Les termes en logique et en programmation
Version préliminaire

Hubert Comon et Jean-Pierre Jouannaud

Laboratoire de Recherche en Informatique
Bât. 490
CNRS et Université de Paris Sud
91405 Orsay, France
email: comon,jouannau@lri.lri.fr
http://www.lri.fr/~jouannau/biblio.html

December 10, 1997

Contents

Chapter 1

Introduction

Les termes et algèbres de termes constituent la notion syntaxique de base utilisée en programmation fonctionnelle, en programmation logique, en programmation par contrainte, mais aussi en logique et donc dans la pluspart des démonstrateurs. Ils servent également à construire des modèles, permettant de montrer la complétude de méthodes de déduction en logique classique. Ils jouent donc un rôle fondamental en programmation et en démonstration.

L'objectif de ce cours est de donner quelques outils et résultats sur les termes qui permettent l'analyse de programmes ou de stratégies de déduction.

Nous étudions dans le chapitre 6 la construction d'ordres bien fondés sur les termes; ceux-ci permettent d'effectuer des preuves de terminaison et de définir des stratégies ordonnées de preuve en logique équationnelle ou en logique clausale. Le résultat important de ce chapitre est le théorème de Kruskal qui énonce que le plongement est un pré-bel ordre, ce qui permet comme nous le montrons, de construire systématiquement des ordres de simplification, ordres qui sont bien fondés sur les termes.

Dans le chapitre 8 nous étudions la structure de treillis des termes: la borne supérieure de deux termes peut être calculée en utilisant un plus général unificateur (lorsqu'il existe). Nous présentons donc des algorithmes d'unification, pour les termes finis et pour les termes infinis. La borne inférieure de deux termes finis peut être calculée à l'aide d'un algorithme d'anti-unification (aussi appelé "de généralisation") également présenté dans ce chapitre. Ces opérations permettent de munir l'ensemble des termes d'une structure de treillis. En fait, on peut étendre cette structure en un treillis Booléen, ce qui sera une conséquence des résultats du chapitre 11.

Dans le chapitre 3 nous montrons les résultats de Birkhoff de complétude en logique équationnelle qui permettent entre autres, de ramener les preuves dans une variété d'algèbres à des preuves dans l'algèbre des termes.

Dans le chapitre 9 nous introduisons les notions de base de la logique classique du premier ordre puis nous montrons un résultat similaire à celui du chapitre précédent pour les preuves de formules en forme clausale: le théorème de Herbrand.

Dans le chapitre 10, nous étudions la résolution comme système de déduction pour les formules sous forme clausale. Nous montrons la complétude de certaines stratégies de preuve et nous appliquons les techniques précédentes aux clauses de Horn; ce sont les clauses utilisées en programmation logique.

Dans le chapitre 11, nous donnons des axiomatisations complètes de la théorie des termes (dans le cas des termes finis sur un alphabet fini, dans le cas des termes finis sur un alphabet infini, dans le cas des termes infinis sur un alphabet fini et dans le cas des termes infinis sur un alphabet infini). Ces axiomatisations complètes résultent d'un théorème de Mal'cev dans le cas des termes finis et de résultats plus récents dans le cas des termes infinis.

Enfin, dans le chapitre 12, nous étudions les outils de théorie des langages d'arbres, dans le but de reconnaitre des ensembles particuliers de termes clos, en particulier les termes clos en forme normale pour un système de réécriture linéaire gauche.

Pour en savoir plus, on pourra consulter les ouvrages [8, 7, 11, 9, 5, 6, 10, 2, 3, 4, 13, 12, 1].

Chapter 2

Algèbres de Termes

La notion de terme fonde tout calcul symbolique, en particulier l'interprétation des langages de programmation évolués.

2.1 Termes

2.1.1 Signatures

Une *signature* est une paire (S, F) où

(i) S est un ensemble non vide de (noms de) *sortes* . Les noms de sortes seront soulignés pour les distinguer des autres objets.

(ii) F est un ensemble non vide de (noms de) *fonctions* , disjoint de S. F sera supposé muni d'une *fonction de typage* τ qui associe à chaque symbole de F une séquence non vide d'éléments de S. Si $\tau(f) = (\underline{s}_1, \ldots, \underline{s}_n, \underline{s})$, on note $f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$. $|f| = n$ est appelé *arité* de f , $\underline{s}_1 \times \ldots \times \underline{s}_n$ est appelé *domaine de* f et $\underline{s}$ est appelé *codomaine* de f . Les fonctions d'arité 0 sont appelées *constantes* .

On désignera par F_n l'ensemble des symboles de F dont l'arité est n et par $F_{\underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}}$ l'ensemble des symboles de F de domaine $\underline{s}_1 \times \ldots \times \underline{s}_n$ et de codomaine $\underline{s}$. On a donc : $F = \cup_n F_n = \bigcup_{\underline{s}_1, \ldots, \underline{s}_n, \underline{s}} F_{\underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}}$.

S et τ pourront être omis lorsque S est un singleton.

Exemple 2.1 *$S = \{\underline{bool}, \underline{nat}\}$, et F est l'ensemble des symboles:*

$0 :$		$\to$	$\underline{nat}$	$s :$	$\underline{nat}$	$\to$	$\underline{nat}$
$+ :$	$\underline{nat} \times \underline{nat}$	$\to$	$\underline{nat}$	$eq :$	$\underline{nat} \times \underline{nat}$	$\to$	$\underline{bool}$
$true, false :$		$\to$	$\underline{bool}$	$\wedge :$	$\underline{bool} \times \underline{bool}$	$\to$	$\underline{bool}$
$\ldots$							

2.1.2 Positions

Soit $\mathbf{N}$ l'ensemble des entiers naturels, $\mathbf{N}_+$ l'ensemble des entiers naturels non nuls, $\mathbf{N}_+^*$ l'ensemble des suites (ou séquences) finies d'entiers naturels non nuls (ou des mots sur le vocabulaire $\mathbf{N}_+$), et $\Lambda \in \mathbf{N}_+^*$ la suite (ou mot) vide. Chaque $i \in \mathbf{N}_+$ est identifié à la séquence contenant l'unique élément i. Si $p, q \in \mathbf{N}_+^*$, $p \cdot q$ dénote leur concaténation, opération interne dont la séquence vide est élément neutre à droite et à gauche.

$\mathbf{N}_+^*$ est muni de la relation d'ordre $\leq_{pref}$ définie par:

$$\forall p, q \in \mathbf{N}_+^*,\ p \leq_{pref} q \Leftrightarrow \exists r \in \mathbf{N}_+^*,\ p \cdot r = q$$

et on dira que p est un *préfixe* de q lorsque $p \leq_{pref} q$. On écrira $p \bowtie q$ lorsque p et q sont incomparables, c'est à dire lorsque p n'est pas préfixe de q ni q de p. L'ordre *hiérarchique* étend l'ordre préfixe en un ordre total comme suit:

$$p \leq_{hier} q \text{ ssi } p \leq_{pref} q \text{ ou } p = r.i.p', q = r.j.q' \text{ et } i <_{\mathbf{N}_+} j$$

Un ensemble de *positions* est un sous-ensemble non vide de $\mathbf{N}_+^*$ fermé par ordre hierarchique, donc contenant Λ, appelée *racine* dans ce contexte, et dont les positions maximales pour l'ordre préfixe sont appellées *feuilles*.

2.1.3 Termes

Etant donnée une signature (S, F), un *arbre étiqueté*, encore appellé *terme*, est une application t d'un ensemble de positions noté $Pos(t)$ dans F, qui respecte l'arité des symboles de fonction:

- $t(p) \in F_0$ ssi p est une feuille de $Pos(t)$
- Si $t(p) = f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$, alors $\forall i \in \mathbf{N},\ p \cdot i \in Pos(t) \ \Leftrightarrow 1 \leq i \leq n$, et $t(p \cdot i)$ a $\underline{s}_i$ pour codomaine.

Un terme est *fini* si son ensemble de positions est fini. On notera par $T(F)$ l'ensemble des termes fini sur le vocabulaire F.

Étant donné un ensemble $X = \cup_{s \in S} X_s$ disjoint de F et S, de (noms de) constantes appellées *variables*, on notera par $T(F, X)$ l'ensemble $T(F \cup X)$. Les termes de $T(F) \subseteq T(F, X)$ seront dits *clos* ou *fermés*.

Exemple 2.2 *Supposons que la signature est celle de l'exemple 2.1, et que la variable x est de sorte $\underline{nat}$. Les trois arbres de la figure 2.1 sont des arbres finis, qui ont respectivement pour ensemble de positions:*

- $Pos(t_1) = \{\Lambda, 1, 2, 1 \cdot 1\}$ *avec* $t_1(\Lambda) = +,\ t_1(1) = s,\ t_1(1 \cdot 1) = t_1(2) = 0$
- $Pos(t_2) = \{\Lambda, 1, 2\}$ *avec* $t_2(\Lambda) = +$ *et* $t_2(1) = t_2(2) = x$.
- $Pos(t_3) = \{\Lambda, 1, 2, 1 \cdot 1, 1 \cdot 2, 1 \cdot 2 \cdot 2, 1 \cdot 2 \cdot 1\}$

Les figures 2.2 et 2.3 montrent des exemples d'arbres infinis. Sur les exemples de la figure 2.2, les ensembles de positions sont respectivement:

- *L'ensemble des séquences finies de 1:* $\{\Lambda, 1, 1 \cdot 1, 1 \cdot 1 \cdot 1, \ldots\} = \{1\}^*$
- *L'ensemble* $\{\Lambda, 1, 1 \cdot 1, 2, 2 \cdot 1, 2 \cdot 2, 2 \cdot 2 \cdot 1, 2 \cdot 2 \cdot 2 \cdot 1, \ldots\}$ *qui peut être représenté par l'expression régulière* $(2 \cdot 2)^*(\Lambda + 1 + 2 + 1 \cdot 1 + 2 \cdot 1)$

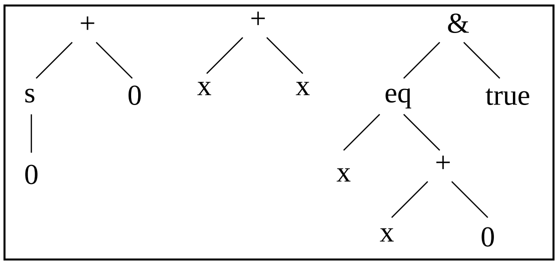

Figure 2.1: Termes finis

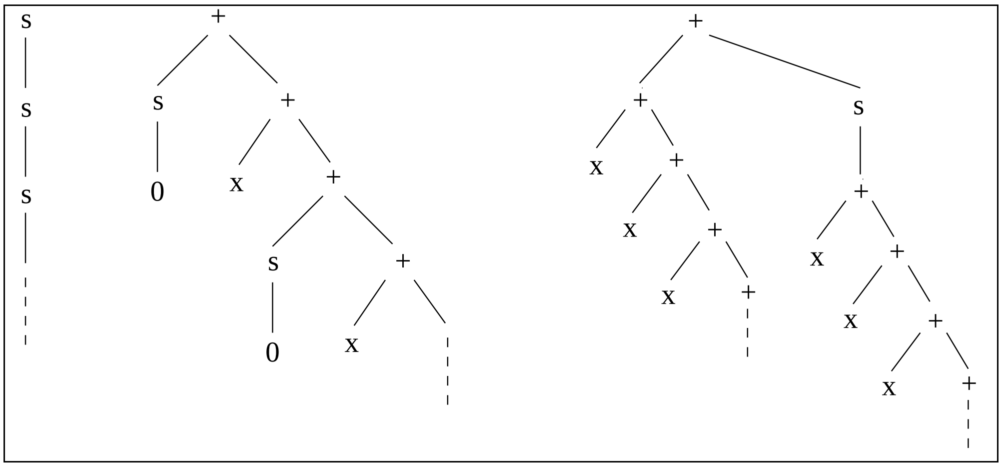

Figure 2.2: Termes infinis rationnels

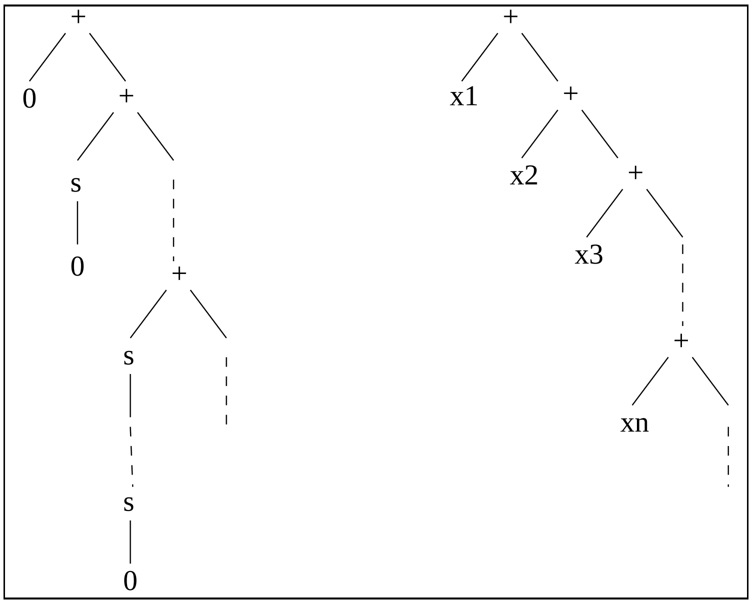

Figure 2.3: Termes infinis non rationnels

- $(\Lambda+2)2^*(\Lambda+1)$

Un terme fini est aussi noté sous forme d'une expression parenthésée. Les termes de la figure 2.1 peuvent ainsi être notés respectivement

$$+(s(0),0), \qquad +(x,x), \qquad \&(eq(x,+(x,0)),true).$$

Les symboles d'arité 2 pourront être utilisés en notation infixée. Par exemple, les termes de la figure 2.1 seront aussi notés $s(0)+0$, $x+x$ et $eq(x,x+0) \wedge true$.

La *profondeur* d'un terme fini ($t \in T(F,X)$) est la longueur de sa plus longue position. Sa *taille* est le cardinal de son ensemble de positions. Si $t \in T(F,X)$, $Var(t)$ désigne l'ensemble des variables apparaissant comme étiquette à une position de t.

La *sorte* d'un terme t est le codomaine de $t(\Lambda)$. Comme nous avons supposé l'existence d'une constante de chaque sorte, il existe donc un terme clos de sorte s pour chaque sorte s.

Enfin, l'égalité "syntaxique" entre termes est notée $\equiv$.

2.1.4 Sous-termes

Définition 2.3 *Étant donnés un terme t et une position $p \in Pos(t)$, le* sous-terme *de t à la position p, noté $t|_p$, est défini par:*

- $Pos(t|_p) = \{q \in \mathbf{N}_+^* \mid p \cdot q \in Pos(t)\}$
- *Pour tout $q \in Pos(t|_p)$, $(t|_p)(q) = t(p \cdot q)$*

On vérifie aisément que, $\forall p \in Pos(t)$, $t|_p$ est un terme.

Exemple 2.4 *Sur l'exemple 2.2, t_1 a pour sous-termes $t_1|_\Lambda \equiv t_1$, $t_1|_1 \equiv s(0)$, $t_1|_2 \equiv t_1|_{1.1} \equiv 0$*

Définition 2.5 *Un* terme rationnel *est un terme n'ayant qu'un nombre fini de sous-termes distincts.*

Les termes finis sont des termes rationnels. Les exemples de la figure 2.2 sont des termes rationnels sans être des termes finis.

Les termes représentés sur la figure 2.3 ne sont pas des termes rationnels (si l'on suppose que $x_1, \ldots, x_n, \ldots$ sont des variables distinctes).

Les termes rationnels permettent de décrire des structures cycliques. Ils sont en fait, des "dépliages" de graphes finis étiquetés (nous ne formalisons pas ici ces notions). La figure 2.4 donne des représentations sous forme de graphes des termes rationnels de la figure 2.2.

Proposition 2.6 *L'ensemble des positions d'un terme rationnel est un langage régulier.*

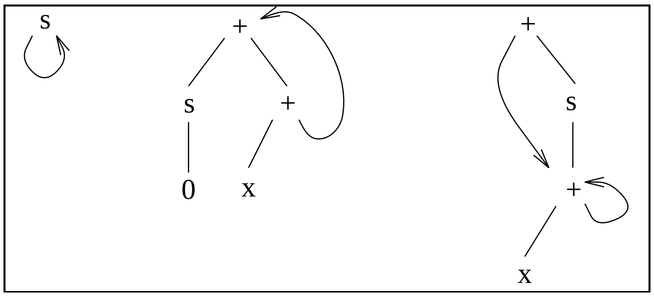

Figure 2.4: Les termes rationnels comme graphes finis

Preuve
Soit t un terme rationnel: soit n le nombre de ses sous-termes distincts. Soit p une position de t de longueur n. Par hypothèse, au moins deux des termes $t|_q$, $q \leq_{pref} p$ sont identiques. Soit $t|_{p_1} \equiv t|_{p_1 \cdot q_1}$. $p_1 \cdot q_1$ est un préfixe de p, donc $p_1 \cdot q_1 \cdot r_1 = p$ et $t|_p \equiv t|_{p_1 \cdot r_1}$. Ainsi tout sous-terme de t à une profondeur supérieure à n est identique à l'un de ses "ancêtres". Soit P l'ensemble des positions de t et, pour chaque position q de t, de longueur inférieure ou égale à n, P_q l'ensemble des positions de $t|_q$. Si $q < n$, P_q vérifie une équation

$$P_q = 1 \cdot P_{q \cdot 1} + \ldots + n_q \cdot P_{q \cdot n_q}$$

si n_q est l'arité de $t(q)$ (ou bien le nombre de descendants directs s'il s'agit d'un symbole sans arité). Si maintenant q est de longueur n, P_q vérifie une équation

$$P_q = P_{p_1 \cdot r_1}$$

où $p_1 \cdot r_1$ est de longueur strictement inférieure à n. Donc les ensembles P_q vérifient un système d'équations linéaires gauche qui possède une unique solution qui est un langage régulier (cf. un résultat classique de langages formels). En particulier, $P_\Lambda = P$ est régulier. $\Box$

L'ensemble des termes rationnels est noté $RT(F, X)$ (ou $RT(F)$ pour les termes rationnels fermés). L'ensemble de tous les termes finis ou infinis est noté $IT(F, X)$ (ou $IT(F)$ pour les termes fermés). $Var(t)$ est encore l'ensemble (fini) des variables du terme rationnel t.

Une *occurrence* d'un terme t dans un terme u est une position $p \in Pos(u)$ telle que $u|_p \equiv t$. Un terme dans lequel toute variable a au plus une occurrence est dit *linéaire* .

2.1.5 Remplacement de sous-termes

Définition 2.7 *Soient u et v deux termes et p une position de u telle que v et $u|_p$ sont de même sorte. Le terme $u[v]_p$ obtenu par* remplacement dans u du sous-terme à la position p par v *est défini par:*

- $Pos(u[v]_p) = \{q \in Pos(u) \ \mid q \not\geq_{pref} p\} \cup \{p \cdot q \mid q \in Pos(v)\}$

- $u[v]_p(q) = u(q)$ *si* $q \in Pos(u)$ *et* $q \not\geq_{pref} p$

- $u[v]_p(p \cdot q) = v(q)$ *si* $q \in Pos(v)$

On vérifie aisément que $u[v]_p$ est bien un terme (et qu'il est fini dès que u et v le sont).

Exemple 2.8 *Dans l'exemple 2.2,* $t_3 \equiv eq(x, x+0) \wedge true$, $t_1 \equiv s(0)+0$ *et* $t_2 \equiv x+x$. $t_3[t_2]_{1\cdot 1} \equiv eq(x+x, x+0) \wedge true$. $t_3[t_2[t_2[t_1]_1]_1]_{1\cdot 2} \equiv eq(x, ((s(0)+0)+x)+x) \wedge true$.

2.1.6 Généralisations

Il est souvent utile d'introduire des symboles d'arité variable. La fonction de typage τ associe aux symboles dont l'arité est variable une paire de symboles de sorte et l'on écrit

$$f : \underline{s}^* \to \underline{s}'$$

le codomaine de f est la sorte $\underline{s}'$.

La définition des termes est alors modifiée de la façon suivante: un terme est une application d'un ensemble de positions $Pos(t)$ dans F telle que, si $t(p) = f$ et f est d'arité variable, alors $\forall i \in \mathbf{N}_+$ tel que $p \cdot i \in Pos(t)$, alors $t(p \cdot i)$ est étiqueté par un symbole de sorte $\underline{s}$.

Toutes les notions définies précedemment s'appliquent sans difficulté à cette nouvelle définition.

2.2 F-algèbres

Nous complétons ici les définitions syntaxiques qui précèdent par des notions sémantiques, en précisant ce que peut être une *interprétation* des symboles de F. En fait, les algèbres de termes sont des interprétations particulières jouant un rôle essentiel qui sera étudié par la suite.

Définition 2.9 *Soit* (S, F) *une signature. Une* F-algèbre $\mathcal{A}$ *est constituée*

- *pour chaque* $\underline{s} \in S$ *d'un ensemble non vide* $\mathcal{A}_{\underline{s}}$

- *pour chaque* $f \in F$, $f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$, *d'une application* $f_{\mathcal{A}}$ *de* $\mathcal{A}_{\underline{s}_1} \times \ldots \mathcal{A}_{\underline{s}_n}$ *dans* $\mathcal{A}_{\underline{s}}$.

Exemple 2.10 $T(F)$, $T(F, X)$, $RT(F)$, $RT(F, X)$, $IT(F)$, $IT(F, X)$ *sont des* F*-algèbres, puisque nous avons supposé que* $T(F)$ *contient au moins un terme de chaque sorte. L'interprétation des symboles de fonction est, par exemple,* $f_{T(F)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. *Ces algèbres étant construites à partir de la signature, on les qualifie généralement de syntaxiques. On parle également d'algèbres de termes.*

Exemple 2.11 *$S = \{\underline{s}\}$ et $F = \{0 : \to \underline{s};\ s : \underline{s} \to \underline{s};\ + : \underline{s} \times \underline{s} \to \underline{s}\}$.*

$(\mathbf{N}, 0_{\mathbf{N}}, succ_{\mathbf{N}}, +_{\mathbf{N}})$ *est une F-algèbre. (Ici $succ_{\mathbf{N}}$ désigne le successeur: $succ_{\mathbf{N}}(n) = n +_{\mathbf{N}} 1$).*

$(\mathbf{Q}_*, 1, \div 2, \div)$ *est une F-algèbre.*

Lorsque la signature contient des symboles $f : \underline{s}^* \to \underline{s}'$ dont l'arité n'est pas définie, leur interprétation $f_{\mathcal{A}}$ dans une F-algèbre $\mathcal{A}$ est une application des suites d'éléments de $\mathcal{A}_{\underline{s}}$ dans $\mathcal{A}_{\underline{s}'}$.

On dit que A est une partie de la F-algèbre $\mathcal{A}$ (et l'on note $A \subseteq \mathcal{A}$ si A est une famille d'ensembles non-vides $\{A_{\underline{s}}\ |\underline{s} \in S\}$ telle que, pour tout $\underline{s} \in S$, $A_{\underline{s}} \subseteq \mathcal{A}_{\underline{s}}$. Une *sous-algèbre* de la F-algèbre $\mathcal{A}$ est une partie $\mathcal{B}$ de $\mathcal{A}$ telle que, pour tout $f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$ dans F et tous $t_1, \ldots, t_n \in \mathcal{B}_{\underline{s}_1} \times \ldots \times \mathcal{B}_{\underline{s}_n}$, $f_{\mathcal{A}}(t_1, \ldots, t_n) \in \mathcal{B}$ de telle sorte que l'on puisse poser $f_{\mathcal{B}}(t_1, \ldots, t_n) = f_{\mathcal{A}}(t_1, \ldots, t_n)$, faisant ainsi de $\mathcal{B}$ uns F-algèbre.

Une *précongruence* est une relation binaire $\sim$ sur une F-algèbre $\mathcal{A}$ telle que, pour tout $f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$ dans F, pour tous termes $t_1, \ldots, t_n, u_1, \ldots, u_n$ (t_i et u_i étant de même sorte pour tout i),

$$t_1 \sim u_1 \wedge \ldots \wedge t_n \sim u_n \quad \Rightarrow \quad f(t_1, \ldots, t_n) \sim f(u_1, \ldots, u_n)$$

La restriction à la sous-algèbre $\mathcal{B}$ d'une précongruence de l'algèbre $\mathcal{A}$ est bien-sûr une précongruence de $\mathcal{B}$.

Une précongruence $\sim$ est une *congruence* si c'est une relation d'équivalence[1].

Lorsque $\sim$ est une congruence sur la F-algèbre $\mathcal{A}$, l'ensemble des classes d'équivalence modulo $\sim$, $\mathcal{A}/_{\sim}$, est muni canoniquement d'une structure de F-algèbre par :

- $(\mathcal{A}/_{\sim})_{\underline{s}}$ est l'ensemble des classes des termes de sorte $\underline{s}$: $(\mathcal{A}/_{\sim})_{\underline{s}} = \{\overline{t} \mid t \in \mathcal{A}_{\underline{s}}\}$, $\overline{t}$ désignant la classe d'équivalence de t,
- $f_{\mathcal{A}/_{\sim}}(\overline{t_1}, \ldots, \overline{t_n}) = \overline{f_{\mathcal{A}}(t_1, \ldots, t_n)}$.

On vérifie aisément que cette définition ne dépend pas des représentants choisis : c'est une conséquence du fait que $\sim$ est une congruence. Par la suite, on utilisera très fréquemment cette construction en l'appliquant à la F-algèbre libre $T(F, X)$.

2.3 Homomorphismes

Définition 2.12 *Soient $\mathcal{A}$ et $\mathcal{B}$ deux F-algèbres. Un* homomorphisme *de $\mathcal{A}$ dans $\mathcal{B}$ est un ensemble d'applications $\{h_{\underline{s}} \mid \underline{s} \in S\}$ tel que, pour toute sorte $\underline{s} \in S$, $h_{\underline{s}}$ est une application de $\mathcal{A}_{\underline{s}}$ dans $\mathcal{B}_{\underline{s}}$ et, si $f : \underline{s}_1 \times \ldots \times \underline{s}_n \to \underline{s}$, alors*

$$\forall a_1, \ldots, a_n \in \mathcal{A},\ \ h_{\underline{s}}(f_{\mathcal{A}}(a_1, \ldots, a_n)) = f_{\mathcal{B}}(h_{\underline{s}_1}(a_1), \ldots, h_{\underline{s}_n}(a_n))$$

Un *endomorphisme* est un homomorphisme d'une F-algèbre dans elle-même. Un *isomorphisme* est un homomorphisme bijectif. Un *automorphisme* est un endomorphisme bijectif.

[1] c'est à dire vérifiant les propriétés $\forall t_1, t_1 \sim t_1$ (réflexivité), $\forall t_1, t_2,\ t_1 \sim t_2 \Rightarrow t_2 \sim t_1$ (symétrie) et $\forall t_1, t_2, t_3,\ t_1 \sim t_2 \wedge t_2 \sim t_3 \Rightarrow t_1 \sim t_3$ (transitivité)

Exemple 2.13 $F = \{0 : \to \underline{s};\;\; g : \underline{s} \to \underline{s};\;\; f : \underline{s} \times \underline{s} \to \underline{s}\}$. *L'application qui à tout $z \in \mathbf{Z}$ associe son opposé est un isomorphisme de $(\mathbf{Z}, 0, +1, +)$ dans $(\mathbf{Z}, 0, -1, +)$. La vérification est laissée en exercice.*

Munie de ses homomorphismes, la classe des F-algèbres forme une catégorie. Nous pourrons à l'occasion utiliser le vocabulaire catégorique. Les algèbres de termes ont une propriété particulière vis à vis des homomorphismes, identique à la propriété des polynômes sur un anneau commutatif unitaire :

Theorem 2.14 (propriété universelle de $T(F, X)$)
Soit $\mathcal{A}$ une F-algèbre. Soit i l'injection (canonique) de X dans $T(F, X)$. Pour toute application f de X dans $\mathcal{A}$, telle que, si x est de sorte $\underline{s}$ alors $f(x)$ est de sorte $\underline{s}$, il existe un unique homomorphisme $\widehat{f}$ de $T(F, X)$ dans $\mathcal{A}$ tel que

$$\forall x \in X,\;\; \widehat{f} \circ i(x) = f(x)$$

La construction de $\widehat{f}$ est effectuée par récurrence structurelle sur les termes, ou, si l'on préfère, par récurrence sur la taille des termes.

Si $\mathcal{A}$ est une F-algèbre, on appelle $\mathcal{A}$-*assignation* toute application σ de X dans $\mathcal{A}$. D'après le théorème 2.14, on confond σ et l'homomorphisme $\widehat{\sigma}$ de $T(F, X)$ dans $\mathcal{A}$ correspondant. On note $t\sigma$ l'application d'une assignation σ à t.

Les endomorphismes de $T(F, X)$ (resp. de $RT(F, X)$, resp. de $IT(F, X)$) sont appelés *substitutions*. On note Σ l'ensemble des substitutions (de façon ambigüe, la même notation est employée pour les termes finis, rationnels et infinis). Les homomorphismes de $T(F, X)$ dans $T(F)$ (resp. de $RT(F, X)$ dans $RT(F)$, resp. de $IT(F, X)$ dans $IT(F)$) sont appelés *substitutions closes*. L'ensemble des substitutions closes est noté Σ_g. Si σ est une substitution, on appelle *domaine* de σ l'ensemble $\{x \in X \mid x\sigma \not\equiv x\}$ noté $Dom(\sigma)$.

2.3.1 Substitutions et subsumption

Une substitution de domaine fini sera souvent noté en extension: $\{x_1 \to t_1, \ldots, x_n \to t_n\}$ est la substitution de domaine $Dom(\sigma) = \{x_1, \ldots, x_n\}$ telle que $\forall i \in [1..n],\; x_i\sigma \equiv t_i$. Bien sûr, on suppose ici que $x_1, \ldots, x_n$ sont des variables distinctes et que, pour tout i, x_i est de même sorte que t_i. Cette notation est aussi employée pour les $\mathcal{A}$-assignations dont les seules valeurs significatives sont celles qui sont prises sur $\{x_1, \ldots, x_n\}$.

Lorsque σ est de domaine fini, on note aussi $VIm(\sigma)$ le sous-ensemble de X :

$$VIm(\sigma) = \bigcup_{x \in Dom(\sigma)} Var(x\sigma)$$

Un *renommage* (ou *conversion* comme en λ-calcul) est une substitution σ qui associe à toute variable une variable et qui est une bijection de $Dom(\sigma)$ sur $VIm(\sigma)$. On supposera toujours implicitement que le domaine d'un renommage contient les variables des termes auxquels il est appliqué.

Une substitution σ est dite *idempotente* si $\sigma \circ \sigma = \sigma$. Toute substitution close est idempotente.

Définition 2.15 *Deux termes t_1, t_2 sont dits* semblables *lorsqu'ils s'échangent par renommage. On écrit dans ce cas $t_1 \doteq t_2$. La relation s'étend aux substitutions comme attendu.*

La relation $\doteq$ est une relation d'équivalence sur les termes, et par conséquent sur les substitutions.

Définition 2.16 *Le terme s est dit* plus général *que le terme t, ce qui est noté $s \preceq t$ s'il existe une substitution ρ telle que $t = s\rho$.. La substitution σ est dite* plus générale *que la substitution τ, encore noté $\sigma \preceq \tau$ s'il existe une substitution ρ telle que $\tau = \sigma\rho$.*

On dit aussi que t (resp. τ) est une instance *de s (resp. de σ).*

Proposition 2.17 *La relation de subsumption $\preceq$ (sur les termes finis ou infinis, sur les substitutions de termes finis ou infinis) est un préordre dont l'équivalence est la similitude $\simeq$.*

Si t est un terme fini (resp. un terme rationnel), on note $[\![t]\!]$ (resp. $[\![t]\!]_{RT(F)}$) l'ensemble des instances closes finies (resp. rationnelles) de t. La propriété qui suit est fondamentale : elle donne la signification d'un terme avec variable comme l'ensemble de ses instances closes. Ce point de vue sera étudié plus en détail dans les chapitres 8 et 12.

Proposition 2.18 *Étant donnés deux termes s et t, on a :*

(i) $s \preceq t$ ssi $[\![t]\!] \subseteq [\![s]\!]$ et

(ii) $s \doteq t$ ssi $[\![t]\!] = [\![s]\!]$.

Notons que la seconde propriété est une conséquence de la première.

Définition 2.19 *Deux termes s et t sont dits* unifiables *par la substitution σ appellée unificateur, si $s\sigma = t\sigma$.*

Définition 2.20 *Étant donné un ensemble E de substitutions, une substitution $\sigma \in E$ est dite* principale *si toute substitution de E est une instance de σ.*

Theorem 2.21 *L'ensemble des unificateurs de deux terms s et t, lorsqu'il n'est pas vide, possède un minimum pour l'ordre de subsumption (à renommage près), appellé* unificateur principal *ou* unificateur plus général *de s et t.*

Ce résultat sera démontré au chapitre 8.1.

L'ordre d'imbrication étend l'ordre de subsumption comme suit:

Définition 2.22 *Le terme s est imbriqué dans le terme t si il existe une position $p \in \mathcal{P}os(t)$ et une substitution σ telles que $t|_p = s\sigma$. On note par $\trianglerighteq$ la relation d'imbrication.*

Proposition 2.23 *La relation d'imbrication surles termes est un préordre dont l'équivalence est la similitude $\simeq$.*

2.4 Exercices

1. Le théorème 2.14 reste-t-il vrai si l'on remplace $T(F, X)$ par $RT(F, X)$? Pourquoi ?

2. Montrer que, pour tous arbres u, v, w et toutes positions p, p', q (telles que les expressions indiquées aient un sens):

   (a) $u[u|_p]_p \equiv u$

   (b) $u[v]_p|_p \cdot q \equiv v|_q$

   (c) $(u[v]_p)[w]_{p \cdot q} \equiv u[v[w]_q]_p$

   (d) $u[v]_p|_{p'} \equiv u|_{p'}$ si $p \bowtie p'$

   (e) $(u[v]_p)[w]_{p'} \equiv (u[w]_{p'})[v]_p$ si $p \bowtie p'$

   (f) $u[v]_{p \cdot q}|_p \equiv (u|_p)[v]_q$

   (g) $(u[v]_{p \cdot q})[w]_p \equiv u[v]_p$

3. Dans quel(s) cas $T(F)$ est il vide ? fini ?

4. Soient $t \in T(F, X)$ et $p \in Pos(t)$ et $p \neq \epsilon$. Montrer qu'il existe un et un seul arbre rationnel u tel que $u \equiv t[u]_p$.

5. Montrer que la réciproque de la proposition 2.6 est fausse: donner un alphabet fini F (où les symboles ont tous une arité fixe) et une terme de $IT(F)$ qui n'est pas rationnel alors que son ensemble de positions est un langage régulier.

6. Soit (S, F) une signature finie. Une sorte $\underline{s}$ est dite *infinitaire* (resp. *finitaire*) s'il existe une infinité (resp. un nombre fini) de termes fermés de sorte $\underline{s}$. Par extension, $t \in RT(F, X)$ est dit infinitaire s'il est de sorte infinitaire. Montrer que

   (a) Si $t \in RT(F)$ possède un sous-arbre strict de même sorte que lui, alors t est infinitaire

   (b) S'il n'y a dans $RT(F)$ qu'un nombre fini d'arbres de même sorte que $t \in RT(F)$, alors $t \in T(F)$

   (c) Si $t \in RT(F) - T(F)$, alors t est infinitaire

   (d) Il existe dans $RT(F)$ une infinité d'arbres de même sorte que t ssi t est infinitaire

7. Montrer que le logarithme népérien est un homomorphisme de F-algèbre de $]0, +\infty[$ dans $\mathbb{R}$, où $F = \{0 :\to \underline{s};\ h : \underline{s} \to \underline{s};\ f : \underline{s} \times \underline{s} \to \underline{s}\}$. On précisera les structures de F-algèbres de $]0, +\infty[$ et de $\mathbb{R}$.

8. Montrer que, s'il existe un homomorphisme de $RT(F)$ dans $T(F)$ qui est l'identité sur $T(F)$, alors $RT(F) = T(F)$. Dans quels cas cela peut-il se produire ?

9. Soit $F = \{0 :\to \underline{s};\ f : \underline{s} \times \underline{s} \to \underline{s}\}$. Donner une opération $f_{\mathbf{Z}}$ sur $\mathbf{Z}$ telle que l'application qui à tout entier associe son opposé soit un morphisme de $(\mathbf{Z}, 0, *)$ dans $(\mathbf{Z}, 0, f_{\mathbf{Z}})$ (considérés comme F-algèbres).

10. Faire la preuve du théorème 2.14.

11. Existe-t-il des renommages idempotents autres que l'identité ?

12. Soit $\mathcal{A}$ une F-algèbre.

    (a) Vérifier qu'une sous-algèbre de $\mathcal{A}$ est une F-algèbre.

    (b) Montrer que l'"intersection" de sous-algèbres, lorsque l'intersection est non vide pour chaque sorte, est une sous-algèbre. On note $[A]$ l'intersection des sous-algèbres de $\mathcal{A}$ qui contiennent A. $A \subseteq \mathcal{A}$ est un *système de générateurs* de $\mathcal{A}$ si $[A] = \mathcal{A}$.

    (c) Montrer que, $T(F)$ a un système de générateurs fini mais que $RT(F)$, s'il est distinct de $T(F)$, n'a pas de système de générateurs fini.

    (d) $A \subseteq \mathcal{A}$ est un *système libre* de $\mathcal{A}$ si, pour tous termes $t, u \in T(F, X)$ et toute $\mathcal{A}$-assignation σ ,

    $$t\sigma =_{\mathcal{A}} u\sigma \quad \text{et} \quad \forall x \in X, x\sigma \in A \quad \Rightarrow \quad t \equiv u$$

    Montrer qu'il existe dans $T(F)$ un système libre infini ssi il existe dans $RT(F)$ un système libre infini.

    (e) $A \subseteq \mathcal{A}$ est une *base* de $\mathcal{A}$ si A est est un système de générateurs libre. Une algèbre $\mathcal{A}$ est *libre* si $\mathcal{A}$ possède une base. Montrer que $T(F)$ et $T(F, X)$ sont des algèbres libres. Dans quels cas (précisément) $RT(F, X)$ et $RT(F)$ sont elles libres ? (Justifier formellement le résultat proposé).

    (f) Une algèbre $\mathcal{A}$ est *localement libre* si toute sous-algèbre de $\mathcal{A}$ finiment engendrée est libre. Montrer que toute algèbre libre est localement libre mais que la réciproque est fausse. $RT(F, X)$ et $RT(F)$ sont-elles localement libres ?

Chapter 3

Logique équationnelle

La logique équationelle est à la base des méthodes de spécification des structures de données, vues comme des structures algébriques.

On appelle équation toute paire de termes (s, t) notée $s \doteq t$.

Définition 3.1 *Une F-algèbre $\mathcal{A}$ est un* modèle *de l'ensemble d'équations E si, pour toute équation $s \doteq t$ de E et pour toute $\mathcal{A}$-assignation σ, $s\sigma =_{\mathcal{A}} t\sigma$.*

Lorsque $\mathcal{A}$ est un modèle de $s \doteq t$, on note $\mathcal{A} \models s \doteq t$.

Pour l'instant, $\doteq$ a la même signification que $=$. Par la suite, il deviendra utile de distinguer les axiomes des conjectures à prouver.

Exemple 3.2 *(S, F) est la signature de l'exemple 2.11. On considère $E = \{0 + x \doteq x;\ s(x) + y \doteq s(x + y)\}$.*

$\mathcal{A}_1 = (\mathbf{N}, 0_{\mathbf{N}}, succ_{\mathbf{N}}, +_{\mathbf{N}})$ et $\mathcal{A}_2 = (\mathbf{Q}, 1_{\mathbf{Q}}, *2_{\mathbf{Q}}, \div_{\mathbf{Q}})$ sont des modèles de E. (Vérification en exercice).

