. Furthermore, the **update** $\theta[y \leftarrow s]$ of θ at y with s is defined by: $\theta[y \leftarrow s](x) := s$ if $x = y$, otherwise $\theta[y \leftarrow s](x) := \theta x$.

A syntactical object O is called an **instance** of a syntactical object O' if there is a substitution θ such that $O = \theta O'$. A syntactical object O is called a **variant** of a syntactical object O' if O is obtainable from O' by consistent variable renaming, that is, there exist substitutions θ and ψ such that $O' = \theta O$ and $O = \psi O'$.

Let $\rightarrow$ be a binary relation on a set M. Then we use $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$. The relation $\rightarrow$ is called **confluent** if for all a, b, and c in M such that $a \rightarrow^* b$ and $a \rightarrow^* c$ there exists a d in M such that $b \rightarrow^* d$ and $c \rightarrow^* d$. Furthermore, $\rightarrow$ is called **terminating** if there is no infinite chain $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots$.

A Σ-**rewriting rule** $s \rightarrow t$ is an equation $s \doteq t$ such that s isn't a variable and every variable occurring in the right hand side t occurs in the left hand side s. A **rewriting system** $\mathcal{R} = (\Sigma, \mathcal{E})$ consists of a signature Σ and a set $\mathcal{E}$ of Σ-rewriting rules. A rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ defines a binary relation $\xrightarrow{\mathcal{R}}$ called the **rewriting relation** of $\mathcal{R}$ on the set of all Σ-terms as follows: $s \xrightarrow{\mathcal{R}} t$ if and only if there exists an occurrence π of s and an instance $u \rightarrow v$ of a rule of $\mathcal{R}$ such that $s/\pi = u$ and $t = s[\pi \leftarrow v]$. A term s is $\mathcal{R}$-**normal** if there is no term t such that $s \xrightarrow{\mathcal{R}} t$. A term t is an $\mathcal{R}$-**normal form** of a term s if $s \xrightarrow{\mathcal{R}}{}^* t$ and t is $\mathcal{R}$-normal. An $\mathcal{R}$-**value** is an $\mathcal{R}$-normal ground term. A rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ is **ground confluent** if the restriction of $\xrightarrow{\mathcal{R}}$ to the set of all ground Σ-terms is confluent.

The **initial algebra** $\mathcal{I}(\mathcal{R})$ specified by a ground confluent and terminating rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ can be defined as follows:

- The carrier of $\mathcal{I}(\mathcal{R})$ is the set of all $\mathcal{R}$-values.
- The denotation $f_{\mathcal{I}(\mathcal{R})}$ of a function symbol in Σ is given by $f_{\mathcal{I}(\mathcal{R})}(s_1, \ldots, s_n) = s$, where s is the $\mathcal{R}$-normal form of $f(s_1, \ldots, s_n)$.

A ground Σ-equation $s \doteq t$ is **valid** in (the initial algebra of) $\mathcal{R}$ if s and t have the same $\mathcal{R}$-normal form. We write $\mathcal{R} \models s \doteq t$ or $s =_{\mathcal{R}} t$ if $s \doteq t$ is valid in $\mathcal{R}$. A ground Σ-equation system is **valid** in (the initial algebra of) $\mathcal{R}$ if each of its equations is valid in $\mathcal{R}$. We write $\mathcal{R} \models E$ if E is valid in $\mathcal{R}$. A Σ-equation $s \doteq t$ is an **inductive consequence** of $\mathcal{R}$ if every ground instance of $s \doteq t$ is valid in $\mathcal{R}$. Two ground Σ-substitutions θ and ψ are **equal in** $\mathcal{R}$ (write $\theta =_{\mathcal{R}} \psi$) if $\mathcal{D}\theta = \mathcal{D}\psi$ and $\theta x =_{\mathcal{R}} \psi x$ for every $x \in \mathcal{D}\theta$.

Let $\mathcal{R} = (\Sigma, \mathcal{E})$ be a ground confluent and terminating rewriting system. Then we have:

- "$s =_{\mathcal{R}} t$" is a congruence on the set of all ground Σ-terms, that is, "$s =_{\mathcal{R}} t$" is an equivalence relation satisfying
$$s_1 =_{\mathcal{R}} t_1 \ \wedge \ \cdots \ \wedge \ s_n =_{\mathcal{R}} t_n \quad \Rightarrow \quad f(s_1, \ldots, s_n) =_{\mathcal{R}} f(t_1, \ldots, t_n).$$
- "$\theta =_{\mathcal{R}} \psi$" is an equivalence relation on the set of all ground Σ-substitutions.
- If $\theta =_{\mathcal{R}} \psi$, then $\theta s =_{\mathcal{R}} \psi s$ for every term s such that $\mathcal{V}(s) \subseteq \mathcal{D}\theta = \mathcal{D}\psi$.

3 The Basic Resolution Rules

In this section we develop a simple equation solving calculus that captures the essence

of Hullot's [80] basic narrowing method. This calculus is the basis for a simple solution procedure whose soundness and completeness we will prove. In the next section we will present several extensions for this calculus, thus obtaining a refined solution procedure with a much smaller search space. In particular, the basic calculus to be presented in this section does not yet incorporate term unification, which will only be added in the next section.

General Assumption. *In the rest of this paper we assume that $\mathcal{R} = (\Sigma, \mathcal{E})$ is a ground confluent and terminating rewriting system; furthermore, we assume that there is at least one ground Σ-term.*

We start by defining the solutions of an equation system in the initial algebra of $\mathcal{R}$. A substitution θ is an $\mathcal{R}$-**assignment** if θx is an $\mathcal{R}$-value for all $x \in \mathcal{D}\theta$. We use $\mathrm{ASS}_{\mathcal{R}}$ to denote the set of all $\mathcal{R}$-assignments. With that we define the set of all $\mathcal{R}$-solutions of an equation system E as

$$\mathrm{SOL}_{\mathcal{R}}(E) := \{\theta \in \mathrm{ASS}_{\mathcal{R}} \mid \mathcal{D}\theta = \mathcal{V}(E) \ \wedge \ \mathcal{R} \models \theta E\}.$$

An equation solving procedure for $\mathcal{R}$ is a procedure that enumerates $\mathrm{SOL}_{\mathcal{R}}(E)$.

For technical reasons that will become apparent soon, we need to relativize the solutions of an equation system with respect to a set of "primary variables". The $\mathcal{R}$-**solutions** of an Σ-equation system E with respect to a set V of vaiables are defined as follows:

$$\mathrm{SOL}^V_{\mathcal{R}}(E) := \{\theta|_V \mid \theta \in \mathrm{ASS}_{\mathcal{R}} \ \wedge \ \mathcal{D}\theta = V \cup \mathcal{V}(E) \ \wedge \ \mathcal{R} \models \theta E\}.$$

Note that $\mathrm{SOL}_{\mathcal{R}}(E) = \mathrm{SOL}^{\mathcal{V}(E)}_{\mathcal{R}}(E)$. For convenience, we write $\mathrm{SOL}^V_{\Sigma}(E)$ for $\mathrm{SOL}^V_{(\Sigma,\emptyset)}(E)$, where $(\Sigma, \emptyset)$ is the rewriting system with signature Σ and no rules. Note that $\mathrm{SOL}^V_{\Sigma}(E)$ can be represented rather explicitly by the most general unifier of E, which can be computed using term unification. This will be discussed in Subsection 4.2.

In the literature, narrowing is usually presented for confluent rewriting systems and completeness is shown with respect to all unifiers, which include nonground substitutions. Since we have weakened the confluence requirement to ground confluence, we have to restrict our attention to ground substitutions. Nevertheless, the ground confluence approach subsumes the conventional approach. To see this, assume a confluent rewriting system is given. We can extend this system by adding infinitely many constants to its signature, one for each variable. Then the solutions with respect to the extended system, which is still ground confluent, exactly correspond to the unifiers with respect to the original system.

The rules of our equation solving calculus, which are given in Figure 3.1, apply to pairs $C \,.\, E$ consisting of two equation systems C and E; C is called the **constraint part** and E is called the **unsolved part**. The division of $C \& E$ into two parts is needed to express the basic narrowing strategy. The calculus will allow us to reduce an **initial pair** $\emptyset \,.\, E$ to **solved pairs** $C_1 \,.\, \emptyset$, $C_2 \,.\, \emptyset$, ... such that

- *(Soundness)* $\forall\, i. \quad \mathrm{SOL}^V_{\Sigma}(C_i) \subseteq \mathrm{SOL}^V_{\mathcal{R}}(C_i) \subseteq \mathrm{SOL}^V_{\mathcal{R}}(E)$
- *(Completeness)* $\forall\, \theta \in \mathrm{SOL}^V_{\mathcal{R}}(\emptyset \,.\, E) \quad \exists\, i. \quad \theta \in \mathrm{SOL}^V_{\Sigma}(C_i)$.

Thus, our calculus "solves" by reducing $\mathcal{R}$-solutions to Σ-solutions. The two rules given in Figure 3.1 are called *resolution rules* because they are the primary rules for solving equation

Blocking

(*B*) $C.\ P\ \&\ E \xrightarrow{r}_{\mathcal{R},V} C\ \&\ P.\ E$

Application

(*A*) $C.\ P\ \&\ E \xrightarrow{r}_{\mathcal{R},V} C\ \&\ (P/\pi \doteq u).\ P[\pi \leftarrow v]\ \&\ E$

if P/π isn't a variable and $u \to v$ is a variant of a rule of $\mathcal{R}$ having no variables in common with $C.\ P\ \&\ E$ or V

Figure 3.1. The basic resolution rules.

systems and because we want to distinguish them from the failure and simplification rules to be presented in the next section. With Robinson's [65] resolution rule our resolution rules have only in common that they resolve something—in our case equations.

The application rule in Figure 3.1 has to introduce new variables to obtain a renamed variant of the employed rewriting rule. The following assumption makes sure that there are always enough new variables left.

General Assumption. *In the rest of this paper we assume that V is a finite set of variables.*

Example. Let $\mathcal{R}$ be the system in Figure 1.1, $V = \{y\}$, and consider the equation $0 + y \doteq 0$, which has the unique solution $\langle y \doteq 0 \rangle$. Then:

$$
\begin{array}{ll}
 & \emptyset\ .\ 0 + y \doteq 0 \\
\xrightarrow{r}_{\mathcal{R},V} & 0 + y \doteq 0 + y'\ .\ y' \doteq 0 \qquad \text{by a resolution step using rule (3)} \\
\xrightarrow{r}_{\mathcal{R},V} & 0 + y \doteq 0 + y'\ \&\ y' \doteq 0\ .\ \emptyset \qquad \text{by a blocking step.}
\end{array}
$$

Theorem 3.1. (Soundness) *If $C.\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by a blocking or an application step, then $\mathrm{SOL}^V_{\mathcal{R}}(C'\ \&\ E') \subseteq \mathrm{SOL}^V_{\mathcal{R}}(C\ \&\ E)$.*

Proof. Let $C\ \&\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ and let θ be an assignment such that $V \cup \mathcal{V}(C'.\ E') = \mathcal{D}\theta$ and $\theta(C'\ \&\ E')$ is valid in $\mathcal{R}$. It suffices to show that $\theta(C\ \&\ E)$ is valid in $\mathcal{R}$.

If $C'\ \&\ E'$ has been obtained from $C.\ E$ by a blocking step, then the claim is trivial. If an application step has been performed, then $C.\ E = (C.\ P\ \&\ E_1)$, $C'.\ E' = (C\ \&\ P/\pi \doteq u.\ P[\pi \leftarrow v]\ \&\ E_1)$, and $u \to v$ is a rule of $\mathcal{R}$. It suffices to show that θP is valid in $\mathcal{R}$. Since $\theta(P/\pi) =_{\mathcal{R}} \theta u$ and $u \to v$ is a rule of $\mathcal{R}$, we have $\theta(P/\pi) =_{\mathcal{R}} \theta v$. Since $\theta(P[\pi \leftarrow v]) = (\theta P)[\pi \leftarrow \theta v]$ is valid in $\mathcal{R}$, we know that $(\theta P)[\pi \leftarrow \theta(P/\pi)] = \theta P$ is valid in $\mathcal{R}$. □

The nondeterministic solution procedure in Figure 3.2 is an operational formulation of the equation solving calculus in Figure 3.1. The procedure can be explained as a two person game played by a don't care player who makes the don't care choices and a don't know player who makes the don't know choices. Given $\mathcal{R}$, a pair $\emptyset.\ E$, $V := \mathcal{V}(E)$, and

solve($C.\ E$) is

1. if E is empty, then return C;
2. choose **don't care** an equation P in E;
3. choose **don't know** $C'.\ E'$ such that $C.\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by a step on P;
4. solve($C'.\ E'$)

Figure 3.2. The basic solution procedure.

a solution $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, the don't care player wins if the procedure terminates with an equation system C such that $\theta \notin \mathrm{SOL}^V_\Sigma(C)$; the don't know player wins if the procedure terminates with an equation system C such that $\theta \in \mathrm{SOL}^V_\Sigma(C)$. We say that the procedure is complete if the don't know player can always win if he makes the right choices. In the following we will show the completeness of the procedure.

An implementation of the basic solution procedure has to explore all alternatives of a don't know choice. In fact, the procedure generates a huge number of don't know alternatives in step 3. One alternative is to block P; the other alternatives are obtained by applying a rule to P, where every nonvariable occurrence of P and every rule of $\mathcal{R}$ have to be considered. To be efficient, it is crucial to eliminate redundant or inconsistent don't know alternatives as early as possible. This will be the theme of the next section.

The application rule needs to introduce new variables to obtain a renamed variant of a rewriting rule. The choice of the new variables is obviously a don't care nondeterminism, but making this fact explicit is technically very tedious. For this reason the choice of new variables appears as a don't know nondeterminism in the procedure in Figure 3.2. This problem will be solved in the next section by the introduction of a simplification rule that can be used to rename variables not occurring in V.

The basic idea behind the completeness proof is a lifting argument. If $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, then this fact can be verified by rewriting θE into a trivial equation system. Now the idea is that a blocking step corresponds to the deletion of a trivial equation in θE and an application step corresponds to an innermost rewriting step on θE.

We start by setting up a calculus for verifying that a ground equation system is valid in $\mathcal{R}$. The two rules of the verification calculus correspond to the blocking and the application rule of the equation solving calculus:

- *(VB)* $P\ \&\ E \xrightarrow{vr}_{\mathcal{R}} E$ if P is a trivial equation
- *(VA)* $P\ \&\ E \xrightarrow{vr}_{\mathcal{R}} P[\pi \leftarrow v]\ \&\ E$ if $P/\pi \to v$ is an instance of a rule of $\mathcal{R}$.

The rule *(VB)* deletes a trivial equation and the rule *(VA)* applies a rewriting step.

Proposition 3.2.

- *(Invariance) If $E \xrightarrow{vr}_{\mathcal{R}} E'$, then E is valid in $\mathcal{R}$ if and only if E' is valid in $\mathcal{R}$.*

- *(Termination) The relation "$E \xrightarrow{vr}_{\mathcal{R}} E'$" is terminating.*
- *(Completeness) E is valid in $\mathcal{R}$ if and only if $E \xrightarrow{vr}{}^{*}_{\mathcal{R}} \emptyset$.*

An **$\mathcal{R}$-triple** $\theta.\ C.\ E$ consists of an $\mathcal{R}$-assignment θ and two equation systems C and E such that $\mathcal{V}(C.\ E) \subseteq \mathcal{D}\theta$, θC is trivial, and θE is valid in $\mathcal{R}$. The assignment θ should be thought of as the solution one wants to find by applying resolution steps to the pair $C.\ E$.

Proposition 3.3. *If $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, then $\theta.\ \emptyset.\ E$ is an $\mathcal{R}$-triple. Furthermore, if $\theta.\ C.\ \emptyset$ is an $\mathcal{R}$-triple and $V \subseteq \mathcal{V}(C)$, then $\theta|_V \in \mathrm{SOL}^{V}_{\Sigma}(C)$.*

We now define a reduction relation on $\mathcal{R}$-triples that links resolution steps with their corresponding verification steps. We write $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if $\theta.\ C.\ E$ and $\theta'.\ C'.\ E'$ are both $\mathcal{R}$-triples and

- θ and θ' agree on V
- $C.\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by a resolution rule σ
- $\theta E \xrightarrow{vr}_{\mathcal{R}} \theta' E'$ by the verification rule corresponding to σ.

Proposition 3.4. (Termination) *The triple relation "$\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$" is terminating.*

A term is called **$\mathcal{R}$-innermost** if each of its proper subterms is $\mathcal{R}$-normal. The proof of the following theorem rests on the idea that for a triple $\theta.\ C.\ E$ a verification step that rewrites an innermost term of θE can be "pushed up" to an application step on $C.\ E$.

Theorem 3.5. (Push Up) *If $\theta.\ C.\ E$ is an $\mathcal{R}$-triple and P is an equation in E, then there exists a triple $\theta'.\ C'.\ E'$ such that $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a resolution step on P.*

Proof. Let $\theta.\ C.\ P\&E$ be an $\mathcal{R}$-triple. Then θP is valid in $\mathcal{R}$. Thus θP is either trivial or can be rewritten.

1. Suppose θP is a trivial equation. Then $C.\ P\ \&\ E \xrightarrow{r}_{\mathcal{R},V} C\ \&\ P.\ E$ by the blocking rule and $\theta(P\ \&\ E) \xrightarrow{vr}_{\mathcal{R}} \theta E$ by the verification rule *VB*. Since $\theta.\ C\&P.\ E$ is an $\mathcal{R}$-triple, this yields the claim.

2. Suppose θP can be rewritten. Then there exist a nonvariable occurrence π of P such that $(\theta P)/\pi$ is $\mathcal{R}$-innermost, a variant $u \to v$ of a rule of $\mathcal{R}$, and a substitution ϕ such that $\phi u = (\theta P)/\pi$. Without loss of generality we can assume that $\mathcal{D}\phi = \mathcal{V}(u \to v)$ and $u \to v$ has no variables in common with V, $C.\ P\&E$, and $\mathcal{D}\theta$. Define $C' := (C\ \&\ P/\pi \doteq u)$ and $E' := (P[\pi \leftarrow v]\ \&\ E)$. Since $\theta P\ \&\ \theta E_1 \xrightarrow{vr}_{\mathcal{R}} (\theta P)[\pi \leftarrow \phi v]\ \&\ \theta E_1$ by the verification rule *VA* and $C.\ P\&E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by the resolution rule *A*, it suffices to show that there exists an $\mathcal{R}$-assignment θ' such that $\mathcal{D}\theta' = \mathcal{D}\theta \cup \mathcal{V}(u \to v)$, θ' agrees with θ on $\mathcal{D}\theta$, $\theta'(P/\pi) = \theta' u$, and $\theta'(P[\pi \leftarrow v])$ is valid in $\mathcal{R}$.

To show this, define θ' as follows: if $x \in \mathcal{D}\phi$ then $\theta' x := \phi x$, otherwise $\theta' x := \theta x$. To show that θ' is an $\mathcal{R}$-assignment, it suffices to show that ϕ is an $\mathcal{R}$-assignment, which holds since $\mathcal{D}\phi = \mathcal{V}(u \to v) = \mathcal{V}(u)$, $\phi u = \theta(P/\pi)$ is $\mathcal{R}$-innermost and ground, and u isn't a variable.

Since $\mathcal{D}\phi = \mathcal{V}(u \to v)$, we have $\mathcal{D}\theta' = \mathcal{D}\theta \cup \mathcal{D}\phi = \mathcal{D}\theta \cup \mathcal{V}(u \to v)$ as required. Since $\mathcal{D}\theta$ and $\mathcal{D}\phi = \mathcal{V}(u \to v)$ are disjoint, θ' and θ agree on $\mathcal{D}\theta$. Furthermore, $\theta'(P/\pi) = \theta' u$ since $\theta(P/\pi) = \phi u$.