Un triplet (S, F, E) où E est un ensemble fini d'équations est appelé *spécification équationnelle.*

3.1 La congruence $=_E$

Soit E un ensemble d'équations. On note $=_E$ la plus petite congruence telle que

$$\forall \sigma \in \Sigma,\ \forall s \doteq t \in E,\ s\sigma =_E t\sigma$$

L'existence et l'unicité d'une telle relation sont laissées en exercice.

Remarquons que, comme $T(F)$ est une sous-algèbre de $T(F, X)$, la restriction de $=_E$ à $T(F)$ définit encore une congruence sur $T(F)$.

Remarquons également que $T(F, X)/_{=_E}$ est (canoniquement) munie d'une structure de F-algèbre, la construction a été donnée dans le paragraphe 2.2.

Exemple 3.3 *Reprenons l'exemple 3.2. $E = \{0 + x \doteq x;\ s(x) + y \doteq s(x + y)\}$. L'application qui associe à tout $n \in \mathbf{N}$ la classe de $s^n(0)$ dans $T(F)/_{=_E}$ est un isomorphisme de F-algèbres. ($\mathbf{N}$ est muni de sa structure "naturellé' de F-algèbre). La preuve est laissée en exercice.*

Proposition 3.4 *Soit (S, F, E) une spécification équationnelle. Pour toute F-algèbre $\mathcal{A}$ telle que $\mathcal{A} \models E$, il existe un unique homomorphisme h de $T(F)/_{=_E}$ dans $\mathcal{A}$.*

Cette proposition établit que $T(F)/_{=_E}$ est une algèbre initiale dans la catégorie des F-algèbres qui sont des modèles de E. C'est pourquoi nous emploierons parfois le mot "algèbre initialé" pour désigner $T(F)/_{=_E}$.

La preuve de cette proposition est laissée en exercice. C'est une conséquence du théorème 2.14 et du fait que $T(F)$ contient au moins un terme de chaque sorte.

On note $E \models s = t$ lorsque toute F-algèbre qui satisfait E, satisfait aussi $s = t$.

3.2 Problèmes du mot, d'unification, inductifs

Certains problèmes de validité présentent un intérêt particulier. Soient (S, F, E) une spécification équationnelle, s et t deux termes de $T(F, X)$ de même sorte.

Un problème du mot consiste à décider si $E \models s = t$, c'est à dire à décider si l'équation $s = t$ est valide dans tout modèle de E.

> **Exemple 3.5** *Avec les notations des exemples précédents,*
>
> $$\{s(x) + y \doteq s(x + y);\ 0 + x \doteq x\} \quad \models \quad s(s(x)) + y = s(s(x + y))$$

Un problème d'unification sémantique consiste à trouver, toutes les substitutions σ telles que

$$E \models t\sigma = s\sigma$$

Dans le cas où $E \neq \emptyset$ on parle généralement de E-unification, le mot "unification" étant réservé au cas où $E = \emptyset$.

> **Exemple 3.6** $F = \{a, b : \to \underline{s};\ + : \underline{s} \times \underline{s} \to \underline{s}\}$ *et* $E = \{x + y \doteq y + x\}$. *Le problème d'unification* $a + x = y + z$ *a pour solutions* $\sigma_0 = \{y \to a;\ z \to x\}$, $\sigma_1 = \{z \to a;\ y \to x\}$ *ainsi que toutes les substitutions de la forme* $\sigma \circ \sigma_0$ *ou* $\sigma \circ \sigma_1$.

Un problème inductif consiste à décider si $T(F)/_{=_E} \models s = t$.

> **Exemple 3.7** $E = \{0 + x \doteq x;\ s(x) + y \doteq s(x + y)\}$. $T(F)/_{=_E} \models x + 0 = x$, *mais* $E \not\models x + 0 = x$. *(cf exercices).*

3.2.1 Raisonnement équationnel

Le raisonnement équationnel, ou remplacement d'égaux par égaux, est un système de déduction très ancien permettant de résoudre le problème du mot. Dans notre logique, les formules sont des équations, implicitement quantifiées universellement, et les règles d'inférence sont données dans la figure 3.1. Les trois premières règles ne font qu'exprimer que l'égalité est une relation d'équivalence. La quatrième règle

$$\frac{}{s = s} \quad \text{Réflexivité}$$

$$\frac{s = t}{t = s} \quad \text{Symétrie}$$

$$\frac{s = t \quad t = u}{s = u} \quad \text{Transitivité}$$

$$\frac{s = t}{u[s\sigma]_p = u[t\sigma]_p} \quad \text{Remplacement}$$

Figure 3.1: Règles d'inférence de la logique équationnelle

est le remplacement d'égaux par égaux proprement dit: si $s = t$, on peut remplacer $s\sigma$ par $t\sigma$ à la position p d'un terme u.

Si E est un ensemble d'équations telle qu'il existe une règle d'inférence dont les prémisses sont dans E et la conclusion est $s = t$, on note $E \vdash \{s = t\} \cup E \vdash$ est généralement appelée "relation de déduction". On ne répète pas toujours E dans la conclusion de la relation de déduction. Si un ensemble d'équations se déduit de E en plusieurs étapes de déduction, on note encore $E \vdash$. (Si bien que $\vdash$ est transitive)

Un résultat ancien (Birkhoff 1933) établit l'adéquation de ce système de règles d'inférence dans le cas où chaque sorte contient au moins un terms clos :[1]

Theorem 3.8

$$s =_E t \;\Leftrightarrow\; E \vdash s = t \;\Leftrightarrow\; E \models s = t$$

Autrement dit, une équation $s = t$ est prouvable en utilisant les règles de la figure 3.1 ssi elle est valide dans tous les modèles de E. Le cas où certains sortes sont vides pose un problème de correction des règles dont la solution demande de manipuler explicitement les quantificateurs.

Preuve

La première équivalence exprime que $=_E$ est le plus petit point fixe de la relation de déduction appliquée à E.

Si $E \vdash s = t$, alors $E \models s = t$. Ce résultat est prouvé par récurrence sur la longueur de la déduction de $s = t$. Si cette longueur est 0, alors $s \equiv t$ et $E \models s = t$. Si $E \vdash E_1 \vdash s = t$, la dernière déduction ayant lieu en une seule étape, alors $E \models E_1$ et $s = t$ se déduit de E_1 par l'une des règles de la figure 3.1. L'égalité dans tout

[1] Un système de règles d'inférence est dit *correct* si $\phi \vdash \psi \Rightarrow \phi \models \psi$, *complet* si $\phi \models \psi \Rightarrow \phi \vdash \psi$ et *adéquat* s'il est correct et complet.

modèle de E est une congruence et, par définition des modèles, on peut appliquer n'importe quelle substitution sur les variables, tout en préservant sa validité. D'où $E \models s = t$.

Si $E \models s = t$, soit alors $\mathcal{A} = T(F,X)/_{=_E}$. $\mathcal{A} \models E$ donc, par hypothèse, $\mathcal{A} \models s = t$ et ainsi $s =_E t$. □

Exemple 3.9
La figure 3.2 montre un exemple de déduction dans la logique équationnelle du théorème bien connu en mathématiques: " Tout groupe d'ordre 2 est commutatif". Dans cet exemple, $F = \{e : \to \underline{s};\ * : \underline{s} \times \underline{s} \to \underline{s}\}$ *et* E *est l'ensemble d'équations:*

$(E1)$	$x * (y * z) \doteq (x * y) * z$	*(associativité)*
$(E2)$	$x * e \doteq x$	*(élément neutre)*
$(E3)$	$e * x \doteq x$	
$(E4)$	$x * x \doteq e$	*(tout élément est d'ordre 2)*

La preuve de la figure 3.2 présente alors la déduction $\{E1, E2, E3, E4\} \vdash x{*}y = y{*}x$.

A la place de la notation par règles d'inférence, nous utiliserons aussi la relation $\leftrightarrow_E$définie par: $\leftrightarrow_E$ est la plus petite précongruence réflexive sur $T(F,X)$ telle que $s \doteq t \in E \Rightarrow \forall \sigma \in \Sigma, s\sigma \leftrightarrow_E t\sigma$.

Remarques

- comme $=$ (et $\doteq$) est symétrique, il en est de même de $\leftrightarrow_E$.
- $=_E$ est la fermeture transitive de $\leftrightarrow_E$. (i.e. la plus petite relation transitive sur $T(F,X)$ qui contienne E).
- Le théorème de Birkhoff peut se reformuler:

$$E \models s = t \quad \Leftrightarrow \quad \exists t_1, \ldots, t_n \in T(F,X),\ s \equiv t_1 \leftrightarrow_E t_2 \leftrightarrow_E \ldots \leftrightarrow_E t_n \equiv t$$

La *théorie équationnelle* définie par la spécification (S, F, E) est l'ensemble des équations $s = t$ de $T(F,X)$ telles que $E \vdash s = t$.

On note TH_E la théorie équationnelle de E. C'est un ensemble récursivement énumérable si X, F, E, S le sont.

3.3 Exercices

1. Soit $F = \{e :\to \underline{s};\ I : \underline{s} \to \underline{s};\ * : \underline{s} \times \underline{s} \to \underline{s}\}$. Soit $\mathcal{A}$ un modèle de $E = \{e * x \doteq x;\ I(x) * x \doteq e;\ x * (y * z) \doteq (x * y) * z\}$. Prouver que $\mathcal{A} \models x * e = x$.

2. Soit $F = \{0 :\to \underline{nat};\ s : \underline{nat} \to \underline{nat};\ + : \underline{nat} \times \underline{nat} \to \underline{nat}\}$ et soit E l'ensemble d'axiomes $E = \{0 + x \doteq x;\ s(x) + y \doteq s(x + y)\}$. Donner une F-algèbre $\mathcal{A}$ telle que $\mathcal{A} \models E$ et $\mathcal{A} \not\models x + 0 = x$. Montrer que $T(F)_{=_E} \models x + 0 = x$

$$\dfrac{\dfrac{\dfrac{E4}{((x*y)*(x*y))*y = e*y} \qquad \dfrac{E3}{e*y = y}}{((x*y)*(x*y))*y = y} \qquad \dfrac{\begin{array}{cc} E1 & E1 \\ \vdots & \vdots \end{array}}{((x*y)*(x*y))*y = ((x*y)*x)*(y*y)}}{((x*y)*x)*(y*y) = y}$$

$$\dfrac{\begin{array}{c} \vdots \\ ((x*y)*x)*(y*y) = y \end{array} \qquad \dfrac{\begin{array}{c} E4 \\ \vdots \end{array}}{((x*y)*x)*(y*y) = ((x*y)*x)*e}}{((x*y)*x)*e = y}$$

$$\dfrac{\begin{array}{c} \vdots \\ ((x*y)*x)*e = y \end{array} \qquad \dfrac{E2}{((x*y)*x)*e = (x*y)*x}}{(x*y)*x = y}$$

$$\dfrac{\dfrac{\begin{array}{c} \vdots \\ (x*y)*x = y \end{array}}{((x*y)*x)*x = y*x} \qquad \dfrac{\begin{array}{cc} E1 & E1 \\ \vdash & \vdash \end{array}}{((x*y)*x)*x = (x*y)*(x*x)}}{(x*y)*(x*x) = y*x}$$

$$\dfrac{\begin{array}{c} \vdots \\ (x*y)*(x*x) = y*x \end{array} \qquad \dfrac{\dfrac{E4}{(x*y)*(x*x) = (x*y)*e} \qquad \dfrac{E2}{(x*y)*e = x*y}}{(x*y)*(x*x) = x*y}}{x*y = y*x}$$

Figure 3.2: Preuve en logique équationnelle du théorème : “tout goupe d’ordre 2 est commutatif”

3. L'intersection d'une famille de relation $(R_i)_{i \in I}$ définies sur l'ensemble $\mathcal{E}$ est la relation R définie sur $\mathcal{E}$ par :

$$aRb \Leftrightarrow \forall i \in I, aR_ib$$

   Montrer que l'intersection de précongruences est une précongruence. En déduire une (autre) définition de $=_E$.

4. $F = \{0 :\to \underline{nat};\ s : \underline{nat} \to \underline{nat}; +\underline{nat} \times \underline{nat} \to \underline{nat}\}$ et $E = \{0 + x \doteq x;\ s(x) + y \doteq s(x + y)\}$. Montrer que l'application qui associe à tout $n \in \mathbf{N}$ la classe de $s^n(0)$ dans $T(F)/_{=_E}$ est un isomorphisme de F-algèbres. ($\mathbf{N}$ est muni de sa structure "naturellé' de F-algèbre).

5. Prouver la proposition 3.4. La condition "E fin' est elle nécessaire ? La condition sur $T(F)$ de contenir au moins un terme de chaque sorte est elle nécessaire ?

6. Soit E un ensemble d'équations.
   (a) Soit $\mathcal{A}$ un modèle de E, f une application de X dans $\mathcal{A}$, i l'injection de X dans $T(F, X)$ et s l'application qui à tout terme de $T(F, X)$ associe sa classe modulo $=_E$. Montrer qu'il existe un unique homomorphisme h de $T(F, X)/_{=_E}$ dans $\mathcal{A}$ tel que $\forall x \in X,\ h(s(i(x))) =_{\mathcal{A}} f(x)$.
   (b) Montrer que, pour tout modèle $\mathcal{A}$ de E, il existe un morphisme de $T(F, X)/_{=_E}$ dans $\mathcal{A}$ mais que celui-ci n'est pas unique.
   (c) Montrer que toutes les algèbres initiales sont isomorphes. C'est à dire montrer que, si $\mathcal{A}_1$ et $\mathcal{A}_2$ vérifient: pour toute F-algèbre $\mathcal{A}$, il existe un unique homomorphisme de $\mathcal{A}_1$ dans $\mathcal{A}$ et un unique homomorphisme de $\mathcal{A}_2$ dans $\mathcal{A}$, alors $\mathcal{A}_1$ et $\mathcal{A}_2$ sont isomorphes.

Chapter 4

Calculs par Réécriture

Les preuves equationelles sont fortement non-déterministes, donc se prètent mal au calcul. Afin de les déterminiser, nous allons orienter les équations en des règles de réécriture. Les preuves auront alors la forme $\longrightarrow^* \longleftarrow^*$.

Définition 4.1 *Une* règle de réécriture *est une paire (l, r) notée $l \to r$. Un ensemble de règles de réécriture $R = \{l_i \to r_i\}_i$ est appellé* système de réécriture. *On notera par $E_R = \{l_i = r_i\}_i$ l'ensemble des équations associées à R.*

Étant donné un système de réécriture R, on dit que le terme s se réécrit en le terme t à la position $p \in \mathcal{P}os(s)$, noté $s \longrightarrow_R^p t$, s'il existe une règle $l \to r \in R$ et une substitution σ telles que $s|_p = l\sigma$ et $t = s[r\sigma]_p$. On pourra omettre p et R, mais aussi préciser la règle $l \to r$ utilisée.

On notera par $\longrightarrow_R^*$ la fermeture réflexive transitive de $\longrightarrow_R$. Les definitions et résultats qui suivent s'appliquent en fait à des relations arbitraires sur des ensembles arbitraires.

Définition 4.2 *Étant donnée une relation de réécriture $\longrightarrow_R$, on dit que s est R-*irreductible, *ou en R-*forme normale, *s'il n'existe aucun t tel que $s \longrightarrow_R t$, et que le terme R-irréductibe t est une forme normale de s si $s \longrightarrow_R^* t$.*

4.1 Propriétés de confluence

En pratique, la réécriture sert à tester l'égalité de deux termes par calcul de leur forme normale. La propriété sous-jacente s'appelle propriété de Church-Rosser. Nous définissons en même temps trois propriétés très proches, la propriŕté de Church-Rosser, la confluence et la confluence locale. Ces propriétés sont représentées à la figure 4.1.

Définition 4.3 *Le système de réécriture R est* Church-Rosser *si pour tout couple de termes (s, t), $s \longleftrightarrow_{E_R}^* t$ si et seulement si il existe un terme v tel que $s \longrightarrow_R^* v$ et $t \longrightarrow_R^* v$.*

Le système de réécriture R est confluent *si pour tout triplet de termes (s, u, t) tel que $u \longrightarrow_R^* s$ et $u \longrightarrow_R^* t$, alors il existe un terme v tel que $s \longrightarrow_R^* v$ et $t \longrightarrow_R^* v$.*

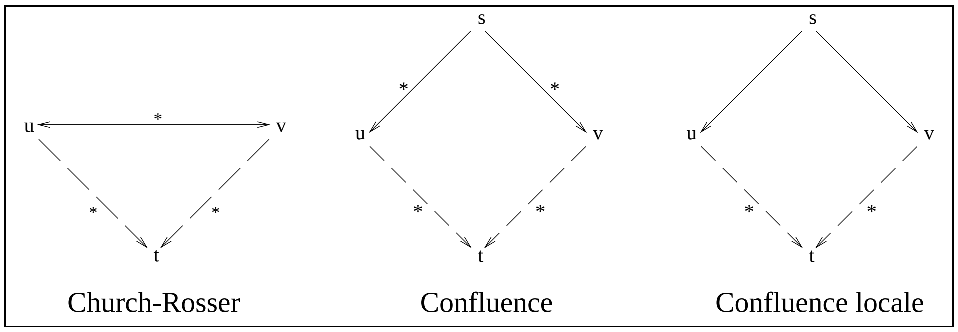

Figure 4.1: Propriétés abstraites des relations

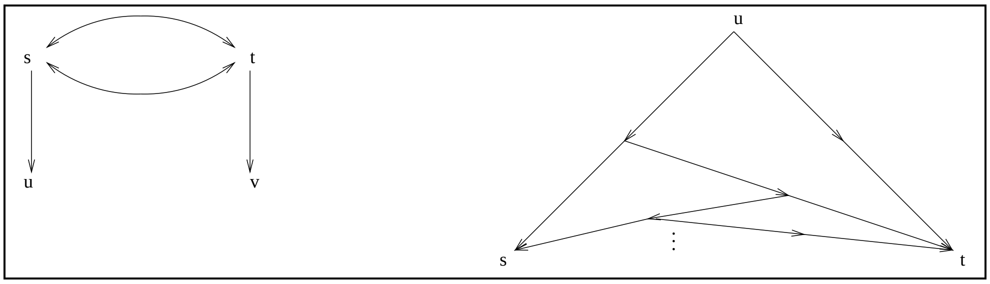

Figure 4.2: Confluence versus confluence locale

Le système de réécriture R est localement confluent *si pour tout triplet de termes (s, u, t) tel que $u \longrightarrow_R s$ et $u \longrightarrow_R t$, alors il existe un terme v tel que $s \longrightarrow_R^* v$ et $t \longrightarrow_R^* v$.*

Lemme 4.4 *Un système de réécriture est Church-Rosser si et seulement si il est confluent.*

Preuve
Par récurrence sur la longueur des preuves équationelles.

Par contre, il n'est pas vrai que confluence et confluence locale soient des propriétés équivalentes, comme le montre l'exemple de la figure 4.2. La non confluence des relations localement confluentes considérées provient de l'existence d'un cycle dans le premier cas et de chemins infinis dans le second.

Définition 4.5 *On dit qu'une relation de réécriture $\longrightarrow_R$* termine*, ou est* noetherienne*, ou est* fortement normalisante *s'il n'existe aucune suite infinie de réécritures de la forme $s_0 \longrightarrow_R s_1 \longrightarrow_R \ldots \longrightarrow_R s_n \longrightarrow_R \ldots$. Une relation à la fois noetherienne et confluente est dite* convergente*.*

Terminaison et confluence sont des propriétés indécidables de la réécriture de termes. Nous verrons toutefois au chapitre 5 comment prouver qu'une relation de réécriture termine. Nous allons maintenant nous intéresser à la relation entre

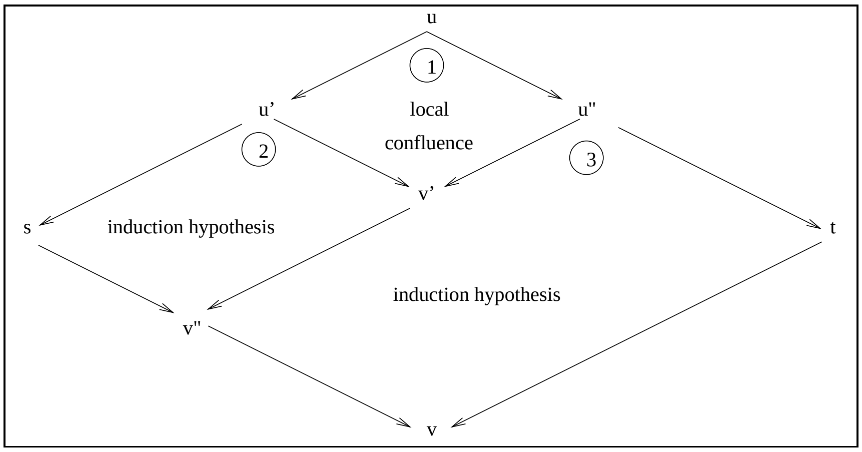

Figure 4.3: Lemme de Newmann

confluence et confluence locale avant d'aborder la technique majeure de preuve de confluence au paragraphe 4.2.

Theorem 4.6 *(Lemme de Newmann) Une relation de réécriture noetherienne est confluente si et seulement si elle est localement confluente.*

Preuve
On montre le cas si par récurrence noetherienne sur la relation de réécriture. Dans le cas général, $u \longrightarrow u' \longrightarrow s$ et $u \longrightarrow u'' \longrightarrow s$. Par propriété de confluence locale, il existe v' tel que $u' \longrightarrow^* v'$ et $u'' \longrightarrow^* v'$. Par hypothèse de récurrence appliquée à u', il existe v'' tel que $s \longrightarrow^* v''$ et $v' \longrightarrow^* v''$. Par hypothèse de récurrence appliquée à u'', il existe v tel que $v'' \longrightarrow^* v$ et $t \longrightarrow^* v$. Cette preuve est représentée à la figure 4.3.

4.2 Paires Critiques

Nous allons maintenant prouver la décidabilité de la confluence locale, donc de la confluence, en supposant la terminaison.

Définition 4.7 *On appelle* paire critique *de la règle $g \to d$ sur un renommage de la règle $l \to r$ à la position $p \in \mathcal{FP}os(l)$ tel que $\mathcal{V}ar(l) \cup \mathcal{V}ar(g) = \emptyset$, l'équation $r\sigma = l\sigma[d\sigma]_p$) telle que σ soit un unificateur principal de g et $l|_p$.*

Exemple 4.8 *Soit $R = \{x + y) + z \to x + (y + z)\}$. Renommant cette règle en $(x' + y') + z' \to x' + (y' + z')$, les deux termes $(x + y) + z$ et $(x' + y')$ s'unifient avec l'unificateur principal $\{x' \mapsto x + y,\ y' \mapsto z\}$. La paire critiques est donc l'équation $(x + y) + (z + z') = (x + (y + z)) + z'$.*

Theorem 4.9 *Un systeme de réécriture R est localement confluent ssi ses paires critiques sont confluentes.*

Preuve
Soit $u \longrightarrow^p_{l\to r} s$ et $u \longrightarrow^q_{g\to d} t$, avec $u|_p = l\sigma$ et $u|_q = g\sigma$, en supposant que $\mathcal{V}ar(l) \cap \mathcal{V}ar(g) = \emptyset$. On montre l'existence d'un terme v tel que $s \longrightarrow^* v$ et $t \longrightarrow^* v$ par cas suivant les positions respectives de p et q. La preuve est représentée (sous forme simplifiée) à la figure 4.4.

- redex disjoints, c'est à dire $p \bowtie q$: les deux réécritures commutent.
- redex ancêtre, c'est à dire $q = pq'q''$ tel que $l|_{q'} = x \in \mathcal{V}ar(l)$. Le problème est que $t|_p$ n'est plus une instance de l si la variable x a plus d'une occurrence dans l. Il faut donc d'abord réécrire les termes $x\sigma$ aux autres occurrences de x dans l. Soit $\{q'_1, \ldots, q'_n\} \subseteq \mathcal{D}om(l)$ l'ensemble des positions différentes de q' telles que $l|_{q'_i} = x$ pour tout $i \in [1..n]$. Soit maintenant la substitution σ' telle que $x\sigma' = x\sigma[d\sigma]_{q'}$ et $y\sigma' = y\sigma$ pour toute variable $y \neq x$. On a: $t \longrightarrow^* v' = t[x\sigma[d\sigma]_{q'}]_{\{q'_1, \ldots, q'_n\}} = t[l\sigma']_p \longrightarrow t[r\sigma']_p = v$. Par ailleurs, $u[l\sigma]_p \longrightarrow u[r\sigma]_p \longrightarrow^* u[r\sigma']_p$, et l'on conclue la preuve en remarquant que $u[\;]_p = t[\;]_p$.
- redex superposés, c'est à dire $q = pq'$ tel que $q' \in \mathcal{FD}om(l)$. Dans ce cas, $u|_q = l\sigma|_{q'} = g\sigma$, et donc σ est unificateur de $l|_{q'}$ et g. Soit maintenant $\sigma = \theta\tau$, où θ est l'unificateur principal de ces deux termes. Alors $s = u[(l\theta)\tau]_p$ et $t = u[(l\theta[g\theta]_{q'})\tau]_p$. Comme $(l\theta) \longrightarrow^* v \longleftarrow^* l\theta[g\theta]_{q'}$, on en déduit que $s \longrightarrow^* s[v]_p = u[v]_p = t[v]_p \longleftarrow^* t$.

Exemple 4.10 *Considérons le système de réécriture $R = \{(x+y)+z \to x+(y+z),\ (x+0) \to x\}$, et calculons ses paires critiques. On voit figure 4.5 que l'une d'elle n'est pas confluente, et donc la réécriture engendrée par les règles d'associative à droite et d'identité ne conflue pas. Au contraire, la seule règle d'associativité est confluente.*

Corollaire 4.11 *Soit R un système de réécriture noetherien. Alors R est confluent si et seulement si pour toute paire critique (p,q), $p{\downarrow} = q{\downarrow}$.*

La confluence d'une relation de réécritures est donc décidable sous l'hypothèse de terminaison.

4.3 Algèbre des formes normales

Lorsque la relation de réécriture est confluente et noetherienne, alors tout terme clos possède une forme normale unique. Mais la propriété de forme normale unique est plus faible: le systeme de règles $\{a \to a, a \to b\}$ possède trivialement cette propriété sans être pour autant noetherien. En pratique, toutefois, il sera difficile de prouver cette proprété sans l'hypothèse de terminaison des calculs, car la confluence est indécidable.

Une propriété remarquable est alors que l'ensemble des formes normales closes peut être munie d'une structure d'algèbre, et que cette algèbre est initiale dans la classe des algèbres qui satisfont les axiomes équationnels considérés:

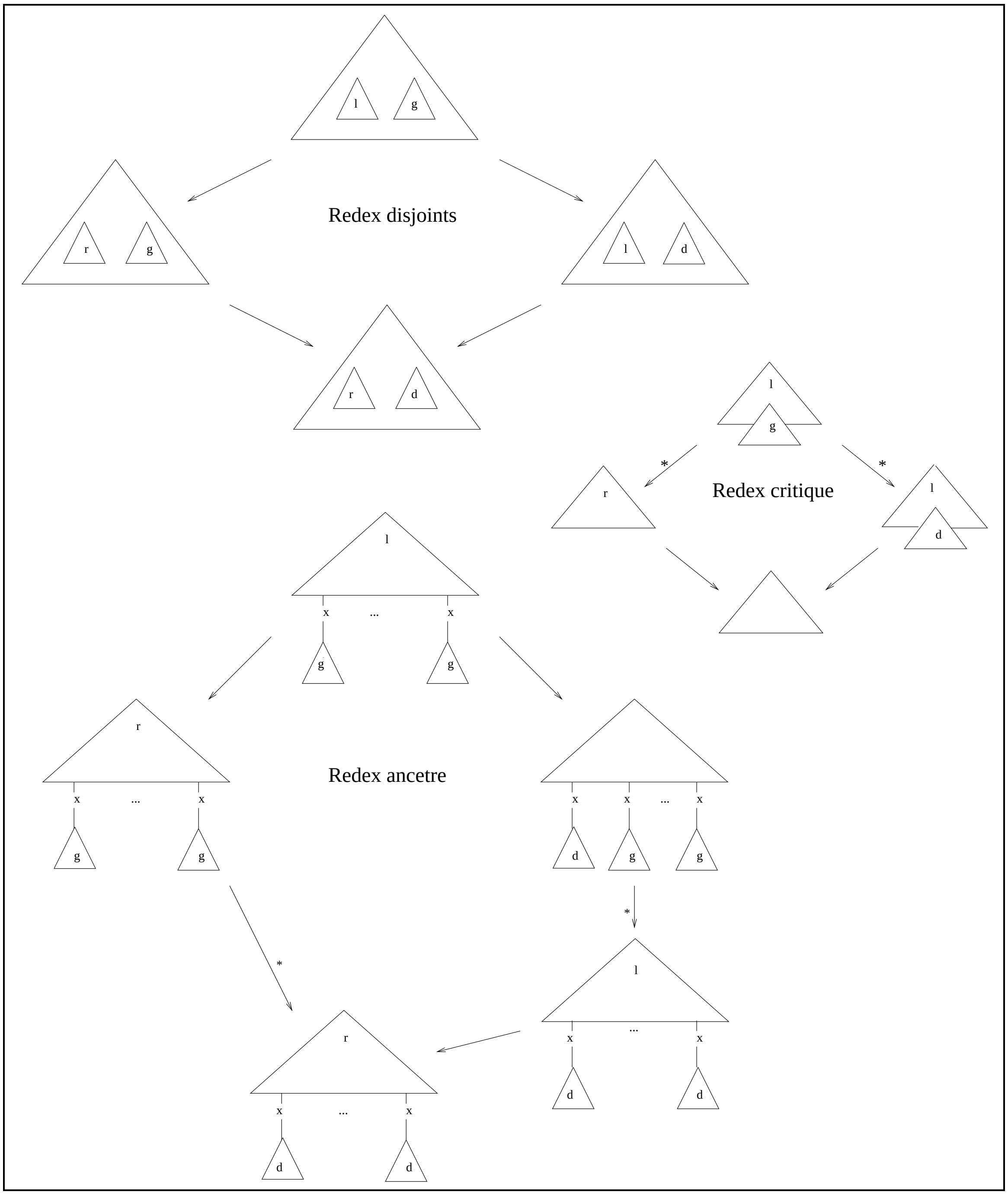

Figure 4.4: Lemme des paires critiques

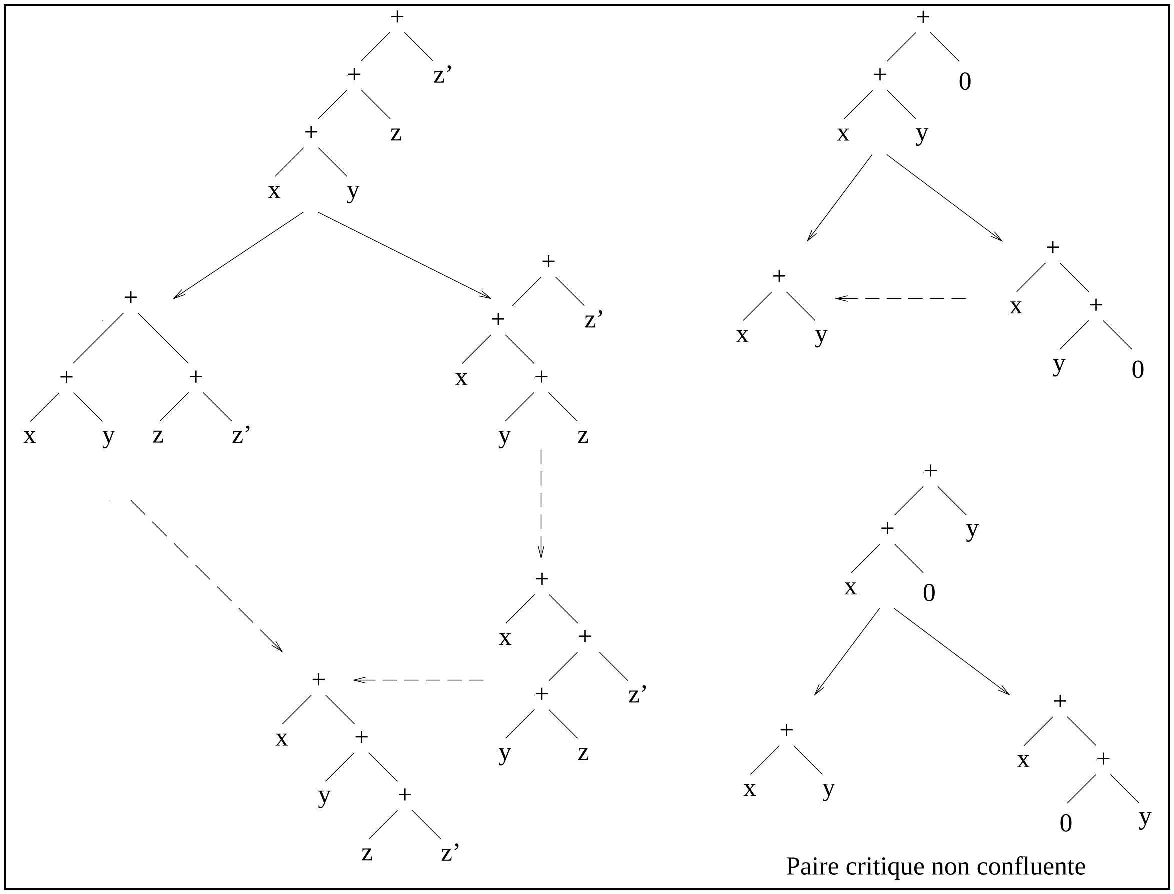

Figure 4.5: Confluence des règles d'associativité et identité

Theorem 4.12 *Soit R un système de réécriture pour lequel tout terme clos de $T(F)$ possède une forme normale unique. Alors la F-algèbre dont le domaine est constitué de l'ensemble des termes clos en forme normale, et dont les opérations sont définies par*

$$f(u_1\downarrow, \ldots, u_n\downarrow) = f(u_1, \ldots, u_n)\downarrow$$

est initiale dans la classe des F-algèbres qui satisfont les axiomes de E_R.

Preuve
Il suffit de vérifier que cette algèbre est isomorphe à l'algèbre quotient $T(F)/=_{E_R}$.

Cette construction a son importance pour la mise en oeuvre des preuves par récurrence. Il se trouve en effet que l'ensemble des termes clos en forme normale est reconnaissable par un automate ascendant d'arbre lorsque les règles sont linéaires gauches, et par un automate ascendant d'arbe avec transitions conditionelles lorsque les règles ne sont pas linéaires gauches. Cet automate permet à son tour de calculer automatiquement un schéma de récurrence, permettant d'automatiser (partiellement) ces preuves.

4.4 Systèmes canoniques

Étant donné un système de réécriture R confluent et noetherien, il est judicieux de réduire les règles de R avec R lui-même:

Définition 4.13 *Un système de règles est dit* interréduit*, si pour toute règle $l \to r \in R$, r est R-irréductible, et l est $(R - \{l \to r\})$-irréductible. Un système de règles est dit* canonique *s'il est neotherien, confluent et interréduit.*

Le mot canonique est justifié par la propriété suivante:

Theorem 4.14 *Soit R et R' deux systèmes canoniques tels que (i) $=_{E_R}==_{E'_R}$, et (ii) $R \cup R'$ termine.*

Alors R et R' sont identiques (modulo renommage des variables).

Preuve
D'après les hypothèses, pour tout $l \to r \in R$, $l \longrightarrow^*_{R'} \longleftarrow^*_{R'} r$. Par propriété de terminaison, $l \longrightarrow_{g \to d \in R'} u \longrightarrow^*_{R'} \longleftarrow^*_{R'} r$. Le même raisonnement appliqué à $g \to d \in R'$ montre que g est réductible par une règle $l' \to r'$ telle que $l' = l\sigma$, où σ est un renommage. D'où $r' =_{E_R \cup E_{R'}} r\sigma = E_{R'} r\sigma$. Comme r' est R'-irréductible par hypothèse, $r\sigma \longrightarrow^*_{R'} r'$, et par le même raisonnement que précedemment, $r\sigma = r'$ puisque r est R-irréductible.

Les systèmes canoniques, et cette propriété, jouent un rôle essentiel en théorie de la réécriture.

4.5 Exercices

1. On dit qu'un système de réécriture est interéduit si tout member droit de règle est irréductible ainsi que tout terme plus petit qu'un membre gauche de règle

dans l'ordre d'imbrication $s \triangleright\!\!\bullet\, t$ iff $s|_p = t\sigma$ et $s \overset{\bullet}{\neq} t$. Montrer que la notion de système canonique est inchangée pour cette nouvelle notion d'interréduction.

2. Soit R un système de réécriture non noetherien, et $\Rightarrow$ une relation sur les termes dont la fermeture transitive soit $=_{E_R}$. On suppose de plus que $\Rightarrow$ est fortement confluente, c'est à dire que $u \Rightarrow s$ et $u \Rightarrow t$ implique l'existence d'un terme v tel que $s \Rightarrow u$ et $t \Rightarrow u$. En déduire que $\longrightarrow_R$ est confluente. Peut-on affaiblir la notion de confluence forte de manière à déduire la confluence des systèmes de réécriture suivants:

   - $\{a \to c,\ a \to b,\ c \to b,\ c \to a\}$. Détailler la preuve.
   - $\{f(f(x)) \to f(x),\ f(a) \to f(b),\ b \to a\}$. Détailler la preuve.

Chapter 5

Terminaison

La terminaison d'un système de règles est indécidable. Due à Huet et Lankford, la pemière preuve codait un problème indécidable avec un nombre non borné de règles. Ultérieurement, Dershowitz donnait un codage avec 5, puis 3 règles. Enfin, Dauchet est arrivé à coder une machine de Türing arbitraire avec uniquement une règle, qui plus est linéaire gauche et sans superposition sur elle-même.

Puisqu'elle ne peut faire l'objet d'un calcul, la terminaison d'un système de règles doit donc faire l'objet d'une *preuve*. Par la suite, nous examinons plusieurs des méthodes standard, qui sont généralement implantées dans les logiciels de récriture.

Prouver la terminaison d'une relation de récriture peut se faire de plusieurs manières. La première méthode consiste à montrer que la relation de récriture est incluse dans un ordre bien-fondé. C'est pourquoi nous seront amenés à étudier les ordres bien-fondés sur les termes au chapitre 6. La seconde revient à appliquer un principe de type "divide and conquer", c'est-à-dire à prouver séparément la terminaison de plusieurs relations de récriture dont la relation de départ est l'union. Cette méthode porte souvent le nom de "modularité" de la terminaison. La troisième consiste à faire une preuve par récurrence. Les preuves à la Tait-Girard utilisées en λ-calcul typé font partie de cette dernière catégorie.

Le but de ce chapitre est d'introduire les deux premières de ces méthodes, en commençant par la plus importante, la première. En fait, la première méthode se subdivise elle-même en deux méthodes, dont la seconde est un raffinement de la première. Ces deux méthodes sont basées sur la notion de pré-ordre de récriture. Par la suite, on désignera en général par $\succeq$ un préordre[1], par $\preceq$ son symmétrique, c'est-à-dire $s \preceq t$ ssi $t \succeq s$, par $\simeq$ l'équivalence associée définie comme l'intersection $\succeq \cap \preceq$, et par $\succ$ l'ordre strict associé défini comme la différence $\succeq \setminus \simeq$.

Définition 5.1 *Une relation $\rightarrow$ sur une algèbre de termes est* monotone *si c'est une précongruence* [2], stable *ssi $\forall s, t, \sigma$ such that $s \rightarrow t$, then $s\sigma \rightarrow t\sigma$, et bien fondée si il n'existe pas de suite infinie $s_0 \rightarrow s_1 \ldots \rightarrow s_n \rightarrow \ldots$.*

[1]Un préordre est une relation réflexive transitive, un ordre est un préordre antisymmétrique

[2]Nous ne faisons qu'employer un nouveau vocabulaire dans le cas des relations d'ordre pour désigner une notion déjà existante, celle de précongruence. Le mot "monotone" est utilisé ici pour des raisons historiques; en fait ce sont les symboles de fonction de F qui sont monotones pour $\succeq$ lorsque $\succeq$ est une précongruence.