Finally, $\theta'(P[\pi \leftarrow v]) = (\theta P)[\pi \leftarrow \phi v]$ is valid in $\mathcal{R}$ since $\phi v =_{\mathcal{R}} \phi u$, $\phi u = \theta(P/\pi)$, and $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$. □

Corollary 3.6. *For every $\mathcal{R}$-triple $\theta.\ C.\ E$ there exist θ' and C' such that $\theta.\ C.\ E \xrightarrow{r}{}^{*}_{\mathcal{R},V} \theta'.\ C'.\ \emptyset$.*

Proof. Suppose that $\theta.\ C.\ E$ is an $\mathcal{R}$-triple. If E is empty, then the claim is trivial. Otherwise, the push up theorem applies and yields $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ for some triple $\theta'.\ C'.\ E'$. Thus, using the termination property of the triple reduction relation, the claim follows by induction. □

Corollary 3.7. (Completeness) *Let $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$. Then there exists an equation system C such that $\emptyset.\ E \xrightarrow{r}{}^{*}_{\mathcal{R},\mathcal{V}(E)} C.\ \emptyset$ and $\theta \in \mathrm{SOL}^{\mathcal{V}(E)}_{\Sigma}(C)$.*

Proof. Let $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$. Then $\theta.\ \emptyset.\ E$ is an $\mathcal{R}$-triple. By the preceding corollary we know that there exist θ' and C' such that $\theta.\ \emptyset.\ E \xrightarrow{r}{}^{*}_{\mathcal{R},\mathcal{V}(E)} \theta'.\ C'.\ \emptyset$. Thus, we know that $\theta = \theta'|_{\mathcal{V}(E)} \in \mathrm{SOL}^{\mathcal{V}(E)}_{\Sigma}(C)$. □

4 Failure and Simplification Rules

In this section we present several optimizations for the basic solution procedure that was discussed in the last section. An implementation of this procedure must explore all alternatives of a don't know choice in step 3, which generates a huge search space. To reduce this search space, it is crucial to detect as early as possible whether a pair $C.\ E$ is consistent, that is, whether there is an assignment that extends it to an $\mathcal{R}$-triple. This is accomplished by so-called failure rules, which are decidable sufficient criteria for the inconsistency of a pair. The second method for cutting down the search space is the addition of so-called simplification rules whose application, in contrast to the application of resolution rules, is don't care nondeterministic. By simplifying a pair with the simplification rules before the application of a resolution step it is often possible to reduce the number of don't know resolution steps needed to reach a solved pair. Furthermore, often a failure rule applies to an inconsistent pair only after it has been simplified. Figure 4.1 shows the extension of the basic solution procedure to failure and simplification rules.

4.1 The Failure Rules

The following definitions are needed to formulate the failure rules.

An equation system E is **Σ-consistent** if there is a substitution θ such that θE is trivial. The Σ-consistency of an equation system can be decided by a term unification algorithm.

A pair $C.\ E$ is **consistent** in $\mathcal{R}$ if there exists a substitution θ such that $\theta.\ C.\ E$ is an $\mathcal{R}$-triple.

A function symbol is called **generating** in $\mathcal{R}$ if it occurs in at least one $\mathcal{R}$-value. A function symbol is called **completely defined** in $\mathcal{R}$ if it is not generating in $\mathcal{R}$. In the

rewriting system in Figure 1.1 the functions 0, s and p are generating and the functions $+$, $-$ and $*$ are completely defined.

Two function symbols f and g are **disjoint** in $\mathcal{R}$ if no ground equation of the form $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$ is valid in $\mathcal{R}$.

A function symbol f is called **reducible** in $\mathcal{R}$ if there is a rule ρ in $\mathcal{R}$ such that f is the top symbol of the left hand side of ρ. A function symbol is called **irreducible** in $\mathcal{R}$ if it isn't reducible in $\mathcal{R}$. The constant 0 is the only irreducible function symbol in the rewriting system in Figure 1.1

Proposition 4.1. *If a function symbol is irreducible in $\mathcal{R}$, then it is generating in $\mathcal{R}$. Furthermore, if f and g are distinct function symbols that are both irreducible in $\mathcal{R}$, then f and g are disjoint in $\mathcal{R}$.*

Proposition 4.2. (Failure Rules) *A pair $C. E$ is inconsistent in $\mathcal{R}$ if one of the following conditions holds:*

1. *C is not Σ-consistent.*
2. *C contains an equation $x \doteq t$ such that t is not $\mathcal{R}$-normal.*
3. *C contains an equation $x \doteq t$ such that t contains a completely defined function symbol.*
4. *$C = [\psi]$ for some substitution ψ and ψE contains an equation $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$ such that f and g are disjoint.*

The requirement that the constraint part of a pair is the equational representation of a substitution is not a real restriction since we will introduce a simplification rule that replaces the constraint part by its most general unifier.

The concept of a completely defined function symbol is of little use for unsorted rewriting systems. For instance, if we add to the system in Figure 1.1 the constants *true* and *false*, the functions $+$, $-$ and $*$ are no longer completely defined. This problem can be avoided by working with many-sorted rewriting systems. Since the power of the failure rule (3) increases with the number of completely defined functions, the presence of sorts, even without subsorts, can lead to smaller search spaces.

4.2 Term Unification and Solved Equation Systems

Term unification will be an important part of our optimized solution procedure. After every resolution step the computation of the most general unifier of the constraint part of the obtained pair is attempted. If this attempt fails, we know by the failure rule (1) that the obtained pair is inconsistent. Otherwise, the constraint part can be replaced with the equational representation of its most general unifier, an optimization that will be expressed by a simplification rule. If no other failure and simplication rules are employed, the thereby obtained solution procedure performs essentially basic narrowing as described in [Hullot 80].

In this subsection we review the necessary notations and results for term unification.

An equation system S is called **solved** if it has the form $x_1 \doteq s_1 \;\&\; \ldots \;\&\; x_n \doteq s_n$ where the variables $x_1, \ldots, x_n$ occur only once. Note that an equation system is solved if and only if it is the equational representation of an idempotent substitution. The letter S will always range over solved systems.

solve($C. E$) is

1. choose **don't care** $C'. E'$ such that $C. E \xrightarrow{s}{}^*_{\mathcal{R},V} C'. E'$ by simplification steps;
2. if a failure rule applies to $C'. E'$, then fail;
3. if E' is empty, then return C';
4. choose **don't care** an equation P in E';
5. choose **don't know** $C''. E''$ such that $C'. E' \xrightarrow{r}_{\mathcal{R},V} C''. E''$ by a resolution step on P;
6. solve($C''. E''$)

Figure 4.1. The extended solution procedure.

The next theorem is the adaption of Robinson's [65] unification theorem to our framework.

Theorem 4.3. *A Σ-equation system E is Σ-consistent if and only if there exists a solved Σ-equation system S such that $\mathcal{D}\langle S\rangle \subseteq V$ and $\mathrm{SOL}^V_\Sigma(E) = \mathrm{SOL}^V_\Sigma(S)$.*

For an example, consider $\mathrm{SOL}^{\{x\}}_\Sigma(x + s(0) \doteq s(0) + y) = \mathrm{SOL}^{\{x\}}_\Sigma(x \doteq s(0))$. The next proposition says that the solved system S is a fairly explicit representation of the solution set $\mathrm{SOL}^V_\Sigma(S)$.

Proposition 4.4. *If $\mathcal{D}\langle S\rangle \subseteq V$, then $\mathrm{SOL}^V_\Sigma(S) = \{\langle\theta\langle S\rangle\rangle|_V \mid \forall x \in V.\ \ \theta\langle S\rangle x$ is ground$\}$.*

4.3 The Simplification Rules

Figure 4.2 and 4.3 show the simplification rules we will discuss in this paper. Three of these rules—the rewriting rule, the unfolding rule and the safe blocking rule *SB1*—did not appear in the literature so far. In conjunction with the don't care selection of the equation to be resolved upon next, the unfolding rule can drastically reduce the don't know alternatives our solution procedure has to explore. The rewriting rule, if used together with the unfolding rule and the safe blocking rule *SB1*, results in a marriage of basic and normalizing narrowing that enjoys the advantages of both approaches.

The key property of the simplification rules is that their application preserves the reachable solutions, that is, if $C. E \xrightarrow{s}_{\mathcal{R},V} C'. E'$ by a simplification step, then every solution that can be reached from $C. E$ can also be reached from $C'. E'$. We postpone the proof of this claim to the next subsection. As a consequence of this preservation property, a pair $C. E$ is inconsistent if it is inconsistent after it has been simplified. This fact greatly enhances the power of the failure rules.

The following definition is needed for the decomposition rule. A function symbol f is **decomposable** in $\mathcal{R}$ if for every ground equation $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ that is valid in $\mathcal{R}$ the equations $s_1 \doteq t_1, \ldots, s_n \doteq t_n$ are valid in $\mathcal{R}$. In the rewriting system in Figure 1.1 the function symbols s and p are decomposable.

Unification

(*Uni*) $C.\ E \xrightarrow{s}_{\mathcal{R},V} S.\ E$

if S is solved, $\mathrm{SOL}^W_\Sigma(C) = \mathrm{SOL}^W_\Sigma(S)$, $\mathcal{D}\langle S\rangle \subseteq W$, and $W = V \cup \mathcal{V}(E)$

Rewriting

(*R*) $S.\ P\ \&\ E \xrightarrow{s}_{\mathcal{R},V} S.\ P[\pi \leftarrow v]\ \&\ E$

if $\langle S\rangle(P/\pi) \to v$ is an instance of a rule of $\mathcal{R}$

Unfolding

(*Unf*) $C.\ P\ \&\ E \xrightarrow{s}_{\mathcal{R},V} C.\ x \doteq P/\pi\ \&\ P[\pi \leftarrow x]\ \&\ E$

if x is a new variable, that is, $x \notin V \cup \mathcal{V}(C.\ P\&E)$,
and both P/π and $P[\pi \leftarrow x]$ contain at least one function symbol

Safe Blocking

(*SB1*) $S.\ x \doteq t\ \&\ E \xrightarrow{s}_{\mathcal{R},V} S\ \&\ x \doteq t.\ E$

if S contains an equation $y \doteq s$ such that $\langle S\rangle t$ is a subterm of s

(*SB2*) $C.\ P\ \&\ E \xrightarrow{s}_{\mathcal{R},V} C\ \&\ P.\ E$

if every function symbol occurring in P is irreducible

Decomposition

(*D*) $C.\ f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n)\ \&\ E \xrightarrow{s}_{\mathcal{R},V} C.\ s_1 \doteq t_1\ \&\ \ldots\ \&\ s_n \doteq t_n\ \&\ E$

if f is decomposable

Figure 4.2. The simplification rules, part 1.

Proposition 4.5. *Every irreducible function symbol is decomposable.*

The following rewriting system will be used in examples.

(1) $app(nil, x) \to x$
(2) $app(x.y, z) \to x.app(y, z)$ ($\mathcal{R}1$)

$\mathcal{R}1$ is a confluent and terminating rewriting system. The function symbols *nil* (the empty

Subsumption

(*S*) $S.\ P\ \&\ Q\ \&\ E \xrightarrow{s}_{\mathcal{R},V} S.\ Q\ \&\ E$

if $\langle S\rangle P = \langle S\rangle Q$

Permutation

(*P1*) $C.\ s \doteq t\ \&\ E \xrightarrow{s}_{\mathcal{R},V} C.\ t \doteq s\ \&\ E$

(*P2*) $S.\ x \doteq s\ \&\ t \doteq u\ \&\ E \xrightarrow{s}_{\mathcal{R},V} S.\ x \doteq s\ \&\ x \doteq u\ \&\ E$

if $\langle S\rangle s = \langle S\rangle t$

(*P3*) $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$

if $C'.\ E'$ is obtainable from $C.\ E$ by replacing all occurrences
of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C.\ E)$

Figure 4.3. The simplification rules, part 2.

list) and '.' (the cons operator) are irreducible and thus generating, decomposable and disjoint. The function symbol *app* (list concatenation) is completely defined.

Example 4.6. (Rewriting) We want to solve the equation $app(app(x, y),\ z) \doteq nil$ in $\mathcal{R}1$ with respect to the variable z. This problem has an infinite search space if only unification is employed for simplification, but it has a finite search space if both the unification and rewriting rule can be used. To see this, consider the derivation

$$\begin{array}{ll} & \emptyset\ .\ app(app(x,y),\ z) \doteq nil \\ \xrightarrow{r}_{\mathcal{R}1,\{z\}} & app(x,y) \doteq app(x'.y', z')\ .\ app(x'.app(y', z'),\ z) \doteq nil \quad \text{by } A \\ \xrightarrow{s}_{\mathcal{R}1,\{z\}} & \emptyset\ .\ app(x'.app(y', z'),\ z) \doteq nil \quad \text{by } Uni, \end{array}$$

which can be continued infinitely often by applying rule (2) to the inner occurrence of *app*. However, if the rewriting rule is available for simplification, we can prune this infinite and inconsistent part of the search space by rewriting the above pair to

$$\xrightarrow{s}_{\mathcal{R}1,\{z\}} \quad \emptyset\ .\ x'.app(app(y', z'),\ z) \doteq nil \qquad \text{by } R.$$

This pair can now be recognized as inconsistent by the failure rule (4) since the function symbols '.' and *nil* are disjoint in $\mathcal{R}1$.

The following derivation shows how the solution of the system can be computed:

$$
\begin{array}{lll}
 & \emptyset \cdot app(app(x,y),\ z) \doteq nil & \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & app(x,y) \doteq app(nil,x') \cdot app(x',z) \doteq nil & \text{by } A \\
\xrightarrow{s}_{\mathcal{R}1,\{x\}} & \emptyset \cdot app(x',z) \doteq nil & \text{by } Uni \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & app(x',z) \doteq app(nil,y') \cdot y' \doteq nil & \text{by } A \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & app(x',z) \doteq app(nil,y') \ \&\ y' \doteq nil \cdot \emptyset & \text{by } SB2 \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & z \doteq nil \cdot \emptyset & \text{by } Uni
\end{array}
$$

Example 4.7. (Unfolding) We want to solve the equation $app(x,\ app(y,z)) \doteq nil$ in $\mathcal{R}1$ with respect to the variable x. This problem has a finite search space if the unfolding rule can be used for simplification, while it has infinite search space otherwise. To see this, consider the derivation

$$
\begin{array}{lll}
 & \emptyset \cdot app(x,\ app(y,z)) \doteq nil & \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & app(y,z) \doteq app(x'.y',z') \cdot app(x,\ x'.app(y',z')) \doteq nil & \text{by } A \\
\xrightarrow{s}_{\mathcal{R}1,\{x\}} & \emptyset \cdot app(x,\ x'.app(y',z')) \doteq nil & \text{by } Uni,
\end{array}
$$

which can be continued infinitely often by applying rule (2) to the inner occurrence of app. However, if we start with the unfolding rule, we can prune this infinite and inconsistent part of the search space and compute the solution as follows:

$$
\begin{array}{lll}
 & \emptyset \cdot app(x,\ app(y,z)) \doteq nil & \\
\xrightarrow{s}_{\mathcal{R}1,\{x\}} & \emptyset \cdot app(x,x') \doteq nil \ \&\ x' \doteq app(y,z) & \text{by } Unf \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & app(x,x') \doteq app(nil,y') \cdot y' \doteq nil \ \&\ x' \doteq app(y,z) & \text{by } A \\
\xrightarrow{s}^{*}_{\mathcal{R}1,\{x\}} & x \doteq nil \ \&\ x' \doteq nil \cdot x' \doteq app(y,z) & \text{by } SB2,\ Uni \\
\xrightarrow{r}_{\mathcal{R}1,\{x\}} & x \doteq nil \ \&\ x' \doteq nil \ \&\ app(y,z) \doteq app(nil,z') \cdot x' \doteq z' & \text{by } A \\
\xrightarrow{s}^{*}_{\mathcal{R}1,\{x\}} & x \doteq nil \cdot \emptyset & \text{by } SB2,\ Uni.
\end{array}
$$

Compared to ordinary narrowing, the basic narrowing strategy achieves a smaller search space by avoiding many derivations that don't correspond to innermost rewriting chains. This becomes apparent in the proof of the push up theorem, where only innermost rewriting steps are pushed up, and with the failure rules (2) and (3). However, as the last example demonstrates, this innermost flavor of the basic narrowing strategy can be weakened by using the unfolding rule without loosing the search space reductions.

The last example also demonstrates that, in conjunction with the don't care selection of the next equation to be resolved upon, the unfolding rule can lead to drastic search space reductions by breaking large equations with many don't know alternatives into small equations with few don't know alternatives. For instance, if the extended solution procedure selects the equation

$$app(app(x,y),\ app(x',y')) \doteq z$$

in step 4, it must explore the following five, not obviously inconsistent, don't know alternatives:

(1) blocking the equation,
(2) applying rule (1) of $\mathcal{R}1$ to the left inner occurrence of app,
(3) applying rule (2) of $\mathcal{R}1$ to the left inner occurrence of app,
(4) applying rule (1) of $\mathcal{R}1$ to the right inner occurrence of app, and
(5) applying rule (2) of $\mathcal{R}1$ to the right inner occurrence of app.

The alternatives (2) and (3) or, alternatively, (4) and (5) seem to be redundant since it shouldn't make a difference whether the left or right inner occurrence of app is considered

first. This idea can be exploited by unfolding the right inner occurrence of app, which yields the equations

$$z' \doteq app(x',y') \ \&\ app(app(x,y),z') \doteq z$$

and thus eliminates the alternatives (2) and (3) if the left equation is considered first.

In conjunction with the don't care selection of the next equation to be resolved upon the unfolding rule can be used to obtain a variety of strategies that reduce the don't know alternatives a solution procedure has to consider. Two examples are the the left-to-right basic narrowing strategy in [Herold 86] and the selection narrowing strategy in [Bosco et al. 87]. Another example is the innermost constructor strategy in [Fribourg 85], which we will discuss in the next section.

Bosco et al. [87] present a translation of basic narrowing into SLD-resolution [Lloyd 84], which gives them implicitly the effect we would obtain by using the unfolding rule as often as possible. Complete unfolding, however, has the disadvantage of reducing the power of the rewriting rule. Nevertheless, Bosco et al.'s [87] paper gave us the idea for the unfolding rule.

The application conditions of the unfolding rule ensure that it can't produce equations of the form $x \doteq y$, a restriction that is needed to preserve the completeness of the extended solution procedure.

Example 4.8. (Safe Blocking) As we have seen in Example 3.1, using the rewriting rule for simplification may cut down an infinite search space to a finite one. A disadvantage of the rewriting rule is, however, that it transfers terms from the constraint part back into the unsolved part, thus increasing the search space again. To see this, let $\mathcal{R}$ be the rewriting system in Figure 1.1 and consider the rewriting step

$$
\begin{array}{lll}
 & y \doteq s(s(s(z))) \cdot x + p(y) \doteq 0 & \\
\xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \cdot x + s(s(z)) \doteq 0 & \text{by } R,
\end{array}
$$

which carries the term $s(s(z))$ from the constraint part into the unsolved part. This disadvantage can be completely avoided by using the unfolding and the safe blocking rule to transfer terms carried over by the rewriting rule back into the constraint part:

$$
\begin{array}{lll}
\xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \cdot x' \doteq s(s(z)) \ \&\ x + x' \doteq 0 & \text{by } Unf \\
\xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \ \&\ x' \doteq s(s(z)) \cdot x + x' \doteq 0 & \text{by } SB1.
\end{array}
$$

Example 4.9. (Naive Rewriting) The following restriction of the application rule, which we will refer to as the *naive rewriting rule*, seems to be a better alternative to the rewriting rule in Figure 4.2 since it doesn't transfer terms from the constraint part to the unsolved part:

$$
\begin{array}{l}
S.\ P\ \&\ E \quad \longrightarrow_{\mathcal{R},V} \quad S\ \&\ (P/\pi \doteq u).\ P[\pi \leftarrow v]\ \&\ E \\
\qquad \text{if } P/\pi \text{ isn't a variable,} \\
\qquad u \to v \text{ is a variant of a rule of } \mathcal{R} \text{ containing only new variables,} \\
\qquad \text{and } \langle S \rangle (P/\pi) \text{ is an instance of } u.
\end{array}
$$

However, this rule cannot be used as a simplification rule since, in general, its application is not don't care nondeterministic. To see this, consider the rewriting system in Figure 1.1 and the initial pair

$$\emptyset \cdot s(p(x+0)) \doteq 0,$$

which has the unique solution $(x \doteq 0)$. By applying the naive rewriting rule to s with rule (2) we obtain the pair

$$s(p(x+0)) \doteq s(p(x')) \,.\, x' \doteq 0,$$

which, after a unification step, becomes

$$x' \doteq x + 0 \,.\, x' \doteq 0.$$

The only resolution step that applies to the unsolved equation of this pair is blocking, which yields

$$x' \doteq x + 0 \;\&\; x' \doteq 0 \,.\, \emptyset,$$

a pair whose constraint part is Σ-inconsistent. This shows that the application of the naive rewriting rule is not don't care nondeterministic.

Example 4.10. (Decomposition) Let $\mathcal{R}$ be the rewriting system in Figure 1.1 and consider the equation $s(x) \doteq s(y)$. Since s is decomposable in $\mathcal{R}$ (note that s is not irreducible in $\mathcal{R}$), we know by the decomposition rule that the equation $x \doteq y$, which is in solved form, has the same solutions in $\mathcal{R}$ as the equation $s(x) \doteq s(y)$. Without the decomposition rule, however, our solution procedure cannot avoid to compute a second solved form that is redundant:

	$\emptyset \,.\, s(x) \doteq s(y)$	
$\xrightarrow{r}_{\mathcal{R},\{y\}}$	$s(x) \doteq s(p(x')) \,.\, x' \doteq s(y)$	by *A*
$\xrightarrow{s}_{\mathcal{R},\{y\}}$	$\emptyset \,.\, x' \doteq s(y)$	by *Uni*
$\xrightarrow{r}_{\mathcal{R},\{y\}}$	$s(y) \doteq s(p(y')) \,.\, x' \doteq y'$	by *A*
$\xrightarrow{s}_{\mathcal{R},\{y\}}$	$y \doteq p(y') \,.\, \emptyset$	by *SB2*, *Uni*.

With the permutation rule *P3* it is possible to rename auxiliary variables, that is, variables that don't occur in V. We have included this rule to show that the introduction of new variables by the application rule (Figure 3.1) is actually a don't care nondeterminism.

4.4 Soundness and Completeness Proofs

Theorem 4.11. (Soundness) *If* $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ *by a simplification step, then* $\mathrm{SOL}^V_{\mathcal{R}}(C' \;\&\; E') \subseteq \mathrm{SOL}^V_{\mathcal{R}}(C \;\&\; E)$.

Proof. Let $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ by a simplification step and let θ be an assignment such that $\mathcal{D}\theta = V \cup \mathcal{V}(C' \,.\, E')$ and $\theta(C' \;\&\; E')$ is valid in $\mathcal{R}$. We have to show that there exists an assignment θ' such that $V \cup \mathcal{V}(C \;\&\; E) \subseteq \mathcal{D}\theta'$, θ' and θ agree on V, and $\theta'(C \;\&\; E)$ is valid in $\mathcal{R}$. Let the simplification rule employed in $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ be:

Uni. Then $C' \,.\, E' = S \,.\, E$, where S is solved, $\mathrm{SOL}^W_\Sigma(C) = \mathrm{SOL}^W_\Sigma(S)$ and $\mathcal{D}\langle S\rangle \subseteq W = V \cup \mathcal{V}(E)$. It suffices to show that there exists a ground substitution $\tilde{\theta}$ such that $V \cup \mathcal{V}(C \;\&\; E) \subseteq \mathcal{D}\tilde{\theta}$, $\theta|_V =_{\mathcal{R}} \tilde{\theta}|_V$, and $\tilde{\theta}(C \;\&\; E)$ is valid in $\mathcal{R}$, since then defining $\theta' x$ as the normal form of $\tilde{\theta} x$ for every $x \in \mathcal{D}\theta$ yields the claim.

Since S is solved, we know that $\langle S\rangle S$ is a trivial equation system, which implies that $\theta\langle S\rangle S$ is a trivial system. This implies $(\theta\langle S\rangle)|_W \in \mathrm{SOL}^W_\Sigma(S) = \mathrm{SOL}^W_\Sigma(C)$. Therefore, there

exists a ground substitution $\tilde{\theta}$ such that $\mathcal{D}\tilde{\theta} = W \cup \mathcal{V}(C) = V \cup \mathcal{V}(E) \cup \mathcal{V}(C)$, $\tilde{\theta}|_W = (\theta\langle S\rangle)|_W$, and $\tilde{\theta}C$ is trivial. In particular, $\tilde{\theta}C$ is valid in $\mathcal{R}$.

Since θS is valid in $\mathcal{R}$, we have $\theta =_{\mathcal{R}} \theta\langle S\rangle$, which yields $\theta|_W =_{\mathcal{R}} (\theta\langle S\rangle)|_W = \tilde{\theta}|_W$. Since $W = V \cup \mathcal{V}(E)$, this yields that $\tilde{\theta}E$ is valid in $\mathcal{R}$ and $\theta|_V =_{\mathcal{R}} \tilde{\theta}|_V$.

R. Then $C \,.\, E = (S \,.\, P \;\&\; E_1)$ and $C' \,.\, E' = (S \,.\, P[\pi \leftarrow v] \;\&\; E_1)$, where $\langle S\rangle(P/\pi) \to v$ is an instance of a rule of $\mathcal{R}$. It suffices to show that $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$, which in turn follows from $\theta(P/\pi) =_{\mathcal{R}} \theta v$, since $(\theta P)[\pi \leftarrow \theta v] = \theta(P[\pi \leftarrow v])$ is valid in $\mathcal{R}$. Since θS is valid in $\mathcal{R}$, we know that $\theta =_{\mathcal{R}} \theta\langle S\rangle$. Hence, $\theta(P/\pi) =_{\mathcal{R}} \theta\langle S\rangle(P/\pi) =_{\mathcal{R}} \theta v$ as required.

Unf. Then $C \,.\, E = (C \,.\, P \;\&\; E_1)$ and $C' \,.\, E' = (C \,.\, x \doteq P/\pi \;\&\; P[\pi \leftarrow x] \;\&\; E_1)$, where x is a new variable. It suffices to show that $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$, which holds since $\theta x =_{\mathcal{R}} \theta(P/\pi)$ and $(\theta P)[\pi \leftarrow \theta x] = \theta(P[\pi \leftarrow x])$ is valid in $\mathcal{R}$.

SB1 or *SB2.* Then the claim is trivial.

D. Then the claim follows from the congruence property of the relation "$s =_{\mathcal{R}} t$".

S. Then $C \,.\, E = (S \,.\, P \;\&\; Q \;\&\; E_1)$ and $C' \,.\, E' = (S \,.\, Q \;\&\; E_1)$, where $\langle S\rangle P = \langle S\rangle Q$. It suffices to show that θP is valid in $\mathcal{R}$. Since θS is valid in $\mathcal{R}$, we have $\theta =_{\mathcal{R}} \theta\langle S\rangle$, and since θQ is valid in $\mathcal{R}$, we know that $\theta\langle S\rangle Q$ is valid in $\mathcal{R}$. This yields that θP is valid in $\mathcal{R}$ since $\langle S\rangle Q = \langle S\rangle P$.

P1. Then the claim is trivial.

P2. Then $C \,.\, E = (S \,.\, x \doteq s \;\&\; t \doteq u \;\&\; E_1)$ and $C' \,.\, E' = (S \,.\, x \doteq s \;\&\; x \doteq u \;\&\; E_1)$, where $\langle S\rangle s = \langle S\rangle t$. It suffices to show that $\theta t =_{\mathcal{R}} \theta u$. Since θS is valid in $\mathcal{R}$, we know that $\theta =_{\mathcal{R}} \theta\langle S\rangle$, which yields that $\theta t =_{\mathcal{R}} \theta\langle S\rangle t = \theta\langle S\rangle s =_{\mathcal{R}} \theta s =_{\mathcal{R}} \theta x =_{\mathcal{R}} \theta u$.

P3. Then $C' \,.\, E'$ has been obtained from $C \,.\, E$ by replacing all occurrences of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C \,.\, E)$. Thus $\theta' := \theta[x \leftarrow \theta y]$ yields the claim. □

Our next goal is to prove the completeness of the extended solution procedure in Figure 4.1. As before, the proof will be based on the notion of a triple reduction relation, which links steps on the resolution level with steps on the verification level. We start by giving the corresponding verification rule for every simplification rule:

- *(VUni), (VP3)* $E \xrightarrow{vs}_{\mathcal{R}} E$
- *(VR)* $P \;\&\; E \xrightarrow{vs}_{\mathcal{R}} P[\pi \leftarrow v] \;\&\; E$ if $P/\pi \to v$ is an instance of a rule of $\mathcal{R}$
- *(VUnf)* $P \;\&\; E \xrightarrow{vs}_{\mathcal{R}} s \doteq P/\pi \;\&\; P[\pi \leftarrow s] \;\&\; E$ if s is the $\mathcal{R}$-normal form of P/π
- *(VSB1), (VSB2)* $P \;\&\; E \xrightarrow{vs}_{\mathcal{R}} E$ if P is a trivial equation
- *(VD)* $f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n) \;\&\; E \xrightarrow{vs}_{\mathcal{R}} s_1 \doteq t_1 \;\&\; \ldots \;\&\; s_n \doteq t_n \;\&\; E$ if f is decomposable
- *(VS)* $P \;\&\; P \;\&\; E \xrightarrow{vs}_{\mathcal{R}} P \;\&\; E$
- *(VP1)* $s \doteq t \;\&\; E \xrightarrow{vs}_{\mathcal{R}} t \doteq s \;\&\; E$
- *(VP2)* $v \doteq s \;\&\; s \doteq u \;\&\; E \xrightarrow{vs}_{\mathcal{R}} v \doteq s \;\&\; v \doteq u \;\&\; E$ if v is $\mathcal{R}$-normal.

Proposition 4.12. (Invariance) *Let $E \xrightarrow{vs}_{\mathcal{R}} E'$. Then E is valid in $\mathcal{R}$ if and only if E' is valid in $\mathcal{R}$.*

The **$\mathcal{R}$-complexity** $\|E\|_{\mathcal{R}}$ of an equation system E is defined as the maximal length of an $\mathcal{R}$-rewriting derivation issuing from E.

Proposition 4.13. (Compatibility) *If $E \xrightarrow{vs}_{\mathcal{R}} E'$, then $\|E\|_{\mathcal{R}} \geq \|E'\|_{\mathcal{R}}$.*

Next we extend the simplification steps to $\mathcal{R}$-triples. We write $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if both $\theta.\ C.\ E$ and $\theta'.\ C'.\ E'$ are $\mathcal{R}$-triples and

- θ and θ' agree on V
- $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ by some simplification rule σ
- $\theta E \xrightarrow{vs}_{\mathcal{R}} \theta' E'$ by the verification rule corresponding to σ.

The next theorem is the counterpart to the push up theorem for the resolution rules. Since the application of the simplification rules is supposed to be don't care nondeterministic, we must be able to push down a simplification step from the resolution level to the verification level.

Theorem 4.14. (Push Down) *If $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ by a simplification step and $\theta.\ C.\ E$ is an $\mathcal{R}$-triple, then there exists an assignment θ' such that $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$.*

Proof. Let $\theta.\ C.\ E$ be an $\mathcal{R}$-triple. Then $\mathcal{V}(C.\ E) \subseteq \mathcal{D}\theta$, θC is trivial, and θE is valid in $\mathcal{R}$. We will show that for every simplification step $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ there exists an assignment θ' such that $\mathcal{V}(C'.\ E') \subseteq \mathcal{D}\theta'$, θ and θ' agree on V, $\theta' C'$ is trivial, and $\theta E \xrightarrow{vs}_{\mathcal{R}} \theta' E'$ by the corresponding verification step. Let the simplification rule employed in $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ be:

Uni. Then $C'.\ E' = S.\ E$, where S is solved, $\mathrm{SOL}^W_\Sigma(C) = \mathrm{SOL}^W_\Sigma(S)$, $\mathcal{D}\langle S\rangle \subseteq W$, and $W = V \cup \mathcal{V}(E)$. Since θC is trivial and $W \cup \mathcal{V}(C) \subseteq \mathcal{D}\theta$, we have $\theta|_W \in \mathrm{SOL}^W_\Sigma(C) = \mathrm{SOL}^W_\Sigma(S)$. Therefore, there exists a ground substitution θ' such that θ' agrees with θ on W, $\theta' S$ is trivial, and $\mathcal{D}\theta' = W \cup \mathcal{V}(S)$. Since $\mathcal{V}(E) \subseteq W$, we know that $\theta' E = \theta E$ is valid in $\mathcal{R}$. Thus, it suffices to show that $\theta' x$ is $\mathcal{R}$-normal for every $x \in W \cup \mathcal{V}(S) = W \cup \mathcal{I}(S)$.

If $x \in W$, then $\theta' x$ is $\mathcal{R}$-normal, since $\theta' x = \theta x$ and θx is $\mathcal{R}$-normal. If $x \in \mathcal{I}\langle S\rangle$, then there is an equation $y \doteq s$ in S such that x occurs in s and $y \in \mathcal{D}\langle S\rangle \subseteq W$. Hence, $\theta' x$ is a subterm of the term $\theta' s$, which is $\mathcal{R}$-normal since $\theta' s = \theta' y = \theta y$. Thus $\theta' x$ is $\mathcal{R}$-normal.

R. Then $C.\ E = (S.\ P\ \&\ E_1)$ and $C'.\ E' = (S.\ P[\pi \leftarrow v]\ \&\ E_1)$, where $\langle S\rangle(P/\pi) \to v$ is an instance of a rule of $\mathcal{R}$. It suffices to prove that $\theta P \xrightarrow{vs}_{\mathcal{R}} \theta(P[\pi \leftarrow v])$ by the verification rule VR, since then we can define $\theta' := \theta$.

Since $\theta.\ S.\ E$ is an $\mathcal{R}$-triple, θS is trivial. Hence $\theta = \theta\langle S\rangle$, which implies that $(\theta P)/\pi = \theta(P/\pi) = \theta\langle S\rangle(P/\pi)$. Thus $\theta P \xrightarrow{vs}_{\mathcal{R}} (\theta P)[\pi \leftarrow \theta v]$ by the verification rule VR, since $\langle S\rangle(P/\pi) \to v$ is an instance of a rule of $\mathcal{R}$.

Unf. Then $C.\ E = (C.\ P\ \&\ E_1)$ and $C'.\ E' = (C.\ x \doteq P/\pi\ \&\ P[\pi \leftarrow x]\ \&\ E_1)$, where x is a new variable. Defining $\theta' := \theta[x \leftarrow s]$, where s is the $\mathcal{R}$-normal form of $\theta(P/\pi)$, yields the claim.

SB1. Then $C.\ E = (S.\ x \doteq t\ \&\ E_1)$ and $C'.\ E' = (S\ \&\ x \doteq t.\ E_1)$, where S contains an equation $y \doteq s$ such that $\langle S\rangle t$ is a subterm of s. It suffices to show that $\theta x \doteq \theta t$ is a trivial equation, since then we can define $\theta' := \theta$.

Since $\theta.\ C.\ E$ is an $\mathcal{R}$-triple, we have that $\theta = \theta\langle S\rangle$, $\theta y = \theta s$, and $\theta x =_{\mathcal{R}} \theta t$. Since $\theta t = \theta\langle S\rangle t$ and $\langle S\rangle t$ is a subterm of s, we know that θt is a subterm of θs. Since $\theta s = \theta y$ is an $\mathcal{R}$-value, θt is an $\mathcal{R}$-value. Since θx is an $\mathcal{R}$-value and $\theta x =_{\mathcal{R}} \theta t$, we conclude that $\theta x = \theta t$.

SB2. Then $C.\ E = (C.\ P\ \&\ E_1)$ and $C'.\ E' = (C\ \&\ P.\ E_1)$, where every function symbol occurring in P is irreducible. It suffices to prove that θP is trivial, since then we can define $\theta' := \theta$. Since θ is normal and every function symbol occurring in P is irreducible, θP cannot be rewritten. Since θP is valid in $\mathcal{R}$, this yields the claim.

D, S, P1 or *P2.* For these rules $\theta' := \theta$ does the job.

P3. Then $C'.\ E'$ has been obtained from $C.\ E$ by replacing all occurrences of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C.\ E)$. Defining $\theta' := \theta[y \leftarrow \theta x]$ yields the claim. □

We write $\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if $\theta.\ C.\ E \xrightarrow{s}{}^{\bullet}_{\mathcal{R},V} \theta''.\ C''.\ E'' \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ for some $\mathcal{R}$-triple $\theta''.\ C''.\ E''$. By the push down and the push up theorem we know that the extended solution procedure builds a derivation

$$\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E' \xrightarrow{sr}_{\mathcal{R},V} \theta''.\ C''.\ E'' \xrightarrow{sr}_{\mathcal{R},V} \cdots,$$

provided the right don't know choices are made. Thus, we know that the procedure is complete if we can show that the triple reduction relation "$\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E'$" is terminating. To do this, we will define a complexity measure on triples that is decreased by resolution steps and not increased by simplification steps. A first attempt to define the complexity of a triple $\theta.\ C.\ E$ could be to use $\|E\|_{\mathcal{R}}$. However, this doesn't work since the resolution step B (blocking) doesn't necessarily decrease $\|E\|_{\mathcal{R}}$.

To define a complexity measure that works, we need a few auxiliary definitions. For a term s, let $|s|$ be the number of function symbols occurring in s. For an equation $s \doteq t$, define $|s \doteq t| := 0$ if s and t are variables and $|s \doteq t| := |s| + |t| - 1$ otherwise. For an equation system E, let $|E| := \sum_{P \in E} |P|$ and $\|E$ be the number of equations occurring in E. With that we define the complexity of an $\mathcal{R}$-triple as a triple of nonnegative integers:

$$|\theta.\ C.\ E| := (\|\theta E\|_{\mathcal{R}}, |E|, \|E).$$

On these complexities we obtain a well founded ordering "$|\theta.\ C.\ E| \geq |\theta'C'.\ E'|$" by extending the usual ordering on integers lexiographically.

Theorem 4.15. (Compatibility)

1. *If $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a resolution step, then $|\theta.\ C.\ E| > |\theta'C'.\ E'|$.*
2. *If $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a simplification step, then $|\theta.\ C.\ E| \geq |\theta'C'.\ E'|$.*

Proof. 1. Since application steps decrease $\|\theta E\|_{\mathcal{R}}$, and since blocking steps increase neither $\|\theta E\|_{\mathcal{R}}$ nor $|E|$, but decrease $\|E$, resolution steps decrease the complexity of a triple.

2. Let $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a simplification step. By proposition 4.12, we know that no simplification step increases $\|\theta E\|_{\mathcal{R}}$. Therefore, it suffices to show that if

a simplification step increases $|E|$, then it decreases $\|\theta E\|_{\mathcal{R}}$, and if a step simplification increases $\sharp E$, then it decreases $|E|$. The only rule that can increase $|E|$ is the rewriting rule, which does decrease $\|\theta E\|_{\mathcal{R}}$. The only rules that can increase $\sharp E$ are the unfolding and the decomposition rule, which do decrease $|E|$. □

Corollary 4.16. *The relation "$\theta \,.\, C \,.\, E \xrightarrow{sr}_{\mathcal{R},\mathcal{V}} \theta' \,.\, C' \,.\, E'$" is terminating.*

Corollary 4.17. *The solution procedure in Figure 4.1 is complete.*

The proof method we have developped in this and the last section can be used to show the completeness of alternative sets of resolution and simplification rules. Given such an alternative set of rules, the first step is to devise for every rule a suitable verification rule. The verification rules are applied to ground equation systems and must leave their validity invariant. The combination of the given rules with their corresponding verification rules then yields a reduction relation on triples. Next one defines a complexity measure on triples that is decreased by resolution steps and not increased by simplification steps. Then one shows with a push up theorem that every unsolved triple can be reduced by a resolution step on any given equation. Finally, one shows with a push down theorem that every triple can be reduced with any given simplification step.

If one uses an alternative set of resolution rules but the same complexity measure we used here, the simplification rules discussed here can be used without reproving anything. If the complexity measure is changed, it is still possible to reuse the push down theorem.

5 Refinements

In this section we discuss two refinements for the extended solution procedure. Both of them depend on additional knowledge about the underlying rewriting system.

5.1 Rewriting with Inductive Consequences

Let $\mathcal{R}$ be the rewriting system in Figure 1.1 and consider the equation $x + 0 \doteq 0$. Although this equation has the unique solution $x = 0$ in $\mathcal{R}_I$, which is easily found, the extended solution procedure nevertheless has an infinite search space for this equation. To see this, consider the derivation steps

$$
\begin{array}{lll}
 & \emptyset \,.\, x + 0 \doteq 0 & \\
\xrightarrow{r}_{\mathcal{R},\{x\}} & x + 0 \doteq s(x') + y' \,.\, s(x' + y') \doteq 0 & \text{by } A \\
\xrightarrow{s}_{\mathcal{R},\{x\}} & x \doteq s(x') \;\&\; y' \doteq 0 \,.\, s(x' + y') \doteq 0 & \text{by } \mathit{Uni},
\end{array}
$$

which can be continued infinitely often by applying rule (4) to the occurrence of $+$. The obtained pair is actually inconsistent, but our simplification and failure rules are too weak to detect this inconsistency.

We can get rid of this annoying problem if we add the rule $x + 0 \to x$ to the rewriting system. Then the extended solution procedure can find the solution of $x + 0 \doteq 0$ by using simplification steps only. Since the equation $x + 0 \doteq x$ is valid in the initial model of $\mathcal{R}$ and the extended rewriting system still terminates, adding this rule doesn't change the solutions of an equation. We will show that the solution procedure stays complete if the new rule is used for simplification with the rewriting rule but is not used for resolution with the application rule.