Le préordre $\succeq$ est un préordre de récriture faible *s'il est monotone, si $\succ$ et $\simeq$ sont stables et si $\succ$ est bien-fondée. C'est un* préordre de récriture *si les relations $\succ$ et $\simeq$ sont monotones et stables et si $\succ$ est bien-fondée.*

On voit que la différence entre les notions de préordre de récriture et préordre de récriture faible est dans la notion de monotonie qui est plus faible dans le second cas.

Dans ce chapitre, nous allons étudier les deux méthodes à base de préordres de récriture (faibles ou non), avant d'aborder brièvement les questions de modularité.

5.1 Méthode de Manna et Ness

L'idée de base est donc de montrer que la relation de récriture est incluse dans un pré-ordre bien fondé $\succeq$ sur les termes : R termine si $\forall s, t\ s \longrightarrow_R t, \quad s \succ t$. Bien sûr, on aimerait n'avoir à vérifier la décroissance de l'ordre bien-fondé $\succeq$ que sur les paires de termes du système de réécriture. Cela est possible pour les préordres de récriture :

Lemma 5.2 *Soit $R = \{l_i \to r_i\}_{i \in I}$ un sytème de récriture. R termine ssi il existe un préordre de récriture $\succeq$ tel que $l_i \succ r_i$ pour tout $i \in I$.*

Proof: On montre que $s \longrightarrow_{l_i \to r_i} t$ implique $s \succ t$ ce qui résulte directement des la stabilité et de la monotonie. On en déduit que la relation $\longrightarrow_R^+$ est incluse dans $\succ$, d'où le résultat puisque $\succ$ est bien fondé.

Des exemples d'utilisation de ce théorème simple sont donnés dans le chapitre 5.

5.2 Méthode de Aarts et Giesl

La méthode précédente est locale, en ce sens qu'elle se préocuppe de chaque pas de récriture. La méthode de Arts et Giesl est globale, en ce sens qu'elle est basée sur une analyse des dérivations. Plus particulièrement, elle est fondée sur un lemme d'existence de dérivations d'une certaine forme. Pour cela, on notera par $\mathcal{D} = \{f \mid f(\bar{l}) \to r \in R\}$ l'ensemble des symboles *définis* et par $\mathcal{C} = F - \mathcal{D}$ l'ensemble des symboles *constructeurs*. L'idée est alors qu'une dérivation issue d'un membre gauche l de règle doit récrire à la seconde étape un sous-terme du membre droit dont le symbole de tête est dans $\mathcal{D}$.

Définition 5.3 *Étant donné un système de récriture R, on appelle* paire de dépendance *de la règle $l \to r \in R$, toute paire de la forme $(l, f(\overline{r})$ où $f(\overline{r})$ est un sous-terme de r tel que $f \in \mathcal{D}$. On appelle paire de dépendance de R une paire de dépendance d'une règle de R, et on note par $DP(R)$ l'ensemble des paires de déoendances de R.*

On appelle maintenant chaîne de dépendance *de R, toute dérivation de la forme $u_0 \longrightarrow_R^{\neq\Lambda} v_0 \longrightarrow_{DP(R)}^{\Lambda} u_1 \longrightarrow_R^{\neq\Lambda} v_1 \ldots$, que l'on notera par $\{u_i, v_i\}_i$.*

Lemma 5.4 *Soit R un système de récriture qui ne termine pas. Alors il existe un terme de la forme $w[f(\overline{u})]_p$ où $f(\overline{u})$ est tel que*

(i) il existe une R-dérivation infinie issue de $f(\overline{u})$,
(ii) toutes les R-dérivations issues d'un quelconque terme $v \in \overline{u}$ sont finies.

Proof: Comme R ne termine pas, il existe un terme w qui est l'origine d'une dérivation infinie. On montre la propriété par récurrence sur la taille de w.

Si w est de la forme $C(\overline{v})$ où $C \in \mathcal{C}$, alors nécessairement, l'un des v_i est l'origine d'une dérivation infinie, et l'on conclue alors par application de l'hypothèse de récurrence appliquée à v_i.

Si w est de la forme $f(\overline{u})$ avec $f \in \mathcal{D}$, il y a deux cas. Dans le premier cas, aucun terme $v \in \overline{u}$ n'est l'origine d'une dérivation infinie. Le lemme est alors démontré. Dans le second cas, un terme $v \in \overline{u}$ est l'origine d'une dérivation infinie. On conclue alors par l'hypothèse de récurrence appliquée à v.

Theorem 5.5 *R termine si ses chaînes de dépendance sont finies.*

Proof: On suppose que les chaînes de dépendance de R sont finies, ainsi que l'existence d'une R-dérivation infinie issue d'un terme s. Posons $s \Rightarrow t$ ssi il existe une chaîne de dépendance $\{u_i, v_i\}_i$ telle que $s = u_i$ et $t = u_j$ avec $i > j$. $\Rightarrow$ est une relation bien fondée sur $T(F, X)$ puisque les chaînes de dépendance sont finies.

D'après le lemme 5.4, s contient un sous-terme non fortement normalisable $f(\overline{u})$ tel que $\overline{u}$ est fortement normalisable. On va maintenant montrer par récurrence sur $\Rightarrow$ que $f(\overline{u})$ est fortement normalisable, ce qui engendrera une contradiction.

Comme $\overline{u}$ est fortement normalisable, on ne peut le réduire indéfiniment, et donc, il existe $\overline{v}$ tel que $\overline{u} \longrightarrow^*_R \overline{v}$ et $f(\overline{v}) \longrightarrow^\Lambda_R t$. Il existe donc une règle $f(\overline{l}) \to r$ telle que $f(\overline{v}) = f(\overline{l})\sigma$ où σ est fortement normalisable, et $t = r\sigma$. Par le lemme, $r\sigma$ contient un sous-terme non fortement normalisable qui occure nécessairement à une position non-variable de r puisque σ est fortement normalisable. Ce sous-terme est donc de la forme $g(\overline{w})\sigma$, où $g(\overline{w})$ est un sous-terme de r. La paire $(f(\overline{l}), g(\overline{w}))$ est donc une paire de dépendence de R, et donc $(f(\overline{u}), f(\overline{l\sigma}), g(\overline{w\sigma}))$ est le début d'une chaîne de dépendance issue de $f(\overline{u})$, et donc $f(\overline{u}) \Rightarrow g(\overline{w})\sigma$. Par hypothèse de récurrence, $g(\overline{w})\sigma$ est fortement normalisable, ce qui aboutit à une contradiction.

On en déduit comme corollaire :

Theorem 5.6 *R termine s'il existe un préordre de récriture faible tel que*
(i) $l \succeq r$ pour toute règle $l \to r \in R$,
(ii) $s \succ t$ pour toute paire de dépendance $(s, t) \in DP(R)$.

Proof: On a $\longrightarrow_R \ \subseteq \succeq$, donc, si R ne terminait pas, d'après le théorème précédent, il existerait des chaînes de dépendances infinies. Soit $u_0 \longrightarrow^{\neq\Lambda}_R v_0 \longrightarrow^\Lambda_{DP(R)} u_1 \longrightarrow^{\neq\Lambda}_R v_1 \ldots$ une telle chaîne. On a immédiatement $u_0 \succeq v_0$ par monotonie de $\succeq$ et $v_0 \succ u_1$ par stabilité de l'ordre strict, etc, et donc $u_0 \succ u_1 \succ u_2 \ldots$, ce qui contredit la bonne fondation de $\succ$.

Exemple 5.7 *Considérons l'exemple de l'addition des entiers en représentation de Peàno.*

Le système :

$$\begin{array}{rcl} x + 0 & \to & x \\ x + s(y) & \to & s(x + y) \end{array}$$

possède une unique paire de dépendance :

$$\langle x + s(y), x + y \rangle$$

La terminaison de ce système peut alors être prouvée très naturellement, en interprétant les termes de l'algèbre libre dans les entiers naturels, le symbole s étant interprété par l'addition de 1, et le symbole $+$ par l'addition des entiers. On vérifie que pour l'ordre bien fondé ainsi induit :

$$\begin{array}{rcl} x+0 & \simeq & x \\ x+s(y) & \simeq & s(x+y) \\ x+s(y) & \succ & x+y \end{array}$$

En pratique, le gain obtenu par rapport à la méthode de Manna et Ness est faible, car le même préordre est utilisé pour les règles de R et les paires de $DP(R)$. Remarquons que c'est maintenant l'orientation stricte des paires de dépendance avec un préordre de récriture faible qui assure la terminaison des récritures. Pour les règles $g \to d$, il suffit que le couple (g, d) soit formé de paires équivalentes dans l'équivalence engendrée par le préordre, comme dans l'exemple ci-dessus.

En fait, on peut améliorer drastiquement ce résultat d'un point de vue pratique de la manière suivante : on va dupliquer l'alphabet $\mathcal{D}$ avec un nouvel alphabet $\check{D} = \{\check{f} \mid f \in \mathcal{D}\}$. Une *paire de dépendance avec duplication* de la règle $f(\bar{l}) \to r \in R$, sera alors une paire de la forme $(\check{f}(\bar{l}), \check{g}(\bar{r})$ où $g(\bar{r})$ est un sous-terme de r tel que $g \in \mathcal{D}$. La notion de chaîne de dépendance n'en est pas affectée puisque les récritures qui interviennent avec R portent sur des sous-termes stricts des membres des paires de dépendance.

Il est facile de vérifier que les résultats précédents restent vrais avec cette nouvelle définition des paires de dépendance. Cela résulte du lemme :

Lemma 5.8 *Les chaînes de dépendance sont finies ssi les chaînes de dépendance avec duplication sont finies.*

Proof: C'est l'argument précédent.

Exemple 5.9 *Considérons l'exemple de l'addition et multiplication des entiers en représentation de Peàno.*

Les paires de dépendance du système :

$$\begin{array}{rcl} x+0 & \to & x \\ x+s(y) & \to & s(x+y) \\ x \times 0 & \to & 0 \\ x \times s(y) & \to & x \times y + x \end{array}$$

sont au nombre de trois, les première et troisième règles n'en créant pas :

$$\langle x \check{+} s(y), x \check{+} y \rangle$$

pour la deuxième règle et

$$\langle x \check{\times} s(y), x \check{\times} y \rangle$$
$$\langle x \check{\times} s(y), (x \times y) \check{+} x \rangle$$

pour la quatrième.

On peut alors conclure grâce à l'interprétation polynomiale simple (voir chapitre 6) :

$$\begin{array}{ll} [\![\check{\times}]\!](x,y) = xy+1; & [\![\check{+}]\!](x,y) = x+y; \\ [\![\times]\!](x,y) = xy; & [\![+]\!](x,y) = x+y; \\ [\![s]\!](x) = x+1; & [\![0]\!] = 0. \end{array}$$

5.3 Modularité

L'union de deux systèmes de règles R_1 et R_2 qui terminent tous deux ne termine pas nécessairement, comme le montre l'exemple simple $R_1 = \{a \to b\}$, $R_2 = \{b \to a\}$. Ce qui est plus surprenant est que cela reste le cas lorsque les deux systèmes de règles ne partagent aucun symbole de fonction, comme le montre l'exemple suivant dû à Toyama :

$$R_1 = \{or(x,y) \to x,\ or(x,y) \to y\},\quad R_2 = \{f(a,b,x) \to f(x,x,x)\}$$

La terminaison de R_1 est évidente, il suffit d'interpréter un terme par sa taille. R_2 termine également, on pourra interpréter un terme par le nombre de radicaux distincts qu'il contient. Par contre, l'union de R_1 et de R_2 ne termine pas, en effet :

$$\begin{array}{c} f(or(a,b), or(a,b), or(a,b)) \longrightarrow f(a, or(a,b), or(a,b)) \longrightarrow \\ f(a,b,or(a,b)) \longrightarrow f(or(a,b), or(a,b), or(a,b)) \longrightarrow \dots \end{array}$$

On peut bien sûr se demander si la non-confluence de R_1 joue un rôle, mais il n'en est rien. Par contre, la duplication des variables de la règle de R_2 joue un rôle essentiel, ainsi que la possibilité de projeter un terme sur l'un de ses sous-termes ce que réalisent les règles de R_1. Avant de traiter ce problème, nous allons donner des conditions suffisantes de modularité d'une union.

5.3.1 Commutation

Définition 5.10 *Soient $\longrightarrow_{R_1}$ et $\longrightarrow_{R_2}$ deux relations sur un ensemble S, et $\longrightarrow = \longrightarrow_{R_1} \cup \longrightarrow_{R_2}$. On dit que R_1* commute *sur R_2 si la propriété suivante est satisfaite :*

$$\forall s,t,u \in S \textit{ such that } s \underset{R_2}{\longrightarrow} u \underset{R_1}{\longrightarrow} t,\ \exists v \in S \textit{ such that } s \overset{+}{\underset{R_1}{\longrightarrow}} v \overset{*}{\longrightarrow} t$$

Proposition 5.11 *Supposons que les deux relations $\longrightarrow_{R_1}$ et $\longrightarrow_{R_2}$ soient bien fondées et que R_1 commute sur R_2, Alors la relation $\longrightarrow = \longrightarrow_{R_1} \cup \longrightarrow_{R_2}$ est bien fondée.*

Preuve
On montre que tout terme w, toute dérivation issue de w pour la relation $\longrightarrow_{R_1} \cup \longrightarrow_{R_2}$ est finie. La preuve est par récurrence sur le couple (w,n), où n est le nombre d'étapes consécutives avec R_2 séparant w de la première étape avec R_2 (on prendra $n = \infty$ au cas où la dérivation ne contient aucune étape R_1). Les paires (w,n) seront comparées lexicographiquement, le premier argument avec la relation $\longrightarrow_{R_1}$, le second avec l'ordre sur les entiers naturels. Il y a trois cas (une fois éliminé le cas trivial d'une dérivation vide) :

1. La dérivation est de la forme $w \longrightarrow_{R_1} s \ldots$. Comme $w \longrightarrow_{R_1} s$, la dérivation issue de s est plus petite que la dérivation initiale, donc elle est finie. Il en est donc de même de la dérivation issue de w.

2. La dérivation ne contient aucune étape R_1. Alors elle ne contient que des étapoes R_2, et comme $\longrightarrow_{R_2}$ est bien fondée, elle est finie.

3. La dérivation est de la forme $w \longrightarrow^n_{R_2} s \longrightarrow_{R_2} u \longrightarrow_{R_1} t \ldots$. Cette dérivation est donc mesurée par le couple $(w, n+1)$. Par la propriété de commutation, il existe v tel que $s \longrightarrow^+_{R_1} v \longrightarrow^* t$. La dérivation $w \longrightarrow^n_{R_2} s \longrightarrow^+_{R_1} v \longrightarrow^* t \ldots$ est mesurée par le couple (w, n). Par hypothèse de récurrence, elle est donc finie, ce qui implique la finitude de la dérivation de départ.

Un cas particulier s'avère d'une importance essentielle:

Corollaire 5.12 *Soit $\longrightarrow_R$ une relation de récriture qui termine. Alors la relation $\longrightarrow_R \cup \rhd$ est bien fondée.*

Proof: On montre simplement que la récriture commute sur sous-terme strict, ce qui est évident.

Ce corollaire sert très souvent en pratique. Notons que la relation obtenue est stable, mais pas monotone.

5.3.2 Unions disjointes

5.4 Exercices

Exercice 5.13 *Donner un codage des machines de Türing par un système de récriture, tel que, si R est le code de la machine M, alors M s'arrète lors du calcul de la donneé u ssi le code de u est fortement normalisable pour R.*

Exercice 5.14 *Soit R un système de récriture sans variables.*

- *Monmtrer qu'il existe toujours une dérivation infinie issue d'un member gauche de règles dans le cas où R ne termine pas.*

- *En déduire que la terminaison de la réécritre avec R est décidable.*

Exercice 5.15 *On définit la relation $s \to t$ si et seulement si il existe une position $p \in \mathcal{P}os(s)$ et une substitution $\sigma : \mathcal{V}ar(s) \longrightarrow \mathcal{X}$ telles que $(s|_p)\sigma = t$.*

Soit maintenant $\longrightarrow_R$ une relation de réécriture qui termine. Montrer que la relation $\longrightarrow_R \cup \to$ termine. Est-elle monotone? stable?

Chapter 6

Ordres bien fondés sur les termes

Les ordres bien fondés sur les termes et leur construction sont l'outil de base pour les preuves de terminaison (des systèmes de réécriture). L'objectif est ici de donner quelques briques comme les extensions lexicographique et multi-ensemble ou les ordres récursifs sur les chemins. Ces "briques" permettent, via des fonctions d'interprétation, de construire des ordres bien fondés adaptés au problème à traiter. Nous donnons tout d'abord quelques rappels sur les préordres bien fondés et les ordinaux. Nous montrons ensuite les extensions lexicographique et multi-ensemble d'un préordre bien fondé. Nous terminons par le théorème de Kruskal, qui permettra de construire des ordres de simplification.

6.1 Préordres bien fondés et beaux préordres

Rappelons qu'une relation de *préordre* R sur un ensemble D est une relation binaire transitive (i.e. pour tous $x, y, z \in D$, xRy & $yRz \Rightarrow xRz$) et réflexive (i.e. pour tout $x \in D$, xRx) et qu'une relation *d'ordre* est de plus anti-symétrique (i.e. pour tous $x, y \in D$, xRy & $yRx \Rightarrow x = y$).

Exemple 6.1 *Sur $T(F)$, la relation $\geq$ définie par*

$$t \geq u \quad ssi \quad |t| \geq_{\mathbf{N}} |u|$$

(i.e. les termes sont comparés suivant leur taille) est une relation de préordre, mais pas une relation d'ordre en général car deux termes peuvent avoir la même taille sans être identiques.

Si $\geq$ est une relation de préordre, la *relation d'équivalence associée* $\simeq$ est définie par $x \simeq y \stackrel{\text{def}}{=} x \geq y \;\&\; y \geq x$. Si $\geq$ est une relation de préordre, l'*ordre strict* associé (relation transitive et irréflexive) est défini par $x > y \stackrel{\text{def}}{=} x \geq y \& y \not\geq x$.

Une relation de préordre $\geq$ est *totale* (sur D) lorsque, pour tous $x, y \in D$, $x \geq y$ ou $y \geq x$.

Définition 6.2 *Étant donnée une relation de préordre $\succeq$ sur D, un élément $x_1 \in D$ est* fortement normalisant *s'il n'existe aucune suite infinie strictement décroissante d'éléments de D issue de x_1, c'est-à-dire de la forme $x_1 \succ x_2 \succ \ldots \succ x_n \succ \ldots$.*

Une relation de préordre $\succeq$ sur D est bien fondée *ou* noetherienne *si tout élément de D est fortement normalisant.*

Une relation de préordre $\preceq$ sur D est un beau préordre *(on dit aussi un* pré-bel ordre*) si pour toute suite infinie $\{s_i\}_{i \in I\!N}$ d'éléments de D, il existe un couple d'entiers naturels $j < k$ tel que $s_i \preceq s_j$.*

Exemple 6.3 *La relation d'ordre habituelle sur $\mathbf{N}$ est bien fondée. Mais la relation d'ordre sur $\mathbf{Q}$ ne l'est pas.*

Exemple 6.4 *Supposons que F ne contient qu'un symbole binaire $\cdot$. On définit l'application ϕ de $T(F,X)$ dans $\mathbf{N}$ par:*

- $\phi(u \cdot v) \stackrel{\text{def}}{=} 2\phi(u) + \phi(v) + 1$
- $\phi(x) = 0$ *pour x variable.*

On définit alors la relation $\geq$ sur $T(F,X)$ par $t \geq u$ ssi $\phi(t) \geq \phi(u)$. La relation ainsi définie est une relation de préordre bien fondée.

Exemple 6.5 *La relation d'égalité sur un ensemble D fini est un bel ordre. Ce n'est pas vrai si D est infini.*

Exemple 6.6 *Tout ordre total bien fondé est un beau préordre; Les préordres d'interprétation dans les entiers sont des beaux préordres (ainsi que tous les ordres définis à partir d'interprétations dans des ensembles munis d'un beau préordre).*

Notons que l'ordre sous-terme est bien fondé, mais n'est pas un beau préordre:

Exemple 6.7 *Soit F contenant f binaire et 0 (constante), on construit les suites t_n et u_n de termes de la façon suivante: $t_0 = 0$ et $t_{n+1} = f(0, t_n)$ et $u_n = f(f(0,0), t_n)$. La suite t_n est croissante pour l'ordre sous-terme, mais dans la suite u_n il n'y a aucune paire de termes comparables pour l'ordre sous-terme.*

Nous énonçons maintenant quelques propriétés simples des beaux préordres et des préordres bien fondés.

Property 6.8 *(i) Soient $\geq$ et $\succeq$ deux préordres sur D tels que $\succ \subseteq >$. Alors, $\succeq$ est un préordre bien fondé si $\geq$ est un préordre bien fondé.*

(ii) Soient $\leq$ et $\preceq$ deux préordres sur D tels que $\preceq \subseteq \leq$. Alors $\leq$ est un beau préordre si $\preceq$ est un beau préordre.

Les ordres bien fondés se conservent donc par restriction, alors que les beaux préordres se conservent par extension. Ce sera une raison de leur utilité. La seconde est qu'ils permettent de construire systématiquement des familles de préordres bien fondés, en combinant la propriété qui suit avec la propriété (ii) ci-dessus.

Property 6.9 *Soit $\preceq$ un beau-préordre sur D. Alors $\succeq$ est un préordre bien-fondé.*

Proof: Soit $s_0 \succ s_1 \ldots$ une suite infinie décroissante d'éléments de D. Par définition des beaux préordres, il existe $i < j$ tels que $s_i \preceq s_j$. Par transitivité de $\succ$, on obtient également $s_i \succ s_j$, aboutissant à une contradiction.

Nous allons conclure par une propriété caractéristique des beaux préordres :

Property 6.10 *Soit $\preceq$ un beau préordre sur D. Alors, pour toute suite infinie $\{s_i\}_{i \in \mathbb{N}}$ d'éléments de D, il existe une suite infinie strictement croissante d'entiers naturels $\{i_j\}_{j \in \mathbb{N}}$ tels que $s_{i_j} \preceq s_{i_{j+1}}$.*

Proof: Soit $\{s_i\}_{i \in \mathbb{N}}$ une suite infinie de D. Si cette suite contient deux éléments équivalents, la propriété est vraie. Sinon, on raisonne par l'absurde. Si la propriété n'est pas vraie, on montre par aplication répetée de la propriété de beau préordre l'existeence d'une suite infinie strictement croissante d'entiers naturels $\{i_j\}_{j \in \mathbb{N}}$ tels que s_{i_j} soit maximal, c'est-à-dire tels que $\forall k > i_j, \neg(s_{i_j} \preceq s_k)$. La suite des éléments maximaux étant ainsi infinie, onpeut lui appliquer l'hypothèse de beau préordre, d'où l'on en déduit qu'un certain élément s_{i_l} n'est pas maximal, résultant en une contradiction.

6.2 Préordres d'interprétation

Le moyen le plus utilisé pour prouver la terminaison d'un programme ou d'un ensemble de règles est de construire, comme dans l'exemple précédent, une fonction d'interprétation ϕ des termes (ou des données du programme) dans un domaine D muni d'un préordre bien fondé $\geq_D$. L'ensemble des termes (ou données) est alors muni du préordre bien fondé défini par:

$$x \geq y \quad \text{ssi} \quad \phi(x) \geq_D \phi(y)$$

D'un point de vue formel, la fonction d'interprétation ϕ est un homomorphisme de la $\mathcal{F}$-algèbre (libre) des termes dans une certaine $\mathcal{F}$-algèbre construite sur le domaine D. Plus précisément, c'est la définition de ϕ qui définit la $\mathcal{F}$-algèbre, car ϕ a pour rôle d'interpréter les symboles de fonction de $\mathcal{F}$ dans le domaine D. Il suffit donc de se donner la structure de $\mathcal{F}$-algèbre sur D d'une part (c'est-à-dire l'interpréation des symboles de fonction), et la valeur de l'homorphisme ϕ pour les variables de $\mathcal{X}$.

Exemple 6.11 *Soit la règle de mise en forme normale disjonctive en calcul propositionnel (on suppose que les négation n'apparaissent que sur les variables propositionnelles):*

$$(P \vee Q) \wedge R \rightarrow (P \wedge R) \vee (Q \wedge R)$$

(où P, Q, R désignent n'importe quelle formule du calcul propositionnel; cette règle pouvant elle aussi être appliqué à n'importe quelle sous-formule). On considère la fonction d'interprétation dans les entiers naturels supérieurs ou égaux à 2 munis de leur ordre habituel définie comme suit:

$$\begin{cases} \phi(P) \stackrel{\text{def}}{=} 2 & \textit{Si } P \textit{ est un littéral} \\ \phi(P \vee Q) \stackrel{\text{def}}{=} \phi(P) + \phi(Q) + 2 & \\ \phi(P \wedge Q) \stackrel{\text{def}}{=} \phi(P) \times \phi(Q) & \end{cases}$$

On peut remarquer alors que:

$$\begin{array}{rcl}\phi((P \vee Q) \wedge R) & = & \phi(P \vee Q) \times \phi(R) \\ & = & \phi(P) \times \phi(R) + \phi(Q) \times \phi(R) + 2 \times \phi(R)\end{array}$$

et que

$$\begin{array}{rcl}\phi((P \wedge R) \vee (Q \wedge R)) & = & \phi(P \wedge R) + \phi(Q \wedge R) + 2 \\ & = & \phi(P) \times \phi(R) + \phi(Q) \times \phi(R) + 2\end{array}$$

Mais comme $\phi(R) \geq 2$, $\phi((P \vee Q) \wedge R) > \phi((P \wedge R) \vee (Q \wedge R))$. De plus, on peut vérifier la monotonie du préordre défini par ϕ avec $\vee$ et $\wedge$:

$$\begin{array}{rcl}(\phi(P) > \phi(Q)) & \Rightarrow & \phi(P \wedge R) > \phi(Q \wedge R) \\ (\phi(P) > \phi(Q)) & \Rightarrow & \phi(P \vee R) > \phi(Q \vee R)\end{array}$$

par croissance de la fonction d'interprétation. L'application de la règle à n'importe quelle sous-formule retourne donc une nouvelle formule strictement plus petite.

On vérifie aussi la compatibilité de ϕ avec l'associativité et la commutativité des connecteurs $\wedge, \vee$, par associativité et commutativité de l'adition et de la multiplication des entiers naturels.

6.3 Ordinaux

Les ordres bien fondés permettent d'effectuer des preuves par récurrence, suivant la règle de déduction:

> Si une propriété P est vraie sur les éléments minimaux de D (i.e. ceux pour lesquels il n'y a pas d'éléments strictement plus petits) et si $\forall y \in D, (\forall x \in D(x < y \Rightarrow P(x))) \Rightarrow P(y)$, alors $\forall x \in D, P(x)$.

La correction de cette règle est laissée en exercice. (Il faudra considérer l'ensemble $\{x \in D \mid P(x) \text{ est faux}\}$).

Nous donnerons des utilisations de cette récurrence puissante quand nous aurons construit des ordres bien fondés.

Les ensembles munis d'un ordre total bien fondé sont aussi appelés *ordinaux*. (NB: en fait, la notion d'ordinal est inséparable de celle d'ensemble; cette "définition" n'est donc pas correcte car elle présuppose la notion d'ensemble). Les ordinaux sont considérés "à isomorphisme près": deux ensembles totalement ordonnés $(A, \geq)$ et $(B, \geq')$ tels qu'il existe une bijection ϕ de A dans B telle que $x < y$ si et seulement si $\phi(x) < \phi(y)$ sont considérés comme le même ordinal.

Si $\alpha(= (D, \geq))$ est un ordinal, $\alpha + 1$ est l'ordinal obtenu en adjoignant à D un élément plus grand que tous ceux de D (qu'on peut identifier à l'ensemble D lui-même). Un ordinal α est un ordinal *successeur* s'il existe un ordinal β tel que $\alpha = \beta + 1$. Un ordinal qui n'est pas un successeur est appelé *ordinal limite*. ω est le plus petit ordinal limite. C'est l'ensemble des entiers naturels munis de son ordre habituel. Le plus petit ordinal (l'ensemble vide) est aussi noté 0.

De même que pour la règle de récurrence ci-dessus, on peut définir une fonction par *récurrence transfinie* de la façon suivante (on admettra la correction de telles définitions, qui en général, n'ont pas de sens en théorie des ensembles, mais qui, pour notre propos dans la suite, ne posent pas de problème):

si h est une fonction de D^2 dans D (où D est un ordinal) et $a \in D$, alors il existe une unique fonction f telle que

- $f(0) = a$
- $f(x+1) = h(f(x), x)$
- $f(x) = \sup\{f(y) \mid y < x\}$ si x est un ordinal limite

On utilise aussi couramment la somme et le produit de deux ordinaux qui sont définis pas récurrence transfinie par:

- $x + 0 \stackrel{\text{def}}{=} x$
- $x + (y+1) \stackrel{\text{def}}{=} (x+y)+1$
- $x + y \stackrel{\text{def}}{=} \sup\{x + z \mid z < y\}$ si y est un ordinal limite
- $x \times 0 \stackrel{\text{def}}{=} 0$
- $x \times (y+1) \stackrel{\text{def}}{=} x \times y + x$
- $x \times y \stackrel{\text{def}}{=} \sup\{x \times z \mid z < y\}$ si y est un ordinal limite.

Exemple 6.12 *On peut réaliser, par exemple $\omega + \omega$ en ordonnant les entiers comme suit:*

> *Si n et m ont même parité, alors $n \leq m$ ssi $n \leq_{\mathbf{N}} m$. Si n est pair et m est impair, alors $n < m$.*

Plusieurs propriétés de cette arithmétique des ordinaux sont données en exercice. Remarquons néanmoins que l'addition des ordinaux n'est pas commutative car, par exemple, $1 + \omega = \sup\{1 + z \mid z < \omega\} = \omega \neq \omega + 1$.

Enfin, on peut de même définir l'exponentiation:

- $x^0 \stackrel{\text{def}}{=} 1$
- $x^{y+1} \stackrel{\text{def}}{=} x^y \times x$
- $x^y \stackrel{\text{def}}{=} \sup\{x^z \mid z < y\}$ si y est un ordinal limite.

Pour terminer avec ces notations, on note ϵ_0 le plus petit ordinal α tel que $\alpha = \omega^\alpha$. C'est aussi la borne supérieure de l'ensemble $\{\omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots\}$.

6.4 Fonctionnelles d'ordres asssociées aux structures de données essentielles

6.4.1 Extension produit

Soient $(D_1, \geq_1), \ldots, (D_n, \geq_n)$ des domaines (non vides) munis de relations de préordre. On munit l'ensemble $D_1 \times \ldots \times D_n$ des n-uplets d'éléments de $D_1, \ldots, D_n$ de la relation de préordre $(\geq_1, \ldots, \geq_n)_\times$, notée ici $\geq_\times$ et appellée *ordre produit*, définie par:

$$(a_1, \ldots, a_n) \geq_\times (b_1, \ldots, b_n) \text{ iff } \forall i \in [1..n] \ a_i \geq_i b_i$$

$$(a_1, \ldots, a_n) \simeq_\times (b_1, \ldots, b_n) \text{ iff } \forall i \in [1..n] \ a_i \simeq_i b_i$$

Notons que la décroissance stricte de $\geq_\times$ implique la décroissante stricte d'au moins une composante.

Proposition 6.13 *La relation $\geq_\times$ est un préordre sur $(D_1, \geq_1), \ldots, (D_n, \geq_n)$ dont l'équivalence est la relation $\simeq_\times$. C'est un beau préordre (resp. un préordre bien-fondé) sur $D_1 \times \ldots \times D_n$ si et seulement si les relations $\geq_i$ sont toutes des beaux préordres (resp. ordres bien fondés) sur D_i.*

Preuve
Par récurrence sur la taille des n-uplets.

6.4.2 Extension lexicographique

Soient $(D_1, \geq_1), \ldots, (D_n, \geq_n)$ des domaines (non vides) munis de relations de préordre. On munit l'ensemble $D_1 \times \ldots \times D_n$ des n-uplets d'élémenmts de $D_1, \ldots, D_n$ de la relation de préordre $(\geq_1, \ldots, \geq_n)_{lex}$ notée ici $\geq_{lex}$ définie comme l'union des deux relations suivantes :

$$(a_1, \ldots, a_n) >_{lex} (b_1, \ldots, b_n) \text{ iff } \exists i \geq 1, \forall j < i, a_j \simeq_j b_j \text{ et } a_i >_i b_i$$
$$(a_1, \ldots, a_n) \simeq_{lex} (b_1, \ldots, b_n) \text{ iff } \forall i \geq 1, a_i \simeq_i b_i.$$

Proposition 6.14 *La composée lexicographique de n relations de préordre est une relation d'ordre si et seulement si chacune des composantes est une relation d'ordre. Elle est totale ssi chacune des composantes est totale,*

La preuve est laissée en exercice.

Proposition 6.15 *$\geq_{lex}$ est un préordre bien fondé (resp. un beau préordre) sur $D_1 \times \ldots \times D_n$ ssi les préordres $\geq_1, \ldots, \geq_n$ sont bien fondés (resp. des beaux préordres).*

Preuve
Supposons que $\geq_{lex}$ est un préordre bien fondé, et soit $\{a_i^j\}_{j \in \mathbf{N}}$ une suite strictement décroissante de D_i. La suite $\{(a_1, \ldots a_{i-1}, a_i^j, a_{i+1}, \ldots, a_n)\}_{j \in \mathbf{N}}$ est une suite strictement décroissante pour $\geq_{lex}$. (où a_k, pour $k \neq i$, est un élément quelconque, fixé, de D_k.)

Réciproquement, on procède par récurrence sur la taille k des k-uplets, et par contradiction. La propriété est triviale pour le cas de base, $k = 1$. Pour le cas général, soit $\{(a_1^k, \ldots, a_n^k)\}_{k \in \mathbb{N}}$ une suite strictement décroissante pour $\geq_{lex}$. Notons que $(a_1, \ldots, a_n) \geq_{lex} (b_1, \ldots, b_n)$ implique $a_1 \geq_1 b_1$, et donc la suite $\{a_1^k\}_{k \in \mathbb{N}}$ est décroissante pour $\geq_1$. Comme $>_1$ est bien fondé, elle est nécessairement stationnaire à partir d'un certain rang n. Il s'ensuit que la suite $\{(a_2^k, \ldots, a_n^k)\}_{k \geq \mathbb{N}^n}$ est une suite infinie strictement décroissante ce qui contradit la bonne fondation de l'ordre pour les $(k-1)$-uplets. $\square$

Exemple 6.16 *Montrons un exemple d'utilisation de la composée lexicographique. $\mathcal{R}$ est le système de réécriture*

$$\begin{cases} (R1) & x \cdot (y \cdot z) & \rightarrow & y \cdot a \\ (R2) & (x \cdot y) \cdot a & \rightarrow & x \cdot (y \cdot b) \end{cases}$$

où a et b sont des constantes distinctes.

On construit les fonctions d'interprétation : $\phi_1(u)$ est la taille de u et $\phi_2(u)$ est le nombre d'ocurrences de a dans u. Soit alors ϕ la composée lexicographique: $\phi(u) = (\phi_1(u), \phi_2(u)) \in \mathbf{N}^2$. On vérifie que:

- *si u se réécrit en v par la règle $R1$, $\phi_1(u) > \phi_1(v)$.*
- *Si u se réécrit en v par la règle $R2$, alors $\phi_1(u) = \phi_1(v)$ et $\phi_2(u) > \phi_2(v)$.*

Donc, dans tous les cas, $\phi(u) >_{lex} \phi(v)$. Ce qui prouve la terminaison de $\mathcal{R}$ puisque $\geq_{lex}$ est bien fondé.

6.4.3 Extension multi-ensemble

Il s'agit cette fois de construire un pré-ordre sur les multi-ensembles finis composés d'éléments d'un ensemble D.

Si D est un ensemble, l'ensemble $\mathcal{M}(D)$ des *multi-ensembles* finis d'éléments de D est formé des applications de D dans $\mathbf{N}$ qui sont nulles sauf pour un nombre fini d'éléments de D. Habituellement, on note un multi-ensemble $\mathcal{M}$ en répétant d $\mathcal{M}(d)$ fois:

Exemple 6.17 *Soit $D = \mathbf{N}$ et $\mathcal{M}$ le multi-ensemble $\mathcal{M}(0) = 2$, $\mathcal{M}(1) = 0$, $\mathcal{M}(2) = 3$ et $\mathcal{M}(n) = 0$ pour $n > 2$. On note aussi*

$$\mathcal{M} = \{0, 0, 2, 2, 2\} = \{0, 2, 0, 2, 2\} \ldots$$

On définit les opérations $+, \cup, \cap, -$ sur les multi-ensembles par:

$$\begin{array}{rrcl} \forall x, y \in \mathcal{M}(D), \forall d \in D, & (x+y)(d) & = & x(d) + y(d) \\ & (x \cup y)(d) & = & \max(x(d), y(d)) \\ & (x \cap y)(d) & = & \min(x(d), y(d)) \\ & (x - y)(d) & = & \max(0, x(d) - y(d)) \\ & \emptyset(d) & = & 0 \end{array}$$

De même,

$$x \subseteq y \Leftrightarrow \forall d \in D, x(d) \leq y(d) \Leftrightarrow \exists z, y = x + z$$

et $d \in x$ si $x(d) > 0$. La *taille* d'un multi-ensemble M est l'entier $\sum_{d \in D} M(d)$.

Il existe de nombreuses définitions de l'extension multi-ensemble d'un pré-ordre $\geq_D$ sur un ensemble D. Nous en donnons une qui se prète bien aux raisonnements. Les autres font l'objet d'exercices. Celle que nous choisissons définit l'extension multi-ensemble d'un pré-ordre comme la fermeture réflexive transitive d'une relation plus primitive $\geq_{\mathcal{M}(D)}$:

Définition 6.18 $\begin{array}{llll} M \cup \{x\} & >_{\mathcal{M}(D)} & M \cup \{y_1, \ldots, y_n\} & \textit{if} \quad \forall i \in [1..n]\ x >_D y_i \\ M \cup \{x\} & \simeq_{\mathcal{M}(D)} & M \cup \{y\} & \textit{if} \quad x \simeq y \end{array}$

On définit maintenant l'ordre multi-ensemble sur $\mathcal{M}(D)$ engendré par un préordre sur D comme la fermeture réflexive transitive de la relation $\geq_{\mathcal{M}(D)} = (>_{\mathcal{M}(D)} \cup \simeq_{\mathcal{M}(D)})$, c'est-à-dire : $\geq_{mul} = (\geq_{\mathcal{M}(D)})^*$

Proposition 6.19 *$\geq_{mul}$ est un préordre dont l'équivalence associée est $(\simeq_{\mathcal{M}(D)})^*$ et l'ordre strict associé est $(>_{\mathcal{M}(D)})^*$.*

La preuve est à faire en exercice. Remarquons que l'extension multiensemble d'un oprdre est monotone vis-à-vis des opérations multi-ensemblistes :

Proposition 6.20 *$\geq_m$ est compatible avec $+$ et $-$:*

$$\forall x, y, z \in \mathcal{M}(D),\ x >_{mul} y \Rightarrow x + z >_{mul} y + z \quad et \quad x - z \geq_m y - z$$

Nous arrivons maintenant au résultat fondamental:

Theorem 6.21 *$\geq_{mul}$ est bien fondé sur $\mathcal{M}(D)$ ssi $\geq_D$ est bien fondé sur D.*

Preuve
Le sens seulement si est trivial, car l'on peut identifier D avec les multi-ensembles singletons. Nous raisonnons par l'absurde pour prouver l'autre direction, en remarquant qu'il suffit de prouver la propriété pour la relation $>_{\mathcal{M}(D)}$. Soit donc $\{M_i\}_{i \in \mathbb{N}}$ uns suite infinie décroissante pour $>_{\mathcal{M}(D)}$. On construit alors par récurrence sur $\mathbb{N}$ une suite infinie $\{A_i\}_{i \in \mathbb{N}}$ d'arbres vérifiant les propriétés suivantes :

(i) M_i est le multi-ensemble des feuilles de A_i,

(ii) Si x est l'etiquette d'un nœud interne de A_i différent de la racine, et y l'étiquette d'un fils de ce nœud, alors $x >_D y$,

(iii) A_i est un arbre à branchement fini.