Two ground confluent and terminating rewriting systems are **equivalent** if they have the same signature and every ground term has the same normal form in both systems. Equivalent rewriting systems define, up to isomorphism, the same initial algebra.

Proposition 5.1. *Let $\mathcal{R}$ and $\mathcal{R}'$ be two equivalent ground confluent and terminating rewriting systems. Then an equation is valid in $\mathcal{R}$ if and only if it is valid in $\mathcal{R}'$.*

Proposition 5.2. *Let $\mathcal{R} = (\Sigma, \mathcal{E})$ be a ground confluent and terminating rewriting system and $s \to t$ be a rewriting rule that is an inductive consequence of $\mathcal{R}$. Then $\mathcal{R}' := (\Sigma, \mathcal{E} \cup \{s \to t\})$ is a ground confluent rewriting system. Furthermore, if $\mathcal{R}'$ is terminating, then $\mathcal{R}$ and $\mathcal{R}'$ are equivalent.*

Theorem 5.3. *Let $\mathcal{R} = (\Sigma, \mathcal{E})$ and $\mathcal{R}' = (\Sigma, \mathcal{E} \cup \mathcal{E}')$ be two equivalent ground confluent and terminating rewriting systems. Then the extended solution procedure in Figure 4.1 is complete if the rules in $\mathcal{E}$ are employed for resolution steps and the rules in $\mathcal{E} \cup \mathcal{E}'$ are employed for simplification steps.*

Proof. It suffices to show that the Push Up Theorem still holds if only the rules in $\mathcal{E}$ are available for application steps. This is the case since every ground term that can be rewritten with a rule in $\mathcal{E} \cup \mathcal{E}'$ can also be rewritten with a rule in $\mathcal{E}$. □

The idea to use inductive consequences for rewriting also appears in Fribourg [85].

5.2 Free Rewriting Systems

A ground confluent and terminating rewriting system $\mathcal{R}$ is called **free** if every function symbol that is reducible in $\mathcal{R}$ is completely defined in $\mathcal{R}$. Recall that a function symbol f is reducible in $\mathcal{R}$ if f is the top symbol of the left hand side of at least one rule of $\mathcal{R}$, and that f is completely defined in $\mathcal{R}$ if f occurs in no $\mathcal{R}$-value. The rewriting system $\mathcal{R}1$ in Subsection 4.3 is an example for a free rewriting system. The irreducible function symbols of a free rewriting system are often called **constructors**. Furthermore, a term is called **canonical** in $\mathcal{R}$ if it doesn't contain a function symbol that is reducible in $\mathcal{R}$.

Proposition 5.4. *Let $\mathcal{R}$ be a free rewriting system. Then a ground term is an $\mathcal{R}$-value if and only if it is canonical.*

The reason we discuss free rewriting systems here is that for these systems the number of don't know alternatives our solution procedure has to explore can be significantly reduced. Given a free rewriting system $\mathcal{R}$, we call a term $f(s_1, \ldots, s_n)$ **simple** in $\mathcal{R}$ if its top symbol f is reducible in $\mathcal{R}$ and its arguments $s_1, \ldots, s_n$ are canonical in $\mathcal{R}$. The solution procedure in Figure 5.1 restricts resolution steps to rule applications to don't care chosen simple subterms. To prove that this procedure is complete for free rewriting systems, we have to show two things. First, it must always be possible to simplify a pair $C \,.\, E$ such that the unsolved part contains only equations that contain at least one simple term. This is the case since an equation that doesn't contain a simple term contains only irreducible function symbols and can thus be blocked with the simplification rule *SB2*. Second, we need a stronger push up theorem:

Theorem 5.5. (Push Up for Free Rewriting Systems) *Let $\mathcal{R}$ be a free rewriting system. Then, if $\theta \,.\, C \,.\, E$ is an $\mathcal{R}$-triple, P is an equation in E, and P/π is a simple subterm*

solve($C.\ E$) is

1. choose **don't care** $C'.\ E'$ such that $C.\ E \xrightarrow{!}_{\mathcal{R},V}{}^{*} C'.\ E'$ by simplification steps and every equation in E' contains at least one simple term;
2. if a failure rule applies to $C'.\ E'$, then fail;
3. if E' is empty, then return C';
4. choose **don't care** an equation P in E' and a simple subterm P/π in P;
5. choose **don't know** $C''.\ E''$ such that $C'.\ E' \xrightarrow{!}_{\mathcal{R},V} C''.\ E''$ by an application step on P at π;
6. solve($C''.\ E''$)

Figure 5.1. A solution procedure for free rewriting system.

of P, there exists a triple $\theta'.\ C'.\ E'$ such that $\theta.\ C.\ E \xrightarrow{!}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by an application step on P at π.

Proof. Let $\theta.\ C.\ P\&E$ be an $\mathcal{R}$-triple and P/π be a simple subterm of P. Then $\theta(P/\pi)$ is an innermost ground term. Thus there exist a variant $u \to v$ of a rule of $\mathcal{R}$ and a substitution ϕ such that $\phi u = (\theta P)/\pi$. From here on the proof is identical with the proof of the push up theorem in Section 3. □

Corollary 5.6. *The solution procedure in Figure 5.1 is complete for free rewriting systems.*

Fribourg [85] discusses a similar solution procedure for free conditional rewriting systems. He has the additional requirement that the left hand sides of all rules be simple terms.

There is actually no need for reproving a stronger version of the push up theorem, since our simplification rules are already strong enough to justify the solution procedure for free rewriting systems. In fact, the solution procedure in Figure 5.1 just realizes one of the many strategies that one can obtain by using the unfolding rule in conjunction of the don't care selection of the next equation to be resolved upon. To see this, first notice that every equation that doesn't contain a simple term can be safely blocked with the simplification rule *SB1*. Secondly, any simple term s contained in an equation can be unfolded into an equation $x \doteq s$, which then can be chosen to be the next equation to be resolved upon. Blocking such an equation immediately yields an inconsistent pair, as we know by failure rule (3) since the top symbol of s is completely defined. Furthermore, any application step to a proper subterm s/π of s yields an inconsistent pair, as we know by failure rule (1) since the top symbol of s/π is irreducible, that is, is different from the top symbol of the left hand side of any rewriting rule. Thus we are left with exactly the don't know alternatives that are considered by the solution procedure for free rewriting systems.

The left-to-right basic narrowing strategy in [Herold 86] and the selection narrowing strategy in [Bosco et al. 87] are two further examples for the strategies that can be obtained by using the unfolding rule.

References

P.G. Bosco, E. Giovannetti, and C. Moiso, Refined Strategies for Semantic Unification. Proc. of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 1987, Springer LNCS 250, 276–290.

N. Dershowitz and D. Plaisted, Logic Programming cum Applicative Programming. Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 54–67.

M. Fay, First Order Unification in an Equational Theory. Proc. of the 4th Workshop on Automated Deduction, Austin 1979, 161–167.

L. Fribourg, SLOG: A Logic Programming Language Inerpreter Based on Clausal Superposition and Rewriting. Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 54–67.

J. Gallier and W. Snyder, A General Complete E-Unification Procedure. Proc. of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, May 1987, Springer LNCS 256, 216–227.

J.A. Goguen and J. Meseguer, Eqlog: Equality, Types, and Generic Modules for Logic Programming. In D. DeGroot and G. Lindstrom (ed.), Logic Programming, Functions, Relations, and Equations. Prentice Hall 1986, 179–210.

A. Herold, Narrowing Techniques Applied to Idempotent Unification. Seki Report SR-86-16, Universität Kaiserslautern, West Germany, 1986.

S. Hölldobler, A Unification Algorithm for Confluent Theories. Proc. of the 14th International Conference on Automata, Languages, and Programming, Karlsruhe, Germany, 1987, Springer LNCS 267, 31–41.

G. Huet, Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. Journal of the ACM 27,4 (1980), 797–821.

G. Huet and D.C. Oppen, Equations and Rewrite Rules: A Survey. In R. Book (ed.), Formal Languagues: Perspectives and Open Problems, Academic Press 1980, 349–405.

J.-M. Hullot, Canonical Forms and Unification. Proc. of the 5th International Conference on Automated Deduction, 1980, Les Arcs, France, Springer LNCS 87, 318–334.

H. Hußmann, Unification in Conditional Equational Theories. Proc. of the EUROCAL '85, Springer LNCS 204, 543–553.

A. Josephson and N. Dershowitz, An Implementation of Narrowing: The RITE Way. Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 187–197.

S. Kaplan, Fair Conditional Term Rewriting Systems: Unification, Termination, and Confluence. Technical Report no. 194, Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, 1984.

C. Kirchner, Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationelles, These d'état de l'Université de Nancy I, 1985.

D.S. Lankford, Canonical Inference, Technical Report ATP-32, Department of Mathematics and Computer Science, University of Texas at Austin, December 1975.

J.W. Lloyd, Foundations of Logic Programming. Springer Verlag, 1984.

A. Martelli, C. Moiso, and G.G. Rossi, An Algorithm for Unification in Equational Theories. Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 187–197.

P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne, NAROWER: A New Algorithm for Unification and its Application to Logic Programming. Proc. of the 1st International Conference on Term Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 141–157.

P. Réty, Improving Basic Narrowing Techniques. Proc. of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, May 1987, Springer LNCS 256, 216–227. Also presented at the 1st Workshop on Unification, Val d'Ajol, France, March 1987.

G.A. Robinson and L. Wos, Paramodulation and Theorem-Proving in First-Order Theories with Equality. Machine Intelligence 4, Edinburgh University Press, 1969, 135–150.

J.A. Robinson, A Machine-oriented Logic Based on the Resolution Principle. Journal of the ACM 12, 1965, 23–41.

J.R. Slagle, Automated Theorem Proving for Theories with Simplifiers, Commutativity, and Associativity. Journal of the Association of Computing Machinery, Vol. 21, No. 4, October 1974, 622–642.

G. Smolka and W. Nutt, Lazy Basic Order-Sorted Narrowing. Presented at the 1st Workshop on Unification, Val d'Ajol, France, March 1987.

G. Smolka, W. Nutt, J. Goguen, and J. Meseguer, Order-Sorted Equational Computation. Presented at the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, May 1987.

J.-H. You and P.A. Subrahmanyam, A Class of Confluent Term Rewriting Systems and Unification. Journal of Automated Reasoning 2, 1986, 319–418.

Chapitre 6

Approche équationnelle de la surréduction basique

Les travaux présentés ici poursuivent ceux du chapitre précédent en suivant deux lignes directrices.

1. Généraliser. Il s'agit de rendre notre méthode de surréduction utilisable dans plus de cas, en particulier dans le cadre de la logique ordo-sortée et/ou relativement à des systèmes de réécriture équationnelle. Dans ces deux cas la surréduction transforme un problème de résolution d'équations modulo un système de réécriture en un problème de résolution d'équations ordo-sorté et/ou modulo des équations, lesquels problèmes ne sont pas unitaires. Cela signifie que leurs solutions ne pourront pas être représentées par un système unique d'équations, mais par plusieurs, chacun d'eux décrivant un espace de solutions, et la réunion de ces espaces formant l'ensemble de toutes les solutions. D'où l'idée de considérer des disjonctions de systèmes d'équations, et pour cela d'introduire l'opérateur de disjonction $\vee$. Nous ne nous limiterons pas à la considération de disjonctions de systèmes d'équations, comme dans [44], mais prendrons d'une manière générale des expressions comportant des équations et les opérateurs $\wedge$ et $\vee$. L'introduction du $\vee$ permet par ailleurs de donner une formulation nouvelle à la surréduction, qui la fait apparaitre, contrairement au chapitre précédent, comme étant une régle "don't care". Cela permet de plus une certaine factorisation dans l'arbre de recherche.

2. Rendre la formulation complètement équationnelle, c'est à dire faire disparaitre la notion de substitution intermédiaire. La simplification par réécriture, telle qu'elle est présentée au chapitre précédent, fait apparaitre des substitutions. D'autre part, appliquer la règle de dépliage peut rendre la réécriture impossible. Par exemple l'équation $f(0) \doteq x$ peut se réécrire par la règle $f(0) \rightarrow 0$, mais mise sous la forme dépliée $f(y) \doteq x \wedge y \doteq 0$ cela n'est plus possible. Or nous avons vu que ces règles réduisent toutes deux l'espace de recherche, il faudrait donc qu'elles ne se contrarient pas. Dans la formulation présentée ici, la réécriture est opérationnelle et n'est plus contrariée par le dépliage.

Ce chapitre fournit un cadre formel qui permettra une extension facile à la logique ordo-sortée dans un premier temps, puis à la surréduction ordo-sortée équationnelle. Il fournit aussi une présentation équationnelle de la réécriture.

6.1 Préliminaires

Soit R un système de réécriture fixé pour tout le chapitre. On définit deux types d'équations et pour chacun d'eux un ensemble de solutions. En fait, seules les solutions normalisées nous intéressent.

1. on appelle équation modulo R ou équation en $\doteq_R$ tout objet de la forme $t \doteq_R t'$. L'ensemble de ses solutions noté $SOL(t \doteq_R t')$ est défini par:
   $SOL(t \doteq_R t') = \{\theta | \theta$ est normalisé et $\theta(t) =_R \theta(t')\}$.

2. on appelle équation syntaxique ou équation en $\doteq$ tout objet de la forme $t \doteq t'$. L'ensemble de ses solutions noté $SOL(t \doteq t')$ est défini par:
   $SOL(t \doteq t') = \{\theta | \theta$ est normalisé et $\theta(t) = \theta(t')\}$.

Soient deux symboles binaires $\wedge$, $\vee$ associatifs, commutatifs, et distributifs l'un sur l'autre. Les **unificandes** sont définis par:

- toute équation $t \doteq t'$ ou $t \doteq_R t'$ est un unificande,

- si S et S' sont des unificandes, alors $S \wedge S'$ et $S \vee S'$ sont des unificandes.

SOL est défini récursivement sur les unificandes par:

$$SOL(S \wedge S') = SOL(S) \cap SOL(S')$$

$$SOL(S \vee S') = SOL(S) \cup SOL(S')$$

La notation $SOL(S) \subseteq SOL(S')$ $[V]$ signifie que pour tout $\theta \in SOL(S)$ il existe $\theta' \in SOL(S')$ telle que $\forall x \in V$, $\theta x = \theta' x$.

Les unificandes peuvent être vus comme des termes, et on utilisera la notion d'occurrence d'un sous-unificande (ou d'une équation) dans un unificande. A ne pas confondre avec les occurrences d'un sous-terme dans équation apparaissant dans l'unificande.
Notation: $S[p \leftarrow S']$ représente l'unificande S dans lequel le sous-unificande à l'occurrence p a été remplacé par S'. S'il est inutile de préciser la valeur de p on écrira $S[S']$, avec la convention qu'au cours d'un même calcul ou dans une règle d'inférence la valeur de p ne change pas.

Lemme 6.1 (stabilité de SOL par plongement)

$$SOL(S') \subseteq SOL(S'') \Longrightarrow SOL(S[S']) \subseteq SOL(S[S''])$$

Preuve: Par récurrence structurelle sur S. Supposons que $SOL(S') \subseteq SOL(S'')$.

1. Si S est l'unificande vide le résultat est trivial.

2. Si $S = S_1 \wedge S_2$, supposons que l'adjonction de S' soit faite dans S_1. En appliquant l'hypothèse de récurrence il vient $SOL(S_1[S']) \subseteq SOL(S_1[S''])$. Or par définition, pour tous unificandes S'_1, S'_2 on a $SOL(S'_1 \wedge S'_2) = SOL(S'_1) \cap SOL(S'_2)$. L'intersection des ensembles étant croissante par rapport à l'inclusion, on a $SOL(S[S']) \subseteq SOL(S[S''])$.

3. Si $S = S_1 \vee S_2$, supposons que l'adjonction de S' est faite dans S_1. En appliquant l'hypothèse de récurrence il vient $SOL(S_1[S']) \subseteq SOL(S_1[S''])$. Or par définition, pour tous unificandes S'_1, S'_2 on a $SOL(S'_1 \vee S'_2) = SOL(S'_1) \cup SOL(S'_2)$. La réunion des ensembles étant croissante par rapport à l'inclusion, on a $SOL(S[S']) \subseteq SOL(S[S''])$. □

Définition 6.1 Un unificande S est dit **trivial** si toute substitution est solution, formellement si $SOL(S)$ est l'ensemble de toutes les substitutions normalisées. ◇

L'unificande trivial a pour ensemble complet d'unificateurs $\{Id\}$.

Remarque: Si θ est une substitution alors

$$\theta \in SOL(S) \Longleftrightarrow \theta(S) \text{ est trivial}$$

Dans la suite, nous prouvons que les règles d'inférences que nous donnons permettent de résoudre modulo R des équations. Pour cela il faut montrer que chaque règle est correcte et complète sur les variables de V, c'est à dire que pour toute règle, $S \longrightarrow S'$ implique $SOL(S) = SOL(S')$ $[V]$. Mais ce n'est pas suffisant car la règle qui transforme tout unificande en lui même est correcte et complète, mais ne permet pas de résoudre des équations modulo R. Il faut montrer en plus que pour toute solution θ, on pourra toujours atteindre, sous une hypothèse d'équité à définir, un unificande S_θ dont θ est solution syntaxique. Mesurons la "distance' qui sépare un unificande donné S de S_θ par une mesure de complexité c. Il suffit alors de montrer qu'au cours d'une dérivation équitable, la complexité décroit et finit par atteindre la valeur nulle. Pour tout unificande S on note par S_{dc} la forme disjonctive-conjonctive de S. Quand on passe de S à S_{dc} par distributivité, certaines équations de S peuvent se trouver multipliées. Appliquer une règle sur une équation de S revient donc à appliquer une ou plusieurs fois cette même règle sur S_{dc}. Si $\theta \in SOL(S)$, alors $\theta \in SOL(S_{dc})$, donc θS_{dc} est trivial. Il possède donc des facteurs conjonctifs triviaux. Soit $S \longrightarrow S'$ une étape obtenue en appliquant la règle r. Par complétude il existe $\theta' = \theta$ $[V]$ telle que $\theta' \in SOL(S')$. Nous montrerons que si θC est un facteur conjonctif trivial de θS_{dc} et qu'il est modifié lors de l'application de r, alors $\theta' S'_{dc}$ contient un facteur conjonctif trivial $\theta' C'$ tel que

- si r est la règle de surréduction, alors $c(\theta', C') < c(\theta, C)$.

- si r est une autre règle, alors $c(\theta', C') \leq c(\theta, C)$.

En résumé, pour chaque règle il faudra prouver trois arguments:

1. correction

2. complétude

3. décroissance stricte ou large (selon le cas) de la complexité.

La complexité définie ci-dessous est la même que celle du chapitre précédent. Nous avions indiqué pourquoi elle était compliquée.

Définition 6.2 1. Pour tout facteur conjonctif C on définit $\|C\|$ par:

- si C est trivial, $\|C\|$ est défini récursivement par:
  - $\|t \doteq t'\| = 0$
  - $\|t \doteq_R t'\|$ est la longueur maximale des dérivations de réécriture issues de $t \doteq_R t'$
  - $\|C_1 \wedge C_2\| = \|C_1\| + \|C_2\|$.

- si C n'est pas trivial, $\|C\| = 0$.

2. On définit $|C|$ par:
- $|t|$ est le nombre d'éléments de $O(t)$
- $|t \doteq t'| = 0$
- $|t \doteq_R t'| = 0$ si t et t' sont des variables, $|t \doteq_R t'| = |t| + |t'| - 1$ sinon
- $|C_1 + C_2| = |C_1| + |C_2|$.

3. Enfin $\#C$ désigne le nombre d'équations en $\doteq_R$ de C.

Etant donné une substitution θ et un facteur conjonctif C, nous définissons la mesure de complexité c par le triplet d'entiers naturels $c(\theta, C) = (\|\theta C\|, |C|, \#C)$. On obtient un ordre nœthérien sur cette mesure par extension lexicographique de la gauche vers la droite de l'ordre sur les naturels. ◇

Nous montrerons dans la suite que si $c(\theta, C) = (0,0,0)$ alors il est facile de calculer θ à partir de C.

6.2 Spécification de la surréduction

La règle ci-dessous prend en compte une équation en $\doteq_R$ et effectue en un seul coup les opérations suivantes. Elle transforme l'équation en une équation en $\doteq$, et calcule par ailleurs toutes les surréductions qui pourraient en être issues. Les différents unificandes obtenus sont intégrés dans un seul unificande grâce à l'opérateur $\vee$. Ainsi donc, elle conserve l'ensemble des solutions, et apparait comme une règle "don't care". Le choix de l'équation prise en compte est lui aussi "don't care".

Dans cette règle, l'unificande S dans lequel est plongé l'équation à surréduire apparait explicitement dans le but de considérer ses variables. Il faut en effet éviter d'introduire des variables qui se trouveraient déjà dans S, ce qui créerait un conflit.

(Surréduction)
$$\frac{S[t \doteq_R t']}{S\Big[\bigvee_{\substack{p \in O(t),\, g \to d \in R \\ V(g) \cap (V(S[t \doteq_R t']) \cup V) = \emptyset}} (g \doteq t[p \leftarrow d] \wedge t[p \leftarrow d] \doteq_R t') \quad \bigvee t \doteq t'\Big]}$$

La relation induite par cette règle est notée $\longrightarrow$. Remarquons que toute équation en $\doteq_R$ est réductible.

Remarque: Soit
$$R = \begin{array}{l} f(0) \to 0 \\ f(1) \to 1 \end{array}$$
et considérons un unificande S de la forme $S = e \wedge f(x) \doteq_R y$ où e est une équation et x, y des variables. En appliquant (Surréduction) on obtient
$$S' = e \wedge \Big((f(x) \doteq f(0) \wedge 0 \doteq_R y) \vee (f(x) \doteq f(1) \wedge 1 \doteq_R y) \vee f(x) \doteq y\Big)$$
Si le $\vee$ n'avait pas été introduit, il aurait fallu calculer, comme au chapitre précédent, trois unificandes:
$$\begin{array}{ll} S_1 = & e \wedge f(x) \doteq f(0) \wedge 0 \doteq_R y \\ S_2 = & e \wedge f(x) \doteq f(1) \wedge 1 \doteq_R y \\ S_3 = & e \wedge f(x) \doteq y \end{array}$$
$S_1 \vee S_2 \vee S_3$ est exactement la forme disjonctive-conjonctive de S'. Grâce au $\vee$, l'équation e est mise en facteur et les opérations qui devront être effectuées par la suite sur e le seront une fois au lieu de trois.

Proposition 6.2 *(correction)*
Si $S \longrightarrow S'$ alors $SOL(S') \subseteq SOL(S)$.

Preuve: Puisque (Surréduction) ne s'applique que sur les équations en $\doteq_R$, supposons que $S = (t \doteq_R t')$. Soit $\theta \in SOL(S')$, il existe au moins un facteur conjonctif de $\theta(S')$ qui soit trivial. Deux cas possibles:

1. Il est de la forme $\theta(g \doteq t[p \wedge t[p \leftarrow d] \doteq_R t')$, alors
$$\theta(t) = \theta(t[p \leftarrow t|p]) = \theta(t[p \leftarrow g]) =_R \theta(t')$$
2. Ou bien il est de la forme $\theta(t \doteq t')$, alors $\theta(t) = \theta(t')$.

D'où $\theta \in SOL(S)$. On en déduit le résultat grâce au lemme 6.1. □

Cherchons à prouver la complétude et la décroissance de la complexité. Commençons en travaillant sur une équation isolée. Nous introduisons ici un ensemble de variables protégées W, qui permettra par la suite de plonger l'équation dans un unificande sans avoir de conflit de variables (W sera l'ensemble des variables de l'unificande).

Lemme 6.3 Soit $t \doteq_R t'$ une équation et W un ensemble de variables. Soit $\theta \in SOL(t \doteq_R t')$ et considérons l'étape de surréduction $t \doteq_R t' \longrightarrow S'$.
Alors il existe $\theta' \in SOL(S')$ telle que
$$\theta' = \theta \ [V \cup W]$$
et $\theta' S'$ contient un facteur conjonctif trivial $\theta' C'$ tel que
$$c(\theta', C') < c(\theta, t \doteq_R t')$$

Preuve: Puisque la relation de réécriture est confluente, ou bien $\theta(t) = \theta(t')$, ou bien $\theta(t \doteq_R t')$ peut se réécrire.

1. Si $\theta(t) = \theta(t')$, en posant $\theta' = \theta$ on a $\theta' \in SOL(S')$. Par définition $c(\theta', t \doteq t') = (0,0,0)$. Par contre il existe deux entiers naturels i, j tels que $c(\theta, t \doteq_R t') = (i, j, 1)$. Donc $c(\theta', t \doteq t') < c(\theta, t \doteq_R t')$.

2. Si $\theta(t \doteq_R t')$ peut se réécrire, elle peut se réécrire par une stratégie de l'intérieur vers l'extérieur. Supposons que $\theta(t) \to_{[p, g \to d, \sigma]} (\theta t)[p \leftarrow \sigma(d)]$ par une telle stratégie, et on peut toujours supposer que $V(g) \cap (V \cup W) = \emptyset$. Posons alors
$$\theta' = \begin{array}{l} \sigma \ [V(g)] \\ \theta \ [V \cup W] \end{array}$$
Puisque θ est normalisé, $p \in O(t)$. L'unificande $g \doteq t[p \wedge t[p \leftarrow d] \doteq_R t'$ est donc un facteur conjonctif de S'. On a $\theta' g = \sigma g = \theta t|p = \theta' t|p$ et
$$(\theta' t)[p \leftarrow \theta' d] = (\theta t)[p \leftarrow \sigma d] =_R (\theta t)[p \leftarrow \sigma g] = \theta t =_R \theta t' = \theta' t'$$
De plus σ est normalisé à cause de la stratégie de réécriture choisie, ce qui implique que θ' est normalisé, et finalement $\theta' \in SOL(S')$. Par définition $\theta' = \theta \ [V \cup W]$. Nous savons donc que $\theta'(g \doteq t[p \wedge t[p \leftarrow d] \doteq_R t')$ est un facteur conjonctif trivial de $\theta' S'$ et $\theta'(t[p \leftarrow d]) = (\theta t)[p \leftarrow \sigma d]$ est issu de θt par réécriture. Par définition de la complexité, $c(\theta', t[p \leftarrow d] \doteq_R t') < c(\theta, t \doteq_R t')$. Puisque les équations en $\doteq$ n'influent pas la complexité, on a même $c(\theta', g \doteq t[p \wedge t[p \leftarrow d] \doteq_R t') < c(\theta, t \doteq_R t')$. □

Proposition 6.4 *(complétude)*
Soit $S \longrightarrow S'$ une étape de surréduction et $\theta \in SOL(S)$. Alors

$$SOL(S) \subseteq SOL(S') \ [V \cup V(S)]$$

Preuve: D'après le lemme précédent en prenant $W = V(S)$. □

Notation: Rappelons que S_{dc} est la forme disjonctive-conjonctive de l'unificande S et que $SOL(S_{dc}) = SOL(S)$. Si e est une équation de S à l'occurrence p (il s'agit d'une occurrence de sous-unificande dans un unificande), elle admet des résidus (défini au chapitre 3) à travers les étapes de distributivité faisant passer de S à S_{dc}. Soient $q_1, \ldots, q_n$ les résidus de p dans S_{dc}. Pour chaque $1 \leq i \leq n$ on a $S|_{q_i} = e$. Par abus de langage on dira que $q_1, \ldots, q_n$ sont les résidus de e dans S_{dc}.

Remarque: Il est facile de montrer par récurrence nœthérienne sur la distributivité que deux résidus de e dans S_{dc} ne peuvent appartenir au même facteur conjonctif de S_{dc}.

Lemme 6.5 Soit S un unificande, p l'occurrence d'une équation de S et $q_1, \ldots, q_n$ les résidus de p dans S_{dc}. Alors

1. $S \longrightarrow_{[p]} S'$ implique $S_{dc} \xrightarrow{*}_{[q_1,\ldots,q_n]} S''$ et $S''_{dc} = S'_{dc}$.
2. $S_{dc} \xrightarrow{*}_{[q_1,\ldots,q_n]} S''$ implique $S \longrightarrow_{[p]} S'$ et $S'_{dc} = S''_{dc}$.

Remarque: $\wedge$ et $\vee$ sont associatifs et commutatifs. Nous allons travailler sur les classes d'équivalence modulo l'associativité et la commutativité de $\wedge$ et $\vee$, en représentant les facteurs conjonctifs et disjonctifs sous forme aplatie. Ainsi les occurrences des équation d'un unificande en forme disjonctive-conjonctive sont des listes d'entiers de longueur deux. Le premier entier représente la position d'un facteur conjonctif dans l'unificande, le deuxième donne la position d'une équation dans ce facteur conjonctif. Par exemple dans $S = e_1 \vee (e_2 \wedge e_3 \wedge e_4) \vee e_5$ l'équation e_2 apparait à l'occurrence 2.1 .

Proposition 6.6 *(décroissance de la complexité)*
Soit S un unificande, $\theta \in SOL(S)$, et e une équation en $\doteq_R$ de S. Considérons l'étape de surréduction $S \longrightarrow S'$ effectuée sur e. D'après la proposition précédente nous savons qu'il existe $\theta' \in SOL(S')$ telle que $\theta' = \theta$ $[V]$.
Alors pour tout facteur conjonctif trivial $\theta S_{dc}|_i$ contenant un résidu q de e (c'est à dire $q = i.j$), l'unificande $\theta' S'_{dc}$ possède un facteur conjonctif trivial $\theta' C'$ tel que

$$c(\theta', C') < c(\theta, S_{dc}|_i)$$

Preuve: Par hypothèse $\theta S_{dc}|_q = \theta e$ est trivial, donc $\theta \in SOL(e)$.

Considérons l'étape de surréduction issue de S_{dc} à l'occurrence q:

$$S_{dc} \longrightarrow_{[q]} S_0 = S_{dc}[q \leftarrow S'_0]$$

D'après le lemme 6.3 il existe $\theta'_0 \in SOL(S'_0)$ tel que $\theta'_0 = \theta$ $[V \cup V(S_{dc})]$ et $\theta'_0 S'_0$ contient un facteur conjonctif trivial $\theta'_0 C'_0$ tel que $c(\theta'_0, C'_0) < c(\theta, e)$. Puisque $\theta'_0 = \theta$ $[V(S_{dc})]$ on a $\theta'_0 \in SOL(S_0)$. Puisque S'_0 a été obtenu par la règle de surréduction, il est en

forme disjonctive-conjonctive et s'écrit $S'_0 = \vee_k C_k \vee C'_0$. Rappelons que $q = i.j$ et étudions le facteur $S_0|_i$. Or $S_{dc}|_i$ est de la forme $S_{dc}|_i = \wedge_m e_m \wedge e$, donc $S_0|_i$ s'écrit:

$$S_0|_i = \wedge_m e_m \wedge S'_0 = \wedge_m e_m \wedge (\vee_k C_k \vee C'_0)$$

En appliquant la distributivité on voit que $(S_0)_{dc}$ contiendra le facteur conjonctif $C_0 = \wedge_m e_m \wedge C'_0$.

Montrons que $\theta'_0 C_0$ est trivial et $c(\theta'_0, C_0) < c(\theta, S_{dc}|_i)$. Rappelons que $S_{dc}|_i = \wedge_m e_m \wedge e$. Par hypothèse $\theta S_{dc}|_i$ est trivial, et puisqu'il est égal à $\theta'_0 S_{dc}|_i$ il résulte que $\theta'_0(\wedge_m e_m)$ est trivial. Nous savons de plus que $\theta'_0 C'_0$ est trivial, donc $\theta'_0 C_0$ est trivial.
Comparons maintenant $c(\theta, S_{dc}|_i)$ avec $c(\theta'_0, C_0)$.

$$c(\theta, S_{dc}|_i) = c(\theta, \wedge_m e_m \wedge e) = \sum_m c(\theta, e_m) + c(\theta, e) = \sum_m c(\theta'_0, e_m) + c(\theta, e)$$

$$c(\theta'_0, C_0) = c(\theta'_0, \wedge_m e_m \wedge C'_0) = \sum_m c(\theta'_0, e_m) + c(\theta'_0, C'_0)$$

Nous avons vu que $c(\theta'_0, C'_0) < c(\theta, e)$, donc $c(\theta'_0, C_0) < c(\theta, S_{dc}|_i)$.

Soient $q_1 = i_1.j_1, \ldots, q_n = i_n.j_n$ les autres résidus de e dans S_{dc}. D'après le lemme précédent

$$S_{dc} \xrightarrow{*}_{[q,q_1,\ldots,q_n]} S_n = S_{dc}[q \leftarrow S'_0][q_1 \leftarrow S'_0] \ldots [q_n \leftarrow S'_0]$$

avec $(S_n)_{dc} = S'_{dc}$. Puisque les occurrences $q, q_1, \ldots, q_n$ sont dans des facteurs conjonctifs de S_{dc} différents, S_n s'écrit

$$S_n = S_n|_i \vee S_n|_{i_1} \vee \ldots \vee S_n|_{i_n} \vee \ldots$$

Donc

$$S'_{dc} = (S_n)_{dc} = (S_n|_i)_{dc} \vee (S_n|_{i_1})_{dc} \vee \ldots \vee (S_n|_{i_n})_{dc} \vee \ldots$$

Puisque $(S_n|_i)_{dc} = (S_0)_{dc}$, l'unificande S'_{dc} contient le facteur conjonctif C_0. Comme de surcroit $\theta'_0 C_0$ est trivial, il vient $\theta'_0 \in SOL(S')$. Il suffit alors de poser $\theta' = \theta'_0$ et $C' = C_0$. □

Remarque: Nous avons écrit d'autres preuves qui ne passent pas par l'intermédiaire des formes disjonctives-conjonctives. Mais la définition de la complexité est alors plus compliquée et les preuves plus difficiles. Nous avons choisi de donner dans cette thèse la méthode de preuve la plus simple.

La définition qui suit caractérise les bonnes stratégies d'application de (Surréduction), c'est à dire celles qui nous donneront des ensembles complets de solutions. Une dérivation est équitable si aucune équation en $\doteq_R$ n'est délaissée indéfiniment.

Définition 6.3 (hypothèse d'équité)
Une dérivation $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ finie ou infinie est dite **équitable** si pour tout unificande S_i de la dérivation et pour toute équation $t \doteq_R t'$ en $\doteq_R$ de cet unificande, il existe $j \geq i$ tel que l'étape $S_j \longrightarrow S_{j+1}$ se fasse sur $t \doteq_R t'$. ◇

Lemme 6.7 Soit $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ une dérivation équitable et $\theta_0 \in SOL(S_0)$. Alors il existe un unificande S_n dans la dérivation, $\theta_n \in SOL(S_n)$ tels que $\theta_n = \theta$ $[V]$ et $\theta_n(S_n)_{dc}$ possède un facteur conjonctif trivial $\theta_n C_n$ tel que $c(\theta_n, C_n) = (0, 0, 0)$.

Preuve: Choisissons dans $\theta_0(S_0)_{dc}$ un facteur conjonctif trivial $\theta_0 C_0$. A chaque unificande S_j de la dérivation, on associe une substitution θ_j et un facteur conjonctif C_j de $(S_j)_{dc}$ tel que $\theta_j C_j$ soit trivial, par la définition récurrente:

- (θ_0, C_0) est associé à S_0,
- si (θ_i, C_i) est associé à S_i, considérons l'étape de (Surréduction) $S_i \longrightarrow_{[p_i]} S_{i+1}$. D'après la proposition 6.6 il existe une substitution $\theta_{i+1} \in SOL(S_{i+1})$ telle que $\theta_{i+1} = \theta_i \ [V]$ et $\theta_{i+1}(S_{i+1})_{dc}$ possède un facteur conjonctif trivial $\theta_{i+1}C_{i+1}$ tel que
  - si le facteur C_i de $(S_i)_{dc}$ contient un résidu de p, alors $c(\theta_{i+1}, C_{i+1}) < c(\theta_i, C_i)$.
  - s'il ne contient pas de résidu, alors $c(\theta_{i+1}, C_{i+1}) = c(\theta_i, C_i)$.

La suite des $(c(\theta_i, C_i))$ est décroissante. Montrons par l'absurde qu'elle atteint la valeur $(0,0,0)$.
Supposons que pour tout unificande S_j de la dérivation, on ait $c(\theta_j, C_j) > (0,0,0)$. Puisque la suite est décroissante, il existe un unificande S_k à partir duquel $c(\theta_k, C_k) > (0,0,0)$ ne change plus. Il en résulte que C_k n'est pas formé uniquement d'équations en $\doteq$, il a au moins une équation en $\doteq_R$, disons e, à l'occurrence q dans $(S_k)_{dc}$. Soit p l'antécédent de q dans S_k. Puisque la dérivation est équitable, il existe une étape de la dérivation portant sur cet antécédent $S_i \longrightarrow_{[p]} S_{i+1}$ avec $i \geq k$. Puisque q est un résidu de p, on a $c(\theta_i, C_i) < c(\theta_k, C_k)$, ce qui crée une contradiction. □

Lemme 6.8 Soit S un unificande, $\theta \in SOL(S)$, et S' l'unificande obtenu en remplaçant dans S toutes les équations en $\doteq_R$ par F. Si θS_{dc} a un facteur conjonctif trivial θC tel que $c(\theta C) = (0,0,0)$, alors $\theta \in SOL(S')$.

Preuve: θC ne contient que des équations triviales, donc les équations en $\doteq_R$ qu'il pourrait contenir auraient une complexité non nulle, et alors $c(\theta, C) > (0,0,0)$. Donc C contient seulement des équations en $\doteq$. Par conséquent C est aussi un facteur conjonctif de S'_{dc}, et puisque θC est trivial, $\theta \in SOL(S')$. □

Théorème 6.9 *Soit S_0 un unificande, $\theta_0 \in SOL(S_0)$, et $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ une dérivation équitable issue de S_0. Alors il existe un unificande S_i de cette dérivation et une substitution θ_i tel que*

- *$\theta_i \in SOL(S'_i)$ où S'_i est l'unificande obtenu à partir de S_i en remplaçant toutes les équations en $\doteq_R$ par F.*
- *$\theta_i = \theta_0 \ [V(S_0)]$*

Preuve: D'après les deux lemmes précédents en prenant $V = V(S_0)$. □

Ainsi le problème de l'unification modulo R est transformé en un problème d'unification syntaxique.

6.3 Spécification de la réécriture

Nous donnons dans ce paragraphe une spécification de la réécriture où n'apparait aucune substitution, et que le dépliage ne contrarie pas.

Pour que la réécriture et le dépliage ne se contrarient pas il faut prendre en compte le fait qu'une équation puisse-être sous forme dépliée, donc considérer son contexte, et replier juste ce qu'il faut pour qu'elle puisse se réécrire. Par exemple dans l'unificande

$$S = \ f(x) \doteq_R t \ \wedge \ x \doteq g(y) \ \wedge \ y \doteq h(z)$$

il faudra remplacer x par $g(y)$ dans l'équation $f(x) \doteq_R t$ pour qu'elle devienne réductible par la règle de réécriture $r = \ f(g(x')) \to x'$. La réductibilité d'une équation est étroitement liée à son contexte puisque dans

$$S' = \ f(x) \doteq_R t \ \wedge \ y \doteq h(z)$$

cette même équation $f(x) \doteq_R t$ n'est plus réductible par r.

Ce qui est présenté ici n'est pas à proprement parler la réécriture, mais la tentative de réécriture d'une équation par une règle de réécriture donnée et à une occurrence donnée. Ce processus comporte deux phases:

1. tentative de filtrage du membre gauche de la règle de réécriture vers le sous-terme considéré de l'équation
2. on a alors deux cas:
   - si le filtrage termine avec succès, remplacement du problème de filtrage par l'équation réécrite (sous une forme dépliée)
   - si le filtrage échoue, ce qui signifie que la réécriture n'est pas possible, remplacement du problème de filtrage par l'équation de départ.

Le filtrage est traité par décomposition et fusion. Mais le membre gauche d'une équation de filtrage $t_1 \ll t_2$ (filtrage de t_1 vers t_2) ne peut être considéré comme clos puisqu'il peut être instancié par le contexte, comme le montre l'exemple précédent. Les solutions d'une équation de filtrage ne peuvent pas être définies de manière intrinsèque, car elles dépendent du contexte considéré. Par exemple soit $S_1 = (f(x) \ll y)$, $S_2 = (y \doteq f(0))$ et $S = S_1 \wedge S_2$. Alors $SOL(S_1) = \emptyset$, $SOL(S_2) = (y/f(0))$ et on voudrait que $SOL(S) = (x/0, y/f(0))$, donc $SOL(S) \neq SOL(S_1) \cap SOL(S_2)$, ce qui est impossible par définition de SOL. Une solution consiste à intégrer le contexte comme paramètre de l'équation de filtrage.

Le filtrage est traité par décomposition et fusion des équations de filtrage. Lorsqu'il échoue, il faut restaurer l'équation que l'on tentait de réécrire. Pour garder une trace de cette équation une solution consiste à l'intégrer comme paramètre des équations de filtrage. Lorsque le filtrage réussit, il faut générer l'équation réécrite. Une solution consiste à garder trace, en plus de l'équation à réécrire, de l'occurrence de réduction et de la règle de réécriture utilisée, toujours sous forme de paramètres.