A_0 est un arbre de racine quelconque dont les feuilles sont étiquettées par les éléments de M_0, satisfaisant ainsi les propriétés annoncées. Soit $M_i >_{\mathcal{M}(D)} M_{i+1}$. Par définition, $M_i = M \cup \{x\}$ et $M_{i+1} = M \cup \{y_1, \ldots, y_n\}$, avec $x >_D y_i$ pour tout $i \in [1..n]$. Par hypothèse de récurrence, x est une feuille de M_i. On obtient alors M_{i+1} en ajoutant n fils à la feuille de M_i étiquettée par x, qui sont étiquettés par $y_1, \ldots, y_n$. Il est clair que cette construction préserve les propriétés (i), (ii) et (iii). Cette suite strictement croissante d'arbres (pour l'ordre $A \subseteq A'$ ssi toute position de A est une position de A' qui a la même étiquette de surcroît) a donc une limite qui est un arbre infini satisfaisant les propriétés (ii) et (iii). Comme cet arbre infini est à branchement fini, le lemme de König implique l'existence d'une branche infinie, notée B, qui est donc une suite infinie de positions de l'arbre infini. Par définition de l'arbre infini, $\forall p \in B$, $\exists i \in \mathbb{N}$ tel que p est une feuille de A_i (i n'est bien sûr pas unique, mais l'étiquette de p est la même pour tous les A_i possibles). Soit maintenant la suite $S = \{x_i \mid \exists p \in B$ tel que p est une feuille de A_i d'étiquette $x_i\}$. Par la propriété (ii), $x_i >_D x_{i+1}$. S est donc une suite strictement décroissante d'éléments de D, ce qui est impossible.

Exemple 6.22 *Considérons le système de réécriture composé de la seule règle*

$$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$$

On considère alors la fonction d'interprétation $\phi(t)$ qui est le multi-ensemble des nombres $|u|$ pour $u \cdot v$ sous-terme de t. Par exemple, $\phi((a \cdot b) \cdot ((a \cdot b) \cdot c)) = \{|a|, |a|, |a| + |b| + 1, |a| + |b| + 1\}$ si l'on suppose que a et b ne contiennent pas d'occurrence de $\cdot$.

Si $t[u]_p$ se réécrit en $t[v]_p$ à la position p, alors $\phi(t[u]_p) = \phi(u) + K$ et $\phi(t[v]_p) = \phi(v) + K$. Il suffit donc de prouver que $\phi(u) >_m \phi(v)$. Par hypothèse, $u \equiv (u_1 \cdot u_2) \cdot u_3$ et $v \equiv u_1 \cdot (u_2 \cdot u_3)$. Donc $\phi(u) = \{|u_1| + |u_2| + 1, |u_1|\} + \phi(u_1) + \phi(u_2) + \phi(u_3)$ et $\phi(v) = \{|u_1|, |u_2|\} + \phi(u_1) + \phi(u_2) + \phi(u_3)$. D'où $\phi(u) >_m \phi(v)$ ce qui achève la preuve de terminaison.

Mais dans cet exemple, une extension lexicographique aurait tout aussi bien permis de prouver la terminaison.

On peut noter que l'extension multi-ensemble d'une relation d'ordre $\geq$ est un ordre total ssi $\geq$ est un ordre total.

6.4.4 Les théorèmes de Higman et de Kruskal

Le théorème de Kruskal est un résultat essentiel sur lequel se fondent la pluspart des méthodes de preuves automatiques de terminaison. Le Théorème de Higman en est le cas particulier lorsque tous les symboles de fonction sont unaires (cas des mots).

Plongement

Pour simplifier, on suppose qu'il n'y a qu'une sorte. De plus F est supposé muni d'un beau préordre $\geq_F$ (par exemple l'égalité lorsque F est fini). Nous définissons le plongement pour les termes clos. Pour les termes avec variables, il suffira de considérer les variables comme des constantes particulières. L'ordre $\geq_F$ sera alors clos par réflexivité en ajoutant les seules paires (x, x) pour les variables x. Il est immédiat de voir que la relation obtenue est un beau préordre si le nombre des variables est fini.

Définition 6.23 *Soient deux termes $u = f(u_1, \ldots, u_m)$ et $v = g(v_1, \ldots, v_n)$. On dit que u* se plonge *dans v, ce que l'on note $u \trianglelefteq v$, ssi l'un des deux cas suivants est satisfait :*

- *$f \leq_F g$ et il existe $j_1 < \ldots < j_m$ sous-suite croissante de $[1..m]$, tels que, pour tout $i \in [1..m]$, $u|_i \trianglelefteq v_{j_i}$*
- *$\exists i \in \{1, \ldots, n\}$ tel que $u \trianglelefteq v|_i$.*

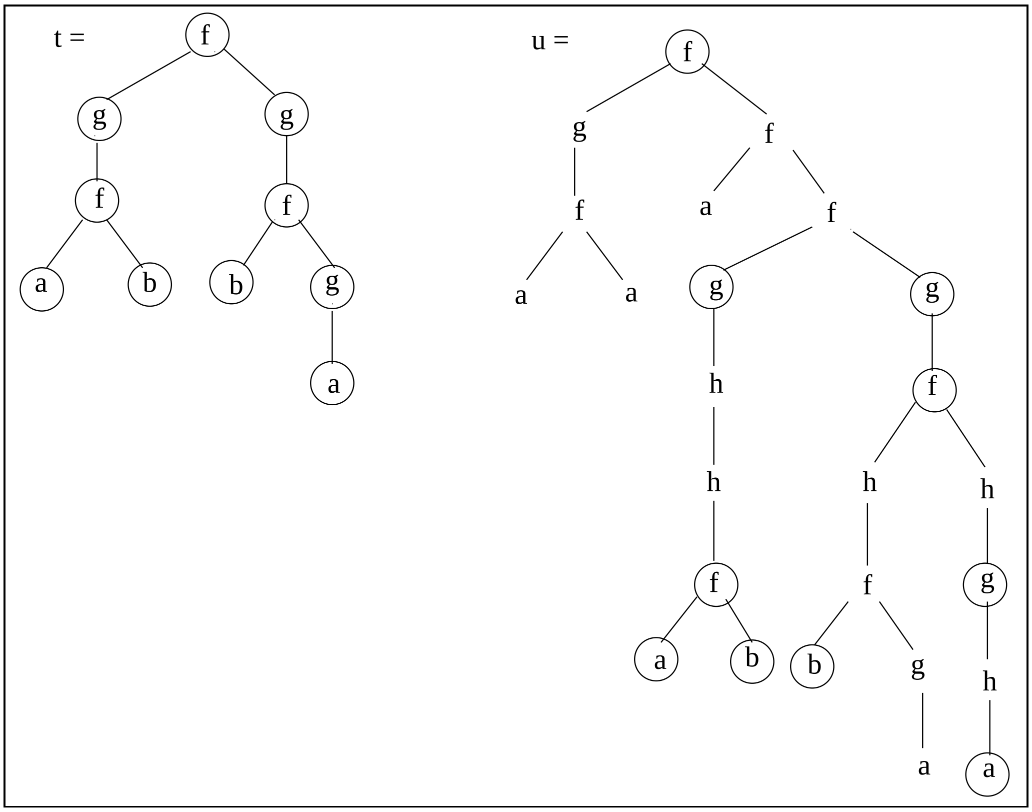

Figure 6.1: Exemple de plongement: $t \trianglelefteq u$

Notons par exemple que si $a, b \in F$ sont deux constantes telles que $a \simeq_F b$, alors $a \trianglelefteq b$.

Exemple 6.24 *Sur la figure 6.1 sont représentés les deux termes*

$$t \equiv f(g(f(a,b)), g(f(b, g(a))))$$

et

$$u \equiv f(g(f(a,a)), f(a, f(g(h(h(f(a,b)))), g(f(h(f(b,g(a))), h(g(h(a)))))))).$$

On a $t \triangleleft u$. Ce résultat peut s'obtenir de plusieurs façons dont, en particulier, celle que suggère la figure 6.1.

Lemme 6.25 *Si s est sous-terme de t, alors $s \trianglelefteq t$.*

Preuve
Par récurrence sur la taille de t: si s et t ont même taille, alors $s \equiv t$ et donc $s \trianglelefteq t$ en appliquant les deux premières règles de la définition. Si $s \equiv t|_{i \cdot p}$, on applique l'hypothèse de récurrence: s est un sous-terme de $(t|_i)|_p$ et donc $s \trianglelefteq t|_i$. Il suffit alors d'appliquer le second cas de la définition. $\square$

Lemme 6.26 $\trianglelefteq$ *est un préordre bien fondé.*

Preuve
Il suffit de remarquer que, lorsque $u \triangleleft v$, alors la taille de u est strictement inférieure à celle de v. □

Le lemme suivant établit que le plongement est un beau préordre dans le cas des mots.

Lemme 6.27 (Higman) *Si $\geq$ est un beau préordre sur D, alors la relation suivante, est un beau préordre sur D^*, ensemble des mots construits sur l'alphabet D:*

1. *$\Lambda \trianglelefteq \Lambda$, où Λ désigne le mot vide,*

2. *$u \trianglelefteq a \cdot v$ si $u \trianglelefteq v$,*

3. *$b \cdot u \trianglelefteq b \cdot v$ si $b \leq a \& u \trianglelefteq v$).*

Preuve
Raisonnons par l'absurde : supposons que l'ensemble E des suites (dites contre-exemples) de mots $\{u_i\}_{i \in \mathbf{N}}$ telles que, si $i < j$, $u_i \not\trianglelefteq u_j$, est non-vide. Cela implique l'existence d'une suite contre-exemple minimale $\{w_i\}_{i \in \mathbb{N}}$ définie comme suit: pour tout $i \in \mathbb{N}$, le mot de rang i, soit w_i, est de taille minimale parmi l'ensemble des mots de rang i de toutes les suites contre-exemple commençant par des mots de taille $\mid w_0 \mid, \ldots, \mid w_{i_1} \mid$.

La suite w_i ne contient pas le mot vide puisqu'elle est dans E et que le mot vide se plonge dans tous les autres mots. Soit alors $w_i = a_i \cdot v_i$ avec $a_i \in D$. Comme $\geq$ est un pré-bel ordre, on peut extraire par le lemme 6.10 une sous-suite a_{i_j} telle que, pour tout j, $a_{i_j} \leq a_{i_{j+1}}$. La suite $(x_i)_{i \in \mathbf{N}} = w_0, \ldots w_{i_0 - 1}, v_{i_0}, v_{i_1}, \ldots$ n'est pas dans E, car sinon la suite $\{w_i\}_{i \in \mathbb{N}}$ ne serait pas minimale. Donc il existe deux indices $i < j$ tels que $x_i \trianglelefteq x_j$. Mais i ne peut être inférieur à i_0 puisque $w_i \trianglelefteq v_i$ entraine $w_i \trianglelefteq w_j$ par définition et que la suite (w_i) est dans E. Il en résulte qu'il existe deux indices $j_1 < j_2$ tels que $v_{i_{j_1}} \trianglelefteq v_{i_{j_2}}$. Mais alors $a_{i_{j_1}} \cdot v_{i_{j_1}} \trianglelefteq a_{i_{j_2}} \cdot v_{i_{j_2}}$ par définition de $\trianglelefteq$, ce qui contredit le fait que (w_i) est dans E. □

Theorem 6.28 (Kruskal) *Étant donné un pré-bel ordre sur F, le plongement $\trianglelefteq$ est un pré-bel ordre sur $T(F)$.*

Preuve
Nous utilisons à nouveau une preuve basée sur le "contre-exemple minimal": raisonnant par l'absurde, l'ensemble E des suites de termes $(t_i)_{i \in \mathbf{N}}$ de $T(F)$ telles que $\forall i < j, t_i \not\trianglelefteq t_j$ est non vide. Il existe alors une suite $(u_i)_{i \in \mathbf{N}}$ minimale dans E, c'est-à-dire telle que, pour tout i, il n'existe aucune suite $u_0, \ldots, u_n, v, \ldots$ telle que v soit strictement plus petit en taille que u_{n+1} (le raisonnement est le même que dans la preuve du lemme 6.27). A nouveau, comme $\geq_F$ est un pré-bel ordre, on peut extraire une sous-suite t_{i_j} telle que la suite $t_{i_j}(\Lambda)$ soit croissante pour $\geq_F$. Soit alors pour tout j, w_j le mot (formé sur l'alphabet $T(F)$) des sous-termes

immédiats de t_{i_j}. Par minimalité de la suite (t_i), la restriction de $\trianglelefteq$ aux lettres de w_j est un pré-bel ordre. Par le lemme 6.27, l'extension de $\trianglelefteq$ aux mots sur cet alphabet est elle-aussi un pré-bel ordre. Il existe donc deux indices j_1 et j_2 tels que $w_{j_1} \trianglelefteq w_{j_2}$. Mais alors $t_{j_1} \trianglelefteq t_{j_2}$, ce qui contredit le fait que w_i est une suite minimale. □

Par exemple, si l'alphabet F est fini, on en déduit le résultat suivant:

Corollaire 6.29 *Tout sous-ensemble infini de $T(F)$ contient une suite strictement croissante pour le plongement.*

En fait, quand F est fini, il est inutile de préciser $\geq_F$ qui est toujours un pré-bel ordre.

6.4.5 Ordres de simplification

Comme nous l'avons annoncé, le théorème de Kruskal permet de construire de façon systématique des ordres bien fondés sur les termes: ce sont les ordres de simplification.

Définition 6.30 *Un (pré)-ordre de* simplification *est un (pré)-ordre sur $T(F, X)$ qui est monotone, stable, et contient sous-terme.*

Proposition 6.31 *Les (pré)-ordres de simplification sont bien fondés.*

Preuve
En effet, ils contiennent le plongement qui est un pré-bel ordre. □

Exemple 6.32 *Les exemples de préordres de simplification abondent. Nous verrons ci-dessous la classe des ordres récursifs sur les chemins. Mais nous pouvons déjà noter que les préordres suivant sont des préordres de simplification:*

- *Le préordre défini sur $T(F, X)$ par la fonction d'interprétation associant à un terme sa taille.*
- *Le préordre défini sur $T(F, X)$ par la fonction associant à un terme sa profondeur.*
- *La composée lexicographique de deux ordres de simplification.*

On suppose l'ensemble des symboles de fonctions partitionné en deux sous-ensembles $F = Mul \uplus Lex$. On dira que $f \in Lex$ a le *statut* lexicographique, et $f \in Mul$ a le statut multi-ensemble. Cette partition doit satisfaire les propriétés suivantes :
(i) si le statut de f n'est pas lexicographiue, alors l'arité de f est définie,
(ii) si $f \simeq_F g$, alors f et g ont le même statut,
(iii) si $f \simeq_F g$ ont le statut lexicographique, alors f et g ont la même arité.

Définition 6.33 (Ordres récursifs sur les chemins) *Supposons que $F = Mul \uplus Lex$ est muni d'un pré-bel ordre $\geq_F$ (ce sera souvent l'égalité dans le cas où F est fini). Le* pré-ordre récursif sur les chemins $\geq_{rpo}$ *étendant $\geq_F$ est défini par:*

$$s \equiv f(\overline{s}) \geq_{rpo} g(\overline{t}) \equiv t$$

ssi et seulement si l'une des conditions suivantes est remplie:

1. *(sous-terme)*
   $\exists u \in \overline{s},\ u \geq_{rpo} t$.

2. *(precedence)*
   $f >_F g$ *et* $\forall u \in \overline{s},\ s >_{rpo} u$.

3. *(multi-ensemble)*
   $f \simeq_F g$ *et* $f \in Mul$ *et* $\overline{s}\ (\geq_{rpo})_{mul}\ \overline{t}$.

4. *(lexicographique)*
   $f \simeq_F g$ *et* $f \in Lex$ *et* $\forall u \in \overline{t} s >_{rpo} u$ *et* $\overline{s}(\geq_{rpo})_{lex}\overline{t}$.

Notons tout d'abord que cette définition est éffective, car les appels récursifs à $\geq_{rpo}$ se font sur des paires de termes plus petits. Plus précisément, si $\geq_{rpo}$ est défini sur les paires de termes (t, u) dont les tailles sont strictement inférieures (disons pour l'extension lexicographique de l'ordre sur les entiers) à (n, m), alors $\geq_{rpo}^*$ est défini sur des paires de multi-ensembles de termes $(\{u_1, \ldots, u_p\}, \{v_1, \ldots, v_k\}$ telles que, pour tous i, j, $|u_i| < n$ et $|v_j| < m$ et $\geq_{rpo}^{lex}$ est défini sur les paires de tuples $((u_1, \ldots, u_k), (v_1, \ldots, v_k))$ tels que pour tous i, j, $|u_i| < n$ et $|v_j| < m$. La définition ci-dessus donne l'ordre $\geq_{rpo}$ sur les paires de termes de tailles respectives n et m.

Lorsque tous les statuts sont lexicographiques, alors $\geq_{rpo}$ est encore noté $\geq_{lpo}$ et appelé *ordre lexicographique sur les chemins*. De même, lorsque tous les statuts sont multi-ensemble, $\geq_{rpo}$ est aussi noté $\geq_{mpo}$ et appelé *ordre multi-ensemble sur les chemins*.

En pratique, il est parfois utile d'adopter un statut lexicographique droite-gauche, ou même un parcours fixé à l'avance des sous-termes d'un symbole lexicographique (ce qui nécessite de définir l'extension lexicographique correspondante). Enfin, on peut aussi mélanger les statuts lexicographiques et multi-ensemble pour un même symbole de fonction. Ces extensions sont simples à définir et à manipuler, mais compliqueraient nos notations. Elles sont laissées en exercice.

Exemple 6.34 *Supposons que F se compose de deux symboles binaires $*$ et $+$ et que $* >_F +$. Alors $(x + y) * z \geq_{mpo} (x * y) + (x * z)$ puisque*

- $* >_F +$ *et il suffit donc de vérifier que* $(x+y)*z >_{mpo} x*y$ *et* $(x+y)*z >_{mpo} x * z$.

- $x + y >_{mpo} x$ *et* $x + y >_{mpo} y$ *donc* $\{x + y, z\} >_{mpo}^* \{x, y\}$. *Et donc* $(x + y) * z >_{mpo} x * y$.

- *de même* $(x+y)*z >_{mpo} x*z$

Lemme 6.35 *Soient* $s \succeq_{rpo} t$ *et* $t = g(\overline{t})$*. Alors,* $\forall u \in \overline{t}$*,* $s \succ_{rpo} u$*.*

Preuve
Soit $s \succeq_{rpo} t$. On montre que $s \succ_{rpo} t_i$ pour tout $i \in [1..n]$ par récurrence sur la somme des tailles de s et t, et on distingue plusieurs cas suivant la preuve de $s \succeq_{rpo} t$.

1. Si $s \succeq_{rpo} t$ par sous-terme, alors $s_j \succeq_{rpo} t$ pour un certain s_j, et on conclue par récurrence que $s_j \succ_{rpo} t_i$, et donc $s \succeq_{rpo} t_i$ par sous-terme. Supposons que $t_i \succeq_{rpo} s$. Alors $t_i \succ_{rpo} s_j$ par hypothèse de récurrence, ce qui donne une contradiction. Donc, $s \succ_{rpo} t_i$.

2. Si $s \succeq_{rpo} t$ par l'un des deux cas precedence ou lexico, alors $s \succ_{rpo} t_i$ par définition de l'ordre.

3. Si $s \succeq_{rpo} t$ par cas multi-ensemble, alors il existe nécessairement un s_j tel que $s_j \succeq_{rpo} t_i$. Donc, $s \succeq_{rpo} t_i$ par sous-terme. Supposons maintenant que $t_i \succeq_{rpo} s$. Alors $t_i \succ_{rpo} s_j$ par hypothèse de récurrence, ce qui donne une contradiction. Donc, $s \succ_{rpo} t_i$. □

Corollary 6.36 *Supposons que* $s \succeq_{rpo} t$ *par sous-terme ou precedence. Alors* $s \succ_{rpo} t$*.*

Preuve
On raisonne par contradiction, supposant que $t \succeq_{rpo} s$, et donc $t \succ_{rpo} s_i$ pour tous les sous-termes immediats s_i de s par le lemme 6.35. Si $s \succeq_{rpo} t$ par sous-terme, c'est-à-dire $s_j \succeq_{rpo} t$ pour un s_j, nous obtenons une contradiction. Si $s \succeq_{rpo} t$ par precedence, alors la seule possibilité d'avoir $t \succeq_{rpo} s$ est par sous-terme, d'où $t \succ_{rpo} s$ par l'argument précédent, ce qui contredit le fait que $s \succeq_{rpo} t$. □

Lemme 6.37 $s \succeq_{rpo} t$ *et* $t \succeq_{rpo} s$ *ssi* $s =_{Mul} t$*.*

Preuve
Remarquons d'abord que $s \succeq_{rpo} t$ si $s =_{Mul} t$. La preuve de la réciproque est par récurrence sur la taille de s. Notons que les preuves que $s \succeq_{rpo} t$ et $t \succeq_{rpo} s$ ne peuvent être par sous-terme ou precedence par le corollaire 6.36. s et t sont donc surmontés du même symbole de fonction, et on conclue aisément par récurrence. □

Corollary 6.38 $\succeq_{rpo}$ *est réflexive.*

Corollary 6.39 $\succ$ *a la propriété de sous-terme (strict).*

Preuve
Notons que nous utilisons pour cela les deux lemmes 6.37 and 6.35. □

Lemme 6.40 $\succeq_{rpo}$ *est transitive.*

Preuve
Supposons que $s = f(\overline{s}) \succeq_{rpo} t = g(\overline{t}) \succeq_{rpo} u = h(\overline{u})$. La preuve est par récurrence sur la somme des tailles des termes s, t, u, et par cas suivant les preuves de $s \succeq_{rpo} t$ et $t \succeq_{rpo} u$. On écrira (p, q) pour indiquer que $s \succeq_{rpo} t$ a lieu par cas p, et $t \succeq_{rpo} u$ par cas q.

1. cas $(1, 2)$. Alors $s_i \succeq_{rpo} t \succeq_{rpo} u$ pour un certain s_i. Par hypothèse de récurrence, $s_i \succeq_{rpo} u$, et donc $s \succeq_{rpo} u$ par sous-terme.

2. cas $(\neq 1, 1)$. Alors, $t_i \succeq_{rpo} u$. Par le lemme 6.35, $s \succ_{rpo} t_i$, et par hypothèse de récurrence, $s \succeq_{rpo} u$.

3. cas (2 ou 3 ou 4, 2). Alors $f \geq_F g >_F h$ et $t \succ_{rpo} u_i$ pour tous les u_i. Par hypothèse de récurrence, $s \succeq_{rpo} u_i$ pour tous les u_i et $s \succeq_{rpo} t$ par cas 2.

4. cas (2, 3 ou 4). Alors $t \succ_{rpo} u_i$ pour tous les u_i par le lemme 6.35. Par hypothèse de récurrence, $s \succeq_{rpo} u_i$ pour tous les u_i. Supposons que $u_i \succeq_{rpo} s$. Alors $u_i \succeq_{rpo} t$ par hypothèse de récurrence, contradisant le fait que $t \succ_{rpo} u_i$. Donc $s \succ_{rpo} u_i$ et $s \succeq_{rpo} u$ par cas 2.

5. cas (3 ou 4, 3 ou 4). On conclue aisément par application de l'hypothèse de récurrence. □

Lemme 6.41 *Supposons la précédence $\geq_{\mathcal{F}}$ totale sur $\mathcal{F}$. Alors $\succeq_{rpo}$ est un préordre total sur les termes clos.*

Preuve
Supposons que $s = f(\overline{s}), t = g(\overline{t})$ soit la paire de termes clos de taille minimale telle que s et t soient incomparables. Par hypothèse de minimalité, les paires $(s, t_j), (s_i, t), (s_i, t_j)$ sont comparables pour tous $s_i \in \overline{s}$ et $t_j \in \overline{t}$. Si $t_j \succeq_{rpo} s$ pour un certain $t_j \in \overline{t}$, alors $t \succ_{rpo} s$, ce qui est une contradiction. Donc, $s \succ t_j$ pour tous $t_j \in \overline{t}$. Par argument de symmétrie, $t \succ s_i$ pour tous $s_i \in \overline{s}$. On distingue maintenant plusieurs cas :

1. $f >_{\mathcal{F}} g$. Alors $t \succ_{rpo} s$ par cas precedence, une contradiction.

2. $g >_{\mathcal{F}} f$ est symmétrique.

3. $f = g \in Mul$. Comme toutes les paires (s_i, t_j) sont comparables, soit $\overline{s}(\succeq_{rpo})_{mul}\overline{t}$, auquel cas $s \succeq_{rpo} t$; soit $\overline{t}(\succeq_{rpo})_{mul}\overline{s}$, auquel cas $t \succeq_{rpo} s$, et nous obtenons donc une contradiction dans les deux cas.

4. $f = g \in Lex$. À nouveau, $\overline{s} \succ_{mul} \overline{t}$, avec $s \succeq_{rpo} t$ puisque $s \succ_{rpo} t_j$ pour tous $t_j \in \overline{t}$, ou bien $\overline{t} \succ_{mul} \overline{s}$, avec $t \succeq_{rpo} s$ puisque $t \succ_{rpo} s_i$ pour tous $s_i \in \overline{s}$, ce qui donne à nouveau une contradiction dans les deux cas. □

Lemme 6.42 *$\succ$ est monotone.*

Preuve
Soient deux termes s et t tels que $s \succ_{rpo} t$. On montre aisément par cas 4 ou 3 suivant le statut de f que $f(\ldots, s, \ldots) \succ_{rpo} f(\ldots, t, \ldots)$. □

Lemme 6.43 $\succ$ *est stable.*

Preuve
Soient $s = f(s_1, \ldots, s_n)$ et $t = g(s_1, \ldots, s_n)$, avec $f, g \in (\mathcal{F} \cup \mathcal{X})$, deux termes tels que $s \succ_{rpo} t$, et soit σ une substitution. On montre que $s\gamma \succ_{rpo} t\gamma$ par récurrence sur $|s| + |t|$. On distingue 5 cas, suivant la preuve que $s \succ_{rpo} t$:

1. $s_i \succeq_{rpo} t$ pour un certain i. Par hypothèse de récurrence, $s_i\gamma \succeq_{rpo} t\gamma$, d'où $s\gamma \succ_{rpo} t\gamma$ par cas sous-terme.
2. $f >_{\mathcal{F}} g$ et $s \succ_{rpo} t_i$ pour tout i. par hypothèse de récurrence, $s\gamma \succ_{rpo} t_i\gamma$, et donc, $s\gamma \succ_{rpo} t\gamma$ par cas precedence.
3. $f \simeq_{\mathcal{F}} g \in Mul$ et $\overline{s} \succ_{mul} \overline{t}$. Par hypothèse de récurrence, $\overline{s\gamma} \succ_{mul} \overline{t\gamma}$, ce qui prouve le résultat par cas multi-ensemble.
4. $f \simeq_{\mathcal{F}} g \in Lex$, $\overline{s} \succ_{lex} \overline{t}$ et $s \succ t_i$ pour tout i. Par hypothèse de récurrence, $\overline{s\gamma} \succ_{lex} \overline{t\gamma}$ et $s\gamma \succ_{rpo} t_i\gamma$, ce qui prouve le dernier cas par cas lexicographique. $\square$

Theorem 6.44 $\geq_{rpo}$ *est un préordre de simplification (donc bien fondé). Il est* stable, *i.e. si* $t \geq_{rpo} u$, *alors, pour toute substitution* σ, $t\sigma \geq_{rpo} u\sigma$.

Preuve
Par récurrence, on montre que $\geq_{rpo}$ contient le plongement. $\square$

Ce résultat est fondamental car il permet de prouver la terminaison de systèmes de réécriture en montrant seulement que pour chaque règle $l \to r$, $l >_{rpo} r$.

Remarquons (la preuve est laissée en exercice) que, lorsque $\geq_F$ est un préordre total, alors il en est de même de $\geq_{rpo}$. On peut alors comparer les notations ordinales fournies par les termes avec les notations ordinales déjà introduites.

Exemple 6.45 *Supposons que* $+$ *est binaire et* $+ >_F 0$ *où* 0 *est une constante. On peut établir la correspondance:*

Terme	0	$0+0$	$0+(0+0)$	$(0+0)+0$	$0+((0+0)+0)$
Ordinal	0	1	2	ω	$\omega+1$

Terme	$(0+0)+(0+0)$	$(0+0)+(0+(0+0))$
Ordinal	$\omega \times 2$	$\omega \times 3$

Terme	$(0+0)+((0+0)+(0+0))$	$((0+0)+0)+0$
Ordinal	ω^2	ω^ω

Terme	$(((0+0)+0)+0)+0$
Ordinal	ϵ_0

L'ordinal défini par $\geq_{rpo}$ *est donc, même dans ce cas simple, bien supérieur à* ϵ_0.

6.5 Exercices

Dans les exercices qui suivent, la signature ne sera pas précisée. Nous supposerons toujours, sauf précision contraire, qu'il n'y a qu'une sorte. L'arité des opérateurs pourra alors toujours être déduite du contexte. Les symboles binaires pourront être utilisés en notation infixée et les symboles unaires en notation préfixée. Les symboles $x, x', y, z, x_1, \ldots$ sont réservés pour désigner des variables.

6.5.1 Ordinaux

1. Montrer les propriétés suivantes de l'addition des ordinaux:

   (a) $0 + x = x + 0$

   (b) Si $x < y$, alors $z + x < z + y$

   (c) Si $z + x = z + y$, alors $x = y$

   (d) Si $x < y$, alors $x + z < y + z$

   (e) Si $x \leq y$, alors il existe un unique ordinal z tel que $x + z = y$

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

2. Montrer les propriétés suivantes de la multiplication des ordinaux:

   (a) $0 \times x = x \times 0 = 0$

   (b) $1 \times x = x \times 1 = x$

   (c) Si $x < y$ et $z > 0$, alors $z \times x < z \times y$ et $x \times z \leq y \times z$.

   (d) Si $z \times x = z \times y$, alors $x = y$

   (e) Si $x \times y = 0$, alors $x = 0$ ou $y = 0$.

   (f) $x \times (y + z) = (x \times y) + (x \times z)$

   (g) $x \times (y \times z) = (x \times y) \times z$

   (h) Si $y \neq 0$, alors pour tout x, il existe un unique q et un unique r tels que $x = q \times y + r$.

3. Montrer les propriétés de l'exponentiation des ordinaux:

   - Si $z > 1$ et $x < y$, alors $z^x < z^y$
   - Si $x < y$ et z est un ordinal successeur alors $x^z < y^z$.
   - Si $n < \omega$, alors $n^\omega = \omega$. (Qu'en déduire pour le cas z ordinal limite dans la propriété ci-dessus ?)
   - $(x^y)^z = x^(y \times z)$.

4. Montrer que $2^{2^{2^{2^{\omega+1}}}} = \omega^{\omega^{\omega^{\omega}}}$

5. Donner un ordre $\geq$ sur les entiers naturels qui soit isomorphe à $\omega \times \omega$. Même question pour ω^ω.

6. Soit b un entier supérieur ou égal à 2 et soit $\phi_b(n)$ la représentation de l'entier n définie par récurrence par:

   - $\phi_b(x) = x$ si $x < b$
   - Si $x \geq b$, soit $x = a_n b^n + \ldots + a_1 b + a_0$ où $a_n, \ldots, a_0 < b$ sa représentation en base b. Alors $\phi_b(x) = a_n b^{\phi(n)} + \ldots + a_1 b^{\phi(1)} + a_0$.

   Si maintenant a est un entier naturel, on définit les suites u_n et b_n par:

   - $u_0 = a$ et $b_0 = 2$
   - Si $u_n > 0$, alors u_{n+1} est obtenu en remplaçant b_n par b_{n+1} dans $\phi_{b_n}(u_n)$ et en retranchant 1. b_{n+1} est un entier strictement supérieur à b_n.

   Montrer que, pour tout a, il existe un indice n tel que $u_n = 0$ (et donc que toutes les suites u_n sont finies)

6.5.2 Extensions lexicographique et multi-ensemble

1. Soit $\simeq_m$ la relation définie sur $\mathcal{M}(D)$ par:

   $$M \simeq_m M' \text{ ssi } \begin{cases} \text{ou bien} & M = M' = \emptyset \\ \text{ou bien} & \exists x \in M, \exists x' \in M', x \simeq x' \text{ et } M - \{x\} \simeq_m M' - \{x'\} \end{cases}$$

   Montrer que $M \geq_{mul} M'$ si et seulement si, il existe des multi-ensembles M_1, M_2, M'_1, M'_2 tels que :

   - $M = M_1 + M_2$ et $M' = M'_1 + M'_2$
   - $M_1 \simeq_m M'_1$
   - $\forall x \in M'_2, \exists y \in M_2, y > x$

2. Montrer la terminaison de la fonction définie sur les entiers naturels par:

```
let rec Ack = function
        (0,x) -> x +1
        | (x,0) -> Ack(x-1,1)
        | (x,y) -> Ack(x-1,Ack(x,y-1))
;;
```

3. Montrer la terminaison de la fonction définie sur les entiers naturels par:

```
let rec f = function
        (0,x) -> 1
        | (x,0) -> 1
        | (x,y) -> f(x-1,x-1) + 2 * f(x-1,y-1) + f(y-1,y-1)
;;
```

4. Montre la terminaison du combat d'Hercule et de l'hydre. Les termes (représentant l'hydre à un instant donné) sont formés à l'aide d'un alphabet F contenant un symbole b d'arité variable (les "branchements" des cous de l'hydre) et d'une constante t (tête de l'hydre). Au départ, l'hydre est représenté par un terme h_0 et, si h_i est la représentation de l'hydre à l'instant i, h_{i+1} est obtenu par application d'une des règles

$$\begin{array}{rcl} h_i[c(x_1,\ldots,x_n,c(y_1,\ldots,y_m))] & \rightarrow & h_i[c(x_1,\ldots,x_n,t,\ldots,t)] \\ h_i \rightarrow 0 & & \end{array}$$

le nombre de têtes ayant "repoussé" dans le membre droit de la première règle est arbitraire. La deuxième règle exprime la fin du combat.

Montrer que, quelle que soit la stratégie d'Hercule (i.e. quelle que soit la façon dont h_{i+1} est obtenu à partir de h_i à l'aide d'une des règles ci-dessus) et quel que soit le nombre de têtes repoussant à chaque étape, Hercule sortira vainqueur du combat.

5. x, y, z sont des multi-ensembles sur D. A quelles conditions a-t-on

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

   (b) $x + y = x + z \Rightarrow y = z$

   (c) $x - y = x - z \Rightarrow y = z$

6. Montrer qu'il n'existe aucune fonction ϕ de $\mathbf{N} \times \mathbf{N}$ dans $\mathbf{N}$ telle que

$$(a_1, a_2) \geq_{lex} (b_1, b_2) \Leftrightarrow \phi(a_1, a_2) \geq \phi(b_1, b_2)$$

($\geq$ désigne ici l'ordre habituel sur $\mathbf{N}$).

7. Même exercice que le précédent avec l'ordre multi-ensemble : montrer qu'il n'existe pas de fonction ϕ de $\mathcal{M}(\mathbf{N})$ dans $\mathbf{N}$ telle que

$$x \geq_m y \Leftrightarrow \phi(x) \geq \phi(y)$$

8. Supposons que l'on restreigne l'ordre multi-ensemble en n'autorisant qu'un nombre borné de remplacements. Plus précisément, soit $\geq_{m,n}$ l'ordre sur $\mathcal{M}(\mathbf{N})$ défini par: $x >_{m,n} y$ s'il existe une suite $z_0 = x, z_1, \ldots, z_k = y$ telle que z_{i+1} s'obtient à partir de z_i en retirant un multi-ensemble non-vide z_i' et en ajoutant un multi-ensemble z_{i+1}'' *de cardinal inférieur à* n tel que tout élément de z_{i+1}'' est strictement inférieur à un élément de z_i'. (La seule différence avec l'ordre multi-ensemble est que l'on impose une borne au cardinal de z_i'').

Montrer que l'on peut trouver une application ϕ de $\mathcal{M}(\mathbf{N})$ dans $\mathbf{N}$ telle que

$$x \geq_{m,n} y \Leftrightarrow \phi(x) \geq \phi(y)$$

9. Soit D un ensemble muni d'un ordre bien fondé $\geq_D$.

(a) Soit D_0 un sous-ensemble fini de D: $D_0 = \{d_1, \ldots, d_m\}$ tel que $j > i \Rightarrow d_j \not>_D d_i$. Montrer que

$$\forall x_1, x_2 \in \mathcal{M}(D_0),\ x_1 \geq_m x_2 \Leftrightarrow (x_1(d_1), \ldots, x_1(d_m)) \geq_{lex} (x_2(d_1), \ldots, x_2(d_m))$$

(b) Supposant que dans D il n'y a qu'un nombre fini d'éléments inférieurs à un d donné (par exemple, $\geq_D$ est un ordre total), montrer que le théorème 6.21 n'est qu'une conséquence de la proposition 6.15.

10. On appelle *multi-ensemble emboité* un multi-ensemble dont les éléments sont soit dans D soit des multi-ensembles d'éléments de D, soit des multi-ensembles de multi-ensembles d'éléments de D, ... et ainsi de suite. Plus formellement, si D est un ensemble muni d'un ordre $\geq_D$, un multi-ensemble emboité est ou bien un élément de D ou bien un multi-ensemble de multi-ensembles emboités sur D. On note $\mathcal{M}^*(D)$ l'ensemble des multi-ensembles emboités construits sur D. On définit alors la relation d'ordre $\geq_m^*$ sur $\mathcal{M}(D)$ par : $x >_m^* y$ ssi

   - ou bien $x, y \in D$ et $x >_D y$
   - ou bien $y \in D$ et $x \notin D$
   - ou bien $x, y \notin D$ et il existe $x',\ \in \mathcal{M}^*(D)$ tels que $\{\} \neq x' \subseteq x$, $y = (x - x') +$ et $\forall d \in ,\exists e \in x', e >_m^* d$

   (a) vérifier que $\geq_m^*$ est bien une relation d'ordre. Montrer que celle-ci est totale si et seulement si $\geq_D$ est totale sur D.

   (b) Classer par ordre croissant les multi-ensembles emboités d'entiers suivants: $\{\{1,1\},\{\{0\},1,2\},0\}$, $\{\{1,0,0\},5,\{\{0\},1,2\},0\}$, $\{\{\{\},1,2\},\{5,5,2\},5\}$.

   (c) On note $\mathcal{M}^i(D)$ l'ensemble des multi-ensembles emboités de profondeur inférieure à i. Autrement dit, $\mathcal{M}^0(D) = D$ et $\mathcal{M}^{i+1}(D)$ est l'ensemble des multi-ensembles dont les éléments sont dans $\mathcal{M}^0(D) \cup \ldots \cup \mathcal{M}^i(D)$ avec au moins un élément $\mathcal{M}^i(D)$. De cette façon, $\mathcal{M}^*(D) = \bigcup_{i \in \mathbf{N}} \mathcal{M}^i(D)$.
   Prouver que si $x, y \in \mathcal{M}^*(D)$ et x est de profondeur strictement supérieure à celle de y, alors $x >_m^* y$.

   (d) Prouver que $\geq_m^*$ est bien fondée si et seulement si $\geq_D$ l'est.

11. Soient $\geq_1, \ldots, \geq_n$ n relations d'ordre. Montrer que la composée lexicographique de ces n relations d'ordre est un ordre total ssi chacun des ordres $\geq_i$ est total. De même, prouver que l'extension multi-ensemble est un ordre total ssi l'ordre de départ est total.