La décomposition d'une équation de filtrage la transforme en plusieurs équations de filtrage. Une tentative de filtrage donnée va donc se retrouver sous la forme d'une conjonction de plusieurs équations. Si on applique la distributivité du $\wedge$ sur le $\vee$, la tentative de filtrage peut se trouver dispersée dans tout l'unificande. Il devient ensuite difficile de la remplacer globalement par l'équation surréduite. Pour éviter cela nous introduisons un nouveau symbole FI qui contient toutes les équations de filtrage d'une même tentative de filtrage. Du fait des considération précédentes il contient aussi l'équation à réécrire, le contexte, l'occurrence de réduction, la règle de réécriture. Ainsi la tentative de réécriture du membre gauche de l'équation $t \doteq_R t'$, à l'occurrence p, par la règle $g \to d$, sous la contexte C est exprimée par l'unificande $FI^{t \doteq_R t', C}_{p, g \to d}(g \ll t|p)$.

Dans un contexte les équations en $\doteq$ et les équations en $\doteq_R$ sont traitées de la même manière. On ne prendra en compte en tant que contexte que des facteurs conjonctifs substituables, c'est à dire sous une forme résolue et n'ayant pas de cycles de variables.

Définition 6.4 On dit que le facteur conjonctif C est **substituable** si

- $C = (x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n) \wedge (x'_1 \doteq_R t'_1 \wedge \ldots \wedge x'_k \doteq_R t'_k)$
- soit $\phi = [x_1 \leftarrow t_1] \ldots [x_n \leftarrow t_n][x'_1 \leftarrow t'_1] \ldots [x'_k \leftarrow t'_k]$, il existe un entier naturel i tel que $\forall j > i, \phi^j = \phi^i$ (C ne contient pas de cycles de variables).

et on pose $< C >= \phi^i$.
$x_1, \ldots, x_n, x'_1, \ldots, x'_k$ sont appelées variables primaires de C. $\diamond$

Remarque: $< C >$ est la substitution obtenue en pliant C. Elle est idempotente. L'ensemble $D(< C >)$ est exactement composé des variables primaires de C.

6.3.1 Les règles

Les règles qui suivent spécifient une tentative de réécriture du membre gauche de l'équation $t \doteq_R t'$, à l'occurrence p, par la règle de réécriture $g \rightarrow d$, et avec le contexte C. Si la réécriture est possible le processus se termine par la règle (Renvoi), sinon l'équation $t \doteq_R t'$ est restaurée avec son contexte par une des règles suivantes: (Collision-$\ll$), (Echec-$\ll$), (Collision-$\cong$), (Echec-$\cong$). De cette manière chaque règle conserve l'ensemble des solutions. Le test qui vérifie que C est substituable n'a pas été explicité. Des algorithmes de détection des cycles de variables existent à ce jour, et pourront être intégrés par la suite.

Ne s'applique qu'en tête.

(Appel) $\dfrac{S[t \doteq_R t' \wedge C]}{S[FI^{t \doteq_R t', C}_{p,g \rightarrow d}(g \ll t|p)]}$ si $\begin{cases} C \text{ est un facteur conjonctif substituable} \\ g \rightarrow d \text{ est une règle de réécriture de } R \\ V(g) \cap (V(S[t \doteq_R t' \wedge C]) \cup V) = \emptyset \end{cases}$

(Décomposition-$\ll$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_n) \wedge C')}{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s_1 \ll t_1 \wedge \ldots \wedge s_n \ll t_n \wedge C')}$

(Collision-$\ll$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(\ldots) \ll g(\ldots) \wedge C')}{t \doteq_R t' \wedge C}$ si $f \neq g$

(Fusion-$< C >$-sur-$\ll$-1) $\dfrac{FI^{t \doteq_R t', C \wedge x \doteq t}_{p,g \rightarrow d}(s \ll x \wedge C')}{FI^{t \doteq_R t', C \wedge x \doteq t}_{p,g \rightarrow d}(s \ll t \wedge C')}$ si s n'est pas une variable

(Fusion-$< C >$-sur-$\ll$-2) $\dfrac{FI^{t \doteq_R t', C \wedge x \doteq_R t}_{p,g \rightarrow d}(s \ll x \wedge C')}{FI^{t \doteq_R t', C \wedge x \doteq_R t}_{p,g \rightarrow d}(s \ll t \wedge C')}$ si s n'est pas une variable

(Echec-$\ll$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s \ll x \wedge C')}{t \doteq_R t' \wedge C}$ si $\begin{cases} s \text{ n'est pas une variable} \\ C \text{ ne contient pas d'équation } x \doteq s \text{ ou } x \doteq_R s \end{cases}$

(Non-linéarité) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \ll t_1 \wedge x \ll t_2 \wedge C')}{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \ll t_1 \wedge t_1 \cong t_2 \wedge C')}$

(Décomposition-$\cong$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(s_1, \ldots, s_n) \cong f(t_1, \ldots, t_n) \wedge C')}{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s_1 \cong t_1 \wedge \ldots \wedge s_n \cong t_n \wedge C')}$

(Collision-$\cong$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(\ldots) \cong g(\ldots) \wedge C')}{t \doteq_R t' \wedge C}$ si $f \neq g$

Les règles suivantes seront appliquées en utilisant le fait que $\cong$ est commutatif.

(Fusion-$< C >$-sur-$\cong$-1) $\dfrac{FI^{t \doteq_R t', C \wedge x \doteq t}_{p,g \rightarrow d}(s \cong x \wedge C')}{FI^{t \doteq_R t', C \wedge x \doteq t}_{p,g \rightarrow d}(s \cong t \wedge C')}$ si $s \neq x$

(Fusion-$< C >$-sur-$\cong$-2) $\dfrac{FI^{t \doteq_R t', C \wedge x \doteq_R t}_{p,g \rightarrow d}(s \cong x \wedge C')}{FI^{t \doteq_R t', C \wedge x \doteq_R t}_{p,g \rightarrow d}(s \cong t \wedge C')}$ si $s \neq x$

(Elimination-$\cong$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \cong x \wedge C')}{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(C')}$

(Echec-$\cong$) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s \cong x \wedge C')}{t \doteq_R t' \wedge C}$ si $\begin{cases} s \text{ n'est pas une variable} \\ C \text{ ne contient pas d'équation } x \doteq s' \text{ et } x \doteq_R s' \end{cases}$

(Renvoi) $\dfrac{FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x_1 \ll t_1 \wedge \ldots \wedge x_n \ll t_n)}{\bigwedge_{x_i \in V(d)} x_i \doteq_R t_i \bigwedge t[p \leftarrow d] \doteq_R t' \bigwedge C}$ si $\begin{cases} FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x_1 \ll t_1 \wedge \ldots \wedge x_n \ll t_n) \\ \text{est irréductible par (Non-linéarité)} \end{cases}$

Dans la surréduction basique telle que l'a définie Hullot, les équations sont partagées en occurrences basiques et en occurrences protégées. Les équations en $\doteq_R$ de notre spécification correspondent aux occurrences basiques et les équations en $\doteq$ aux occurrences protégées. La réécriture que nous venons de spécifier, considère que toutes les occurrences des termes à réécrire sont basiques, même si ce n'est pas vrai. Ce n'est donc pas optimal. La règle présentée ici comble cette lacune. Elle correspond à la règle "$SB1$" du chapitre précédent.

(Basification-de-réécriture) $\dfrac{x \doteq_R t \wedge y \doteq s}{x \doteq_R t[v \leftarrow y] \wedge y \doteq s}$ si $t|v = s$

En n'appliquant pas (Basification-de-réécriture) de façon systématique, mais de façon adéquate, il est possible de spécifier la réécriture faiblement basique du chapitre 3.

6.3.2 Les preuves

Définissons les solutions et la complexité des équations en FI.

Définition 6.5 Posons $A = SOL(FI^{t \doteq_R t', C}_{p,g \rightarrow d}(\bigwedge_{i \in I} s_i \ll t_i \bigwedge_{j \in J} s'_j \cong t'_j))$. L'ensemble $SOL(A)$ des solutions de A est défini par

1. Si
   - il existe une substitution σ telle que $\forall i \in I,\ \sigma s_i =< C > t_i$
   - $\forall j \in J,\ < C > s'_j =< C > t'_j$

   alors $SOL(A) = SOL(t[p \leftarrow d] \doteq_R t' \wedge C \bigwedge_{x \in V(d)} x \doteq_R \sigma x)$.
2. Sinon $SOL(A) = SOL(t \doteq_R t' \wedge C)$.

Soit θ une substitution, la complexité $c(\theta, A)$ est définie par

1. Si
   - il existe une substitution σ telle que $\forall i \in I,\ \sigma s_i =< C > t_i$
   - $\forall j \in J,\ < C > s'_j =< C > t'_j$
   alors $c(\theta, A) = c(\theta, t[p \leftarrow d] \doteq_R t' \ \wedge\ C \bigwedge_{x \in V(d)} x \doteq_R \sigma x)$.

2. Sinon $c(\theta, A) = c(\theta, t \doteq_R t' \ \wedge\ C)$.

$\diamond$

Le lemme suivant permet de justifier cette définition.

Lemme 6.10 Pour tout unificande $FI^{t \doteq_R t', C}_{p,g \rightarrow d}(\bigwedge_i s_i \ll t_i \ \bigwedge_j s'_j \cong t'_j)$ issu de l'application de (Appel) suivie éventuellement de l'application d'autres règles, on a $\cup_i V(s_i) = V(g)$.

Preuve: Il est facile de vérifier que les règles qui transforment FI conservent les variables qui sont à gauche des $\ll$. La règle (Appel) a forcément été utilisée pour créer un unificande du type $FI^{t \doteq_R t', C}_{p,g \rightarrow d}(\ldots)$, d'où le résultat. $\square$

La définition précédente est correcte car $\sigma|_{V(d)}$ est unique. En effet $V(d) \subseteq V(g)$ et d'après ce lemme $V(g) = \cup_{i \in I} V(s_i)$. Or il est bien clair que $\sigma|_{V(\cup_{i \in I} V(s_i))}$ est unique.

Avec cette définition des ensemble de solutions, il est facile de prouver que les règles que nous avons données sont correctes et complètes, c'est à dire qu'elles laissent inchangé l'ensemble des solutions. Seule les preuves de (Appel) et (Renvoi) posent quelques difficultés. D'autre part comme nous montrons que l'expression de l'ensemble des solutions est inchangée à chaque étape, sauf pour (Appel) et (Renvoi), il est évident que la complexité est inchangée. Nous détaillerons la preuve seulement pour ces deux règles.

Il est d'abord nécéssaire d'établir quelques propriétés sur les substitutions définies par les facteurs substituables

Lemme 6.11 Soit C un unificande substituable. Pour toute équation $x \doteq t$ ou $x \doteq_R t$ de C on a $< C > x =< C > t$.

Preuve: ϕ et i étant ceux de la définition précédente, on a

$$< C > x = \phi^{i+1} x = \phi^i(\phi x) = \phi^i(t) =< C > t$$

$\square$

Lemme 6.12 Si C est substituable et $\theta \in SOL(C)$, alors pour toute variable x on a

$$\theta < C > x =_R \theta x$$

Preuve: ϕ est définie comme dans la définition précédente. Montrons par récurrence que pour tout k on a

$$\theta \phi^k x =_R \theta x \ \text{ pour toute variable x}$$

1. $k = 1$. Soit une variable x quelconque. Si x n'est pas primaire alors $\phi x = x$, d'où $\theta \phi x = \theta x$.
Si x est primaire:

- si C contient l'équation $x \doteq t$, alors par définition $\phi x = t$. Or $\theta x = \theta t$, d'où $\theta \phi x = \theta t = \theta x$.
- sinon C contient l'équation $x \doteq_R t$, alors par définition $\phi x = t$. Or $\theta x =_R \theta t$, d'où $\theta \phi x = \theta t =_R \theta x$.

2. Supposons que pour tout x on ait $\theta \phi^k x =_R \theta x$. Soit une variable y quelconque. Alors

$$\begin{array}{lll} \theta \phi^{k+1} y = \theta \phi^k(\phi y) & =_R \theta \phi y & \text{par hypothèse de récurrence} \\ & =_R \theta x & \text{d'après le cas 1.} \end{array}$$

On en déduit la propriété sur $< C >$. $\square$

Proposition 6.13 *Si $S \longrightarrow S'$ par une des règles précédentes, alors $SOL(S) = SOL(S')$ sur les variables de V. Ainsi pour tout $\theta \in SOL(S)$, il existe $\theta' \in SOL(S')$ avec $\theta' = \theta\ [V]$, et de plus $c(\theta', S') \leq c(\theta, S)$.*

Preuve: (Appel)
Supposons qu'il existe σ telle que $\sigma g =< C > t|p$ et montrons que $SOL(t \doteq_R t' \wedge C) = SOL(t[p \leftarrow d] \doteq_R t' \ \wedge\ C \bigwedge_{x \in V(d)} x \doteq_R \sigma x)$.

1. Correction. Soit $\theta \in \cap_{x \in V(d)} SOL(x \doteq_R \sigma(x)) \cap SOL(t[p \leftarrow d] \doteq_R t') \cap SOL(C)$. Montrons que $\theta \in SOL(t \doteq_R t') \cap SOL(C)$.

$$\theta t = \theta t[p \leftarrow \theta t|p] =_R \theta t[p \leftarrow \theta < C > t|p] \ \text{ d'après le lemme 6.12}$$

$$= \theta t[p \leftarrow \theta \sigma g] =_R \theta t[p \leftarrow \theta \sigma d] = \theta t[p \leftarrow \theta d] \ \text{ car } \theta \in \cap_{x \in V(d)} SOL(x \doteq_R \sigma(x))$$

$$= \theta(t[p \leftarrow d]) =_R \theta t' \ \text{ car } \theta \in SOL(t[p \leftarrow d] \doteq_R t')$$

D'autre part il est évident que $\theta \in SOL(C)$.

2. Complétude. Soit $\theta \in SOL(t \doteq_R t' \ \wedge\ C)$. Posons

$$\begin{array}{ll} \theta' & = \theta\ [V(S[t \doteq_R t' \ \wedge\ C]) \cup V] \\ & = (\theta.\sigma)\downarrow\ [V(g)] \end{array}$$

Soit $x \in V(d)$, on a

$$\begin{array}{rcl} \theta' x & = & (\theta \sigma x)\downarrow \\ \theta'(\sigma x) & = & \theta(\sigma x) \ \text{ car } V(\sigma(x)) \subseteq V(t \doteq_R t' \ \wedge\ C_1) \end{array}$$

D'où $\theta' \in SOL(x \doteq_R \sigma(x))$. Par conséquent $\theta' \in \cap_{x \in V(d)} SOL(x \doteq_R \sigma x)$.
Par ailleurs

$$\theta' t[p \leftarrow \theta' d] = \theta t[p \leftarrow (\theta \sigma)\downarrow d] =_R \theta t[p \leftarrow \theta \sigma d] =_R \theta t[p \leftarrow \theta \sigma g]$$

$$= \theta t[p \leftarrow \theta < C > t|p] =_R \theta t[p \leftarrow \theta t|p] \ \text{ d'après le lemme 6.12}$$

$$= \theta t =_R \theta t' = \theta' t'$$

Donc $\theta' \in SOL(t[p \leftarrow d] \doteq_R t')$. D'autre part il est évident que $\theta' \in SOL(C)$.

3. Complexité. Soit

$$k_1 = \| \wedge_{x \in V(d)} \theta' x \doteq_R \theta' \sigma x \| = \sum_{x \in V(d)} \|\theta \sigma x\| \ \text{ car } \theta' \text{ est normalisé}$$

Posons $k = \|\theta t[p \leftarrow (\theta\sigma)d] \doteq_R \theta t'\|$. Alors

$$\|\theta' t[p \leftarrow \theta' d] \doteq_R \theta' t'\| = \|\theta t[p \leftarrow (\theta\sigma)\downarrow d] \doteq_R \theta t'\| \leq k - k_1$$

Donc

$$\|\theta'(\wedge_{x \in V(d)} x \doteq_R \sigma(x) \ \wedge \ t[p \leftarrow d] \doteq_R t')\| \leq k_1 + k - k_1 = k$$

Or on a vu dans 2. que

$$\theta t = \theta t[p \leftarrow \theta\sigma g] \rightarrow_{[g \rightarrow d]} \theta t[p \leftarrow \theta\sigma d]$$

Donc $k < \|\theta(t \doteq_R t')\|$, et par conséquent dans le cas non trivial où nous nous sommes placé, (Appel) fait décroitre strictement la complexité.

(Décomposition-$\ll$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_n) \ \wedge \ C')$$

$$B = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s_1 \ll t_1 \ \wedge \ \ldots \ \wedge \ s_n \ll t_n \ \wedge \ C')$$

Or pour toute substitution σ on a

$$\sigma f(s_1, \ldots, s_n) =< C > f(t_1, \ldots, t_n) \Longleftrightarrow \sigma s_1 =< C > t_1 \text{ et } \ldots \text{ et } \sigma s_n =< C > t_n$$

Donc $SOL(A) = SOL(B)$.

(Collision-$\ll$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(\ldots) \ll g(\ldots) \ \wedge \ C')$$

avec $f \neq g$

$$B = \ t \doteq_R t' \ \wedge \ C$$

Il n'existe pas de σ telle que $\sigma f(\ldots) =< C > g(\ldots)$. Donc $SOL(A) = SOL(B)$.

(Fusion-$< C >$-sur-$\ll$-1)
Posons

$$A = FI^{t \doteq_R t', C'' \wedge x \doteq t}_{p,g \rightarrow d}(s \ll x \ \wedge \ C')$$

où s n'est pas une variable.

$$B = FI^{t \doteq_R t', C'' \wedge x \doteq t}_{p,g \rightarrow d}(s \ll t \ \wedge \ C')$$

et $C = C'' \ \wedge \ x \doteq t$. D'après le lemme 6.12, $< C > x =< C > t$. Donc

$$\sigma x =< C > x \Longleftrightarrow \sigma x =< C > t$$

D'où $SOL(A) = SOL(B)$.

(Fusion-$< C >$-sur-$\ll$-2)
Même chose.

(Echec-$\ll$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s \ll x \ \wedge \ C')$$

en supposant que s n'est pas une variable et que C ne contient pas d'équation du type $x \doteq s$ ou $x \doteq_R s$.

$$B = \ t \doteq_R t' \ \wedge \ C$$

On en déduit que $x \notin D(< C >)$, donc $< C > x = x$. Donc s n'est pas filtrable vers $< C > x$. D'où $SOL(A) = SOL(B)$.

(Non-linéarité)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \ll t_1 \ \wedge \ x \ll t_2 \ \wedge \ C')$$

$$B = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \ll t_1 \ \wedge \ t_1 \cong t_2 \ \wedge \ C')$$

Alors

$$\sigma x =< C > t_1 \text{ et } \sigma x =< C > t_2 \Longleftrightarrow \sigma x =< C > t_1 \text{ et } \ < C > t_1 =< C > t_2$$

Donc $SOL(A) = SOL(B)$.

(Décomposition-$\cong$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(s_1, \ldots, s_n) \cong f(t_1, \ldots, t_n) \ \wedge \ C')$$

$$B = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(s_1 \cong t_1 \ \wedge \ \ldots \ \wedge \ s_n \cong t_n \ \wedge \ C')$$

Alors

$$< C > f(s_1, .., s_n) =< C > f(t_1, \ldots, t_n) \Longleftrightarrow < C > s_1 =< C > t_1 et \ldots et < C > s_n =< C > t_n$$

Donc $SOL(A) = SOL(B)$.

(Collision-$\cong$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(f(\ldots) \cong g(\ldots) \ \wedge \ C')$$

avec $f \neq g$

$$B = \ t \doteq_R t' \ \wedge \ C$$

Or $< C > f(\ldots) \neq < C > g(\ldots)$, donc $SOL(A) = SOL(B)$.

(Fusion-$< C >$-sur-$\cong$1
Posons

$$A = FI^{t \doteq_R t', C'' \wedge x \doteq t}_{p,g \rightarrow d}(s \cong x \ \wedge \ C')$$

$$B = FI^{t \doteq_R t', C'' \wedge x \doteq t}_{p,g \rightarrow d}(s \cong t \ \wedge \ C')$$

et $C = C'' \ \wedge \ x \doteq t$. D'après le lemme 6.11, $< C > x =< C > t$. Donc

$$< C > s =< C > x \Longleftrightarrow < C > s =< C > t$$

D'où $SOL(A) = SOL(B)$.

(Application-$< C >$-sur-$\cong$1
Même chose.

Elimination-$\cong$
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(x \cong x \ \wedge \ C')$$

$$B = FI^{t \doteq_R t', C}_{p,g \rightarrow d}(C')$$

Le résultat est évident.

(Echec$\cong$)
Posons

$$A = FI^{t \doteq_R t', C}_{p,g \to d}(s \cong x \wedge C')$$

$$B = t \doteq_R t' \wedge C$$

et supposons que s n'est pas une variable et C ne contient pas d'équation du type $x \doteq s'$ ou $x \doteq_R s'$. On en déduit que $x \notin D(< C >)$, donc $< C > s \neq < C > x$. D'où $SOL(A) = SOL(B)$.

(Renvoi)
1. Posons

$$A = FI^{t \doteq_R t', C}_{p,g \to d}(x_1 \ll t_1 \wedge \ldots \wedge x_n \ll t_n)$$

$$B = \bigwedge_{x_i \in V(d)} x_i \doteq_R t_i \ \wedge t[p \leftarrow d] \doteq_R t' \wedge C$$

et supposons que A est irréductible par (Non-linéarité). Donc $\forall i, j \in \{1, \ldots, n\}$, $i \neq j \Longrightarrow x_i \neq x_j$. Il existe donc une substitution σ telle que

$$\sigma x_1 =< C > t_1 \text{ et} \ldots \text{et } \sigma x_n =< C > t_n$$

D'où par définition

$$SOL(A) = SOL(t[p \leftarrow d] \doteq_R t' \wedge C \bigwedge_{x_i \in V(d)} x_i \doteq_R < C > t_i)$$

Soit $\theta \in SOL(C)$ et $y \in V(t_i)$. D'après le lemme 6.12, $\theta y =_R \theta < C > y$. Donc $\theta t_i =_R \theta < C > t_i$. D'où

$$\forall \theta \in SOL(C),\ (\theta x =_R \theta t_i \Longleftrightarrow \theta x =_R \theta < C > t_i)$$

Il en résulte $SOL(A) = SOL(B)$.

2. Complexité. Soit θ une substitution. Par définition

$$c(\theta, A) = c(\theta, \bigwedge_{x_i \in V(d)} x_i \doteq_R < C > t_i \ \wedge t[p \leftarrow d] \doteq_R t' \wedge C$$

$$c(\theta, B) = c(\theta, \bigwedge_{x_i \in V(d)} x_i \doteq_R t_i \ \wedge t[p \leftarrow d] \doteq_R t' \wedge C$$

Or $\|t_i\| \leq \| < C > t_i \|$, $|t_i| \leq | < C > t_i |$ et il y a le même nombre d'équations en $\doteq_R$. Par conséquent $c(\theta, B) \leq c(\theta, A)$.

(Basification-de-réécriture)
1. Correction. Soit $\theta \in SOL(x \doteq_R t[v \leftarrow y] \wedge y \doteq s)$. Alors

$$\theta x =_R \theta t[v \leftarrow \theta y] = \theta t[v \leftarrow \theta s] = \theta t[v \leftarrow \theta t[v]] = \theta t$$

2. Complétude. Soit $\theta \in SOL(x \doteq_R t \wedge y \doteq s)$. Alors

$$\theta x =_R \theta t = \theta t[v \leftarrow \theta t[v]] = \theta t[v \leftarrow \theta s] = \theta t[v \leftarrow \theta y] = \theta(t[v \leftarrow y])$$

3. Complexité. Cette règle n'augmente pas le nombre maximal d'étapes de réécriture, ni le nombre de symboles de fonction, ni le nombre d'équations. Elle n'augmente donc pas la complexité. □

Pour les besoins de la preuve introduisons ici une règle de mise en forme disjonctive-conjonctive:

(Distributivité) $\dfrac{S_1 \wedge (S_2 \vee S_3)}{(S_1 \wedge S_2) \vee (S_1 \wedge S_3)}$

Toutes les règles qui ont été introduites ici, hormis (Distributivité), modifient des équations ou des facteurs conjonctifs. Or un facteur conjonctif donné d'un unificande n'est pas détruit ou séparé lors de la mise en forme disjonctive-conjonctive, mais simplement recopié en un ou plusieurs exemplaires. Formellement, considérons (Distributivité) comme une règle de réécriture, les unificandes qu'elle contient étant vus comme des variables. Il est connu que le système de réécriture formé de cette seule règle est confluent et nœthérien, et qu'une expression est irréductible si et seulement si elle est en forme disjonctive-conjonctive. Pour mettre un unificande en forme disjonctive-conjonctive, il suffit donc de lui appliquer (Distributivité) en suivant n'importe qu'elle stratégie et jusqu'à obtenir un unificande irréductible. Ceci étant, si on l'applique de manière à ne pas couper les facteurs conjonctifs, les occurrences d'équation ou de conjonction d'équation admettent des résidus à travers cette réécriture. Par exemple soit $S = (e_1 \wedge e_2 \wedge (e_3 \vee e_4)$. Si on réduit $S \longrightarrow_{[\text{Distributivité}]} e_1 \wedge ((e_2 \wedge e_3) \vee (e_2 \wedge e_4))$, le facteur $e_1 \wedge e_2$ de S a été divisé, tandis que dans $S \longrightarrow_{[\text{Distributivité}]} (e_1 \wedge e_2 \wedge e_3) \vee (e_1 \wedge e_2 \wedge e_4)$ il ne l'est pas et admet deux antécédents. D'où le lemme qui généralise le lemme 6.5.

Lemme 6.14 Soit S un unificande, $p \in D(S)$ l'occurrence d'une conjonction d'équations de S, et $p_1, \ldots, p_n$ les résidus de p dans S_{dc}. Alors

$$S \longrightarrow_{[p]} S' \Longrightarrow S_{dc} \longrightarrow_{[p_1]} \ldots \longrightarrow_{[p_n]} S''$$

et $S''_{dc} = S'_{dc}$.

Définition 6.6 Soit $D = S_0 \longrightarrow_{[p_0]} S_1 \longrightarrow_{[p_1]} \ldots \longrightarrow_{[p_{n-1}]} S_n \longrightarrow \ldots$ une dérivation. A chaque étape $S_i \longrightarrow_{[p_i]} S_{i+1}$ de D on associe grâce au lemme précédent la dérivation $(S_i)_{dc} \xrightarrow{*}_{[p_{i,1}, \ldots, p_{i,k}]} S'_{i+1} \xrightarrow{*}_{[\text{Distributivité}]} (S_{i+1})_{dc}$ où $p_{i,1}, \ldots, p_{i,k}$ sont les résidus de p_i dans $(S_i)_{dc}$.
On appelle dérivation sur les formes disjonctives-conjonctives de D la dérivation $D_{dc} = (S_0)_{dc} \xrightarrow{*} (S_1)_{dc} \xrightarrow{*} \ldots$. Soit C_0 une disjonction de facteurs conjonctifs de $(S_0)_{dc}$. A chaque unificande S_i de la dérivation D on associe un élément C_i par la définition récurrente suivante:

- C_0 est associé à S_0
- si C_i est associé à S_i alors si p_i admets des résidus $q_1, \ldots, q_n$ dans C_i alors C_{i+1} est défini par $C_i \xrightarrow{*}_{[q_1, \ldots, q_n]} C'$ et $C_{i+1} = C'_{dc}$; sinon $C_{i+1} = C_i$ et on dit que C_{i+1} est obtenu par une **étape blanche**.

La dérivation $C_0 \longrightarrow C_1 \longrightarrow \ldots \longrightarrow C_i \longrightarrow \ldots$ est appelée sous-dérivation issue de C_0 sur les formes disjonctives-conjonctives de D. ◇

Définition 6.7 (hypothèse d'équité)
Une dérivation qui utilise les règles présentées jusqu'ici $D = S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ est dite **équitable** si pour tout unificande S_i de la dérivation et pour tout facteur conjonctif C_i de S_i, il existe un élément C' dans la sous-dérivation issue de C_i sur les formes disjonctives-conjonctives de D, et obtenu par des étapes blanches ou de simplification, tel que:

- ou bien la sous-dérivation issue de C' contient une première étape qui est une étape de (Surréduction),

- ou bien C' ne contient que des équations en $\doteq$.

$\diamond$

Lemme 6.15 Soit $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ une dérivation équitable et $\theta_0 \in SOL(S_0)$. Alors il existe un unifiande S_n dans la dérivation, $\theta_n \in SOL(S_n)$ tels que $\theta_n = \theta \ [V]$ et $\theta_n(S_n)_{dc}$ possède un facteur conjonctif trivial $\theta_n C_n$ tel que $c(\theta_n, C_n) = (0,0,0)$.

Preuve: Choisissons dans $\theta_0(S_0)_{dc}$ un facteur conjonctif trivial $\theta_0 C_0$. A chaque unifiande S_j de la dérivation, on associe une substitution θ_j et un facteur conjonctif C_j de $(S_j)_{dc}$ tel que $\theta_j C_j$ soit trivial, par la définition récurrente:

- (θ_0, C_0) est associé à S_0,
- si (θ_i, C_i) est associé à S_i, considérons l'étape $S_i \longrightarrow_{[p,i]} S_{i+1}$. S'il s'agit d'une étape de (Surréduction), d'après la proposition 6.6 il existe une substitution $\theta_{i+1} \in SOL(S_{i+1})$ telle que $\theta_{i+1} = \theta_i \ [V \cup V(S_i)]$ et $\theta_{i+1}(S_{i+1})_{dc}$ possède un facteur conjonctif trivial $\theta_{i+1}C_{i+1}$ tel que
  - si le facteur C_i de $(S_i)_{dc}$ contient un résidu de p, alors $c(\theta_{i+1}, C_{i+1}) < c(\theta_i, C_i)$.
  - s'il ne contient pas de résidu, alors il suffit de prendre $C_{i+1} = C_i$, d'où $c(\theta_{i+1}, C_{i+1}) = c(\theta_i, C_i)$.

  S'il s'agit d'une étape de simplification, d'après les résultats sur les règles de simplification il existe $\theta_{i+1} \in SOL(S_{i+1})$ telle que $\theta_{i+1} = \theta_i \ [V \cup V(S_i)]$ et le facteur conjonctif C_{i+1} de la sous-dérivation issue de C_i en un coup $C_i \longrightarrow C_{i+1}$ vérifie $\theta_{i+1}C_{i+1}$ est trivial et $c(\theta_{i+1}, C_{i+1}) \leq c(\theta_i, C_i)$.

La suite des $(c(\theta_i, C_i))$ est décroissante. Montrons par l'absurde qu'elle atteint la valeur $(0,0,0)$.
Supposons que pour tout unifiande S_j de la dérivation, on ait $c(\theta_j, C_j) > (0,0,0)$. Puisque la suite est décroissante, il existe un unifiande S_k à partir duquel $c(\theta_k, C_k) > (0,0,0)$ ne change plus. Puisque la dérivation est équitable, il existe (θ_i, C_i) avec $i \geq k$, tel que
- ou bien C_k ne contient que des équations en $\doteq$, donc $c(\theta_i, C_i) = (0,0,0)$, ce qui fournit une contradiction
- ou bien $C_k \longrightarrow_{[\text{Surréduction}]} \ldots$, donc l'élément (θ_{k+1}, C_{k+1}) de la suite satisfait $c(\theta_{k+1}, C_{k+1}) < c(\theta_k, C_k)$, ce qui fournit une contradiction. $\square$

Lemme 6.16 Soit S un unifiande, $\theta \in SOL(S)$, et S' l'unifiande obtenu en remplaçant dans S toutes les équations en $\doteq_R$ et en FI par F. Si θS_{dc} a un facteur conjonctif trivial θC tel que $c(\theta C) = (0,0,0)$, alors $\theta \in SOL(S')$.

Preuve: Par définition, les équations en FI triviales ont une complexité non nulle. Or θC ne contient que des équations triviales, donc les équations en $\doteq_R$ ou en FI qu'il pourrait contenir auraient une complexité non nulle, et alors $c(\theta.C) > (0,0,0)$. Donc C contient seulement des équations en $\doteq$. Par conséquent C est aussi un facteur conjonctif de S'_{dc}, et puisque θC est trivial, $\theta \in SOL(S')$. $\square$

Théorème 6.17 *Soit S_0 un unifiande, $\theta_0 \in SOL(S_0)$, et $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ une dérivation équitable issue de S_0 utilisant les règles présentées jusqu'ici. Alors il existe un unifiande S_i de cette dérivation et une substitution θ_i tel que*
- $\theta_i \in SOL(S'_i)$ où S'_i est l'unifiande obtenu à partir de S_i en remplaçant toutes les équations en $\doteq_R$ et en FI par F.
- $\theta_i = \theta_0 \ [V(S_0)]$

Preuve: D'après les deux lemmes précédents en prenant $V = V(S_0)$. $\square$

6.4 Autres règles

Résolution des équations en $\doteq$

Pour utiliser le théorème précédent, il reste maintenant à résoudre les équations en $\doteq$. Il s'agit donc d'un problème d'unification syntaxique. La méthode retenue est celle de Martelli et Montanari [56]. Nous introduisons un nouveau symbole F (F comme faux) vérifiant $SOL(F) = \emptyset$.

(Décomposition-$\doteq$) $\dfrac{f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)}{s_1 \doteq t_1 \ \wedge \ \ldots \ \wedge \ s_n \doteq t_n}$

(Collision-$\doteq$) $\dfrac{f(\ldots) \doteq g(\ldots)}{F}$ si $f \neq g$

(Fusion-$\doteq$) $\dfrac{x \doteq t \ \wedge \ x \doteq t'}{x \doteq t \ \wedge \ t \doteq t'}$ si t n'est pas une variable

Ces règles retournent ou bien F, ou bien un système en forme substituable à condition qu'il n'y ait pas de cycle de variables.

L'équation $x \doteq t$ n'a pas de solution normalisée si t n'est pas en forme normale, donc $SOL(x \doteq t) = \emptyset$. D'où la règle

(Non-normal-$\doteq$) $\dfrac{x \doteq t}{F}$ si t n'est pas en forme normale

Cette règle prend en compte deux faits que nous avions évoqués au chapitre 3: les solutions de l'équation à résoudre non normalisées sont inutiles; les branches qui possèdent des équations ne satisfaisant pas la propriété de grandeur suffisante sont inutiles.

Simplifications

(Décomposition-$\doteq_R$) $\dfrac{f(s_1, \ldots, s_n) \doteq_R f(t_1, \ldots, t_n)}{s_1 \doteq_R t_1 \ \wedge \ \ldots \ \wedge \ s_n \doteq_R t_n}$ si f est un symbole décomposable

(Collision-$\doteq_R$) $\dfrac{f(\ldots) \doteq_R g(\ldots)}{F}$ si $f \neq g$ et f et g sont des symboles décomposables

Application en tête seulement

(Dépliage) $\dfrac{S[t \doteq_R t']}{S[x \doteq_R t|p \ \wedge \ t[p \leftarrow x] \doteq_R t']}$ si $\begin{cases} x \notin V \cup V(S[t \doteq_R t']) \\ t|p \text{ et } t[p \leftarrow x] \text{ ne sont pas des variables} \end{cases}$

(Evaluation-F-1) $\dfrac{S \wedge F}{F}$

(Evaluation-F-2) $\dfrac{S \vee F}{S}$

Conclusion

Nous obtenons une spécification équationnelle de la surréduction basique et de ses simplifications connues. La règle de surréduction apparait comme une règle d'application "don't care" et peut être facilement étendue au cadre ordo-sorté et/ou dans les systèmes de réécriture équationnelle. La simplification par réécriture est spécifiée de manière équationnelle.

Cette spécification peut être vue comme une procédure indéterministe, et décrit différentes procédures selon le choix d'une stratégie. En particulier on obtient:

- la surréduction basique (de Hullot) si seule la règle (Surréduction) est appliquée
- la SL-basic S-narrowing du chapitre 3 en appliquant (Surréduction) et (Non-normal-$\doteq$)
- la surréduction basique avec décomposition des symboles décomposables et détection des collisions en appliquant (Surréduction), (Décomposition-$\doteq_R$), (Collision-$\doteq_R$)
- la surréduction basique de gauche à droite (de Herold) en appliquant (Surréduction) et (Dépliage)
- la surréduction normalisante basique de gauche à droite, appelée aussi "left-to-right SL-basic N-narrowing" au chapitre 3, en appliquant (Surréduction), Dépliage, (Non-normal-$\doteq$), et les règles qui spécifient la réécriture
- une stratégie qui consiste à ne pas calculer les substitutions surréductrices immédiatement si on n'applique pas systématiquement les règles Décomposition-$\doteq$, Collision-$\doteq$, Fusion-$\doteq$. Cette stratégie pourra être intéressante lorsque ce calcul sera couteux, comme dans le cas ordo-sorté ou dans le cas de la réécriture équationnelle.

Conclusion

La surréduction est une méthode générale et complète de résolution d'équations dans les systèmes de réécriture confluents et noethériens. Etant à la fois générale et complète, elle présente un grand intérêt.

Récapitulons les principaux apports de ce travail sur la surréduction, ainsi que leurs limites, et regardons les perspectives de recherche.

Les apports

Méthodes (ou optimisations) nouvelles et complètes

- Description des solutions par un langage régulier. Cette idée permet de décrire finiment un ensemble infini d'unificateurs. Elle peut donc fournir une méthode d'unification complète pour les théories dont l'unification est de type infinitaire ou zéro, ce qui était impossible avec les méthodes de surréduction connues jusqu'ici. Elle apporte aussi une amélioration à la terminaison de la surréduction. La principale limitation de cette méthode est qu'elle ne s'applique que lorsque l'arbre des surréductions est rationnel.
- Suppression des branches instances.
- La surréduction normalisante basique de gauche à droite.

La surréduction basique de gauche à droite est particulièrement intéressante car nous avons vu qu'elle assurait la minimalité du multiensemble des solutions de rang maximal lorsque le système de réécriture est régulier et sans paire critique. Si θ est une solution de l'équation $t \doteq t'$, elle élimine en fait toutes les redondances dues aux multiples chemins de mise en forme normale du théorème $\theta(t \doteq t')$. On obtient même une procédure minimale de filtrage de deux termes dont les variables sont disjointes, en interdisant l'instanciation de certaines variables. Pour un système de réécriture quelconque, la suppression des branches instances et l'introduction d'étapes de réécriture dans la surréduction basique de gauche à droite apportent des améliorations intéressantes à la minimalité. Elles peuvent, en outre, et quelles que soient les hypothèses satisfaites par le système de réécriture, améliorer la terminaison du processus.

Une limitation de ces trois méthodes est que toutes les fois que le système de réécriture possède une règle récursive stricte, i.e. le membre gauche est un sous-terme strict d'une instanciation du membre droite, elles ne termineront pas. Par exemple dans la spécification des entiers, la règle

$$s(x) + y \to s(x + y)$$

est récursive. Et on a:

$$x + y \doteq 0 \leadsto_{[x/s(x)]} s(x + y) \doteq 0 \leadsto s(s(x + y)) \doteq 0 \leadsto \ldots$$

Cette surdérivation est basique de gauche à droite. En utilisant la surréduction normalisante, on obtiendrait la même surdérivation. La suppression des branches instances ne s'applique pas sur cet exemple. La méthode de description des solutions par un langage régulier ne

s'applique pas non plus car cette surdérivation est un arbre non rationnel. Les règles de décomposition et de collision, qui sont intégrées dans le formalisme des chapitres 5 et 6, permettraient de voir que l'équation $s(x+y) \doteq 0$ n'a pas de solution, donc qu'il est inutile de la surréduire.

Apports théoriques

• Preuve de complétude de la surréduction normalisante et de son implantation. Par rapport à la preuve de Fay, ces preuves ont l'avantage de bien faire ressortir les concepts de surréduction et de surréduction normalisante. D'autre part, les raisons pour lesquelles la surréduction normalisante est complète y apparaissent de manière plus explicite.

• Commutation de la surréduction. Ce résultat montre sous quelques hypothèses que deux étapes de surréduction peuvent être commutées, sans que la composée des substitutions surréductrices ni le terme résultant de ces deux étapes soient changés.

Implantation

• En modifiant la procédure de complétion de Knuth-Bendix, il a été possible d'implanter la surréduction dans le logiciel de réécriture REVE. Deux versions ont été crées:

- Une implantation de la surréduction basique normalisante.