12. Donner un exemple d'un ensemble de suites d'entiers qui ne contienne pas de suite minimale pour l'ordre suivant:

$$(u_i)_{i \in \mathbf{N}} < (v_i)_{i \in \mathbf{N}} \quad \text{ssi} \quad u_0 = v_0, \ldots, u_n = v_n, u_{n+1} < v_{n+1}$$

13. Prouver la proposition 6.14.

14. Soit le programme suivant qui calcule la fonction ("fonction 91 de McCarth")

```
let rec  g = function
          (0,x) -> x
        | (n,x) ->  if x > 100 then g(n-1,x-10)
                    else g(n+1,x+11)
;;
let f = function x -> g(1,x);;
```

Montrer que le calcul de f se termine toujours. Quelle est la fonction calculée ?

15. Quel est l'ordinal correspondant à la composée lexicographique des ordinaux $\alpha_1, \ldots, \alpha_n$? à l'extension multi-ensemble de α ?

6.5.3 Ordres de simplification

1. Montrer que $\geq$ est un (pré)ordre de simplification si et seulement si il est monotone et contient la relation de sous-terme. (On suppose ici F fini).

2. Montrer que la relation de plongement $\trianglelefteq$ est la plus petite relation d'ordre sur $T(F, X)$ qui soit monotone et qui possède la propriété de sous-terme.

3. On définit sur $T(F, X)^2$ la relation $\sqsubseteq$ de la façon suivante:

   $t \sqsubseteq u$ s'il existe une application injective ϕ de $Pos(t)$ dans $Pos(u)$ telle que:

   (a) ϕ est croissante par rapport à $\geq_{lex}$ et à $\geq_{pref}$. ($\geq_{lex}$ est l'extension lexicographique de l'ordre habituel sur les entiers: c'est aussi un ordre sur les ensembles de positions. $\geq_{pref}$ est l'ordre de préfixe sur les positions: $p \geq_{pref} q$ ssi il existe r tel que $p \cdot r = q$)

   (b) pour toute position $p \in Pos(t)$, $t(p) = u(\phi(p))$

   (a) Montrer que $\sqsubseteq$ est une relation d'ordre sur les termes.

   (b) Montrer que $\sqsubseteq$ est monotone et stable

   (c) Comparer $\sqsubseteq$ avec $\trianglelefteq$. (i.e. dire si elles sont égales, ou si l'une est strictement contenue dans l'autre. Les résultats négatifs devant être justifiés par un exemple)

   (d) $\sqsubseteq$ est elle une relation bien fondée ?

4. Montrer que, lorsque la précédence sur F est l'égalité, l'ordre $\geq_{rpo}$ se réduit à l'ordre de plongement.

5. Un système de réécriture $\mathcal{R}$

   tourne en rond si

$$\exists t,\ t \rightarrow^+_{\mathcal{R}} t$$

boucle si

$$\exists t, t', \ t \to^+_{\mathcal{R}} t' \text{ et } t \text{ est un sous-terme de } t'.$$

est auto-imbriqué Si

$$\exists t, t', \ t \to^+_{\mathcal{R}} t' \text{ et } t \trianglelefteq t'$$

termine faiblement si $\forall t, \ \exists t', \ t \to^+_{\mathcal{R}} t'$ et t' est irréductible.

(a) Montrer qu'on a les implications suivantes:
Tourne en rond $\Rightarrow$ Boucle $\Rightarrow$ Ne termine pas $\Rightarrow$ Auto-imbriqué

(b) Pour chacune des implications ci-dessus, donner un contre-exemple prouvant que l'implication inverse est fausse.[1]

(c) Donner un exemple montrant que $\mathcal{R}$ peut faiblement terminer sans être à terminaison finie.

(d) Donner un exemple de système de réécriture qui ne boucle pas et ne termine pas faiblement.

6. F n'est composée que d'un symbole binaire f et un symbole de constante a. Pour tout entier n, t_n désigne l'arbre binaire complet de profondeur n étiqueté par F. (t_n contient donc 2^n feuilles étiquetées par a et $2^n - 1$ noeuds étiquetés par f.) Combien y a-t-il (en fonction de n et k) de plongements de t_n dans t_{n+k} ?

7. Montrer que la composée lexicographique de n ordres de simplification est un ordre de simplification. Qu'en est-il de l'extension multi-ensemble d'un ordre de simplification ?

8. On suppose que F contient deux constantes $0, 1$ et un symbole binaire $+$ avec $+ >_F 1 >_F 0$. Donner dans le cas où $+$ a le statut lexicographique et dans le cas où $+$ a le statut multi-ensemble le terme correspondant aux ordinaux 5, ω, $\omega \times 2$, ω^2, ω^ω, ω^{ω^ω}.

9. Montrer la terminaison du calcul de la fonction

```
let rec f = function
        (0,x) -> 0
        | (x,0) -> f(f(x-1,x-1),x-1)
        | (x,y) -> f(y,f(x-1,x-1))
;;
```

définie sur les entiers naturels.

[1]Les exemples de la section précédente pourront être utiles

6.5.4 Exemples de SdR : preuves de terminaison

La plupart des exemples sont tirés de l'article de N. Dershowitz *Termination of Rewriting* paru dans le numéro spécial du *Journal of Symbolic Computation* consacré à la réécriture et datant de Février 87. On trouvera donc la plupart des solutions dans cet article. Nous donnerons aussi assez souvent des indications ou références bibliographiques en bas de page.

Dire si les systèmes de réécriture suivants sont à terminaison finie ou non. Justifier la réponse.

1. $\left\{ \begin{array}{lcl} f(a) & \rightarrow & f(b) \\ g(b) & \rightarrow & g(a) \end{array} \right.$ [2]

2. $f(f(x)) \rightarrow f(g(f(x)))$ [3]

3. $f(g(x)) \rightarrow g(f(x))$ [4]

4. $f(g(g(f(x)))) \rightarrow g(f(f(g(x))))$

5. (a) $f(g(x)) \rightarrow g(g(f(f(x))))$
   (b) $f(g(g(x))) \rightarrow g(f(f(g(x))))$
   (c) $f(f(g(x))) \rightarrow g(g(f(f(f(x)))))$
   (d) Plus généralement, pour quelles valeurs de n, k, m, p le système formé de la seule règle $f^n(g^k(x)) \rightarrow g^m(f^p(x))$ termine-t-il ? (Question difficile).

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

7. $if(if(x,y,z),x',) \rightarrow if(x, if(y,x',), if(z,x',))$

8. $\left\{ \begin{array}{rcl} a(0,x) & \rightarrow & s(x) \\ a(s(x),0) & \rightarrow & a(x,s(0)) \\ a(s(x),s(y)) & \rightarrow & a(x,a(s(x),y)) \end{array} \right.$ [5]

9. $\left\{ \begin{array}{rcl} x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ (x+y) \times z & \rightarrow & (x \times z) + (y \times z) \\ x \times 1 & \rightarrow & x \\ 1 \times x & \rightarrow & x \\ 0 \times x & \rightarrow & 0 \\ x \times 0 & \rightarrow & 0 \end{array} \right.$

[2] Utiliser une preuve directe
[3] On pourra utiliser un ordre d'interprétation
[4] On pourra utiliser un ordre récursif sur les chemins
[5] Vous l'avez reconnu, je suppose...

10. $\left\{\begin{array}{lcl} x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ (x+y) \times z & \rightarrow & (x \times z) + (y \times z) \\ (x \times y) \times z & \rightarrow & x \times (y \times z) \end{array}\right.$

11. $\left\{\begin{array}{rcl} --x & \rightarrow & x \\ -(x+y) & \rightarrow & -x \times -y \\ -(x \times y) & \rightarrow & -x + -y \\ x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ (x+y) \times z & \rightarrow & (x \times z) + (y \times z) \end{array}\right.$

12. $\left\{\begin{array}{rcl} g(0,x) & \rightarrow & s(x) \\ g(s(x),0) & \rightarrow & s(x) \\ g(s(x),s(y)) & \rightarrow & f(f(x,s(y)),y) \\ f(x,0) & \rightarrow & s(x) \\ f(x,s(y)) & \rightarrow & s(g(x,y)) \end{array}\right.$

13. $\left\{\begin{array}{rcl} f(g(0,s(x)),x) & \rightarrow & f(h(0),g(0,x)) \\ f(h(x),h(s(x))) & \rightarrow & f(g(x,0),h(x)) \\ h(0) & \rightarrow & s(0) \\ g(0,0) & \rightarrow & s(0) \end{array}\right.$

14. $\left\{\begin{array}{rcl} D_X X & \rightarrow & 1 \\ D_X a & \rightarrow & 0 \\ D_X(x+y) & \rightarrow & D_X x + D_X y \\ D_X(x \times y) & \rightarrow & y \times D_X x + x \times D_X y \\ D_X(x-y) & \rightarrow & D_X x - D_X y \\ D_X - x & \rightarrow & -D_X x \\ D_X(x/y) & \rightarrow & (D_X x)/y - x \times (D_X y)/y^2 \\ D_X(\ln x) & \rightarrow & (D_X x)/x \\ D_X(x^y) & \rightarrow & y \times x^{y-1 \times D_X x + x^y \times (\ln x) \times D_X y} \end{array}\right.$

15. $\left\{\begin{array}{rcl} --x & \rightarrow & x \\ -(x+y) & \rightarrow & ---x \times ---y \\ -(x \times y) & \rightarrow & ---x + ---y \end{array}\right.$

16. $(((C \cdot x) \cdot y) \cdot z) \cdot x' \quad \rightarrow \quad (x \cdot z) \cdot (((x \cdot y) \cdot z) \cdot x')$

17. $\left\{\begin{array}{lcl} x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ x + (y \times z) & \rightarrow & (x+y) \times (x+z) \end{array}\right.$

18. $\left\{\begin{array}{rcl} (x \cdot y) \cdot z & \rightarrow & x \cdot (y \cdot z) \\ (x+y) \cdot z & \rightarrow & (x \cdot z) + (y \cdot z) \\ x \cdot (y + f(z)) & \rightarrow & g(x,z) \cdot (y+a) \end{array}\right.$

19. $f(a,b,x) \quad \rightarrow \quad f(x,x,x)$

20. $\left\{\begin{array}{lcl} x \cdot (y \cdot z) & \rightarrow & y \cdot y \\ (x \cdot y) \cdot a & \rightarrow & x \cdot (y \cdot b) \end{array}\right.$

21. $\left\{\begin{array}{rcl} or(x,y) & \rightarrow & x \\ or(x,y) & \rightarrow & y \\ f(a,b,x) & \rightarrow & f(x,x,x) \end{array}\right.$

Qu'en concluez-vous?

22. $\left\{\begin{array}{rcl} h(x,x,y) & \rightarrow & g(x) \\ g(a) & \rightarrow & h(a,b,a) \\ i(x) & \rightarrow & f(x,x) \\ f(x,y) & \rightarrow & x \end{array}\right.$

23. $\left\{\begin{array}{rcl} --x & \rightarrow & x \\ -(x+y) & \rightarrow & ---x \times ---y \\ -(x \times y) & \rightarrow & ---x + ---y \\ x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ (x+y) \times z & \rightarrow & (x \times z) + (y \times z) \end{array}\right.$

24. $\left\{\begin{array}{rcl} --x & \rightarrow & x \\ -(x+y) & \rightarrow & --x \times --y \\ -(x \times y) & \rightarrow & --x + --y \\ x \times (y+z) & \rightarrow & (x \times y) + (x \times z) \\ (x+y) \times z & \rightarrow & (x \times z) + (y \times z)^{6} \end{array}\right.$

25. $\left\{\begin{array}{rcl} g(a) & \rightarrow & b \\ f(a,x) & \rightarrow & f(x,g(a)) \end{array}\right.$

26. $\left\{\begin{array}{rcl} f(a,x) & \rightarrow & f(b,x) \\ g(b) & \rightarrow & g(a) \end{array}\right.$

27. $\left\{\begin{array}{rcl} h(f(x),y) & \rightarrow & f(g(x,y)) \\ g(x,y) & \rightarrow & h(x,y) \end{array}\right.$

28. $\left\{\begin{array}{rcl} f(x) \cdot f(y) & \rightarrow & f(x \cdot y) \\ (x \cdot y) \cdot z & \rightarrow & x \cdot (y \cdot z) \\ f(x) \cdot (f(y) \cdot z) & \rightarrow & f(x \cdot y) \cdot z \end{array}\right.$

29. $\left\{\begin{array}{rcl} (x \cdot y) + (x \cdot z) & \rightarrow & x \cdot (y+z) \\ (x+y)+z & \rightarrow & x+(y+z) \\ (x \cdot y) + ((x \cdot z) + x') & \rightarrow & (x \cdot (y+z)) + x' \end{array}\right.$

30. $$\left\{\begin{array}{rcl} g(x_1 \cdot x_2) \cdot (g(x_3) \cdot g(x_1)) & \rightarrow & (g(x_1) \cdot x_2) \cdot g(x_3 \cdot x_4) \\ (g(x_1) \cdot g(x_2)) \cdot g(x_3 \cdot x_4) & \rightarrow & g(x_1 \cdot x_2) \cdot (g(x_3) \cdot x_4) \\ (x_1 \cdot x_2) \cdot (g(x_3) \cdot x_4) & \rightarrow & g(x_1 \cdot x_2) \cdot g(x_3 \cdot x_4) \end{array}\right.$$

31. $$\left\{\begin{array}{rcl} f(\perp \cdot y) & \rightarrow & f(y) \\ f(f(\perp \cdot y) \cdot z) & \rightarrow & c(n, y, z) \end{array}\right.$$

Où n est un entier naturel arbitraire (le système est donc infini) et $c(n, y, z)$ est défini par récurrence par:

$$\left\{\begin{array}{rcl} c(0, y, z) & \stackrel{\text{def}}{=} & f(z) \\ c(n+1, y, z) & \stackrel{\text{def}}{=} & c(n, y, f(y) \cdot z) \end{array}\right.$$

32. $$\left\{\begin{array}{rcl} (x \cdot y) \cdot z & \rightarrow & x \cdot (y \cdot z) \\ x \cdot (y \cdot z) & \rightarrow & y \cdot y \end{array}\right.$$

Chapter 7

Completion

Lorsque qu'un système de réécriture n'est pas confluent, on peut essayer de le rendre confluent en orientant des conséquences équationelles de R, tout simplement ses paires critiques. L'idée est de construire un système de règles canonique R' tel que pour tout couple (s,t), $s =_R t$ ssi $s{\downarrow_{R'}} = t{\downarrow_{R'}}$. Mais ajouter de nouvelles règles à un système de réécriture R ajoute de nouvelles paires critiques, et il faut donc vérifier que ces nouvelles paires sont confluentes.

7.1 règles de complétion

Le processus de complétion est décrit à la figure 7.1 sous forme d'un ensemble de règles d'inférence. Chaque règle d'inférence est une relation entre paires $(E;R)$, où E et R désignent respectivement un ensemble d'èquations et un ensemble de règles. Soit $\succ$ un ordre de réduction, et $\succ_{rgle}$ l'ordre induit sur les règles, défini par $l \to r \succ_{rgle} g \to d$ si (i) $l \rhd g$, ou sinon (ii) $l \doteq g$ and $r \succ d$.

On écrit $(E;R) \vdash_{KB} (E';R')$ si $(E';R')$ peut être obtenu à partir du premier par application d'une règle de KB.

Définition 7.1 *On appelle procedure de completion standard un programme qui prend comme données un ensemble fini d'équations E_0 et un ordre de réduction $\succ$, et retourne comme résultat un ensemble $(E;R)$ tel que $(E_0;\emptyset) \vdash^*_{KB} (E;R)$.*

Étant donnée une séquence $(E_0,\emptyset) \vdash_{KB} (E_1;R_1) \vdash_{KB} \cdots \vdash_{KB} (E_n;R_n) \vdash_{KB} \cdots$, on définit sa limite $(E^*;R^*)$ et sa limite inductive $(E^\infty;R^\infty)$ comme suit:

$$E* = \bigcup_i E_i, \quad R* = \bigcup_i R_i, \quad E^\infty = \bigcup_i \bigcap_{j \geq i} E_i, \quad R^\infty = \bigcup_i \bigcap_{j \geq i} R_i$$

Les équations de E^∞ et les règles de R^∞ sont dites *persistantes*. On dit qu'une séquence de complétion est *réussie* si l'ensemble des équations persistantes est vide et l'ensemble des règles persistantes est canonique. Lorsque le succès est obtenu après un nombre fini d'étapes, le système de règles résultant R^∞ définit une procédure de décision pour la théorie engendrée par les équations de E_0. Mais la complétion peut diverger en produisant un ensemble fini ou infini de règles persistantes. Cela est le cas à partir de la seule équation $f(g(f(x))) = g(f(x))$. Orientée de la seule manière

Paire critique:

$$\frac{(E;R)}{(E \cup \{e\};R)}$$

`if` e is a critical pair of R

Orientation:

$$\frac{(E \cup \{p = q\};R)}{(E;R \cup \{p \rightarrow q\})}$$

`if` $p \succ q$

Elimination:

$$\frac{(E \cup \{s = s\};R)}{(E;R)}$$

Simplification:

$$\frac{(E \cup \{e\};R \cup \{l \rightarrow r\})}{(E \cup \{e'\};R \cup \{l \rightarrow r'\})}$$

`if` $e \longrightarrow e'$ and/or $r \longrightarrow_R r'$

Métamorphose:

$$\frac{(E;R \cup \{l \rightarrow r\})}{(E \cup \{l' = r\};R)}$$

`if` $l \longrightarrow_{p \rightarrow q} l'$ and $(l \rightarrow r) \succ_{rgle} (p \rightarrow q)$

Figure 7.1: Ensemble KB de règles de complétion

possible en la règle $f(g(f(x))) \to g(f(x))$, cette règle se superpose à elle-même, produisant la paire critique $g(f(g(f(x)))) = f(g(g(f(x))))$, qui se simplifie en l'équation $f(g(g(f(x)))) = g(g(f(x)))$, orientée en la règle $f(g(g(f(x)))) \to g(g(f(x)))$. Le processus continue de la même manière, en engendrant l'ensemble infini de règles $\{f(g^i(f(x))) \to g^i(f(x)) \mid i \geq 1\}$.

La figure 7.1 décrit la complétion d'un fragment de la théorie des groupes. Partant des trois axiomes

$$\boxed{\begin{array}{rcl} x \cdot 1 & \simeq & x \\ 1 \cdot x & \simeq & x \\ x^- \cdot (x \cdot y) & \simeq & y \end{array}}$$

où 1 est une constante, "$^-$" est un symbole unaire postfixé et "." est un symbole binaire infixe, la complétion engendre le système de règles canonique

$$\boxed{\begin{array}{rclcrcl} 1 \cdot x & \to & x & \quad & x \cdot 1 & \to & x \\ x^- \cdot x & \to & 1 & & x \cdot x^- & \to & 1 \\ 1^- & \to & 1 & & (x^-)^- & \to & x \\ x^- \cdot (x \cdot y) & \to & y & & x \cdot (x^- \cdot y) & \to & y \end{array}}$$

en utilisant la taille des termes pour ordre de réduction. Dans cette figure, le premier champs, i, indique le nombre d'étapes de complétion. Le second champs indique l'ensemble des règles obtenu à l'étape i. À l'étape 3, par exemple, R_3 est composé de toutes les règles de R_2 et de la règle orientée à l'étape 4 et située sous R_2. Comme l'ensemble des règles n'est pas changé par l'étape 4, les étapes 3 et 4 sont montrées entre les même lignes horinzontales. Le troisième champs montre l'ensemble des équations de E_i sur la même ligne que le nombre d'étapes i. Le dernier champs indique le nom de la règle d'inférence utilisée à l'étape i.

Même lorsqu'un système canonique existe pour une théorie équationelle donnée, il se peut très bien que la complétion ne le trouve pas, au cas où elle engendre des équations intermédiaires dont l'orientation n'est pas possible avec l'ordre $\succ$. Soit par exemple $E_0 = \{f(c) = c, b = d, c = d, f(d) = a\}$, que nous allons compléter avec l'ordre rpo engendré par la précédence $f > d > c > a$ et $d > b > a$ (mais b et c incomparables). Parmi toutes les complétions possibles figurent une complétion réussie:

$$\begin{array}{rl} (E_0; \emptyset) & \vdash^+_{KB} (\{c = d, f(d) = a\}; \{d \to b, f(c) \to c\}) \\ & \vdash^+_{KB} (\emptyset; \{f(a) \to a, b \to a, c \to a, d \to a\}) \end{array}$$

et une complétion qui échoue:

$$\begin{array}{rl} (E_0; \emptyset) & \vdash^+_{KB} (\{c = d, f(d) = a\}; \{d \to b, f(c) \to c\}) \\ & \vdash^+_{KB} (\{b = c\}; \{f(a) \to a, d \to b, f(c) \to c\}) \end{array}$$

7.2 Correction de l'algorithme de complétion

Il s'agit d'exprimer, et de montrer, que d'une part la complétion ne change pas la théorie équationelle engendrée par l'ensemble E_0 de règles, d'autre part que le

i	R_i	E_i	inference
0		$1 \cdot x \simeq x$	
		$x \cdot 1 \simeq x$	
		$x^- \cdot (x \cdot y) \simeq y$	
1	$x \cdot 1 \to x$	$1 \cdot x \simeq x$	orient
		$x^- \cdot (x \cdot y) \simeq y$	
2	R_1	$x^- \cdot (x \cdot y) \simeq y$	orient
	$1 \cdot x \to x$		
3	R_2		orient
4	$x^- \cdot (x \cdot y) \to y$	$x^- \cdot x \simeq 1$	deduce (1,3)
5	R_3		orient
6	$x^- \cdot x \to 1$	$1^- \simeq 1$	deduce (1,5)
7	R_5		orient
8	$1^- \to 1$	$(x^-)^- \cdot y \simeq x \cdot y$	deduce (3,3)
9	R_7		orient
10	$(x^-)^- \cdot y \to x \cdot y$	$1^- \cdot y \simeq 1 \cdot y$	deduce (7,9)
11	R_9		orient
	$1^- \cdot y \to 1 \cdot y$		
12	R_9		compose (11,2)
	$1^- \cdot y \to y$		
13	R_9	$1 \cdot y \simeq y$	collapse (12,7)
14		$y \simeq y$	simplify (2)
15			delete
16		$(x^-)^- \simeq x$	deduce (1,9)
17	R_9		orient
	$(x^-)^- \to x$		
18	R_7	$x \cdot y \simeq x \cdot y$	collapse (9,17)
19	$1^- \cdot y \to y$		delete
20	$(x^-)^- \to x$	$x \cdot (x^- \cdot y) \simeq y$	deduce (3,17)
21	R_{18}		orient
22	$x \cdot (x^- \cdot y) \to y$	$x \cdot x^- \simeq 1$	deduce (1,21)
23	R_{21}		orient
	$x \cdot x^- \to 1$		

Table 7.1: Completion d'un fragment de la théorie des groupes

$$
\begin{array}{ccc}
s \leftrightarrow_{E^*} s & \Rightarrow & s \\
s \rightarrow_{R^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftarrow_{R^*} t \\
 & \text{where} & \left\{ \begin{array}{c} s \rightarrow_{R^*} t \text{ by } l \rightarrow r \\ s \rightarrow_{R^*} v \text{ by } l' \rightarrow r' \\ l \rightarrow r \succ_{rgle} l' \rightarrow r' \end{array} \right. \\
s \leftrightarrow_{E^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftrightarrow_{E^*} t \\
s \leftrightarrow_{E^*} t & \Rightarrow & s \rightarrow_{R^*} t \\
s \rightarrow_{R^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftrightarrow_{E^*} t \\
 & \text{where} & \left\{ \begin{array}{c} s \rightarrow_{R^*} t \text{ by } l \rightarrow r \\ s \rightarrow_{R^*} v \text{ by } l' \rightarrow r' \\ l \rightarrow r \succ_{rgle} l' \rightarrow r' \end{array} \right. \\
s \leftarrow_{R^*} u \rightarrow_{R^*} t & \Rightarrow & s \leftrightarrow_{E^*} t \\
s \leftarrow_{R^*} u \rightarrow_{R^*} t & \Rightarrow & s \rightarrow^*_{R^*} v \leftarrow^*_{R^*} t
\end{array}
$$

Figure 7.2: Règles de réécriture de preuves

système de règles obtenu en cas de succés est canonique, et permet donc de décider le problème du mot dans E_0. La première partie de cette affirmation est simple à montrer, puisque chaque application d'une règle d'inférence conserve la théorie équationelle, c'est à dire:

Lemme 7.2 $=_{E_i \cup E_{R_i}} = =_{E_{i+1} \cup E_{R_{i+1}}}$.

La seconde partie de cette affirmation est plus complexe, en particulier à cause de la possibilité de divergence. La preuve consiste à montrer que les règles d'inférence ont pour effet de transformer des preuves arbitraires en preuves par réécriture.

On appellera *preuve* une séquence d'étapes équationelles avec E et de réécritures avec R, ces dernières étant orientées dans les deux sens. Une *preuve par réécriture* sera une preuve de la forme $s \longrightarrow^*_R v \longleftarrow^*_R t$. Un *pic* est une preuve de la forme $s \longleftarrow_R u \longrightarrow_R t$, et $s \longleftrightarrow_E t$ est une *étape équationelle*. Une preuve par réécriture est donc une preuve sans pic ni étape équationelle. Le but de la complétion est de transformer toute preuve en une preuve par réécriture, ce qui suppose donc l'élimination des pics et des étapes équationelles. Certains des pics peuvent être facilement éliminés, il s'agit des pics correspondants à des redex disjoints et des pics correspondant à des redex ancêtres. La réécriture de ces pics se fait conformément à la preuve du lemme des paires critiques, et est donc implicitement représentée

à la figure 4.4. Il s'agit en fait d'une propriété de l'algèbre des preuves manipulées qui ne dépend pas de l'ensemble des axiomes de départ. Les pics correspondant à des redex critiques vont au contraire en général nécessiter une réécriture mettant en jeu une équation obtenue par inférence: la paire critique. Il s'agit donc d'une propriété de l'algèbre des preuves de $E*$ et $R*$. L'ensemble des règles de réécriture de preuve est représenté à la figure 7.2.

Il nous reeste à montrer qu'une preuve ne peut être réécrite indéfiniment, et donc qu'elle atteint nécéssairement une forme normale qui sera une preuve par réécriture. Pour cela, nous allons tout d'abord montrer que les réécritures de preuves terminent.

Nous définissons donc la complexité d'une preuve comme un multiensemble de paires formées d'une multiensemble de termes et d'une règle:

$C(s) = \{\}$, où s dénote la preuve vide issue de s,
$C(s \longrightarrow_{l\to r\in R^*} u \longleftrightarrow^*_{E*\cup R*}) = \{< \{s\}, l \to r >\} \cup C(u \longleftrightarrow^*_{E*\cup R*})$,
$C(s \longleftrightarrow_{E^*} u \longleftrightarrow^*_{E*\cup R*}) = \{< \{s,t\}, - >\} \cup C(u \longleftrightarrow^*_{E*\cup R*})$,
$C(s \longleftrightarrow^*_{E*\cup R*} u \longleftrightarrow^*_{E*\cup R*} t) = C(s \longleftrightarrow^*_{E*\cup R*} u) \cup C(u \longleftrightarrow^*_{E*\cup R*} t)$.

On associe à cette complexité l'ordre de réécriture $((\succ_{mul}, \succ_{rgle})_{lex})_{mul}$ et on vérifie que les règles de réécriture de preuves décroissent dans cet ordre. La preuve est technique, mais ne présente aucune difficulté.

Cela ne suffit pas, toutefois, à assurer l'obtention des formes normales souhaitées, car rien n'oblige une séquence de complétion à effectuer les inférences nécessaires à la réduction d'une preuve particulière. C'est le rôle de la définition suivante:

Définition 7.3 *Une séquence de complétion* $(E_0, \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots \vdash_{KB} (E_n; R_n) \vdash_{KB} \cdots$ *est équitable, si pour toute preuve* $s \longleftrightarrow^*_{E_i \cup R_i} t$ *qui est réductible par les règles de preuve, il existe une étape* j *telle que* $s \longleftrightarrow^*_{E_i\cup R_i} t \Rightarrow^+ s \longleftrightarrow^*_{E_j \cup R_j} t$.

On a maintenant le théorème de correction de la complétion:

Theorem 7.4 *Soit* $(E_0, \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots \vdash_{KB} (E_n; R_n) \vdash_{KB} \cdots$ *une séquence de complétion équitable. Alors* $s \longleftrightarrow_{E_i \cup R_i} t$ *si et seulement si* $s \longrightarrow^*_{R^\infty} u \longleftarrow^*_{R^\infty} t$.

Preuve
Par induction sur la relation de réécriture. Si la preuve $s \longleftrightarrow_{E_i \cup R_i} t$ n'est pas une preuve par réécriture, elle contient nécessairement un pic ou une étape équationelle réductible par $\Rightarrow$. Si elle est une preuve par réécriture dont certaines étapes utilisent des règles non persistantes, elle est à nouveau réductible par $\Rightarrow$. Par hypothèse d'équité, $s \longleftrightarrow_{E_i\cup R_i} t \Rightarrow^+ s \longleftrightarrow_{E_j \cup R_j} t$ et par hypothèse d'induction, $s \longrightarrow^*_{R^\infty} u \longleftarrow^*_{R^\infty} t$.

L'équité est bien sûr indécidable, mais il est facile d'obtenir des séquences équitables: il suffit de retarder au maximum le calcul des paires critiques, et de calculer ces dernières d'un seul coup par exemple, voir les exercices.

7.3 Exercices

7.3.1 CiME

CiME est un logiciel de complétion développé au sein de l'équipe DEMONS du LRI. Le premier exercice consiste à se familiariser avec CiME en faisant tourner les nombreux exemples disponibles, en particulier la théorie des groupes donnée par les trois équations suivantes: $\{x + (y + z) = (x + y) + z,\ x + 0 = x,\ x + x^- = 0\}$.

7.3.2 Confluence de la complétion

Montrer que la complétion est confluente, c'est à dire que, étant donné un ordre de réécriture sur les termes $\succ$, le système de règles retourné par la complétion, lorsque cette dernière n'échoue pas, est unique.

7.3.3 Complétion close

Soit E_0 un ensemble d'équations closes sur un alphabet, et $\succ$ un ordre de réduction total sur les termes clos.

1. Montrer que la complétion de E_0 termine.
2. Quelle est la complexité des règles de complétion dans ce cas ?
3. Donnez une procédure de complétion close polynomiale (il en existe une en $n\ log\ n$.

7.3.4 Équité

Montrer que tout contrôle pour lequel on ne calcule des paires critiques (et on les calcule alors toutes à la fois) que lorsqu'aucune autre inférence n'est possible, est équitable.

7.3.5 Divergence

On se donne un ensemble $E_0 = \{ggx \to gx,\ fgfx \to fgx\}$ de deux équations où f et g sont deux symboles unaires. On se contentera d'orienter ces équations et toutes celles engendrées par complétion avec l'ordre de plongement. On appelle simplification d'abord tout contrôle qui rejette en dernier les deux règles d'orientation et de calcul des paires critiques.

1. Montrer que l'ensemble des règles de complétion moins la règle de calcul des paires critiques termine pour toute donnée.
2. Montrer que la complétion de E_0 réussit en temps fini avec un contrôle simplification d'abord qui rejette le calcul des paires critiques lorsque plus rien d'autre n'est possible. Que vaut R_∞ ?
3. Trouver une séquence de complétion équitable qui ne termine pas.
4. Montrer que si la complétion diverge avec un contrôle simplification d'abord, alors le système de règles obtenu est nécessairement infini.
5. Montrer que si la complétion termine pour un certain contrôle équitable, alors il existe un contrôle simplification d'abord pour lequel elle termine également.

7.3.6 Réécriture ordonnée

Étant donnés un ensemble d'équations E tel que $r = l \in E$ si $l = r \in E$, et un ordre de réduction $\succ$ total sur les termes clos, la réécriture ordonnée sur les termes clos est d/'efinie comme $s \longrightarrow^p_{l\to r\in E} t$ si il existe une substitution close σ telle que $s|_p = l\sigma$, $t = s[r\sigma]$, et $l\sigma \succ r\sigma$. Cette définition permet donc de réécrire avec des instances closes ordonnées d'équations. On appellera paire critique de $g = d$ sur $l = r$ à la position $p \in \mathcal{FP}os(l)$, l'équation $r\sigma = l\sigma[d\sigma]_p$. On dit qu'une équation $s = t$ est confluente si $s\sigma = t\sigma$ pour toute susbtitution close σ.

1. Soit $E = \{x + y = y + x\}$, et deux constantes a, b. Trouver un ordre de réduction bien fondé $\succ$ tel que $a + b \longrightarrow_{\{x+y=y+x\}} b + a$.

2. Montrer que la relation de réécriture conditionelle est confluente ssi toutes ses paires critiques sont confluentes.

3. En déduire que la commutativité est confluente.

4. $E = \{x + y = y + x,\ (x + y) + z = x + (y + z),\ x + (y + z) = (x + y) + z\}$ est-il confluent?

5. Donner une procédure de complétion pour la réécriture ordonnée.

7.3.7 Réécriture conditionelle

On suppose que les règles sont de la forme $l \to r$ **if** $\overline{u} \downarrow \overline{v}$ où $\overline{u} \downarrow \overline{v}$ dénote l'expression $u_1 = v_1 \wedge \ldots \wedge u_n = v_n$. Le symbole $\downarrow$ sera interprété comme $\exists w$ tel que $\longrightarrow_R^* w \longleftarrow_R^*$. La réécriture est d'efinie comme $s \longrightarrow_{l\to r \text{ if } \overline{u}\downarrow\overline{v}}^p t$ si il existe σ telle que $s|_p = l\sigma$, $t = s[r\sigma]$, et $\overline{u}\sigma \downarrow \overline{v}\sigma$. Le système de réécriture R est dit réductif s'il existe un ordre de réduction $\succ$ tel que $l \succ r,\ l(\succ \cup \rhd)u_i,\ l(\succ \cup \rhd)v_i$ pour tout $i \in [1..n]$.

1. Montrer que $\succ \cup \rhd$ est bien fondé.

2. Montrer que la relation $s \longrightarrow_R t$ est décidable.

3. On appelle paire critique conditionelle de $g \to d$ **if** $\overline{s} \downarrow \overline{t}$ sur $l \to r$ **if** $\overline{u} \downarrow \overline{v}$ à la position $p \in \mathcal{FP}os(l)$, l'équation conditionelle $r\sigma = l\sigma[d\sigma]_p$ **if** $\overline{u}\sigma = \overline{v}\sigma \wedge \overline{s}\sigma = \overline{t}\sigma$. On dit qu'une équation conditionelle $s = t$ **if** $\overline{u} = \overline{v}$ est confluente si $\overline{s}\sigma = \overline{t}\sigma$ pour toute susbtitution σ telle que $\overline{u}\sigma = \overline{v}\sigma$.

   Montrer que la relation de réécriture conditionelle est confluente ssi toutes ses paires critiques conditionelles sont confluentes.

4. Donner une procédure de complétion conditionelle.

Chapter 8

Le treillis des termes

En fait, les termes avec variables servent souvent à représenter l'ensemble de toutes leurs *instances.* Par exemple, si les entiers naturels sont représentés en unaire à l'aide des symboles 0 et s (successeur), le terme $s(x)$ représente l'ensemble infini $\{s(0), s(s(0)), \ldots\}$. Certaines opérations sur les termes sont alors fondamentales (voir les chapitres suivants pour certaines applications). Par exemple, l'intersection des ensembles d'instances de deux termes est obtenue grâce à l'*unification.* Nous montrons l'opération d'unification (paragraphe 8.1) qui correspond d'une part à l'intersection des instances comme nous l'avons dit, mais aussi à la borne supérieure de deux termes (modulo similarité) pour le préordre de généralité (ou d'inclusion des instances). Dans le paragraphe 8.2, nous montrons l'opération inverse dans le cas des arbres finis: comment calculer la borne inférieure de deux termes dans ce treillis.

8.1 Unification des termes finis ou infinis

8.1.1 Problèmes d'unification

Définition 8.1 *Un* problème d'unification *est ou bien la formule $\perp$ ou bien une formule*

$$t_1 = u_1 \wedge \ldots \wedge t_n = u_n$$

où $t_1, \ldots, t_n, u_1, \ldots, u_n$ sont des termes. Lorsque $n = 0$, le problème obtenu est par convention la formule $\top$.

Le signe "=" est symétrique, si bien que nous ne faisons aucune différence entre $s = t$ et $t = s$.

Une substitution (resp. une $RT(F, X)$-assignation) σ est une *solution* de $t_1 = u_1 \wedge \ldots, \wedge t_n = u_n$ lorsque, pour tout indice i, $t_i\sigma \equiv u_i\sigma$. Plus généralement,

Définition 8.2 *Si $\mathcal{A}$ est une F-algèbre et σ un morphisme de $T(F, X)$ dans $\mathcal{A}$, σ est* solution *encore appellée* unificateur *du problème d'unification $t_1 = u_1 \wedge \ldots \wedge t_n = u_n$ si $\sigma(t_i) =_{\mathcal{A}} \sigma(u_i)$ pour tout i.*

Tout morphisme σ est une $\mathcal{A}$-solution de $\top$ et aucun morphisme n'est solution de $\perp$.

Exemple 8.3 *Le problème d'unification* $f(x, g(x)) = f(h(y, z), y)$ *n'a pas de solution dans* $T(F, X)$*. Il a par contre plusieurs solutions dans* $RT(F, X)$*, comme par exemple* $x = t_1, y = t_2, z = g(z')$ *où*

- t_1 *est le terme dont les positions sont* $1^* + (1 \cdot 1)^* \cdot (2 + 2 \cdot 1)$ *et tel que* $t_1(1^{2n}) = h, t_1(1^{2n+1}) = g, t_1(1^{2n}2) = g, t_1(1^{2n}21) = z'$
- t_2 *est le terme dont les positions sont* $1^* + (11)^*1(2 + 21)$ *et tel que* $t_2(1^{2n}) = g, t_2(1^{2n+1}) = h, t_2(1^{2n+1}2) = g, t_2(1^{2n+1}21) = z'$.

Si P est un problème d'unification, on note $Var(P)$ l'ensemble de ses variables (libres).

Définition 8.4 *Étant donnée une* F*-algèbre* $\mathcal{A}$*, deux problèmes d'unification sont* $\mathcal{A}$-équivalents *s'ils ont mêmes ensembles de solutions dans* $\mathcal{A}$*. (Parfois* $\mathcal{A}$ *est omise quand elle peut être aisément déduite du contexte).*

8.1.2 Formes résolues dans les termes finis

Les *formes résolues* sont des problèmes d'unification plus simples pour lesquels le calcul des solutions (qui forment toujours un ensemble non-vide) est immédiat; plusieurs formes résolues sont interessantes dans le cas des termes finis: nous verrons les arbre-formes résolues et les DAG-formes résolues. Dans tous les cas, nous montrerons dans le paragraphe 8.1.6 que tout problème d'unification est équivalent à une forme résolue, et nous montrerons dans les paragraphes suivants comment calculer ces formes résolues.

Définition 8.5 *Dans le cas des termes finis, on dira qu'un problème d'unification est une* arbre-forme résolue *(ou forme résolue tout court) si c'est* $\perp$ *ou s'il peut s'écrire:*

$$x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

où $x_1, \ldots, x_n$ *sont des variables distinctes qui n'ont pas d'autre occurrence dans le problème.*

Lemme 8.6 *Les solutions d'une arbre-forme résolue* $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ *sont les instances d'une substitution* $\sigma = \{x_1 \to t_1 \ldots, x_n \to t_n\}$ *appellée* unificateur principal *de la forme résolue.*

Preuve
On vérifie aisément que σ est une solution, car $x_i\sigma = t_i$ et $t_i\sigma = t_i$ d'après la condition sur les variables. De plus, si θ est une solution arbitraire, $x_i\theta = t_i\theta = (x_i\sigma)\theta$, d'où $\theta = \sigma\theta$, ce qui montre que σ est principale.

Définition 8.7 *Si* $s = t$ *est équivalent dans* $T(F, X)$ *à la forme résolue* $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$*, alors* $\{x_1 \to t_1, \ldots, x_n \to t_n\}$ *est appelé* plus général unificateur de s et t.

Cette définition est justifiée dans la mesure où un plus général unificateur de deux termes est un élément minimal pour le préordre $\preceq$ sur les substitutions. En particulier un plus général unificateur, lorsqu'il existe, est unique modulo $\doteq$.

Définition 8.8 *Dans le cas des termes finis, on dira qu'un problème d'unification est une DAG-forme résolue si c'est $\top, \bot$ ou s'il peut s'écrire:*

$$x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

où $x_1, \ldots, x_n$ sont des variables distinctes telles que, pour tout $i \leq j$, $x_i \notin Var(t_j)$.

Ces formes résolues ne font que représenter un plus général unificateur de façon compacte:

Lemme 8.9 *Si $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ est une DAG-forme résolue équivalente à $s = t$, alors $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$, où $\sigma_i = \{x_i \to t_i$ pour tout $i \in [1..n]$, est un plus général unificateur de s et t.*

Preuve
On montre d'abord que σ est un unificateur. On a $x_i\sigma = x_i\sigma_i \ldots \sigma_n = t_i\sigma_{i+1} \ldots \sigma_n = t_i\sigma$ grâce à la condition sur les variables de la forme résolue.

On montre ensuite que σ est principale par récurrence sur n. Le cas de base $n = 0$ est trivial. Dans le cas général, $\tau = \sigma_1\sigma_2 \ldots \sigma_{n_1}$ est unificateur principal de la forme résolue $P' = x_1 = t_1 \wedge \ldots \wedge x_{n-1} = t_{n-1}$. Mais tout unifiicateur γ de $s = t$ est en particulier unificateur de P', et donc $\gamma = \tau\theta$. Comme γ unifie aussi l'équation $x_n = t_n$, on a $x_n\tau\theta = t_n\tau\theta$, d'ou $x_n\theta = t_n\theta$, et donc θ est une instance de la substitution $\{x_n \mapsto t_n\}$ qui est unificateur prioncipal de cette équation par le lemme 8.6. Et donc γ est une instance de σ, ce qui termine la preuve.

8.1.3 Formes résolues dans les termes rationnels

Dans le cas des arbres infinis, il nous faut d'abord établir quelques propriétés de base des systèmes d'équations.

Définition 8.10 *Un* cycle de variables *est un problème d'unification de la forme*

$$x_1 = x_2 \wedge \ldots \wedge x_{n-1} = x_n \wedge x_n = x_1$$

où $x_1, \ldots, x_n$ sont des variables.

Theorem 8.11 *Un problème d'unification $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ tel que $x_1, \ldots, x_n$ sont des variables distinctes et $Var(t_1, \ldots, t_n) = \{x_1, \ldots, x_n\}$ et ne contenant pas de cycle de variable a une solution unique dans $RT(F, X)$.*

Preuve
Appliquons tout d'abord la règle d'élimination de variables de la figure 8.1 au problème P:

$$x = y \wedge P \to x = y \wedge P\{x \to y\}$$

Lorsque x et y ont tous deux au moins une occurrence dans P.

L'application de cette règle termine (Pour le voir, il suffit de considérer l'interprétation qui associe à un problème d'unification le nombre d'équations $x = y$ où x et y sont des variables apparaissant toutes deux dans le reste du problème. Comme P ne contient pas de cycle de variables $x \not\equiv y$ et cette propriété est conservé par élimination des variables).

Considérons maintenant le problème (équivalent) P' obtenu après une normalisation par la règle ci-dessus. P' peut se décomposer en

$$P' \equiv P_0 \wedge P_1$$

où P_0 ne contient que des équations entre variables et P_1 ne contient aucune équation entre variables. Notons que $Var(P_0)$ contient $n - m$ équations, alors que P_1 contient m variables et m équations avec $m \geq 1$ (Ce sont des invariants de la règle d'élimination de variables). Montrons tout d'abord par récurrence sur m que $P_1 \equiv x_1 = u_1 \wedge \ldots \wedge x_m = u_m$ possède une unique solution dans $RT(F, X)$, dès que $u_1, \ldots, u_m$ sont des termes rationnels non réduits à une variable.

Si $m = 1$, soient Q l'ensemble des positions de x_1 dans u_1 (peut-être a-t-on $Q = \emptyset$). D'après la proposition 2.6, Q est défini par une expression régulière. L'ensemble des positions P d'un terme t tel que $\{x \to t\}$ est solution de P_1 doit vérifier l'équation

$$P = Q \cdot P + Pos(u_1)$$

dans $\mathbf{N}^*$. Une telle équation a pour seule solution dans $\mathbf{N}^*$

$$P = Q^* \cdot Pos(u_1)$$

car $\Lambda \notin Q$ (cf. le cours de langages formels). De plus (par récurrence sur r), $t(p_{i_1} \cdots p_{i_r} q) = u(q)$ si $q \in Pos(u_1)$ et $p_{i_1}, \ldots, p_{i_r} \in Q$. Ceci définit t de manière unique. Inversement, $\{x \to t\}$ est bien une solution de P.

Si $m > 1$, considérant $x_1, \ldots, x_{m-1}$ comme des constantes, d'après ce que nous venons de voir, $x_m = u_m$ a une unique solution $\{x_m \to v_m(x_1, \ldots, x_{m-1}\}$ où $v_m \in RT(F, \{x_1, \ldots, x_{m-1}\})$. En remplaçant x_m par v_m dans le reste du problème, on obtient un système qui, par hypothèse de récurrence a une unique solution dans $RT(F)$. Reportant enfin cette solution dans v_m on obtient le résultat souhaité.

Il reste maintenant à reporter la solution de P_1 dans P_0. Comme, par hypothèse, chaque équation de P_0 contient exactement une variable qui apparait dans P_1 on obtient le résultat souhaité $\square$

Définition 8.12 *Une RT-forme résolue est $\perp$ ou $\top$ ou bien un problème d'unification*

$$x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

ne contenant pas de cycle de variables et où $x_1, \ldots x_n$ sont des variables distinctes.

Pour un tel problème, appelons *paramètres* les variables qui apparaissent dans $t_1, \ldots, t_n$ et qui sont distinctes de $x_1, \ldots, x_n$.

Lemme 8.13 *Soit $P \equiv x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ une RT-forme résolue de paramètres $y_1, \ldots, y_m$. Pour tous termes rationnels $t_1, \ldots, t_m$, il existe une unique solution σ de P telle que $y_i\sigma \equiv t_i$.*

Preuve
C'est une conséquence du théorème 8.11. □

8.1.4 Règles de transformation

Nous donnons maintenant des règles de transformation qui permettent de réécrire un problème d'unification en un problème d'unification équivalent jusqu'à obtenir une forme résolue.

Les règles de la figue 8.1 sont ici volontairement redondantes (nous verrons pourquoi dans le paragraphe 8.1.6). Ce sont des (schémas de) règles de réécriture dans l'algèbre des problèmes d'unification. En particulier elles peuvent s'appliquer à n'importe quel sous-problème, remplaçant le membre gauche par le membre droit dès que la condition d'application est vérifiée. Le symbole $\wedge$ est ici supposé associatif et commutatif.

Proposition 8.14 *Chacune des règles de la figure 8.1 transforme un problème d'unfication dans les termes finis en un problème équivalent.*

Chacune des règles de la figure 8.1 à l'exception du test d'occurrence transforme un problème d'unification dans les termes rationnels en un problème équivalent.

Preuve
Les règles d'élimantion ou de remplacement de variables, d'élimination des équations triviales et de fusion sont des conséquences des axiomes de l'égalité, ou, si l'on préfère, ce sont des conséquence du fait que l'égalité des termes est une congruence.

Les règles de décomposition et d'incompatibilité sont une conséquence des axiomes des termes.

Enfin, le test d'occurrence exprime qu'un terme fini ne peut être égal à un de ces sous-termes stricts. En effet, si c'était le cas, par propriété de congruence de l'égalité, on obtiendrait par récurrence un arbre infini. □

8.1.5 Terminaison

Les règles de la figure 8.1 ne précisent pas de stratégies de d'application: à partir de ces règles on peut fabriquer de nombreux algorithmes d'unification en spécifiant un peu plus dans quel ordre et à quelle sous-problème appliquer les règles. Mais nous allons montrer que, quelle que soit la stratégie d'utilisation des règles, l'algorithme obtenu termine toujours. Ceci permet de factoriser les preuves de terminaison de différents algorithmes d'unification; par exemple la terminaison de l'unification dans les termes finis comme celle de l'unification dans les termes infinis sont des cas particuliers du résultat suivant:

Équations triviales	$s = s$ $\longrightarrow$ $\top$
Decompose	$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ $\longrightarrow$ $s_1 = t_1 \wedge \ldots \wedge s_n = t_n$
Incompatibilité	$f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$ $\longrightarrow$ $\perp$ Si $f \neq g$
Remplacement de variable	$x = y \wedge P$ $\longrightarrow$ $x = y \wedge P\{x \to y\}$ Si $x, y \in Var(P)$
Elimination de variable	$x = s \wedge P$ $\longrightarrow$ $x = s \wedge P\{x \to s\}$ Si $x \in Var(P)$ et $x \notin Var(s)$ et $s \in X \to s \in Var(P)$
Fusion	$x = s \wedge x = t$ $\longrightarrow$ $x = s \wedge s = t$ Si $x \in X$, $s \notin X$ et $\|s\| \leq \|t\|$
Test d'occurrence	$x_1 = t_1[x_2]_{p_1} \wedge \ldots \wedge x_n = t_n[x_1]_{p_n}$ $\longrightarrow$ $\perp$ Si $p_1 \cdot \ldots \cdot p_n \neq \Lambda$

Figure 8.1: Règles d'unification

Theorem 8.15 *Le système de règles de la figure 8.1 définit un ordre strict bien fondé sur les problèmes d'unification.*

Preuve
On considère les fonctions d'interprétation suivantes des problèmes d'unification:

- Une variable x est dite *résolue* dans un problème P si $P \equiv x = s \wedge Q$ et $x \notin Var(s)$ et $x \notin Var(Q)$.

  $\phi_1(P)$ est le nombre de variables non résolues de P.

- Si $P \equiv s_1 = t_1 \wedge \ldots \wedge s_n = t_n$, alors $\phi_2(P)$ est le multi-ensemble $\{m_1, \ldots, m_n\}$ où $m_i = \max(|s_i|, |t_i|)$.

- $\phi_3(P)$ est le nombre d'équations de P dont l'un des membres au moins est une variable.

$\phi(P)$ est le triplet $(\phi_1(P), \phi_2(P), \phi_3(P))$. L'ensemble de ces triplets est muni de la composée lexicographique de trois ordres:

1. L'ordre habituel sur les entiers naturels

2. L'extension multi-ensemble de l'ordre sur les entiers naturels

3. L'ordre habituel sur les entiers naturels

Il est ainsi muni d'un ordre bien fondé puisque construit à partir d'extensions multi-ensemble et lexicographiques d'ordres bien fondés.

Il suffit maintenant de montrer que si $P \longrightarrow Q$ alors $\phi(P) > \phi(Q)$. Le sens de variations des trois fonctions d'interprétation est résumé dans le tableau ci-dessous:

	ϕ_1	ϕ_2	ϕ_3
Équations triviales	$\preceq$	$\prec$	
Decompose	$\preceq$	$\prec$	
Incompatibilité	$\preceq$	$\prec$	
Remplacement de variables	$\prec$		
Élimination de variable	$\prec$		
Fusion	$\preceq$	$=$	$\prec$
Test d'occurrence	$\preceq$	$\prec$	

Il n'est pas difficile de vérifier qu'aucune règle ne crée de nouvelle variable non-résolue. Les conditions d'application des deux règles d'élimination et de remplacement de variables garantissent par ailleurs qu'une variable non résolue (x dans la formulation des règles doit apparaitre dans P) devient résolue après leur application.

Notons également que la règle de décomposition fait bien décroître ϕ_2 puisqu'elle remplace $m = \max(|f(s_1, \ldots, s_n)|, |f(t_1, \ldots, t_n)|)$ par n entiers strictement inférieurs.

La règle de fusion conserve bien ϕ_2 à cause de la condition $|s| \leq |t|$: $\max(|x|, |t|) = \max(|s|, |t|)$. Enfin, ϕ_3 est bien décroissante par fusion à cause de la condition $s \notin x$ qui, avec $|s| \leq |t|$, garantit que $s = t$ est une équation dont aucun des membres n'est une variable.

Bien entendu certaines fonctions d'interprétation (ϕ_2 ou ϕ_3) peuvent croître par application de certaines règles, mais la composée lexicographique des trois interprétations est toujours décroissante comme le montre le tableau ci-dessus. □

8.1.6 Complétude

Nous extrayons successivement 3 ensembles de règles de la figure 8.1 et nous montrons que dans chaque cas, l'ensemble de règles permet d'obtenir les formes résolues souhaitées, ce qui fournit trois algorithmes d'unification.

Proposition 8.16 *Si P est un problème d'unification auquel aucune règle de la figure 8.1 ne s'applique, alors P est une arbre-forme résolue. (NB: les règles de fusion et de remplacement de variables sont inutiles pour obtenir ce résultat).*

Preuve
Comme les règles d'incompatibilité et de décomposition ne s'appliquent pas, toutes les équations ont une variable x dans un de leurs membres. Comme le test d'occurrence et la règle des équations triviales ne s'applique pas, si $x = t$ est une des équations de P, x n'est pas une variable de t. Comme la règle d'élimination de variable ne s'applique pas, x ou t doit de plus être une variable résolue, ce qui conduit au résultat souhaité. □

Définition 8.17 *Deux termes finis s et t sont dits* unifiables *si $s = t$ possède au moins une solution dans les termes finis.*

Corollaire 8.18 *Deux termes s et t sont unifiables si et seulement si ils ont un plus général unificateur.*

Corollaire 8.19 *Si l'on adjoint à $T(F, X)$ un élément $\perp$ supérieur pour $\lessdot$ à tous les termes, alors $(T(F, X)/ \doteq, \lessdot)$ possède une structure de sup-demi treillis.*

Preuve
On suppose que s et t n'ont pas de variable en commun, ce qui est toujous possible par renommage. La borne inférieure de s et t est ou bien $\perp$ si s et t ne sont pas unifiables, ou bien $s\sigma$ si σ est un plus général unificateur de s et t. □

Une deuxième famille d'algorithmes d'unification (plus efficace, cf. exercices) peut être obtenue par les mêmes règles:

Proposition 8.20 *Si P est un problème d'unification auquel la seule règle (peut-être) applicable parmi les règles de la figure 8.1 est l'élimination de variables, alors P est une DAG-forme résolue.*

Preuve
De même que ci-dessus, si la seule règle applicable à P est l'élimination de variable, alors toutes les équations de P sont de la forme $x = t$ où $x \notin Var(t)$. Considérons

alors la relation *d'occurrence* $\geq_{occ}$ sur les variables de P définie comme la plus petite relation de préordre telle que, si $x = t[y]_p$ est dans P, alors $x \geq_{occ} y$. Comme le test d'occurrence ne s'applique pas, la relation d'équivalence associée à ce préordre est la plus petite relation d'équivalence $=_S$ qui contient $x =_S y$ si $x = y$ est dans P. Mais, si $x =_S y$, x ou y est une variable résolue puisque la règle de remplacement de variables ne s'applique pas. On peut donc construire un ordre $\geq$ sur les variables de P tel que: les variables résolues sont minimales et $x >_{occ} y \Rightarrow x > y$. En complétant $\geq$ en un ordre total, on obtient le résultat souhaité. □

Enfin, les mêmes règles fournissent à nouveau des algorithmes d'unification dans les arbres rationnels (ou infinis).

Proposition 8.21 *Si P est un problème d'unification auquel la seule règle de la figure 8.1 peut-être applicable est le test d'occurrence, alors P est une RT-forme résolue.*

Preuve
Pour les mêmes raisons que précédemment, si seul le test d'occurrence peut (peut-être) s'appliquer à P, ce problème ne contient que des équations $x = t$ où x est une variable. De plus, comme l'élimination de variables ne s'applique pas, P ne contient pas de cycle de variable. □

Exemple 8.22

$$\begin{array}{lrl} x = f(f(x)) \wedge x = f(x) & \rightarrow_{Fusion} & x = f(x) \wedge f(f(x)) = f(x) \\ & \rightarrow_{Décomposition} & x = f(x) \wedge x = f(x) \\ & \rightarrow_{Fusion} & x = f(x) \wedge f(x) = f(x) \\ & \rightarrow_{Équations\ triviales} & x = f(x) \end{array}$$

ce dernier problème étant en forme résolue pour les arbres rationnels.

8.2 Généralisation

Soit Φ est une bijection de $T(F,X) \times T(F,X)$ dans X (resp. de $RT(F,X) \times RT(F,X)$ dans X)

L'application AU est définie récursivement de $T(F,X) \times T(F,X)$ dans $T(F,X)$ par

$$\begin{cases} \mathrm{AU}(u,v) = f(\mathrm{AU}(u_1,v_1),\ldots,\mathrm{AU}(u_n,v_n) & \text{Si } u \equiv f(u_1,\ldots,u_n) \text{ et } v \equiv f(v_1,\ldots,v_n) \\ \mathrm{AU}(u,v) = \Phi(u,v) & \text{sinon} \end{cases}$$

Theorem 8.23 *$AU(u,v)$ termine et son résultat est une borne inférieure de u,v pour le préordre de généralité.*

Preuve
La terminaison résulte du fait que les appels récursifs sont effectués sur des termes plus petits (en d'autres termes AU est défini par récurrence structurelle).

$\mathrm{AU}(u,v) \preceq u$ et $\mathrm{AU}(u,v) \preceq v$. En effet, par récurrence sur les tailles de u et v: si u et v sont deux constantes ou variables identiques, alors $\mathrm{AU}(u,v) \equiv u \equiv v$. Si ce sont deux constantes ou variables distinctes alors $\mathrm{AU}(u,v) \in X$. Si maintenant u et v sont deux termes quelconques, ou bien leurs symboles de tête sont distincts et $\mathrm{AU}(u,v) \in X$ est bien plus général que u et v. Ou bien $u \equiv f(u_1,\ldots,u_n)$ et $v \equiv f(v_1,\ldots,v_n)$. Dans ce cas, par hypothèse de récurrence, pour tout i, $t_i \equiv \mathrm{AU}(u_i,v_i)$ est plus général que u_i et que v_i. Soit σ_i et θ_i telles que $u_i \equiv t_i\sigma_i$ et $v_i \equiv t_i\theta_i$ et $Dom(\sigma_i) = Var(t_i) = Dom(\theta_i)$. Si $x \in Var(t_i) \cap Var(t_j)$, alors $x\sigma_i \equiv \Phi^{-1}(x) \equiv x\sigma_j$. La substitution σ de domaine $\bigcup_{i=1}^{n} Dom(\sigma_i)$ et telle que $x\sigma \equiv x\sigma_i$ si $x \in Dom(\sigma_i)$ est donc définie indépendemment de l'indice i tel que $x \in Dom(\sigma_i)$. De même on définit θ par $x\theta \equiv x\theta_i$ si $x \in Dom(\theta_i)$. On obtient alors $(\mathrm{AU}(u,v))\sigma \equiv u$ et $(\mathrm{AU}(u,v))\theta \equiv v$.

Il reste à montrer que pour tout terme t tel que $t \preceq u$ et $t \preceq v$, on a $t \preceq \mathrm{AU}(u,v)$. Supposons donc $t\sigma_i 1 \equiv u$ et $t\sigma_2 \equiv v$. On construit θ par récurrence sur t: si t est une variable ou une constante, la construction est immédiate. Si $t \equiv f(t_1,\ldots,t_n)$ alors u et v doivent pouvoir s'écrire $f(u_1,\ldots,u_n)$ et $f(v_1,\ldots,v_n)$ respectivement et, pour tout i, $t_i\theta_i \equiv \mathrm{AU}(u_i,v_i)$. De plus si $x \in Var(t_i) \cap Var(t_j)$, $x\theta_i \equiv x\theta_j$ car $x\sigma_1 \equiv x\sigma_2$ et $t_i \preceq u_i$ et $t_i \preceq v_i$. On peut donc définir θ comme ci-dessus par $x\theta \equiv x\theta_i$ si $x \in Dom(\theta_i)$. □

Dans les termes rationnels, la définition est la même. Cependant, pour obtenir un algorithme qui termine, il faut utiliser une représentation finie des termes rationnels.

Supposons que X est décomposé en deux sous-ensembles disjoints infinis X_1 et X_2 et que l'on dispose d'une bijection Φ de $RT(F,X_1)^2$ dans X_1. On peut calculer un anti-unificateur de s et t en utilisant les règles de la figure 8.2.

Dans cette figure, les variables introduites par les règles sont supposées appartenir à X_2. De plus, les formules considérées contiennent des conjonctions d'équations ainsi qu'un *environnement* E qui est aussi une conjonction d'équations et noté entre accolades pour le distinguer du reste de la formule.

Quels que soient les termes rationnels s et t, en partant du problème $z = \mathrm{AU}(s,t)$, les règles de la figure 8.2 terminent (ceci est laissé en exercice). De plus, dans tous les problèmes obtenus par réduction de la formule de départ, les variables de X_2 apparaissent au plus une fois comme membre d'une équation dont l'autre membre est non variable. (La preuve de cet invariant est laissée en exercice). La formule irréductible obtenue par normalisation de $z = \mathrm{AU}(s,t)$ est de la forme

$$\exists z_1,\ldots,z_n.\ z = t_0 \wedge x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

où $x_i \in \{z_1,\ldots,z_n\}$ pour tout i, $Var(t_1,\ldots,t_n) \subseteq \{z,x_1,\ldots,x_n\}$ et $t_1,\ldots,t_n \in T(F,X)$. Il n'y a de plus pas de cycle de variable. La formule ainsi obtenue représente un unique terme rationnel qui est la borne inférieure pour le préordre de généralité de s et t. (Ceci est à nouveau laissé en exercice).

$$z = \mathrm{AU}(s,t) \wedge P \wedge \{E\} \quad \rightarrow \quad z = \Phi(s,t) \wedge P \wedge \{E\}$$

Si $s \in X$ ou $t \in X$ ou s et t n'ont pas même symbole de tête

$$\begin{array}{l} z = \mathrm{AU}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)) \wedge P \wedge \{E\} \\ \quad \rightarrow \quad \exists \vec{z}. \bigwedge_{i=1}^{n} z_i = \mathrm{AU}(s_i,t_i) \wedge z = f(z_1,\ldots,z_n) \wedge P \\ \qquad\qquad \wedge \{E \wedge z = \mathrm{AU}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)\} \end{array}$$

Si $\vec{z} \subset X_2$, $\vec{z} \cap Var(P) = \emptyset$ et aucune équation de la forme $z' = \mathrm{AU}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n))$ n'apparait dans E.

$$\begin{array}{l} z = \mathrm{AU}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)) \wedge P \wedge \{E\} \\ \quad \rightarrow \quad z = z' \wedge P \wedge \{E\} \end{array}$$

Si $z' = \mathrm{AU}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n))$ apparait dans E

Figure 8.2: Règles d'anti-unification dans les termes rationnels

8.3 Exercices

1. Donner une DAG-forme résolue et une arbre-forme résolue pour les problèmes d'unification suivants:

   (a) $f(f(x_1,x_2), f(x_3,x_4)) = f(f(f(x_2,x_2), f(x_3,x_3)), f(f(x_4,x_4), f(a,a)))$

   (b) $f(x) = y \wedge f(f(y)) = f(z) \wedge x = f(z)$

   (c) $f(x, f(x,x)) = f(f(y,y), f(f(f(z,z), f(a,x')), f(f(a,a), f(a,a))))$

2. Résoudre dans $RT(F,X)$ les problèmes d'unification suivants:

   (a) $x = f(f(x)) \wedge x = f(f(f(x)))$

   (b) $x = f(f(y)) \wedge x = f(z) \wedge z = f(f(x)) \wedge y = f(f(z))$

3. Montrer par des exemples que toutes les conditions d'application des règles de la figure 8.1 sont nécessaires pour assurer la terminaison.

4. Si $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ est une arbre-forme résolue, on appelle *paramètres* les variables autres que $x_1,\ldots,x_n$. Montrer que deux formes résolues quelconques d'un même problème d'unification ont même nombre de variables résolues et même nombre de paramètres. On parlera donc dans la suite du *nombre de paramètres d'un système d'équations*

5. Soient P et P' deux problèmes d'unification. Montrer que, si toute solution de P est une solution de P' et que, de plus, P et P' ont même nombre de paramètres, alors P et P' ont même ensemble de solutions.

6. Montrer que, si E possède N paramètres et $E_1, \ldots, E_k$ sont des problèmes d'unification ayant chacun strictement moins de N paramètres, alors E possède au moins une solution qui n'est solution d'aucun des problèmes E_i.

7. Montrer que deux arbre-forme résolues équivalentes ont même taille (i.e. même nombre de symboles)

8. Donner un exemple de DAG-forme résolue dont la taille est $O(n)$ alors que l'arbre-forme résolue correspondante est de taille $O(2^n)$.

9. Donner un anti-unificateur de $f(x, f(x, f(a, b)))$ et $f(a, f(a, f(a, a)))$.

10. Montrer que tout ensemble (éventuellement infini) de termes possède une borne inférieure et une borne supérieure pour le préordre de généralité.

11. Montrer que l'anti-unification est associative, i.e. $\mathrm{AU}(s, \mathrm{AU}(t, u)) = \mathrm{AU}(\mathrm{AU}(s, t), u)$.

12. Montrer (dans le cas des termes finis comme dans le cas des termes infinis) que si un problème d'unification E est équivalent à une disjonction $E_1 \vee \ldots \vee E_n$ de problèmes d'unification, alors E est équivalent à l'un des E_i.

13. Montrer que les règles de la figure 8.2 terminent. Montrer qu'elles conduisent bien à transformer $z = \mathrm{AU}(s, t)$ en une forme résolue pour les arbres rationnels (avec quelques variables quantifiées existentiellement). Montrer enfin que l'unique arbre rationnel ainsi défini est bien une borne inférieure de s et de t pour le préordre de généralité.

14. Montrer que, pour les termes finis, l'anti-unificateur de s et t a toujours une taille inférieure à celles de s et de t. En est-il toujours de même pour les termes rationnels ? (si leur taille est mesurée comme leur nombre de sous-termes distincts, ou, si l'on veut, comme le nombre de nœuds du graphe fini qui les représente).

Chapter 9

Logique du premier ordre. Forme clausale

9.1 Syntaxe

Un langage du premier ordre est constitué à partir d'un vocabulaire comprenant

- un ensemble infini de *symboles de variables* $\mathcal{X} = \{x, y, z, \ldots\}$,

- un ensemble de *symboles de fonctions* $\mathcal{F} = \{f, g, h, \ldots, a, b, c, \ldots\}$ munis d'*arités*.

- un ensemble de *symboles de prédicats* $\mathcal{P} = \{P, Q, R, \ldots, p, q, r, \ldots\}$ munis d'arités de telle sorte que pour chaque entier n, $\mathcal{P}_n$ est l'ensemble éventuellement vide des symboles de prédicats d'arité n. Les symboles $\{p, q, r, \ldots\}$ de $\mathcal{R}_0$ sont aussi appellés *symboles* (ou *variables*) *propositionelles.*

des *connecteurs logiques* usuels : $\wedge, \vee, \neg, \Rightarrow$, des constantes $0, 1$, et des quantificateurs $\exists$, $\forall$. On utilisera également des symboles auxiliaires de désambiguation comme $(,), [,], \{$ et $\}$.

Le langage logique est formé de quatre entités syntaxiques, les symboles de variables (ou *variables*), les *termes*, les *atomes* et les *formules*, construites successivement comme suit :

1. l'ensemble $T(\mathcal{F}, \mathcal{X})$ des termes finis est défini au chapitre 2

2. L'ensemble des atomes est l'ensemble des expressions de la forme $P(t_1, \ldots, t_n)$ où $t_1, \ldots, t_n$ sont des termes finis.

3. Le langage $\mathcal{F}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ des *formules du premier ordre* bati sur le vocabulaire $\mathcal{R}, \mathcal{F}, \mathcal{X}$ est le plus petit ensemble contenant les constantes logiques 0 et 1 et les atomes, et clos par les opérations qui

   - à la formule A associent sa négation $\neg A$,

   - aux formules A, B associent leur conjonction $A \wedge B$, leur disjonction $A \vee B$, et l'implication $A \Rightarrow B$,

   - à la formule A et à la variable x associent la formule *existenciellement quantifiée* $\exists x A$, et la formule *universellement quantifiée* $\forall x A$.

Le langage des *formules propositionnelles* est obtenu pour $\mathcal{X} = \emptyset, \mathcal{F} = \emptyset, \mathcal{P} = \mathcal{P}_0$.

Un *littéral* est un atome (auquel cas il est dit *positif*) ou la négation d'un atome (auquel cas il est dit *négatif*). Une formule sans variable est dite *close*. Une formule dont les quantificateurs apparaissent tous en tête est dite *prénexe*. Une formule *universelle* (resp. *existentielle*) est une formule prénexe close dont les seuls quantificateurs sont universels (resp. existentiels).

On appelle *clause* toute formule universelle dont les sous formules non quantifiées sont des littéraux ou des disjonctions de littéraux. Une clause a donc la forme générale

$$\forall x_1 \ldots \forall x_n A_1 \vee \ldots \vee A_p \vee \neg B_1 \vee \ldots \vee \neg B_q$$

où $A_1, \ldots, A_p, B_1, \ldots, B_q$ désignent des atomes.

Les quantificateurs *lient* les variables sur lesquelles ils portent. On définit donc les variables *libres* $\mathcal{V}ar()$ et les variables *liées* $\mathcal{BV}ar()$ d'une formule par récurrence sur la structure de la formule :

Variables $\mathcal{V}ar(x) = \{x\}$ si x est une variable
$\mathcal{BV}ar(x) = \emptyset$ si x est une variable

Termes $\mathcal{V}ar(f(t_1, \ldots, t_n)) = \mathcal{V}ar(t_1) \cup \ldots \cup \mathcal{V}ar(t_n)$
$\mathcal{BV}ar(f(t_1, \ldots, t_n)) = \emptyset$

Atomes $\mathcal{V}ar(P(t_1, \ldots, t_n)) = \mathcal{V}ar(t_1) \cup \ldots \cup \mathcal{V}ar(t_n)$
$\mathcal{BV}ar(P(t_1, \ldots, t_n)) = \emptyset$

Formules $\mathcal{V}ar(A \wedge B) = \mathcal{V}ar(A) \cup \mathcal{V}ar(B)$
$\mathcal{BV}ar(A \wedge B) = \mathcal{BV}ar(A) \cup \mathcal{BV}ar(B)$
$\mathcal{V}ar(A \vee B) = \mathcal{V}ar(A) \cup \mathcal{V}ar(B)$
$\mathcal{BV}ar(A \vee B) = \mathcal{BV}ar(A) \cup \mathcal{BV}ar(B)$
$\mathcal{V}ar(A \Rightarrow B) = \mathcal{V}ar(A) \cup \mathcal{V}ar(B)$
$\mathcal{BV}ar(A \Rightarrow B) = \mathcal{BV}ar(A) \cup \mathcal{BV}ar(B)$
$\mathcal{V}ar(\neg A) = \mathcal{V}ar(A)$
$\mathcal{BV}ar(\neg A) = \mathcal{BV}ar(A)$
$\mathcal{V}ar(\exists x A) = \mathcal{V}ar(A) - \{x\}$
$\mathcal{BV}ar(\exists x A) = \mathcal{BV}ar(A) \cup \{x\}$
$\mathcal{V}ar(\forall x A) = \mathcal{V}ar(A) - \{x\}$
$\mathcal{BV}ar(\forall x A) = \mathcal{BV}ar(A) \cup \{x\}$

On représente souvent les formules comme des arbres, et on utilise des mots sur le vocabulaire des entiers naturels pour désigner les positions des sous formules. Cela permet de parler plus précisément des occurrences libres ou liées d'une variable x. Par exemple, la variable x a une occurrence libre (à la position 1.1) et une occurence liée (à la position 2.1.1) dans la formule $P(x) \vee \forall x Q(x)$, à condition de considérer $\forall x$ comme un opérateur unaire indexé par la variable x. Cela a un sens informatique très précis : la variable liée x est représentée par un pointeur sur le noeud correspondant. Deux occurrences d'une même variable liées par le même quantificateur ($\forall$ ou $\exists$) pointeront donc sur le même noeud (étiquetté par $\forall$ ou $\exists$). Deux variables distinctes ou bien liées par des quantificateurs différents pointeront sur des noeuds distincts. Les variables libres pointeront sur des noeuds ($\exists$ ou $\forall$ selon les cas) fictivement rajoutés au dessus du terme. Cette représentation des termes est due à De Bruijn.

L'application d'une substitution à un terme est étendue aux formules:

Variables $x\sigma = \sigma(x)$

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

Atomes $P(t_1, \ldots, t_n)\sigma = P(t_1\sigma, \ldots, t_n\sigma)$

Formules
$$
\begin{aligned}
(A \wedge B)\sigma &= A\sigma \wedge B\sigma \\
(A \vee B)\sigma &= A\sigma \vee B\sigma \\
(A \Rightarrow B)\sigma &= A\sigma \Rightarrow B\sigma \\
(\neg A)\sigma &= \neg(A\sigma) \\
(\exists x A)\sigma &= \exists x(A\sigma_{\neq x}) \\
(\forall x A)\sigma &= \forall x(A\sigma_{\neq x})
\end{aligned}
$$

9.2 Sémantique

La sémantique de Tarski a pour but d'associer une valeur de vérité parmi (T pour *true* et F pour *false*) à toute formule close, et une fonction à valeur dans les valeurs de vérité à toute formule possédant des variables libres. Certaines sémantiques utilisent trois valeurs de vérité (en ajoutant la valeur *indéfini* par exemple). On munit ces valeurs de vérité d'une structure via une table de vérité qui décrit l'action des connecteurs Booléens sur les valeurs de vérité. Ces connecteurs Booléens expriment les propriétés des connecteurs logiques via les notions fondamentales de struture et d'interprétation.

$$\neg_{Bool}(T) = F, \qquad \neg_{Bool}(F) = T$$

$$\wedge_{Bool}(T,T) = T, \qquad \wedge_{Bool}(T,F) = F, \qquad \wedge_{Bool}(F,T) = F, \qquad \wedge_{Bool}(F,F) = F$$

$$\vee_{Bool}(T,T) = T, \qquad \vee_{Bool}(T,F) = T, \qquad \vee_{Bool}(F,T) = T, \qquad \vee_{Bool}(F,F) = F$$

$$\Rightarrow_{Bool}(T,T) = T, \qquad \Rightarrow_{Bool}(T,F) = F, \qquad \Rightarrow_{Bool}(F,T) = T, \qquad \Rightarrow_{Bool}(F,F) = T$$

Définition 9.1 *Une $\mathcal{F},\mathcal{P}$-*structure *(ou simplement* structure*) est une paire (D, I) où D est un ensemble appelé* domaine de discours *et I est une* interprétation *qui consiste d'une part en un homomorphisme de $T(\mathcal{F})$ dans une $\mathcal{F}$-algèbre de domaine D et d'autre part en une interprétation des symboles de prédicats: pour chaque symbole de prédicat $P \in \mathcal{P}_n$, $I(P)$ (aussi noté P_I) est un sous ensemble de D^n.*

On abrégera souvent (D, I) en D lorsque l'interprétation est déterminée sans ambigüité par le contexte.

Lorsque le symbole Q est d'arité nulle, la relation Q_I est alors une valeur de vérité*, T (pour true) ou F (pour false).*

Quelques exemples de structures importantes :

Exemple 9.2

Domaine	*Opérations*	*Relations*	*Nom*	*Notation*
N	$0, S, +$	$=, \leq$	*Arithmétique de Presburger*	*Presburger*
N	$+, *, 0, S$	$=, \leq$	*Arithmétique de Peano*	*Peano*
R	$+, *, 0, S$	$=, \leq$	*Théorie des Réels*	*Tarski*

Nous allons maintenant interpréter les formules, et il y a deux façons de procéder. La première consiste à interpréter une formule comme une fonction Booléenne de ses variables libres. Une formule close sera donc interprétée par une valeur de vérité. Techniquement, l'ensemble des variables libres d'une formule n'étant pas identique à celui de ses sous-formules, il va être nécessaire de calculer la fonction d'interprétation d'une formule vis à vis d'un ensemble quelconque de variables contenant ses variables libres. La seconde consiste à se donner à priori des valeurs du domaine d'interprétation pour toutes les variables de $\mathcal{X}$. Techniquement, cela nécessite l'introduction d'une fonction (arbitraire) de $\mathcal{X}$ dans le domaine D, appellée *valuation.* Une valuation étant donnée, l'interprétation d'une formule relativement à cette valuation devient une valeur de vérité. Cette solution évite donc la gestion d'un ensemble de variables dans la définition des interprétations, au prix d'un concept supplémentaire. Dans ce cours, nous avons choisi la première solution.

On notera par $[A]_{I,\{x_1,\ldots,x_n\}}(d_1,\ldots,d_n)$ la fonction d'interprétation de la formule A vis à vis de l'interprétation I et de l'ensemble de variables $\{x_1,\ldots,x_n\}$ contenant $\mathcal{V}ar(A)$. On omettra l'indice I lorsque cela sera sans ambiguité. On utilisera la notation $\overline{x}$ pour la liste de variables $\{x_1,\ldots,x_n\}$ prises dans cet ordre, la lettre d pour désigner un élément du domaine S, et $\overline{d}$ pour la liste de valeurs $\{d_1,\ldots,d_n\}$ prises dans cet ordre.

Variable $[y]_{\{\ldots,y,\ldots\}}(\ldots,d,\ldots) = d$

Terme $[f(t_1,\ldots,t_p)]_{\overline{x}}(\overline{d}) = f_I([t_1]_{\overline{x}}(\overline{d}),\ldots,[t_p]_{\overline{x}}(\overline{d}))$

Atome $[R(t_1,\ldots,t_p)]_{\overline{x}}(\overline{d}) = R_I([t_1]_{\overline{x}}(\overline{d}),\ldots,[t_p]_{\overline{x}}(\overline{d}))$

Formules $[0]_{\overline{x}}(\overline{d}) = F, \qquad [1]_{\overline{x}}(\overline{d}) = T$

$[\neg A]_{\overline{x}}(\overline{d}) = \neg_{Bool}[A]_{\overline{x}}(\overline{d})$

$[A \wedge B]_{\overline{x}}(\overline{d}) = [A]_{\overline{x}}(\overline{d}) \wedge_{Bool} [B]_{\overline{x}}(\overline{d})$

$[A \vee B]_{\overline{x}}(\overline{d}) = [A]_{\overline{x}}(\overline{d}) \vee_{Bool} [B]_{\overline{x}}(\overline{d})$

$[A \Rightarrow B]_{\overline{x}}(\overline{d}) = [A]_{\overline{x}}(\overline{d}) \Rightarrow_{Bool} [B]_{\overline{x}}(\overline{d})$

$[\exists x A]_{\overline{x}}(\overline{d}) = T$ ssi il existe $d \in S$ tel que $[A]_{x,\overline{x}}(d,\overline{d}) = T$

$[\forall x A]_{\overline{x}}(\overline{d}) = T$ ssi pour tout $d \in S$, $[A]_{x,\overline{x}}(d,\overline{d}) = T$

Dans la définition ci-dessus, notons que x ne doit pas apparaître dans la liste $\overline{x}$. Si cela était le cas, il faudrait renommer la variable liée x, ce qui est toujours possible. Par ailleurs, l'utilisation du ssi implique que l'interprétation est égale à F lorsque les conditions indiquées ne sont pas satisfaites.