- Une implantation de la surréduction normalisante, avec une forme simplifiée de la description des solutions par un langage régulier. Voici un exemple d'exécution de ce logiciel:

```
Rewrite rules:

1.   f(0, x) -> x
2.   g(f(x, y)) -> g(y)
```

Ce système de réécriture est dû à Fages et Huet [15]. C'est une théorie de type zéro. L'utilisateur tape "narrowing" puis rentre l'équation à résoudre.

```
-> narrowing
Please enter the equation you would like to solve, terminated with <ESC>:
g(x)==g(y)
```

La procédure termine et répond:

```
The equation

g(x) == g(y)

has the set of R-unifiers as follows:
1.
  x  /  x
  y  /  x

and has the infinite sets of R-unifiers, defined by the recursive definitions
as follows:
```



```
If
    x / t1
    y / t2
is a R-unifier of the equation, the substitutions
1.
  x  /  f(all, t1)
  y  /  t2
2.
  x  /  t1
  y  /  f(all, t2)

are R-unifiers of the equation.
(all is what you want)
```

Formulation équationnelle

• Les deux derniers chapitres donnent une nouvelle formulation à la surréduction basique. Elle présente les particularités suivantes:

- Elle est équationnelle.

- Elle est sous la forme de règles d'inférence élémentaires, donc plus facilement implantable.

- Elle peut être fortement parallélisée.

- Elle fournit une procédure indéterministe, dont les instances décrivent plusieurs stratégies.

- Elle fournit, par le chapitre 6 (introduction du $\vee$), un formalisme propice à une extension au cadre de la logique ordo-sortée et aux systèmes de réécriture équationelle.

Les perspectives

• Etendre la méthode de description des solutions par un langage régulier aux cas où l'arbre des surréductions n'est pas rationnel. Puisque la surréduction peut être vue comme une complétion particulière, l'idée consiste à utiliser les travaux effectués sur la procédure de complétion, et en particulier la notion de méta-terme [49]. D'autres travaux [28] donnent des conditions suffisantes de non terminaison de la complétion, et peuvent donner un éclairage utile sur les différents cas de non-terminaison. On pourra aussi utiliser les travaux sur la méta-surréduction [48].

• Avant de passer à une implantation, il serait utile de poursuivre les travaux du chapitre 6 pour améliorer la spécification de la réécriture, et atteindre les objectifs suivants:

- rendre la spécification de la réécriture terminante, ce qui devrait permettre une spécification du contrôle (hypothèse d'équité) plus aisée.

- intégrer dans le processus de filtrage le calcul de la partie protégée, en utilisant la définition de la réécriture faiblement basique du chapitre 3. Cela permettrait de supprimer la règle (Basification-de-réécriture), donc de gagner en efficacité.

- au cours du processus de réécriture, aucune règle d'inférence ne peut être appliquée sur l'équation à réécrire et sur son contexte. On peut voir cela comme étant l'application d'un certain contrôle. Pour une plus grande généralité, il serait souhaitable de permettre l'application de règles sur l'équation à réécrire et son contexte. Il est d'ailleurs remarquable que la relation ainsi crée n'est pas la réécriture, mais quelque chose de plus complexe.

• Etendre cette thèse au cadre des signatures ordo-sortée et à la réécriture équationnelle. L'introduction du $\vee$ au chapitre 6 devrait faciliter une telle extension.

• L'étude de la minimalité du multiensemble des solutions générées par différentes méthodes de surréduction est un moyen d'évaluer la redondance interne de ces méthodes. Il reste à étudier et à apporter des réponses au problème de la non-terminaison. Une première piste de de recherche, dont nous avons déjà parlé, consiste à étendre la méthode de description des solutions par un langage régulier aux cas où l'arbre des surréductions n'est pas rationnel. Une deuxième piste est fournie par les travaux de Sivakumar et Dershowitz [69], qui étendent les règles de décomposition et de collision.

• La surréduction transforme un problème d'unification modulo des équations en un problème plus simple d'unification syntaxique. Peut-on s'en servir pour transformer, d'une manière générale, un problème modulo des équations, en un problème modulo un ensemble d'équations vide? En particulier, peut-elle servir à la résolution d'inéquations ou de diséquations? La résolution de diséquations ou disunification est étudiée dans [8,9,47,10]. Regardons ce que la surréduction pourrait apporter à ce problème. Soit σ une substitution normalisée, σ est une solution de la diséquation $t \neq t'$ ssi pour toute dérivation:

$$(\sigma t \neq \sigma t') \xrightarrow{*} (s_n \neq s'_n)$$

on a $s_n \neq s'_n$. D'après la proposition 1.5, on a alors

$$(t \neq t') \overset{*}{\underset{[\theta]}{\rightsquigarrow}} t_n \ (t_n \neq t'_n)$$

et il existe σ_n telle que

- $\sigma = \sigma_n.\theta \ [V(t) \cup V(t')]$
- $\sigma_n(t_n) = s_n$ et $\sigma_n(t'_n) = s'_n$.

La réciproque est vraie aussi. Par conséquent σ est une solution de $t \neq t'$ ssi pour toute surdérivation:

$$(t \neq t') \overset{*}{\underset{[\theta]}{\rightsquigarrow}} t_n \ (t_n \neq t'_n)$$

telle que $\theta \leq \sigma$, la substitution σ_n définie par $\sigma_n.\theta = \sigma$ satisfait $\sigma_n(t_n) \neq \sigma_n(t'_n)$. Le problème de la disunification modulo des équations est ramené a un problème de disunification syntaxique. Par exemple, soit le système de réécriture:

$$R = \{ \begin{array}{ll} r_1 : & f(x,x) \rightarrow x \\ r_2 : & f(x,a) \rightarrow a \ \} \end{array}$$

Résolvons la diséquation $e = (f(b,y) \neq b)$. On a:

$$\begin{array}{ll} f(b,y) \neq b & \underset{[y/b]}{\rightsquigarrow} b \neq b \ (1) \\ & \underset{[y/a]}{\rightsquigarrow} a \neq b \ (2) \end{array}$$

La substitution y/b n'est pas solution de e puisqu'elle est la substitution surréductrice de la branche (1), qui conduit à une diséquation non satisfaite. Par contre y/a est solution, puisque la branche (2) abouti à une diséquation satisfaite. Toute autre substitution σ est solution de e puisqu'elle n'est pas instance d'une substitution surréductrice et que l'équation $\sigma(e)$ est satisfaite.

Remarquons que contrairement au problème de l'unification, il faut ici générer entièrement l'arbre des surréductions pour être sûr que σ soit une solution. En d'autres termes, la correction d'une telle procédure impose le parcours complet de l'espace de recherche, ce qui nécessite qu'il soit fini.

Bibliographie

[1] A. Baader. Unification in idempotent semigroups is of type zero. *Journal of Automated Reasoning*. 2(x):283–286, 1986.

[2] L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, MCC, 3500 West Balconies Center Drive, Austin, Texas 78759-6509, May 4-6 1987.

[3] R. Barbuti, M. Bellia, G. Levi, and M. Martelli. LEAF: a language which integrates logic, equations and functions. In D. DeGroot and G. Lindstrom, editors, *Logic Programming. Funstions, relations and equations*, Prentice Hall, 1986.

[4] P.G. Bosco, E. Giovannetti, and C. Moiso. Refined strategies for semantic unification. In *Proc. of the International Joint Conference on Theory and Practice of Software Development*, Springer LNCS 250, 276-290, Pisa, Italy, March 1987.

[5] G. Boudol. Computational semantics of term rewriting systems. In M. Nivat and J. Reynolds, editors, *Application of Algebra to Language definition and Compilation*, Prentice Hall, 1985.

[6] H. Bürckert, A. Herold, and M. Schmidt-Schauß. On equational theories, unification and decidability. In *Proceedings of the Second Conference on Rewriting Techniques and Applications*, Springer-Verlag, Bordeaux (France), May 1987.

[7] H.J. Bürckert, A. Herold, and M. Schmidt-Schauss. On equational theories, unification and decidability. *Journal of Symbolic Computation*, to appear in 1988.

[8] A. Colmerauer. Equations and inequations on finite and infinite trees. In *FGCS'84Proceedings*, pages 85–99, November 1984.

[9] H. Comon. Sufficient completeness, term rewriting system and anti-unification. In J. Siekmann, editor, *Proceedings 8th Conference on Automated Deduction, Oxford*, pages 128–140, Springer Verlag, Oxford (England), 1986.

[10] H. Comon. These d'etat. 1988.

[11] R.J. Cunningham and A.J.J. Dick. *Rewrite Systems on a Lattice of Types*. Technical Report, Imperial College, Department of Computing, 1983.

[12] N. Dershowitz. *Applications of the Knuth-Bendix Completion Procedure*. Technical Report ATR-83(8478), Laboratory Operation, Aerospace Corporation, El Segundo, CA 90245, May 1983.

[13] N. Dershowitz and A. Josephson. Logic programming by completion. In *2nd Int. Conf. on Logic Programming*, pages 313–320, Uppsala, Sweden, 1984.

[14] F. Fages. *Le système KB : manuel de référence : présentation et bibliographie, mise en œuvre.* Technical Report Greco de programmation 10.84, INRIA, 1984.

[15] F. Fages and G. Huet. Unification and matching in equational theories. In *Proceedings 5th Conference on Automata, Algebra and Programming, L'Aquila*, Springer Verlag, Lecture Notes in Computer Science, 1983.

[16] M. Fay. First-order unification in an equational theory. In *4th Workshop on Automated Deduction*, pages 161–167, Austin, Texas, 1979.

[17] M. Fay. *First-Order Unification in An Equational Theory.* Master Thesis 78-5-002, U. of California at Santa Cruz, May 1978.

[18] L. Fribourg. Handling function definitions through innermost superposition and rewriting. In *1st Conference on Rewriting Techniques and Applications*, pages 325–344, Springer Verlag, Lecture Notes in Computer Science, Dijon, France, 1985.

[19] L. Fribourg. A narrowing procedure for theories with constructors. In R. E. Shostak, editor, *7th international Conference on Automated Deduction*, Springer Verlag, Lecture Notes in Computer Science, 1984.

[20] L. Fribourg. Oriented equational clauses as a programming language. *Journal of Logic Programming*, 1(2):165–177, 1984.

[21] L. Fribourg. Prolog with simplification. In *First France-Japan Symposium on Artificial Intelligence and Computer Science*, Tokyo, 1986.

[22] L. Fribourg. SLOG: a logic programming language interpreter based on clausal superposition and rewriting. In *IEEE Symposium on Logic Programming*, Salt Lake City, Utah, 1985.

[23] J. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Journal of Theorical Computer Science*, 1988.

[24] J. Gallier and W. Snyder. A general complete E-unification procedure. In P. Lescanne, editor, *Proc. 2nd Rewrite Techniques and Applications*, pages 216–227, Springer-Verlag, Bordeaux (France), 1987.

[25] I. Gnaedig, C. Kirchner, and H. Kirchner. *Equational Completion in Order-sorted Algebras.* Technical Report 87R086, Centre de Recherche en Informatique de Nancy, 1987.

[26] J.A. Goguen and J. Meseguer. EQLOG: equality, types and generic modules for logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming. Functions, relations and equations*, Prentice Hall, 1986.

[27] J. Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la Soc. des Sciences et des Lettres de Varsovie, Classe III*, 33(128), 1930.

[28] M. Hermann and I. Privara. *On nontermination of Knuth-Bendix algorithm.* Technical Report VUSEI-AR-OPS-3/85, Institute of Socio-Economic Information and Automation in Management Department of Programming Systems, Bratislava, Czechoslovakia, november 1985.

[29] A. Herold. *Narrowing Techniques Applied to Idempotent Unification.* Technical Report SR-86-16, SEKI, Août 1986.

[30] A. Herold. *Some basic notions of first-order unification theory.* SEKI-MEMO 15/83, Universitat Karlsruhe, 1983.

[31] S. Hölldobler. A unification algorithm for confluent theories. In *14th Int. Coll. on Automata, Languages and Programming*, 1987.

[32] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In *Int. Coll. on Automata Languages and Programming*, 1987.

[33] G. Huet. Résolution d'equations dans les langages d'ordre 1,2, ...,ω, Thèse d'état de l'Université de Paris 7, 1976.

[34] G. Huet and J.J. Levy. *Call By Need Computations in Non-Ambiguous Linear Term Rewriting Systems.* Technical Report 359, Laboria, Août 1979.

[35] J.M. Hullot. Canonical forms and unification. In *Proc. of the 5th Conf. on Automated Deduction*, pages 318–334, Springer Verlag, Lecture Notes in Computer Science, Les Arcs, France, July 1980.

[36] J.M. Hullot. Compilation de formes canoniques dans les théories equationelles. 1980. Thèse de 3eme cycle, Université de Paris Sud, Orsay, France.

[37] H. Hussmann. Unification in conditional equational theories. In *Proc. of EUROCAL*, Springer Verlag, 1985.

[38] A. Josephson and N. Dershowitz. An implementation of narrowing: the rite way. In *Third IEEE Symposium on Logic Programming*, Salt Lake City, Utah, 1986.

[39] J-P. Jouannaud, C. Kirchner, and H. Kirchner. Incremental construction of unification algorithms in equational theories. In *Automata, Languages and Programming, Barcelona, 1983*, pages 361–373, Springer Lecture Notes in Computer Science, 1983.

[40] J-P. Jouannaud and B. Waldmann. Reductive conditional term rewriting systems. In M. Wirsing, editor, *3rd IFIP Conf. on Formal Description of Programming Concepts*, Elsevier Science Publishers, Ebberup, (Danemark), 1986.

[41] T. Kanamori. *Computation by meta-unification with constructors.* Technical Report, Mitsubishi Electric Corporation, 1987.

[42] C. Kirchner. Computing unification algorithms. In *Proceeding of the first symposium on Logic In Computer Science, Boston (USA)*, pages 206–216, 1986.

[43] C. Kirchner. Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles. Thèse d'état de l'Université de Nancy I, 1985.

[44] C. Kirchner. *Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles.* PhD thesis, Université de Nancy 1, 1985.

[45] C. Kirchner. Methods and tools for equational unification. In *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, BP 239, 54506 Vandœuvre-les-Nancy, France, May 4-6 1987.

[46] C. Kirchner. Order-sorted equational unification. 1988.

[47] C. Kirchner and P. Lescanne. Solving disequations. In *Proceeding of the second symposium on Logic In Computer Science*, pages 347–352, 1987.

[48] H. Kirchner. Preuves par complétion dans les variétés d'algèbres. Thèse d'état de l'Université de Nancy I, 1985.

[49] H. Kirchner. Schematization of infinite sets of rewrite rules. application to the divergence of completion processes. In *Proceedings Second Conference on Rewriting Techniques and Applications*, Springer Verlag, Bordeaux (France), May 1987.

[50] D. Knuth and P. Bendix. *Simple Word Problems in Universal Algebras*, pages 263–297. Pergamon Press, 1970.

[51] D.S. Lankford and A.M. Ballantyne. *Decision Procedures for Simple Equational Theories with Permutative Axioms: Complete sets of Permutative Reductions.* Technical Report Atp-37, Dept. of Mathematics and Computer Science U. Texas, Austin, Avril 1977.

[52] D.S. Lankford and A.M. Ballantyne. The refutation completeness of blocked permutative narrowing and resolution. In *Proceedings of the 4th Conference on Automated Deduction, Lecture Notes in Computer Science, Volume 87*, pages 53–59, Springer-Verlag, Berlin, West-Germany, 1979.

[53] P. Lescanne. Computer experiments with the reve term rewriting systems generator. In *Proceedings, 10th ACM Symposium on Principles of Programming Languages*, ACM, 1983.

[54] A. Martelli, C. Moiso, and G-F. Rossi. An algorithm for unification in equational theories. In *Third IEEE Symposium on Logic Programming*, Salt Lake City, Utah, 1986.

[55] A. Martelli, C. Moiso, and G-F. Rossi. Lazy unification algorithms for canonical rewrite systems. In *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, MCC, 3500 West Balconies Center Drive, Austin, Texas 78759-6509, May 4-6 1987.

[56] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM T.O.P.L.A.S*, 4(2):258–282, 1982.

[57] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions On Programming Languages And Systems*, 4(2):258–282, 1982.

[58] P. Padawitz. Strategy-controlled reduction and narrowing. In *Proc. 2nd Conference on Rewriting Techniques and Applications*, pages 242–255, Springer Verlag, Lecture Notes in Computer Science, Bordeaux (France), 1987.

[59] G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[60] U.S. Reddy. Narrowing as the operational semantics of functional languages. In *IEEE Symposium on Logic Programming*, Salt Lake City, Utah. 1985.

[61] P. Réty. Improving basic narrowing techniques. In *Proc. 2nd International Conference on Rewriting Techniques and Applications*, pages 228–241, Springer Verlag, Lectures notes in Computer Science, Bordeaux, France, 1987.

[62] P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne. Narrower: a new algorithm for unification and its application to logic programming. In *Proc. 1rst Conf. on Rewriting Techniques and Applications*, pages 141–157, Springer Verlag, Lecture Notes in Computer Science, Dijon (France), 1985.

[63] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12, 1965.

[64] M. Sato and Sakurai T. QUTE: a functional language based on unification. In D. DeGroot and G. Lindstrom, editors, *Logic Programming. Funstions, relations and equations*, Prentice Hall, 1986.

[65] M. Schmidt-Schauss. *Computational Aspects of an Order-Sorted Logic with Term Declarations.* PhD thesis, FB Informatik, Universitat Kaiserslautern, 1987.

[66] M. Schmidt-Schauss. Unification under associativity and idempotence is of type nullary. *Journal of Automated Reasoning*, 2(3):277–282, 1986.

[67] J. Siekmann. Unification theory. *Journal of Symbolic Computation*, to appear in 1988.

[68] J. Siekmann and P. Szabo. Universal unification and classification of equational theories. *Proceedings 6th Conference on Automated DEduction, Lecture Notes in Computer Science*, 138, 1982.

[69] G. Sivakumar and N. Dershowitz. *Goal directed equation solving.* Technical Report xxx87, University of Illinois, Urbana-Champaign, USA, 1987.

[70] J.R. Slagle. Automated theorem proving for theories with simplifiers, commutativity and associativity. *Journal of the Association for Computing Machinery*, 21(4):662–642, 1969.

[71] G. Smolka and W. Nutt. Order-sorted equational computation. In H. Ait-Kaci, editor, *Coll. on Resolution of Equations in Algebraic Structure*, Austin, Texas, 1987.

[72] G. Smolka, W. Nutt, J.A. Goguen, and J. Meseguer. Order sorted equational computation. In *Proceedings of the Colloquium on Resolution of Equations in Algebraic Structures*, May 1987.

[73] P.A. Subrahmanyam and J.H. You. FUNLOG: a computational model integrating logic programming and functional programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming. Funstions, relations and equations*, Prentice Hall, 1986.

[74] J.H. You. Enumerating outer narrowing derivations for constructor-based term rewriting systems. In *Int. Coll. on Automata Languages and Programming*, 1988. en cours de publication.

[75] J-H. You and P.A. Subrahmanyam. A class of confluent term rewriting systems and unification. *Journal of Automated Reasoning*, 2:391–418, 1986.

[76] J-H. You and P.A. Subrahmanyam. *On the Completeness of Narrowing for E-Unification.* Technical Report, University of Alberta, 1986.

NOM DE L'ETUDIANT : RETY Pierre

NATURE DE LA THESE : DOCTORAT DE L'UNIVERSITE DE NANCY I en informatique

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 18 MARS 1988 461

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

R. MAINARD

Résumé

L'objet de cette thèse est l'étude et l'amélioration de la surréduction, en vue de son application à l'unification, c'est à dire à la résolution d'équations dans les théories équationnelles.

Le premier objectif est d'améliorer l'efficacité et la terminaison de la surréduction, en utilisant des optimisations de la surréduction qui restent complètes dans les systèmes de réécriture confluents et nœthériens.

- Si l'espace de recherche est infini, nous montrons que dans certains cas il peut être décrit finiment, et les solutions peuvent alors être décrites par un langage régulier.

- Nous définissons une nouvelle stratégie de surréduction: la surréduction normalisante basique de gauche à droite, et établissons sa complétude. Par l'intermédiaire d'une propriété de commutation de la surréduction, nous donnons un résultat sur la minimalité de l'ensemble des solutions générées par cette méthode.

Ces optimisations ont été implantées dans une version expérimentale du logiciel de réécriture REVE, en utilisant le fait que la surréduction peut être vue comme une complétion un peu particulière.

Le deuxième objectif est de donner une formulation équationnelle à la surréduction basique, c'est à dire une formulation basée sur des systèmes d'équations et dans laquelle n'apparait jamais la notion de substitution. Les opérations sont décrites par des règles d'inférence. Ce travail a été conduit en cherchant:

- à ce que les règles d'inférence puissent être, autant que possible, exécutées sans contrôle et en particulier de manière parallèle

- à intégrer dans le processus de nombreuses règles de simplification

- à ce que la méthode puisse être facilement étendue à la logique ordo-sortée et/ou à des systèmes de réécriture équationnelle.