Lemme 9.3 *Soient deux ensembles de variables $\overline{x}$ et $\overline{y}$ tels que $\mathcal{V}ar(A) \subseteq \overline{x}$. Alors la restriction de $[A]_{\overline{x}\cup\overline{y}}$ à $\overline{x}$ coincide avec $[A]_{\overline{x}}$.*

Preuve
Par récurrence sur la structure de A. $\square$

Définition 9.4 *On appelle interprétation de A dans la structure (S, I) la fonction $[A]_{I,\mathcal{V}ar(A)}$ encore notée $[A]_I$ ou tout simplement $[A]$ lorsque cela n'est pas ambigu.*

Le lemme suivant indique que les notions d'interprétation et de substitution sont compatibles :

Lemme 9.5 $[A\sigma]_{\overline{y}}(\overline{d}) = [A]_{\overline{x}}([x_1\sigma]_{\overline{y}}(\overline{d}), \ldots, [x_n\sigma]_{\overline{y}}(\overline{d}))$, *avec les conventions habituelles $\mathcal{V}ar(A) \subseteq \overline{x}$, et $\mathcal{V}ar(x_i\sigma) \subseteq \overline{y}$ pour tout $1 \leq i \leq n$.*

Définition 9.6 *Soit (S, I) uns structure interprétant le langage du premier ordre $\mathcal{CP}1(\mathcal{F}, \mathcal{P}, \mathcal{X})$, et A une formule de n variables libres. On dit que :*

- A est satisfiable dans (S, I) si il existe $\overline{d} \in S^n$ tel que $[A](\overline{d}) = T$,

- A est satisfiable s'il existe une structure (S, I) dans laquelle A est satisfiable, et insatisfiable dans le cas contraire,

- A est valide dans (S, I), noté $(S, I) \models A$, si pour tout $\overline{d} \in S^n$, alors $[A](\overline{d}) = T$, et on dit dans ce cas que (S, I) est un modèle de A,

- A est (universellement) valide, noté $\models A$, si A est valide dans toute structure (S, I), et on dit alors également que A est une tautologie.

Une formule est donc satisfiable si sa clôture existentielle $\exists\mathcal{V}ar(A)A$ s'interprète en T dans une certaine structure (S, I), et valide si sa clôture universelle $\forall\mathcal{V}ar(A)A$ s'interprète en T dans toute structure (S, I). On peut donc toujours raisonner sur des formules closes.

Notons qu'une formule peut être satisfiable sans posséder de modèle. Notons également qu'une formule A est valide ssi sa négation $\neg A$ est insatisfiable. Celle dernière remarque est à la base des méthodes de preuve par réfutation.

Notons qu'une formule peut être satisfiable sans posséder de modèle.

Définition 9.7 *B est* conséquence sémantique *de A, noté $A \models B$, si pour toute structure (S, I), $[B]_{\overline{x}}(\overline{d}) = T$ chaque fois que $[A]_{\overline{x}}(\overline{d}) = T$. La relation de conséquence sémantique s'étend naturellement à des ensembles de formules aussi bien à droite qu'à gauche du symbole $\models$.*

Pour des formules closes, $A \models B$ ssi $(S, I) \models B$ pour tout modèle (S, I) de A.

La relation de conséquence sémantique est un préordre dont l'équivalence associée est l'*équivalence sémantique* notée $\equiv$.

Exemple 9.8 $A \Rightarrow B \equiv B \vee \neg A$

Cette équivalence nous aurait permis de définir un calcul des prédicats sans l'implication, puis de rajouter l'implication $A \Rightarrow B$ comme une abréviation de la formule $\equiv B \vee \neg A$. C'est pour cette raison que nous nous sommes dispensés de quelques autres opérateurs logiques importants, comme l'équivalence, $A \leftrightarrow B$ étant l'abbréviation de $(A \Rightarrow B) \wedge (B \Rightarrow A)$, le ou exclusif, $A \otimes B$ étant l'abbréviation de $(A \vee B) \wedge \neg(A \wedge B)$, etc ...

L'équivalence sémantique permet de munir l'ensemble des formules d'une structure d'algèbre de Boole. Le choix d'un ensemble d'opérateurs adéquat (basé sur le ou exclusif noté $\otimes$) résulte même en une structure d'anneau Booléen. Nous donnons ci-dessous les axiomes de l'algèbre de Boole:

$$\begin{array}{rcl@{\qquad}rcl}
A \vee (B \vee C) & \equiv & (A \vee B) \vee C & A \wedge (B \wedge C) & \equiv & (A \wedge B) \wedge C \\
A \vee B & \equiv & B \vee A & A \wedge B & \equiv & B \wedge A \\
A \vee 0 & \equiv & A & A \wedge 0 & \equiv & 0 \\
A \vee 1 & \equiv & 1 & A \wedge 1 & \equiv & A \\
A \vee A & \equiv & A & A \vee A & \equiv & A \\
A \vee (B \wedge C) & \equiv & (A \vee B) \wedge (A \vee C) & A \wedge (B \vee C) & \equiv & (A \wedge B) \vee (A \wedge C) \\
\neg\neg A & \equiv & A & \neg(A \wedge B) & \equiv & (\neg A) \vee (\neg B) \\
\neg(A \vee B) & \equiv & (\neg A) \wedge (\neg B) & \neg 0 & \equiv 1 & \\
\neg 1 & \equiv & 0 & & &
\end{array}$$

Définition 9.9 *Deux structures (S, I) et $(S', I,)$ sont* isomorphes *s'il existe une bijection ξ de S dans S' (étendue aux produits cartésiens S^n) telle que :*

- pour tout symbole de fonction $f \in \mathcal{F}_n$ et tout $\overline{d} \in S^n$, $f_I(\overline{d}) = c$ ssi $f_{I'}(\xi(\overline{d})) = \xi(c)$,

- pour tout symbole de relation $R \in \mathcal{P}_n$ et tout $\overline{d} \in S^n$, $\overline{d} \in R_I$ ssi $\xi(\overline{d}) \in R_{I'}$,

Par exemple, deux graphes qui ne diffèrent que par le nom des sommets sont isomorphes.

Définition 9.10 *La* théorie *d'une structure (S, I) est l'ensemble des formules de $CP1$ vraies dans (S, I). On la note $Th(S, I)$.*

Un ensemble Γ de formules closes est une théorie *ssi il est clos par conséquence sémantique, cad $A \in \Gamma$ ssi $\Gamma \models A$.*

Une théorie Γ est consistante *si elle ne contient pas à la fois une formule A et sa négation $\neg A$.*

Une théorie Γ est complète *si pour toute formule A, $A \in \Gamma$ ou $(\neg A) \in \Gamma$.*

Une théorie Γ est finiment axiomatisable *s'il existe un sous-ensemble fini de Γ dont Γ soit conséquence sémantique.*

Une théorie Γ est récursivement axiomatisable *s'il existe un sous-ensemble récursif (reconnu par un algorithme qui termine toujours) de Γ dont Γ soit conséquence sémantique.*

Une théorie Γ est décidable *s'il existe une procédure qui termine toujours, telle que pour toute formule A la procédure retourne oui si $A \in \Gamma$ et non dans le cas contraire.*

Une théorie Γ est semi-décidable *s'il existe une procédure telle que pour toute formule A la procédure retourne oui si $A \in \Gamma$ et non ou ne termine pas dans le cas contraire.*

Exemple 9.11 *L'aritmétique de Presburger (resp. Peano, Tarski) est la théorie de la structure Presburger (resp. Peano, Tarski) de l'exemple 9.2.*

Lemme 9.12 *La théorie de Presburger est décidable. La théorie de Peano est indécidable. La théorie de Tarski est décidable.*

Les preuves de ces assertions ne sont pas immédiates, la dernière étant difficile. La décidabilité de l'arithmétique de Presburger peut s'obtenir par des techniques d'automates.

Exemple 9.13 *La théorie du graphe* $(\{1,2\},\{(1,2),(2,1)\})$ *contient les formules* $\forall xy(xEy \Rightarrow yEx)$, $\forall x.\neg xEx$, $\forall xy(xEy \vee \neg xEy)$, ..., *où* $E \in \mathcal{P}_2$.

Exemple 9.14 *Le prédicat d'égalité est en général implicitement présent dans le vocabulaire d'un langage du premier ordre. Sa théorie (dite de l'égalité) est définie par les axiomes :*

$$\textit{réflexivité :}\ \ \forall x\ x = x$$

$$\textit{symmétrie :}\ \ \forall xy\ x = y \Rightarrow y = x$$

$$\textit{transitivité :}\ \ \forall xyz\ x = y \wedge y = z \Rightarrow x = z$$

Pour chaque entier n et chaque symbole de fonction $f \in \mathcal{F}_n$ *:*

$$\textit{congruence :}\ \ \forall \overline{x}\overline{y}\ \overline{x} = \overline{y} \Rightarrow f(\overline{x}) = f(\overline{y})$$

Pour chaque entier n et chaque symbole de prédicat $R \in \mathcal{P}_n$ *:*

$$\forall \overline{x}\overline{y}\overline{x} = \overline{y} \Rightarrow R(\overline{x}) \leftrightarrow R(\overline{y})$$

Cette théorie possède donc un ensemble fini d'axiomes si le langage $CP1$ *possède un vocabulaire fini.*

Exemple 9.15 *La théorie des groupes est engendrée par exemple par les trois axiomes suivants :*

$$\forall xyz\ x * (y * z) = (x * y) * z, \quad \forall x\ e * x = x, \quad \forall x \exists y\ x * y = e$$

La théorie des groupes n'est pas complète, car il existe des groupes commutatifs et d'autres qui ne le sont pas.

Exemple 9.16 *L'arithmétique de Peano est engendrée par les axiomes suivants :*

$$\forall x\ \neg(S(x) = 0), \quad \forall xy\ S(x) = S(y) \Rightarrow x = y$$

$$\forall x\ x + 0 = x, \quad \forall xy\ x + S(y) = S(x + y)$$

$$\forall x\ x * 0 = 0, \quad \forall xy\ x * S(y) = x * y + x$$

$$\forall x\ \neg(x < 0), \quad \forall x\ 0 < S(x), \quad \forall xy\ S(x) < S(y) \Rightarrow x < y$$

Pour toute formule A, et pour toute variable x libre dans A,

$$[A(0) \wedge \forall x(A(x) \Rightarrow A(S(x)))] \Rightarrow \forall x A(x)$$

Cette dernière expression est le schéma d'axiomes de récurrence. Il y a donc un axiome de récurrence par formule et variable libre de la formule. L'arithmétique de Peano n'est donc pas finiment axiomatisable, mais elle est récursive car il existe une procédure qui termine toujours qui, étant donnée une formule, réponde oui s'il s'agit d'un axiome de récurrence et non dans le cas contraire.

Lemme 9.17 *L'arithmétique de Peano est incompléte.*

La preuve d'incomplétude est due à Gödel. Sa lecture est recommandée.

Nous avons donc maintenant deux définitions de l'arithmétique de Peano. Montrer qu'elles coincident est laissé à la sagacité (ou à l'érudition) du lecteur.

Exemple 9.18 *La théorie des arbres finis étiquetés sur un alphabet fini est fondamentale à l'informatique. Elle joue un rôle clé en programmation logique en particulier, où elle sous-tend la notion d'unification et les axiomes de Clarke pour le traitement de la négation. Elle sera étudiée plus en détail dans le chapitre 11. Cette théorie est engendrée par les axiomes de l'égalité que nous avons déjà vus ainsi que par les axiomes suivants qui expriment la "liberté" de l'algèbre $T(F)$:*

Décomposition : *Pour chaque entier n et chaque symbole $f \in \mathcal{F}_n$:*

$$\forall \overline{xy}[f(\overline{x}) = f(\overline{y}) \leftrightarrow \overline{x} = \overline{y}]$$

Conflit : *Pour chaque couple d'entiers m, n et chaque couple de symboles $f \in \mathcal{F}_m, g \in \mathcal{F}_n$:*

$$\forall \overline{xy}[f(\overline{x}) = g(\overline{y}) \leftrightarrow 0]$$

Occurence :

$$\forall xy[x[y]_{p \neq \Lambda} = y \leftrightarrow 0]$$

où $x[y]_{p \neq \Lambda}$ est une notation signifiant que y est un sous-arbre de x à une position p différente de la racine. La définition axiomatique de cette notation est laissée au lecteur.

Monde clos :

$$\forall x(\exists \overline{y}\ x = f(\overline{y})) \vee \ldots \vee (\exists \overline{y}\ x = g(\overline{z}))$$

la disjonction précédente énumérant tous les symboles de fonction de $\mathcal{F}$. Cet axiome est donc finitaire si $\mathcal{F}$ est fini.

Notons que cet ensemble de formules est infini, à cause des axiomes d'occurrence, mais récursif.

Theorem 9.19 *La théorie des arbres finis étiquetés est complète.*

Ce résultat est dû à Mal'cev. On verra au chapitre 11 une axiomatisation récursive semblable (due à Maher) pour les arbres infinis, ainsi qu'une axiomatisation complète dans le cas des arbres infinis. Une preuve récente du théorème, qui se généralise volontiers à des structures d'arbres quotients par des axiomes comme la commutativité d'un symbole, est due à Comon.

Quoique non immédiates sauf pour le premier, les preuves des lemmes généraux qui suivent sont laissées au plaisir du lecteur.

Lemme 9.20 *Deux structures isomorphes sont modèles des même formules.*

Lemme 9.21 *Les deux structures $(Q, \{<_Q\})$ et $(R, \{<_R\})$ sont modèles des même formules du premier ordre.*

Lemme 9.22 *Deux structures finies qui valident le même ensemble de formules sont isomorphes.*

Lemme 9.23 *Si un ensemble de formules a un modèle unique à isomorphisme près, alors sa clôture par conséquence sémantique est une théorie complète.*

Le problème dual, la définissabilité, est par essence informatique : il consiste à programmer dans une structure au moyen du langage du calcul des prédicats du premier ordre. Ce problème a été très étudié dans le cadre des langages de bases de données, pour lesquelles les structures peuvent être considérées comme finies.

Définition 9.24 *Une relation R d'arité n sur une structure (S, I) est* définissable *s'il existe une formule $A \in \mathcal{CP}1(\mathcal{F}, \mathcal{P}, \mathcal{X})$ telle que $[A]_{I,\overline{x}}(\overline{d}) = T$ ssi $\overline{d} \in R$. La définition s'étend aux classes de structures.*

Exemple 9.25 *La relation de la structure Peano $R(x)$ ssi x est premier est définissable par la formule :*

$$\forall yz\ x = y * z \Rightarrow (y = x \wedge z = S(0)) \vee (z = x \wedge y = S(0))$$

L'interprétation de cette formule dans la structure Peano vaut en effet T pour tout nombre premier et F pour tout nombre non premier.

La logique du premier ordre est donc assez pauvre puisqu'elle ne permet pas de définir de nombreux concepts simples. Pour remédier à ce problème, la solution consiste à enrichir le langage logique, en autorisant la quantification sur des variables de fonctions ou de prédicat. On obtient ainsi la logique du second ordre, dont une restriction, la logique monadique du second ordre, n'autorise que la quantification sur les symboles de prédicats unaires, cad sur des variables interprétées comme des sous-ensembles du domaine d'interprétation. Ce fragment de la logique a une grande importance en informatique théorique, puisqu'il permet de caractériser les langages (de mots ou d'arbres) reconnaissables par un automate fini (de mot ou d'arbre). Il a une grande importance pratique également, puisque les automates sont utilisés de manière routinière pour modéliser les circuits logiques.

9.3 Mise sous forme clausale

La suite de ce cours a pour but d'introduire des méthodes calculatoires qui trouvent leur origine dans des travaux de Herbrand et de Robinson. Ces méthodes reposent sur la transformation de formules en clauses.

Définition 9.26 *Une* formule clausale *est une formule prénexe sans quantificateur existentiel.*

Une clause *est une formule clausale de la forme $\forall\overline{x} A_1 \wedge \ldots \wedge A_n \Rightarrow B_1 \vee \ldots \vee B_p$, ou, de manière équivalente $\forall\overline{x}(\neg A_1) \vee \ldots \vee (\neg A_n) \vee B_1 \vee \ldots \vee B_p$. Les quantificateurs d'une clause étant tous universels, ils sont souvent omis. Il sera parfois utile de distinguer la version quantifiée d'une clause de sa version non quantifiée en écrivant $\forall\overline{x}C$ pour la première et C pour la seconde.*

Lemme 9.27 *Toute formule clausale est équivalente à une conjonction de clauses.*

Preuve
La preuve et constructive. Il suffit d'appliquer les règles de l'algèbre de Boole (sauf les axiomes de commutativité et associativité), et de remarquer que ces règles terminent lorsqu'elles sont appliquées de gauche à droite sur les classes d'équivalence de formules modulo la commutativité et l'associativité de la conjonction et de la disjonction. La forme normale est une conjonction de clauses. On peut aussi faire une preuve par récurrence. $\square$

L'importance de la notion de clause vient du théorème de Skolem :

Theorem 9.28 *Pour toute formule A, il existe un ensemble S de clauses tel que A est insatisfiable ssi S l'est.*

Preuve
Par récurrence sur le nombre de quantificateurs existentiels de A supposée en forme prénexe sans perte de généralité, on démontre l'existence d'une formule clausale telle que $\models A$. Et donc est insatisfiable puisque A l'est. Mais il ne serait pas correct de dire que A et sont équivalentes, car on va voir que les deux formules ne sont pas construites sur le même langage logique.

Si A ne possède pas de quantificateur existentiel, c'est terminé. Sinon, soit $A = \forall \overline{x} \exists y\ B$, où B est donc une formule prénexe. Soit f un nouveau symbole de fonction (dit *symbole de Skolem*) dont l'arité est égale au nombre de variables dans $\overline{x}$, et C la formule prénexe $\forall \overline{x} B\{y \mapsto f(\overline{x})\}$. On vérifie que $C \models A$ grâce au lemme 9.5. C contient un quantificateur existentiel de moins que A, donc il existe une formule clausale telle que $\models C$. On conclue par transitivité de la relation de conséquence sémantique, et application du lemme précédent.

Notons que si C était satisfiable dans une interprétation (S, I), alors on en déduirait aisément une interprétation (S, I') (prenant pour I' la restriction de I au langage de A) telle que A soit satisfiable. $\square$

Exemple 9.29 *Mise en forme clausale de la formule* $\neg(\forall x \exists y.P(x,y)) \vee \exists z.Q(z)$*:*

$$\begin{aligned} \neg(\forall x \exists y.P(x,y)) \vee \exists z.Q(z) &\rightarrow \exists x \forall y \exists z.\neg P(x,y) \vee Q(z) \\ &\rightarrow \forall y.\neg P(a,y) \vee Q(f(y)) \end{aligned}$$

Mais la forme clausale n'est pas unique; on peut aussi effectuer la transformation:

$$\begin{aligned} \neg(\forall x \exists y.P(x,y)) \vee \exists z.Q(z) &\rightarrow \forall x(\exists y.P(x,y) \vee \exists y.Q(y)) \\ &\rightarrow \forall x.P(x, f(x)) \vee Q(f(x)) \end{aligned}$$

Démontrer qu'une formule A est un théorème sous les hypothèses $H_1, \ldots, H_n$ se ramène donc à la preuve d'inconsistance d'un ensemble de clauses comme suit :

1. $H_1, \ldots, H_n \models A$ ssi $H_1 \wedge \ldots \wedge H_n \wedge \neg A$ est insatisfiable,

2. Mettre chaque formule de la conjonction sous forme prénexe,

3. Skolemiser chaque formule obtenue à l'étape précédente,

4. Mettre chaque formule obtenue à l'étape précédente sous forme clausale

5. Montrer que l'ensemble de clauses ainsi ontenu est insatisfiable.

On peut bien sûr également s'intéresser à minimiser le nombre de clauses, ou tout autre mesure de la taille du problème final à résoudre.

9.4 Théorèmes de Herbrand

La notion de satisfiabilité est une notion sémantique, qui fait appel aux calculs de toutes les interprétations possibles dans tous les modèles possibles. Le théorème de Herbrand ramène ces calculs sémantiques infaisables à des calculs dans des structures particulières appellées syntaxiques, ou libres, ou de Herbrand, qui vont être construites à partir du vocabulaire figurant dans les clauses.

Soit $\mathcal{T}(\mathcal{F})$ l'ensemble des termes clos, encore appellé univers de Herbrand et noté $\mathcal{G}$. Soit $\mathcal{A}(\mathcal{P}, \mathcal{F})$ l'ensemble des atomes clos, encore appellé base de Herbrand et noté $\mathcal{B}$

Définition 9.30 *Une interprétation H de Herbrand*
- a pour domaine l'univers de Herbrand $\mathcal{G}$,
- interprète le symbole $f \in \mathcal{F}_n$ par l'opération F_H d'arité n telle que $f_H(\overline{t}) = f(\overline{t})$,
- interprète le symbole de prédicat $R \in \mathcal{P}_n$ par un sous-ensemble arbitraire R_H de $\mathcal{G}^n$.

Theorem 9.31 *((Herbrand)) Un ensemble S de clauses est satisfiable ssi il l'est dans une interprétation de Herbrand. Il est donc insatisfiable ssi il l'est dans toute interprétation de Herbrand.*

Preuve
Soit I une interprétation qui satisfait S. On considère l'interprétation de Herbrand H telle que $R_H(\overline{t}) = [R(\overline{t})]_I$, dont on montre qu'elle satisfait S. On utilise pour cela le lemme 9.5. □

La base de Herbrand étant dénombrable, (l'ensemble des termes clos de taille n est fini si $\mathcal{F}$ et $\mathcal{P}$ sont finis et dénombrable sinon), il est possible d'énumérer les atomes clos, donc $\mathcal{B} = \{A_i\}_{i \in N}$. On peut donc organiser l'ensemble des interprétations de Herbrand en un arbre binaire dont les branches sont en correspondance biunivoque avec les interprétations de Herbrand, appellé *arbre des interprétations*; jusqu'à la profondeur n, l'arbre décrit toutes les interprétations des n premiers termes de la base de Herbrand. Les fils d'un nœud donné, de profondeur n correspondent aux deux prolongements possibles de l'interprétation au $n + 1$ième élément de la base.

Un exemple est donné dans la figure 9.1: on suppose dans cet exemple qu'il y a un symbole de prédicat binaire P, une constante 0 et un symbole de fonction unaire s. Les éléments de la base de Herbrand sont ordonnés suivant la taille des

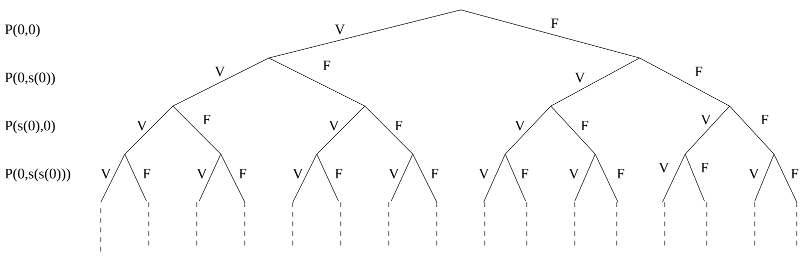

Figure 9.1: Exemple d'arbre des interprétations

atomes, puis, en cas d'égalité des tailles suivant l'ordre lexicographique des tailles des arguments de P.

Un chemin de la racine à un noeud $p \in (N^+)^*$ de l'arbre sera noté H_p et appellé une *interprétation partielle* des atomes A_0 à $A_{|p|-1}$. On dira que l'interprétation I_q *prolonge* l'interprétation I_p si le chemin p est un préfixe du chemin q. On peut définir un calcul des interprétations partielles dans une logique trivaluée (avec une nouvelle valeur $\perp$ qui se veut représenter la valeur indéfinie) comme nous avons défini un calcul des interprétations (totales) en logique binaire. Dans la mesure où il ne s'agit que d'un outil technique, nous nous contenterons de définir les interprétations (de Herbrand) partielles de la manière suivante :

- $H_p(A) = T$ si $H(A) = T$ dans toute interprétation totale qui prolonge H_p.
- $H_p(A) = F$ si $H(A) = F$ dans toute interprétation totale qui prolonge H_p.
- $H_p(A) = \perp$ dans le cas contraire.

Définition 9.32 *p est un* noeud d'échec *pour l'ensemble $S = \{C_1, \ldots, C_n\}$ de clauses s'il existe une substitution close γ telle que $[C\gamma]_{H_p} = F$ et la propriété n'est vrai pour aucun prédécesseur de p dans l'arbre des interprétations.*

On appelle arbre sémantique *d'un ensemble S de clauses l'arbre obtenu à partir de l'arbre des interprétations en élaguant les sous-arbres issus d'un noeud d'échec et en étiquettant ce dernier par les instances des clauses réfutées.*

Un arbre sémantique sera dit clos *si toute branche se termine en un noeud d'échec.*

Exemple 9.33 *Supposons que $\mathcal{F}$ est réduit à un symbole de constante 0 et à un symbole unaire s, $\mathcal{P}$ étant réduit au symbole de prédicat unaire A. Soit S l'ensemble de clauses:*

$$\begin{aligned} S = \quad & \{\neg A(0) \vee A(s(s(0))), \neg A(s(0)) \vee A(s(s(s(0)))), \ldots, \\ & \neg A(s^n(0)) \vee A(s^{n+2}(0)), \ldots, A(0), A(s(0)), \neg A(s(s(s(0)))) \vee \neg A(s(0))\} \end{aligned}$$

On obtient l'arbre sémantique clos de la figure 9.2.

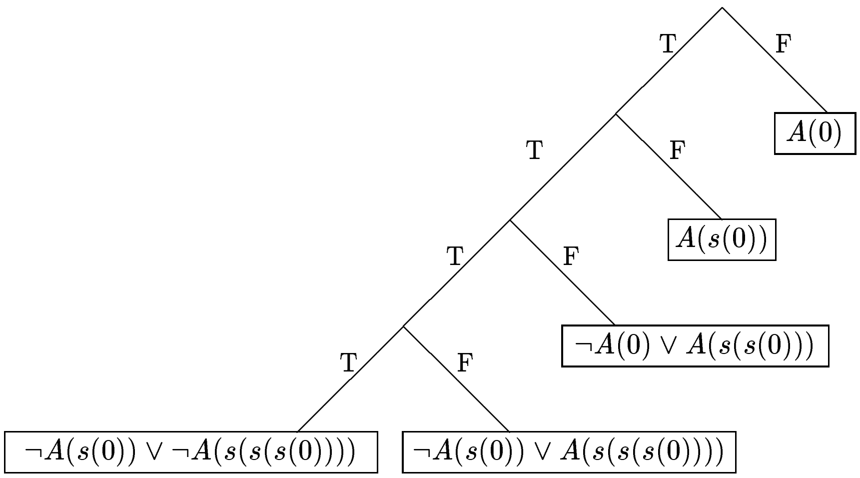

Figure 9.2: Exemple d'arbre sémantique clos

Un arbre sémantique clos est bien sûr fini, cela est une conséquence de l'énumération des atomes de la base de Herbrand. Si nous avions choisi d'énumérer les atomes de la base selon un ordre transfini, donc non isomorphe à l'ordre des entiers naturels, alors l'arbre sémantique ne serait plus nécessairement fini. La notion d'arbre clos, toutefois, est en fait suffisament générale pour traiter ce cas là également.

Theorem 9.34 *Un ensemble S de clauses est insatisfiable ssi il possède un arbre sémantique clos.*

Preuve
Si un ensemble S de clauses possède un arbre sémantique clos, alors toute interprétation I prolonge une interprétation partielle J qui étiquette un noeud d'échec. Il existe donc au moins une instance $C\gamma$ d'une clause C de S dont l'interprétation est fausse dans J donc dans I.

Réciproquement, si S possède un arbre sémantique non clos, alors il existe une interprétation I dont aucune interprétation partielle nétiquette un noeud d'échec. Supposons qu'il existe une clause C de S telle que $[\forall \overline{x} C]_I = F$, cad $[C\gamma]_I = F$ pour une certaine substitution close γ. Les atomes de $C\gamma$ étant en nombre fini, ils sont tous énumérés avant un certain entier k. Il existe donc une interprétation partielle H_p que prolonge I, telle que $[C\gamma]_{H_p} = F$. Donc il existe un noeud préfixe de p qui est noeud d'échec, ce qui est contraire à l'hypothèse. Donc I est un modèle de S. $\square$

9.5 Exercices

9.5.1

Décrire un algorithme qui engendre comme précédemment un ensemble de clauses insatisfiables pour lequel le nombre de symboles de Skolem ajoutés soit minimal.

9.5.2

On désire étudier une classe de graphes considérés comme des structures finies $\{E, \{r, S\}, \{=\}\}$, où r est un symbole de constante et S un symbole de fonction unaire. r est interprété comme la racine (supposée unique) du graphe, et S comme une fonction injective donnant le successeur de chaque noeud (élément de E) dans la relation du graphe, qui est donc fonctionnelle. Ces propriétés sont axiomatisées par les trois axiomes:

$$\text{A}: \ \forall x \ \neg(S(x) = r)$$

$$\text{B}: \ \forall x \ (\forall y \neg(S(x) = y)) \Rightarrow x = r$$

$$\text{C}: \ \forall xy \ (S(x) = S(y) \Rightarrow x = y$$

On considère les structures :

$$N = \{N, \{0, succ\}, \{=\}\}, \quad Z = \{Z, \{succ\}, \{=\}\}, \quad Z/p = \{[0..(p-1)], \{succ\}, \{=\}\}$$

et appellerons $NZ * Z/p*$ l'ensemble des structures formées d'une copie unique de N, d'un nombre fini ou infini (de cardinal arbitraire) de copies de Z, et d'un nombre fini ou infini (de cardinal arbitraire) de copies de Z/p.

1. Montrer que les axiomes A, B, C n'ont pas de modèle fini.

2. Montrer que toute structure de $NZ * Z/p*$ est un modèle de A, B, C.

On se propose maintenant de montrer que tout modèle de A, B, C est isomorphe à une structure de $NZ*Z/p*$. Pour cela, on considère les trois sortes de composantes connexes d'un modèle, infini avec racine, infini sans racine, et fini.

1. Considérons une composante connexe infinie avec racine r. Soit ξ l'application de N dans les sommets de la composante connexe définie par $\xi(0) = r$ et $\xi(succ(x)) = S(\xi(x))$ pour tout entier $x \geq 0$. Montrer successivement que ξ est une application de N dans les sommets de la composante connexe de r, puis qu'elle est injective, enfin qu'elle est surjective. En déduire que la composante connexe est isomorphe à N.

2. Considérons une composante connexe infinie sans racine. Soit ξ l'application de Z dans les sommets de la composante connexe définie par $\xi(0) = a$, où a désigne un sommet arbitraire de la composante connexe, $\xi(succ(x)) = S(\xi(x))$ pour tout entier $x \geq 0$, et $\xi(pred(x)) = y$ tel que $S(y) = x$ pour tout entier $x \leq 0$, et En déduire que la composante connexe est isomorphe à Z.

3. Considérons une composante connexe finie ayant p sommets. Montrer que cette composante connexe est en fait un cycle Hamiltonien élémentaire (passant une fois et une seule par chaque sommet). En déduire qu'elle est isomorphe à Z/p.

4. En déduire que tout modèle de A, B, C est isomorphe à une structure de $NZ * Z/p*$.

On se propose maintenant d'éliminer certains modèles superflus pour se rapprocher du modèle "standerd" des entiers qui est formé d'une unique composante connexe. Pour cela, on va rajouter le schéma d'axiomes d'induction :

$$\text{D} : \Phi(0) \wedge [\forall x(\Phi(x) \Rightarrow \Phi(S(x)))] \Rightarrow \forall x \Phi(x)$$

pour toute formule Φ et variable libre x de Φ. Soit Ψ la formule $\forall x \neg S^p(x) = x$, où S^p désigne l'application p fois de S.

1. Montrer que Ψ n'est pas valide dans les modèles de Z/p.

2. Montrer que $\neg S^p(0) = 0$ est valide dans les modèles de A, B, C, D.

3. Montrer que $\forall x \ \neg S^p(S(x)) = S(x)$ est valide dans les modèles de A, B, C.

4. En déduire que Ψ est valide dans les modèles de A, B, C, D.

5. En déduire que les modèles de A, B, C, D sont formés d'une unique copie de N, et d'un nombre arbitraire de copies de Z.

6. Peut-on éliminer les copies de Z en rajoutant de nouveaux axiomes ou schémas d'axiomes?

9.5.3

Soit la structure propositionnelle $(S, \{\}, \{p_1, \ldots, p_n\})$.

1. Soit $n = 3$. On considère la formule $A = (p_1 \wedge p_2 \wedge \neg p_3) \vee (p_1 \wedge \neg p_2 \wedge p_3) \vee (\neg p_1 \wedge p_2 \wedge p_3)$.

   A et $\neg A$ sont-elles satisfiables? valides?

2. Soit B la formule où p_1 est remplaçé par $\neg p_1$.

   Est-ce que $A \models B$ ou $B \models A$?

3. Soit maintenant $n = 4$. Trouver une formule Ψ qui s'interprète en T dans la structure propositionnelle ssi le nombre de symboles propositionnels égaux à T dans la structure est égal au nombre de symboles propositionnels égaux à F.

4. Peut-on généraliser à $n = 2k$?

9.5.4

Trouver une formule A batie sur le langage défini par la constante 0, le symbole de fonction unaire S, le prédicat d'égalité $=$ et le prédicat ternaire $Plus$ telle que la structure N soit un modèle de A ssi le symbole de prédicat $Plus$ est interprété par la relation $P+$ définie par $(x, y, z) \in P+$ ssi $z = x + y$.

Afin de montrer que $P+$ est la seule relation qui convient, on pourra considérer un plus petit triplet (x, y, z) pour lequel R et $P+$ diffèrent.

9.5.5

On considère la structure $(R, \{0, 1, +, -, *, \div\}, \{<\})$.

1. Evaluer la formule $A = \forall xy \exists z \ (x < y) \Rightarrow (x < z \wedge z < y)$.
2. La structure est-elle un modèle de A?
3. Trouver les relations Qa telles que $(R, \{0, 1, +, -, *, \div\}, \{<, Qa\}) \models \forall xy(Q(x, y) \Rightarrow Q(y, x)) \Rightarrow (x \neg y)$, où Q désigne un symbole de prédicat binaire.
4. Construire une formule B exprimant la transitivité d'une relation binaire, et une formule C exprimant son antisymmétrie. Est-ce que $B \wedge C \models A$?
5. Construire une formule exprimant que l'ordre partiel défini par B et C est en fait un ordre total.

9.5.6

Peut-on caractériser la connexité d'un graphe quelconque par une formule de $CP1$?

9.5.7

Soient $\mathcal{F} = \{a\}, \mathcal{P} = \{P\}$ et ϕ la formule $\neg P(a) \vee \exists x P(x)\}$. ϕ est elle satisfaisable ? Possède-t-elle un modèle de Herbrand ? Que faire?

9.5.8

Construire les arbres sémantiques pour les ensembles de clauses suivants et dire s'ils sont insatisfiables:

1. $S_1 = \{A(x) \vee \neg A(s(s(x))), \ A(0) \vee A(s(0)), \ \neg A(0) \vee \neg A(s(0))\}$
2. $S_2 = S_1 \cup \{\neg A(s(s(s(0)))) \vee \neg A(s(0))\}$
3. $S_3 = S_2 \cup \{\neg A(0) \vee \neg A(s(s(0)))\}$

Avec $\mathcal{F} = \{0, s\}$ et $\mathcal{P} = \{A\}$.

Chapter 10

Résolution

10.1 Résolution close

Nous présentons tout d'abord la règle de résolution pour des clauses closes, et considérons ultérieurement le cas des variables.

$$\frac{A \vee C \qquad \neg A \vee D}{C \vee D}$$

Dans cette *règle d'inférence*, A dénote un atome, et C et D dénotent des clauses (éventuellement vides). Les clauses $A \vee C$, $\neg A \vee D$, $C \vee D$ sont supposées en forme normale vis à vis des règles de l'algèbre de Boole. En conséquence, ni A ni $\neg A$ ne peuvent apparaître dans $C \vee D$.

Lemme 10.1 *(Correction) Si $[A \vee C]_I = [\neg A \vee C]_I = T$, alors $[C \vee D]_I = T$.*

Preuve
On considère les deux cas suivant que l'interprétation de A est vraie ou fausse. □

On note par $Res(S)$ l'ensemble des clause inférées en une étape de résolution à partir de S, et par $Res^*(S)$ la fermeture par inférence de cet ensemble, cad $Res^0(S) = S$, $Res^n(S) = Res(Res^{n-1}(S))$, et $Res^*(S) = \bigcup_{n=0}^{n=\infty} Res^n(S)$.

Lemme 10.2 *(Complétude) La résolution close est réfutationellement complète, cad S est insatisfiable ssi la clause vide appartient à $Res^*(S)$.*

Preuve
Si la clause vide est dans $Res^*(S)$, alors $Res^*(S)$ est insatisfiable, et la correction de la résolution assure qu'il en est de même de S.

Réciproquement, si S est insatisfiable, alors il en est de même de $Res^*(S)$ qui possède donc un arbre sémantique clos par le second théorème de Herbrand. Supposons que l'arbre sémantique n'est pas réduit à sa racine. Comme il est binaire donc à branchement fini, il possède nécessairement un noeud interne J dont les deux successeurs L et R sont des feuilles, donc des noeuds d'échec.

Soit A l'atome interprété en T dans L, F dans R, et non interprété dans J. Comme J et L ne diffèrent que par l'interprétation de l'atome A et que L est noeud d'échec, L est nécessairement étiquetté par une clause de la forme $\neg A \vee C$, telle que $[\neg A \vee C]_L = F$. Mais comme ni A ni $\neg A$ ne peuvent apparaître dans la clause C qui ne serait plus en forme normale, C est donc évaluable dans J, et on a donc nécessairement $[C]_J = [C]_L = F$ puisque L prolonge J. De même, R est étiquetté par une clause de la forme $A \vee D$ telle que $[D]_J = F$. Donc $[C \vee D]_J = F$.

Comme $C \vee D$ est inférée par résolution à partir des clauses $\neg A \vee C$ et $A \vee D$ qui appartiennent toutes deux à $Res^*(S)$ par hypothèse, $C \vee D$ appartient à $Res^*(S)$. Cela contredit la définition d'un arbre sémantique, aucune clause ne pouvant être réfutée par l'interprétation associée à un noeud interne de l'arbre. Donc l'arbre sémantique était réduit à sa racine. Dans ce cas, il est clair que la clause vide est dans $Res^*(S)$, puisqu'elle seule réfute l'interprétation vide. □

10.2 Relèvement et complétude de la résolution

Nous pouvons maintenant passer au cas général des clauses avec variables. L'idée est de se ramener à la règle de résolution close par instanciation adéquate des deux clauses à résoudre, de façon à faire apparaître deux atomes égaux, l'un sans négation dans l'une des clauses, l'autre avec négation dans l'autre clause. C'est là où l'unification va entrer en jeu.

$$\text{Résolution :} \quad \frac{P_1 \vee \ldots \vee P_p \vee C \qquad \neg N_1 \vee \ldots \neg N_n \vee D}{C\sigma \vee D\sigma}$$

Dans cette *règle d'inférence* de résolution, σ désigne l'unificateur principal du problème d'unification

$$P_1 = P_2 \wedge \ldots \wedge P_1 = P_p \wedge P_1 = N_1 \wedge N_1 = N_2 \wedge \ldots \wedge N_1 = N_n$$

Par la suite, on notera par P la clause $P_1 \vee \ldots \vee P_p$, par N la clause $\neg N_1 \vee \ldots \vee \neg N_n$, et par $P = N$ le problème d'unification ci-dessus.

Lemme 10.3 *(Correction) Si $[\forall X \;\; A \vee C]_I = [\forall Y \;\; \neg A \vee C]_I = T$, alors $[\forall Z \;\; C\sigma \vee D\sigma]_I = T$.*

Preuve
Analogue au cas clos, avec utilisation du lemme 9.5. □

On montre maintenant un lemme de relèvement, qui va permettre de ramener la complétude du cas avec variables à la complétude du cas clos.

Lemme 10.4 *Soeint $A \vee C\gamma$ et $\neg A \vee D\eta$ deux instances closes (en forme normale disjonctive) des clauses $P \vee C$ et $N \vee D$, telles que les atomes de $P\gamma$ soient tous égaux à A et ceux de $N\eta$ à $\neg A$. Alors $P = N$ possède un plus grand unificateur σ tel que $C\sigma \vee D\sigma \in Res(P \vee C, N \vee D)$, et $C\gamma \vee D\eta = (C\sigma \vee D\sigma)\tau$ pour une certaine substitution τ*

Preuve
Comme on peut toujours supposer que les clauses $P \vee C$ et $N \vee D$ ont des variables toutes distinctes, on en déduit que la substitution close $\gamma \cup \eta$ égale à γ pour les variables libres de $P \vee C$ et à η pour celles de $N \vee D$ unifie le problème $P = N$. Soit σ l'unificateur principal du problème, et τ la substitution telle que $\gamma \cup \eta = \sigma\tau$. On vérifie sans peine la propriété annoncée. □

Theorem 10.5 *(Robinson) Un ensemble S de clauses est insatisfiable ssi la clause vide appartient à $Res^*(S)$.*

Preuve
Elle utilise le second théorème de Herbrand, la complétude dans le cas clos et le lemme précédent.

Par le second théorème de Herbrand, S est insatisfiable ssi cela est le cas d'un ensemble fini d'instances closes de S (en en prenant par exemple une par feuille de l'arbre sémantique clos). On peut donc inférer la clause vide par résolution (close) à partir de ces clauses closes. Mais le lemme de relèvement nous dit que toute inférence sur des instances de clauses d'un ensemble S fournit une instance d'une clause obtenue par inférence à partir de clauses de l'ensemble S. La clause vide ne pouvant être l'instance que d'elle même, on en déduit le résultat. Notons que l'utilisation du lemme de relèvement impose l'élimination par résolution de tous les atomes qui deviennent égaux dans l'unification. □

10.3 Résolution binaire

La règle de résolution que nous avons donné a le désavantage d'unifier des termes en nombre arbitraire plutôt que des couples de termes. Nous allons donc donner une nouvelle version de la résolution à l'aide de règles d'inférences dites binaires, appellées RFB, pouvant coder la résolution générale.

$$\text{Résolution Binaire : } \quad \frac{A \vee C \qquad \neg B \vee D}{C\sigma \vee D\sigma}$$

où σ est l'unificateur principal des atomes A et B.

$$\text{Factorisation Binaire : } \quad \frac{A \vee B \vee C}{B\sigma \vee C\sigma}$$

où σ est l'unificateur principal des littéraux A et B.

Theorem 10.6 *Résolution et factorisation binaires sont correctes et complètes, cad un ensemble de clauses S est insatisfiable ssi la clause vide appartient à $RFB^*(S)$.*

Preuve
On montre que toute résolution se code comme une séquence de résolutions et factorisations binaires. □

10.4 Résolution négative

Une preuve par résolution consiste à explorer l'ensemble $Res^*(S)$ organisé généralement sous forme d'un arbre. Cet ensemble étant en général infini, et la clause vide pouvant se trouver loin de la racine, il est essentiel d'en réduire la taille d'une part, et d'adopter des stratégies d'exploration efficaces d'autre part. Pour cela, une méthode consiste à contraindre l'application de la règles de résolution. Parmi les nombreuses restrictions de la résolution, l'une d'elle joue un rôle essentiel, la résolution négative, pour laquelle au moins l'une des deux clauses de départ ne contient que des littéraux négatifs.

$$\text{Résolution négative :} \quad \frac{P \vee C \qquad N \vee D}{C\sigma \vee D\sigma}$$

où σ est l'unificateur principal du problème $P = N$, et la clause $N \vee D$ ne contient que des littéraux négatifs.

Theorem 10.7 *La résolution négative est correcte et complète.*

Preuve
Le lemme de relèvement s'appliquant sans changement, il suffit de prouver la complétude dans le cas clos. Pour cela, on reprend la preuve précédente en précisant le noeud J. Pour cela, on considère un chemin particulier dans l'arbre sémantique clos de $Res^*(S)$, appellé chemin gauche, qui consiste à descendre toujours à gauche dans l'arbre, sauf si le successeur gauche est un noeud d'échec, auquel cas on descend à droite. On appelle $J_1, \ldots, J_n$ les noeuds du chemin où la descente s'est poursuivie à droite, A_{i_k} l'atome interprété dans les interprétations partielles successeurs de J_k, et C_k la clause réfutée par l'interprétation fils gauche de J_k. Comme précédemment, le noeud d'échec R fils droit de J_n est étiquetté par une clause de la forme $A_{i_n} \vee C$, et le noeud d'échec L fils gauche de J_n est étiquetté par une clause de la forme $\neg A_{i_n} \vee D$. On va maintenant montrer que C peut être choisie purement négative, et on pourra alors faire une résolution négative à partir de ces deux clauses, entrainant une contradiction.

Pour cela, on remarque que les seuls atomes positifs pouvant apparaître dans C sont ceux qui sont interprétés en F dans l'interprétation L, cad $A_{i_1}, \ldots, A_{i_{n-1}}$. On utilise maintenant au plus $n^2/2$ inférences négatives entre les clauses C_i de manière à faire apparaitre des clauses négatives à la place des clauses C_i. On peut ensuite éliminer les atomes positifs de $\neg A_{i-n} \vee C$ par au plus n résolutions négatives avec les clauses obtenues. $\Box$

On peut bien sûr se restreindre à une règle de résolution négative binaire, pourvu que l'on conserve la règle de factorisation binaire.

La stratégie dite *input* résout toujours la dernière clause engendré par résolution avec une clause de l'ensemble S de départ. Cette stratégie a la propriété de ne pas faire exploser l'espace de recherche, elle a donc une importance fondamentale. Elle est malheureusement incomplète, comme on pourra s'en convaincre avec l'exemple $S = \{A \vee B,\ A \vee \neg B,\ \neg A \vee B,\ \neg A \vee \neg B\}$.

10.5 Clauses de Horn et stratégie input

Définition 10.8 *Une clause de Horn est une clause comportant au plus un littéral positif.*

Une remarque essentielle est que le résultat d'une inférence négative mettant en jeu des clauses de Horn est une clause de Horn.

L'étape suivante consiste à préciser ce que sont un programme et une requête en logique de Horn.

Définition 10.9 *Un* programme *en logique de Horn est un ensemble fini de clauses comportant chacune exactement un littéral positif.*

Une requête *est un ensemble fini (disjonction) de littéraux négatifs.*

On en déduit facilement:

Theorem 10.10 *La résolution négative input est complète pour les programmes en logique de Horn.*

Preuve
Il suffit de remarquer que la règle de factorisation est inutile dans le cas des clauses de Horn, et que la règle de résolution négative est en même temps input. □

En pratique, étant donnée une requête $R_1, \ldots, R_p$ (donc chaque R_i est un littéral négatif), la stratégie négative input consiste à unifier l'un des R_i avec le littéral positif H d'une clause $H \vee A_1 \vee \ldots \vee A_n$ du programme. Si σ est l'unificateur principal de H et R_i, on infère $A_1\sigma \ldots, A_n\sigma$ qui peuvent être considérés comme de nouvelles requêtes. Comme, par la stratégie, la clause but qui vient d'être engendrée est toujours une des prémisses de la nouvelle inférence, le but R_i peut être effacé. Ceci revient ainsi à réécrire le but (ou plutôt un des littéraux du but) en utilisant une des clauses du programme, vu comme une réécriture de la tête (littéral positif) dans le corps (littéraux négatifs).

On appelle *fonction de sélection* un ordre total sur les littéraux négatifs qui spécifie le but à réécrire en priorité: on résoud toujours sur le but minimal. Cette stratégie est complète (i.e. à chaque étape il suffit de considérer un seul but). Par contre, lors de la recherche d'une preuve, toutes les clauses permettant de réécrire ce but devront être considérées.

La règle de résolution avec une fonction de sélection et pour les programmes logiques est appelée *SLD-résolution.* Des considérations ci-dessus il résulte que:

Theorem 10.11 *La SLD-résolution est complète.*

Enfin, en programmation logique, on s'intéresse aussi au *résultat* d'une requête et, pour des raisons d'efficacité, on ne souhaite pas appliquer les substitutions. Pour ces raisons, on considère des *clauses contraintes.* Une clause contrainte est une paire formée d'une clause C et d'un problème d'unification ϕ et notée $C \mid \phi$. Une telle clause contrainte représente en fait l'ensemble des clauses $\{C\sigma \mid \sigma \models_{T(F)} \phi\}$.

($\sigma \models_{T(F)} \phi$ si σ est une substitution close des variables libres de ϕ et $\phi\sigma$ est valide dans la théorie des termes finis). En général, on remplace aussi $\models_{T(F)}$ par $\models_{RT(F)}$ afin d'éviter d'avoir à faire le test d'occurrence dans l'unification. Dans ce contexte, la règle de résolution peut s'écrire:

$$\frac{P(t) \vee C \qquad \neg B(u) \vee D \mid \phi}{C \vee D \mid \phi \wedge B(u) = P(t)}$$

Les variables de t sont supposées ne pas apparaître libres dans u. De plus les variables n'apparaissant pas dans le but initial et n'apparaissant plus non plus dans le but sont implicitement quantifiées existentiellement (ce qui reflète le fait qu'on ne s'intéresse qu'aux valeurs des variables du but initial).

La *réponse* à une requête $\neg B$ est alors la contrainte ψ telle que l'on ait inféré la clause $\Box \mid \psi$. La contrainte ψ est donnée en forme résolue. Ses solutions sont des substitutions σ telles que $P \models B\sigma$.

Le fait que les contraintes soient ici des équations ne joue de rôle que pour la mise en forme résolue et on peut donc employer d'autres systèmes de contraintes plus expressifs (c'est souvent le cas en pratique). De même les clauses du programme P peuvent être contraintes.

Exemple 10.12 *Soit le programme logique*

$$\begin{array}{lll} I(x, [], x.[]). & & \\ I(x, x.l, x.l). & & \\ I(x, y.l, y.l') & \leftarrow & I(x, l, l') \mid x \neq y \end{array}$$

Qui a pour but d'insérer un élément dans une liste sans doublons.

Soit la requête $I(0, s(0).(x.[]), y)$. On obtient deux dérivations possibles conduisant à $\Box \mid \psi$ avec une contrainte ψ satisfiable: la première est (par souci de simplicité, nous n'écrivons la contrainte complète que dans cette séquence, puis nous écrivons une contrainte simplifiée).

$$\begin{array}{lll} I(0, s(0).(x.[]), y) & \rightarrow & I(x_1, l, l') \mid 0 = x_1 \wedge y_1.l = s(0).(x.[]) \wedge y = y_1.l' \wedge x_1 \neq y_1 \\ & \rightarrow & \Box \mid \exists x_2, l', l'', \ \wedge y = s(0).l' \wedge x_2 = 0 \wedge x_2.l'' = x.[] \wedge x_2.l'' = l' \\ & \rightsquigarrow & \Box \mid x = 0 \wedge y = s(0).(0.[]) \end{array}$$

La deuxième est:

$$\begin{array}{llll} I(0, s(0).(x.[]), y) & \rightarrow & I(x_1, l, l') \mid & x_1 = 0 \wedge l = x.[] \wedge y = s(0).l' \\ & \rightarrow & I(x_2, l_1, l_1') \mid & \exists y_2, l, l', \ l = x.[] \wedge y = s(0).l' \wedge x_2 = 0 \\ & & & \wedge l = y_2.l_1 \wedge l' = y_2.l_1' \wedge x_2 \neq y_2 \\ & \rightarrow & \Box \mid & \exists x_2, l_1, l_1', y_2, l, l', \ l_1 = [] \wedge l_1' = x_2.[] \\ & & & \wedge l = x.[] \wedge y = s(0).l' \wedge x_2 = 0 \wedge l = y_2.l_1 \\ & & & \wedge l' = y_2.l_1' \wedge x_2 \neq y_2 \\ & \rightsquigarrow & \Box \mid & \wedge y = s(0).(0.(x.[]))) \wedge x \neq 0 \end{array}$$

Nous reformulons ci-dessous les résultats de correction et de complétude de la SLD-résolution en termes de requêtes et de programmes:

Proposition 10.13 *Si la réponse ψ à une requête R pour un programme logique P a pour mgu σ, alors, pour toute substitution close θ, $R\sigma\theta, P$ est insatisfiable.*

Proposition 10.14 *Si R est une requête pour un programme logique P et si $I\!R\theta, P$ est insatisfiable, alors il existe une réponse ψ à R telle que ψ a pour mgu σ et il existe une substitution τ telle que $\sigma\tau = \theta$.*

10.6 Plus petit modèle de Herbrand et sémantique des programmes logiques

Si P est un programme logique. On note $\mathcal{B}$ la base de Herbrand et $2^{\mathcal{B}}$ l'ensemble des parties de $\mathcal{B}$, que l'on confond aussi avec l'ensemble des interprétations de Herbrand (les éléments de $I \in 2^{\mathcal{B}}$ correspondent aux éléments de la base qui s'interprètent en vrai, les autres s'inteprétant à faux). On note M_P le "plus petit modèle de Herbrand de P": $M_P = \bigcap_{I \in 2^{\mathcal{B}}, I \models P} I$.

Exemple 10.15 *Soit P le programme:*

$$\begin{array}{c} Q(s(x)) \vee \neg Q(x) \\ P(0) \vee \neg Q(s(x)) \end{array}$$

Alors $M_P = \emptyset$ car l'interprétation vide (tout est faux) satisfait bien P.

Proposition 10.16 *Pour tout programme logique P, $M_P \models P$.*

Preuve
Il suffit de remarquer que l'intersection de modèles de Herbrand de P est aussi un modèle de Herbrand de P. $\Box$

Ainsi M_P consititue une interprétation "standard" du programme logique P. De façon classique, on peut aussi construire M_P à l'aide de l'opérateur de conséquence immédiate:

Si P est un programme logique, on note T_P l'*opérateur de conséquence immédiate* défini sur $2^{\mathcal{B}}$ par

$$T_P(I) = \{A \in \mathcal{B} \mid A \vee \neg A_1 \vee \ldots \vee \neg A_n \text{ est une instance close d'une clause de } P, \\ \text{et} A_1, \ldots, A_n \in I\}$$

Exemple 10.17 *Reprenons l'exemple 10.15. Alors $T_P(\emptyset) = \emptyset$, $T_P(\mathcal{B}) = \{P(0)\} \cup \{Q(s(t)) \ \ t \in T(\mathcal{F})\}$, $T_P(T_P(\mathcal{B})) = T_P(\mathcal{B})$.*

$2^{\mathcal{B}}$ est muni d'une structure de treillis complet pour l'ordre $\subseteq$ (i.e. l'ordre de prolongement des interprétations).

Proposition 10.18 *T_P est continu sur les interprétations de Herbrand.*

Preuve
Si $S \subseteq 2^{\mathcal{B}}$ est une partie dirigée de $2^{\mathcal{B}}$, alors $T_P(\bigcup_{I \in S} I) = \bigcup_{I \in S} T_P(I)$. En effet,

$$\begin{aligned} T_P(\bigcup_{I \in S} I) = \; & \{A \in 2^{\mathcal{B}} \mid A \vee \neg A_1 \vee \ldots \vee \neg A_n \text{ est une instance close} \\ & \text{d'une clause de } P \text{ et } A_1, \ldots, A_n \in \textstyle\bigcup_{I \in S} I\} \\ = \; & \{A \in 2^{\mathcal{B}} \mid \exists I \in S, A \vee \neg A_1 \vee \ldots \vee \neg A_n \text{ est une instance close} \\ & \text{d'une clause de } P \text{ et } A_1, \ldots, A_n \in I\} \end{aligned}$$

□

Il est possible de généraliser la définition de T_P aux clauses arbitraires :

$$T_P(I) = \{A \in \mathcal{B} \mid A \vee C \text{ est une instance close d'une clause de } P \text{ et } I \models \neg C\}$$

Mais dans ce cas T_P n'est plus nécessairement continu.

Exemple 10.19 *Supposons P réduit à la seule clause $A \vee B$. $T_P(\emptyset) = \{A, B\}$ et $T_P(\{A, B\}) = \emptyset$, $T_P(\{A\}) = \{A\}$ et $T_P(\{B\}) = \{B\}$. Par exemple pour la partie dirigée $S = \{\emptyset, \{A\}\}$, $T_P(\bigcup_{I \in S} I) = T_P(\{A\}) = \{A\} \neq \{A, B\} = T_P(\emptyset) = \bigcup_{I \in S} T_P(I)$.*

Proposition 10.20 *Soit I une interprétation de Herbrand et P un programme logique. I est un modèle de P si et seulement si $T_P(I) \subseteq I$.*

Preuve
$I \models P$ ssi pour toute instance close $A \vee \neg A_1 \vee \ldots \vee \neg A_n$ d'une clause de P, $I \models A_1, \ldots, A_n$ implique $I \models A$. □

Theorem 10.21 *Si P est un programme logique, $M_P = T_P^{\omega}(\emptyset)$ est le plus petit point fixe de T_P.*

Preuve
M_P est la borne inférieure des modèles de Herbrand de P, d'après la proposition précédente, c'est aussi la borne inférieure des interprétations de Herbrand I telles que $T_P(I) \subseteq I$. M_P étant un modèle de P, $T_P(M_P) \subseteq M_P$. D'autre part, d'après la proposition 10.18 on a aussi $T_P^2(M_P) \subseteq T_P(M_P)$ et donc $T_P(M_P) \models P$ par la proposition 10.20. Par minimalité de M_P on a donc $T_P(M_P) = M_P$ et M_P est donc bien un point fixe de T_P. Mais tout point fixe de T_P est un modèle de P par la proposition 10.20 donc tout point fixe de T_P contient M_P par minimalité de M_P. Il en résulte que M_P est bien le plus petit point fixe de T_P. □

De façon générale, lorsque T_P est monotone, T_P^{α} pour un ordinal α est défini par récurrence transfinie par: $T_P^0 = \emptyset$, si $T_P^{\alpha+1} = T_P(T_P^{\alpha})$ et, si α est un ordinal limite, T_P^{α} est la borne supérieure des T_P^{β} pour $\beta < \alpha$.

10.7 Exercices

1. La stratégie *unitaire* consiste à ne considérer que des résolutions dont une des prémisses est une clause réduite à un littéral.

   Montrer que la stratégie unitaire est en général incomplète. (Autrement dit, donner un ensemble de clauses insatifiable tel que factorisation + résolution unitaire ne permet pas de dériver la clause vide).

   Plus généralement, montrer qu'il existe une réfutation d'un ensemble S de clauses par la résolution unitaire si et seulement si il existe une réfutation de S par la stratégie "input".

   La stratégie unitaire est donc complète pour les clauses de Horn. Est-ce une stratégie intéressante ?

2. Donner un ensemble de clauses tel que T_P^{ω} n'est pas un point fixe alors que $T_P^{2\times\omega}$ est un point fixe.

Chapter 11

Théorie des arbres finis ou infinis

Nous avons vu le treillis des termes au chapitre 8. Nous avons vu qu'on peut ainsi calculer l'intersection de deux ensembles infinis de termes représentés par des instances d'un terme: $[\![t]\!] \cap [\![u]\!] = [\![t\sigma]\!]$ si σ est un plus général unificateur de t et u. Une autre opération est fondamentale: le complémentaire. Par exemple, dans le cas d'un

** Introduction: opération sur les ensembles de termes, transformation de programmes fonctionnels ou logiques **

11.1 Axiomes des algèbres de termes

** Cas des termes finis, cas d'un alphabet fini ou non

11.2 Formules équationnelles

** regles de normalisation

11.3 Formes résolues

11.4 Règles de transformation

11.5 Terminaison et complétude

Chapter 12

Langages de termes et automates d'arbres

Le but de ce chapitre est de décrire des ensembles de termes clos par des outils de théorie des langages, plus précisément les automates ascendants d'arbres.

12.1 Automates ascendants d'arbres

Définition 12.1 *Un* automate fini non déterministe ascendant d'arbres *ou* AFA *est un quadruplet $\mathcal{A} = (\mathbf{Q}, \mathcal{F}, \mathbf{Q}_f, \Delta_{\mathcal{A}})$, où $\mathbf{Q}$ est un ensemble de symboles unaires appellés* états, *$\mathcal{F}$ est le* vocabulaire *(disjoint de $\mathbf{Q}$) des symboles de fonctions, $\mathbf{Q}_f \subseteq Q$ est le sous-ensemble des états* acceptants *ou* finaux, *et $\Delta_{\mathcal{A}}$ est l'ensemble des règles de transition, qui sont des règles de réécriture de la forme*

$$\begin{array}{rcl} f(q_1(x_1), \ldots, q_n(x_n)) & \rightarrow & q(f(x_1, \ldots, x_n)) \\ & \text{où} & \left\{ \begin{array}{l} f \in \mathcal{F}_n \\ q_1, \ldots, q_n, q \in \mathbf{Q} \\ x_1, \ldots, x_n \in \mathcal{X} \text{ avec } x_i \neq x_j \text{ for } i \neq j \end{array} \right. \\ q & \rightarrow & q' \quad \text{(transition vide)} \end{array}$$

Étant donné un terme (clos) t de $\mathcal{T}(\mathcal{F})$, un calcul *est une dérivation de la forme $t \longrightarrow^*_{\Delta_{\mathcal{A}}} q(t)$. Un calcul est* réussi *si $q \in \mathbf{Q}_f$. Le langage $L(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \longrightarrow^*_{\Delta_{\mathcal{A}}} q(t)$ avec $q \in \mathbf{Q}_f\}$ est dit* reconu *par l'automate $\mathcal{A}$. Deux automates $\mathcal{A}$ et $\mathcal{A}'$ sont dits* équivalents *s'ils reconnaissent le même langage.*

Notons que $\Delta_{\mathcal{A}}$ possède la propriété de terminaison si et seulement si le sous-système des transitions vides termine, c'est-à-dire s'il n'y a pas de boucle dans le graphe des transitions vides.

Exemple 12.2 *$\mathcal{F}_0 = \{a\}$, $\mathcal{F}_1 = \{g\}$, $\mathcal{F}_2 = \{f\}$, $\mathbf{Q} = \{q_a, q_g, q_f\}$, $\mathbf{Q}_f = \{q_f\}$, et $\Delta = \{a \rightarrow q_a(a),\ g(q_a(x)) \rightarrow q_g(g(x)),\ g(q_g(x)) \rightarrow q_g(g(x)),\ f(q_g(x), q_g(g(y)) \rightarrow q_f(f(x,y)))\}$. Suivent successivement une dérivation qui n'est pas un calcul, un calcul qui échoue, et un calcul réussi.*

$$f(a,a) \underset{\Delta}{\longrightarrow} f(q_a(a), a) \underset{\Delta}{\longrightarrow} f(q_a(a), q_a(a))$$

$$g(a) \xrightarrow[\Delta]{} g(q_a(a)) \xrightarrow[\Delta]{} q_g(g(a))$$

$$f(g(a), g(a)) \xrightarrow[\Delta]{2} f(q_a(a), q_a(a)) \xrightarrow[\Delta]{2} f(q_g(g(a)), q_g(g(a))) \xrightarrow[\Delta]{} q_f(f(g(a), g(a)))$$

Les automates ascendants d'arbres n'ont pas d'état initial, ce sont en fait les constantes de $\mathcal{F}_0$ qui en tiennent lieu, puisque les transitions non vides associées sont de la forme $c \to q_c(c)$.

Il est possible de considérer les états comme des symboles de constantes, et dans ce cas, les transitions non vides sont des règles closes de la forme $f(q_1, \ldots, q_n) \to q$. La présentation précédente, qui permet de ne pas détruire le terme réécrit, s'avère nécessaire pour la pluspart des généralisations, en particulier pour les automates à contraintes. Par la suite, nous adopterons la présentation simplifiée, qui, pour notre exemple, aurait donné :

Exemple 12.3 $\Delta = \{a \to q_a,\ g(q_a) \to q_g,\ g(q_g) \to q_g,\ f(q_g, q_g) \to q_f\}$. *Les calculs deviennent :*

$$f(a, a) \xrightarrow[\Delta]{} f(q_a, a) \xrightarrow[\Delta]{} f(q_a, q_a)$$

$$g(a) \xrightarrow[\Delta]{} g(q_a) \xrightarrow[\Delta]{} q_g$$

$$f(g(a), g(a)) \xrightarrow[\Delta]{2} f(q_a, q_a) \xrightarrow[\Delta]{2} f(q_g, q_g) \xrightarrow[\Delta]{} q_f$$

Définition 12.4 *Un automate ascendant d'arbres $\mathcal{A}$ est dit*

- réduit *si tous les états sont accessibles, c'est-à-dire si pour tout état $q \in \mathbf{Q}$, il existe un terme $t \in \mathcal{T}(\mathcal{F})$ tel que $t \longrightarrow^*_{\Delta_\mathcal{A}} q(t)$,*

- déterministe *ou AFAD, si tout terme clos possède au plus un calcul. Une condition suffisante (équivalente dans le cas où l'automate est réduit) est qu'il ne possède pas de transition vide ni deux règles de même membre gauche à renommage près de leur variables.*

- complet *si tout terme clos possède au moins un calcul. Une condition suffisante (équivalente dans le cas où l'automate est réduit) est que pour tout $n \in I\!N$, pour tout $f \in \mathcal{F}_n$, pour tous $q_1, \ldots, q_n \in \mathbf{Q}$, il existe $q \in \mathbf{Q}$ et une règle $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ dans $\Delta_\mathcal{A}$.*

L'automate de l'exemple 12.2 est réduit, déterministe, mais incomplet.

Par la suite, nous ne considererons que des automates réduits. Énonçons deux propriétés simples :

Lemme 12.5 *Si $\mathcal{A}$ est un automate complet déterministe, alors tout terme clos possède un calcul unique.*

Si $\mathcal{A}$ est un automate incomplet, alors il existe un automate complet équivalent $\mathcal{A}'$ qui est déterministe si $\mathcal{A}$ l'est.

Les réciproques des propriétés précédentes nécessitent que $\mathcal{A}$ soit réduit.

12.2 Réduction du non déterminisme

Lemme 12.6 *Pour tout AFA avec transitions vides, il existe un AFAD sans transitions vides équivalent.*

La preuve est routinière. Elle consiste, comme dans le cas des mots, à prendre pour nouveaux états les clôtures des états de départ par les transitions vides, et à copier les transitions entre les états de départ sous forme de transitions entre leur clôtures.

Théorème 12.7 *Pour tout AFA, il existe un AFAD équivalent obtenu en temps non déterministe linéaire.*

On commence par éliminer les transitions vides, puis on déterminise de la manière habituelle en passant aux parties de l'ensemble des états.

12.3 Pompage

Théorème 12.8 *Pour tout automate fini ascendant d'arbre $\mathcal{A}$, il existe une constante $K_{\mathcal{A}} > 0$ telle que, pour tout terme clos $t \in L(\mathcal{A})$ de hauteur supérieure à $K_{\mathcal{A}}$, il existe des contextes clos $U[]_p$ et $V[]_q$ et un terme clos w tels que :*
- $\mid p \mid < K_{\mathcal{A}}$,
- $0 < \mid q \mid \leq K_{\mathcal{A}}$,
- $\forall n \in I\!N,\ U[V^n[w]] \in L(\mathcal{A})$.

La preuve est la même que d'habitude, relativement à une branche du terme t de hauteur supérieure à $K_{\mathcal{A}}$.

Soit $\mathcal{F}$ contenant un symbole binaire f. On déduit aisément du lemme précédent que le langage d'arbre $\{f(t,t) \mid t \in \mathcal{T}(\mathcal{F})\}$ n'est pas reconnaissable. Il suffit en effet de pomper un tel terme de hauteur quelconque pour engendrer un terme qui ne fasse pas partie du langage.

On en déduit aussi la propriété essentielle suivante :

Corollaire 12.9 *Le vide d'un AFA est décidable.*

Preuve
En effet, d'après le résultat de pompage, il suffit de tester tous les termes clos de hauteur au plus égale au nombre d'états. Une méthode beaucoup plus efficace consiste à nettoyer l'automate en éliminant les états inutiles (non productifs ou non atteignables). □

12.4 Clôture par les opérations Booléennes

Théorème 12.10 *La classe des langages d'arbres reconnaissables est close par les opérations Booléennes.*

Preuve
On fait la preuve pour l'union, l'intersection et la complémentation, en utilisant des transformation qui préservent la propriété pour un automate d'être déterminisme et complet. Soient $\mathcal{A}' = \{\mathbf{Q}', \mathcal{F}, \mathbf{Q}'_f, \Delta'\}$ et $\mathcal{A}" = \{\mathbf{Q}", \mathcal{F}, \mathbf{Q}"_f, \Delta"\}$ deux AFA complets. On définit l'ensemble $Q = Q \times Q'$ des paires d'états, et le système de règles $\Delta = \{f((q'_1, q"_1), \dots, (q'_n, q"_n)) \to (q', q") \mid f(q'_1, \dots, q'_n) \to q'$ et $f(q"_1, \dots, q"_n) \to q"\}$.

On vérifie alors aisément que les automates $(\mathbf{Q}, \mathcal{F}, Q'_f \times Q", \Delta)$ et $(\mathbf{Q}, \mathcal{F}, \mathbf{Q}'_f \times \mathbf{Q}"_f, \Delta)$ reconnaissent respectivement les langages $\text{Ł}(\mathcal{A}') \cup \text{Ł}(\mathcal{A}")$ et $\text{Ł}(\mathcal{A}') \cap \text{Ł}(\mathcal{A}")$, et sont déterministes si et seulement si $\mathcal{A}'$ et $\mathcal{A}"$ le sont tous deux.

Pour l'automate complémentaire de $\mathcal{A}'$, on procède comme d'habitude en échangeant états acceptants et non acceptants. □

12.5 Clôture par homomorphisme

La situation va différer du cas des mots.

Définition 12.11 *Soient $\mathcal{F}$ et $\mathcal{F}'$ deux alphabets de symboles de fonctions, non nécessairement disjoints. À chaque $n > 0$, on associe un alphabet de variables $\mathcal{X}_n = \{x_1, \dots, x_n\}$ disjoint de $\mathcal{F}$ et $\mathcal{F}'$. Soit $h_{\mathcal{F}}$ une application qui à un symbole $f \in \mathcal{F}_n$ associe un terme de $\mathcal{T}(\mathcal{F}', \mathcal{X}_n)$. L'*homomorphisme d'arbres *$h : \mathcal{T}(\mathcal{F}) \to \mathcal{T}(\mathcal{F})'$ défini par $h_{\mathcal{F}}$ est l'unique application qui vérifie*

$$h(f(t_1, \dots, t_n) = h_F(f)\{x_1 \mapsto h(t_1), \dots, x_n \mapsto h(t_n)\}$$

L'homomorphisme h sera dit linéaire *si $h_{\mathcal{F}}(f)$ est un terme linéaire de $\mathcal{T}(\mathcal{F})'$ pour tout $f \in \mathcal{F}$.*

Exemple 12.12 *$\mathcal{F}_0 = \{a, b\}$, $\mathcal{F}_3 = \{f\}$, $\mathcal{F}'_0 = \{a, b\}, \mathcal{F}'_2 = \{f\}$, $h_{\mathcal{F}}(a) = a, h_{\mathcal{F}}(b) = b$, and $h_{\mathcal{F}}(g) = f(x_1, f(x_2, x_3))$. Soit $t = g(a, g(b, b, b), a)$. Alors $h(t) = f(a, f(f(b, f(b, b)), a))$.*

Les homomorphismes d'arbres ne préservent par la reconnaissabilité. En effet, il suffit de prendre les alphabets $\mathcal{F} = \mathcal{F}' = \{f(,), a\}$, et l'homomorphisme h engendré par $h_{\mathcal{F}}(a) = a, h_{\mathcal{F}}(f) = f(x_1, x_1)$. L'homomorphisme h transforme le langage reconnaissable $\{f(t, t') \mid t, t' \in \mathcal{T}(\mathcal{F})\}$ en le langage $\{f(t, t) \mid t \in \mathcal{T}(\mathcal{F})\}$ dont on a vu qu'il ne l'est pas. Toutefois,

Théorème 12.13 *La classe des langages d'arbres reconnaissables est close par homomorphisme linéaire d'arbres, et par homomorphisme inverse arbitraire.*

La preuve est un exercice non trivial pour la reconnaissabilité par homomorphisme linéaire. Elle est standard pour les homomorphismes inverses.

12.6 Clôture par normalisation

La reconnaissabilité Le résultat essentiel de ce paragraphe est que la reconnaissabilité est préservée par normalisation par un système de réécriture linéaire (droit et gauche).

En fait, ce qui s'avère intéressant en pratique est que le langage des termes clos en forme normale pour un système de règles linéaire gauche est reconnaissable. Les membres droit n'interviennent évidemment pas dans la définition des termes en forme normale, ce qui explique qu'aucune condition de linéarité ne soit requise pour eux. Nous montrons ce résultat à l'aide d'un lemme préliminaire simple, mais important.

Lemme 12.14 *Le langage L_t des termes dans lequel un terme t est imbriqué, est reconnaissable si et seulement si t est linéaire.*

Preuve
Soit $\mathrm{L}_t = \{u[t\gamma]\}$ où u désigne un contexte clos arbitraire et γ une substitution close arbitraire. On reconnait L_t avec l'automate non déterministe $(\mathbf{Q}, \mathcal{F}, \{q_f\}, \Delta)$, où

- $\mathbf{Q} = \{q_0, q_f\} \cup \{t|_p\tau \mid p \in \mathcal{FP}os(t)\}$, où τ désigne la substitution telle que $\forall x \in \mathcal{V}ar(t)\ x\tau = q_0$,

- $\Delta = \Delta_0 \cup \Delta_f \cup \Delta_t \cup \{t\tau \to q_f\}$, où Δ_0 est l'ensemble des règles d'un automate complet pour l'état q_0, $\Delta_f = \{f(q_1, \ldots, q_{i_1}, q_f, q_{i+1}, \ldots, q_n) \to q_f \mid \forall n \in \mathbb{N}\ \forall f \in \mathcal{F}_n\ \forall i \in [1..n]\ \forall q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \in \mathbf{Q}\}$, et $\Delta_t = \{f(q_1, \ldots, q_n) \to q \mid \forall q = f(q_1, \ldots, q_n) \in \mathbf{Q}\}$.

Pour la réciproque, on sait déja que la condition de linéarité est essentielle, puisque l'on a vu que le langage $\{f(t,t) \mid t \in \mathcal{T}(\mathcal{F})\}$ n'est pas reconnaissable. De manière plus générale, cela est vrai de tout terme t non linéaire par le même argument. □

On en déduit maintenant le théorème :

Théorème 12.15 *Le langage des formes normales closes d'un système de réécriture linéaire gauche fini est reconnaissable.*

Preuve
Soit $\mathbb{R} = \{l_i \to r_i\}$. À chaque membre gauche l_i, on associe l'automate $\mathcal{A}_{l_i}$ qui reconnait les termes réductibles par la ième règle. On conclue par union, puis complémentation.

Ce résultat a de nombreuses applications importantes, en particulier en démonstration automatique.

Définition 12.16 *Étant donné un système de réécriture R, un terme t est dit inductivement réductible par R si toutes ses instances closes le sont.*

Corollaire 12.17 *La réductibilité inductive d'un terme linéaire t est décidable pour les systèmes de règles linéaires gauches.*

Preuve
En effet, un terme t est inductivement réductible si le langage de ses instances a une intersection non vide avec le langage des formes normales pour le système de règles, ce qui est décidable. $\square$

En fait, on peut montrer que le résultat reste vrai pour les termes non linéaires. On peut aussi montrer qu'il est vrai pour tout système de règles, linéaire gauche ou pas, mais cela nécessite l'utilisation d'automates à contraintes qui sortent du cadre de ce cours.

Une autre application, intimement liée à la précédente, consiste à "libérer" une algèbre quotient par un système de réécriture R linéaire gauche et convergent. En effet, pour de tels systèmes, l'algèbre quotient est isomorphe à l'algèbre des formes normales. L'ensemble des formes normales étant reconnaissable, on va interpréter les états de l'automate comme des sortes, et ses transitions comme des déclarations de profil des opérations. Cela nous fournit une nouvelle algèbre qui est isomorphe à la précédente. L'avantage de cette nouvelle algèbre est qu'elle est librement engendrée, et que l'on peut donc raisonner aisément par récurrences structurelle. Cette technique a des applications, à la démonstration automatique d'une part, mais aussi en théorie des types, pour la génération des règles d'élimination pour les types inductifs quotient.

Nous présentons ci-dessous un exemple simple, la spécification classique des entiers par prédecesseur et successeur. La construction de l'automate de formes normales (par un algorithme plus efficace que celui esquissé plus haut) nous donne l'automate de la figure 12.1, dont tous les états sont acceptant excepté Int. Nous utilisons la convention que les flèches étiquettées par des ϵ sont des transitions vides.

```
obj INTEGERS
sorts Nat<Int
op S,P : Int → Int
op S : Nat → Nat
op 0 : Nat
eq P(S(x)) → x if x : Int
eq S(P(x)) → x if x : Int
end fmod
```

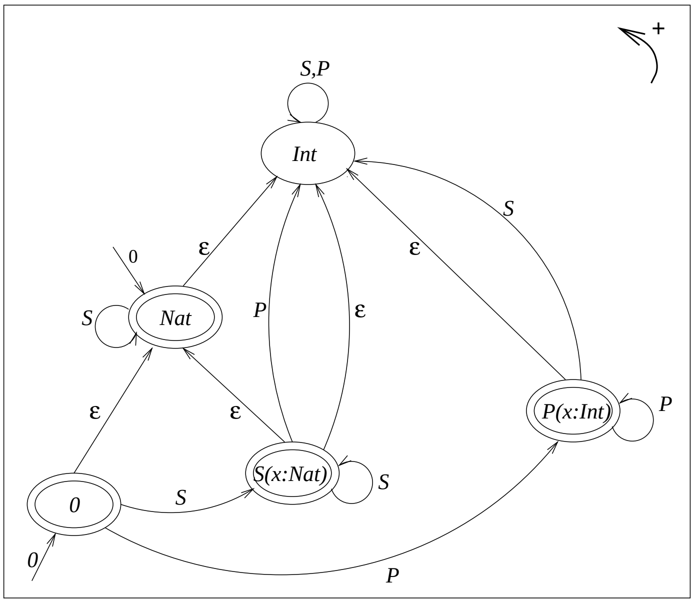

Figure 12.1: Une spécification avec sortes ordonnées des entiers et son automate associé

Bibliography

[1] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3), July 1982.

[2] Hubert Comon. Disunification: a survey. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1991.

[3] Hubert Comon. Constraints in term algebras (short survey). In T. Rus M. Nivat, C. Rattray and G. Scollo, editors, *Proc. Conf. on Algebraic Methodology and Software Technology*, Workshop in Computing, Univ. of Twente, 1993. Springer-Verlag. Invited talk.

[4] Max Dauchet. Rewriting and tree automata. In Hubert Comon and Jean-Pierre Jouannaud, editors, *Term Rewriting*, volume 909 of *Lecture Notes in Computer Science*. French Spring School of Theoretical Computer Science, Springer-Verlag, 1994.

[5] Nachum Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, March 1982.

[6] Nachum Dershowitz. Trees, ordinals and termination. In *Proc. CAAP 93, LNCS 668*, 1993.

[7] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[8] Gérard Huet and D. Oppen. Equations and rewrite rules: a survey. In R. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405. Academic Press, 1980.

[9] Jean-Pierre Jouannaud. Rewrite proofs and computations. In Helmut Schwichtenberg, editor, *Proof and Computation*, volume 139 of *F: Computer and Systems Sciences*, pages 173–218. Springer-Verlag, 1995. NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 20 – August 1, 1993.

[10] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In Jean-Louis Lassez and Gor-

don Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT-Press, 1991.

[11] Jan Willem Klop. Term Rewriting Systems. In S. Abramsky, Dov.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Clarendon Press, 1992.

[12] Robert A. Kowalski and M. H. van Emden. The semantics of predicate logic as a programming language. *J. of the Association for Computing Machinery*, 23:733–742, October 1976.

[13] Martin Wirsing. *Handbook of Theoretical Computer Science*, volume B, chapter Algebraic Specification, pages 675–780. North-Holland, 1990. J. van Leeuwen ed.