(9/9)

NOM : AUFORT Prénom : William Jury :

~~Algèbre~~ ← ~~Entourez l'épreuve~~ → ~~Analyse~~ Informatique

Sujet choisi : Unification : algorithmes et applications.

Autre sujet :

I] Unification

a) Termes, substitution et unification syntaxique

On se donne :

- Σ un ensemble de constantes d'arité donnée
- V un ensemble infini de variables.

Déf 1 : L'ensemble des termes $T(\Sigma, V)$ est défini par induction :

- $\forall x \in V,\ x \in T(\Sigma, V)$
- $\forall f \in \Sigma$ d'arité n, $\forall t_1, \dots t_n \in T(\Sigma, V)$, $f(t_1, \dots, t_n) \in T(\Sigma, V)$.

Exemple 2 : Si $V = \{x, y, z, \dots\}$ et $\Sigma = \{f, g\}$, $f(a, a)$, $f(f(x, a), x)$ sont des termes.

Déf 3 : Une substitution est un ensemble fini de couples $(x, t) \in V \times T(\Sigma, V)$ tel que toute variable apparaît au plus une fois comme première composante d'un couple.

Exemple : $(x, f(a, a))$ est une substitution.

Déf 4 : Une substitution σ est plus générale que σ' s'il existe une substitution ξ telle que $\sigma' = \xi \circ \sigma$.

On note $\sigma \leq \sigma'$.

syntaxique

Déf 5 : Un problème d'unification est un ens. fini d'équations syntaxiques entre termes : $\{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\} = S$

- Un unificateur de S est une substitution σ telle que $\sigma(s_i) = \sigma(t_i)$ $\forall i$. Ainsi, $\sigma(t)$ est le terme obtenu en remplaçant chaque variable de t par $\sigma(x) = \begin{cases} u \text{ si } (x,u) \in \sigma \\ x \text{ sinon.} \end{cases}$ d'où des unificateurs : $\mathcal{U}(S)$.
- Un unificateur le plus général de S est un unificateur $\sigma \in \mathcal{U}(S)$ tel que $\forall \sigma' \in \mathcal{U}(S)$ $\sigma \leq \sigma'$.

Exemple 6 : $x \stackrel{?}{=} f(y)$ admet comme unificateurs $\{(x, f(y))\}$, $\{(x, f(a)), (y, a)\}$, ... unificateur le plus général ($\sigma' = (x,a) \circ \sigma$)

Théorème 7 : Deux unificateurs les plus généraux sont égaux à un renommage près des variables.

b) Algorithme de Martelli-Montanari

Notations :

- $\forall (t_1, u_1), \dots (t_n, u_n) \in S$ désigne le problème d'unif. S de composant en $\{(t_1, u_1), \dots\}$ et S'
- $\mathcal{R}$ désigne les règles de transformation suivantes

Algorithme 8 (Martelli-Montanari)

On applique les règles de transformation suivantes tant qu'elles sont applicables :

$\{t =?\ t\} \uplus S \longrightarrow S$ (1)

$\{x =?\ t\} \uplus S \longrightarrow \{x =?\ t\} \cup S[t/x]$ (2)
si $x \notin Var(t)$

$\longrightarrow \bot$ sinon (3)

$\{f(t_1,\dots t_n) =?\ f(u_1,\dots,u_m)\} \uplus S$

$\longrightarrow \{t_1 =?\ u_1, \dots, t_m =?\ u_m\} \uplus S$ (4)

$\{f(t_1,\dots,t_n) = g(u_1 \dots u_p)\} \uplus S \longrightarrow \bot$ (5)

Proposition 10 : L'algorithme termine sur toute entrée.

Exemple 11 $\{f(x,x) \stackrel{?}{=} f(y, g(y))\}$

$\rightarrow \{x =?\ y ;\ x =?\ g(y)\}$ (4)

$\rightarrow \{x, y\} \cup \{y =?\ g(y)\}$ (2) $\longrightarrow \bot$ (3)

Proposition 12 : Si l'algorithme applique à S renvoit une substitution σ, alors σ est un unificateur le plus général de S. Inversement, si l'algorithme renvoit $\bot$, alors $\mathcal{U}(S) = \emptyset$.

c) Aspect complexité

Exemple 13 : Le problème d'unification S_n :

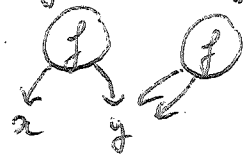
$\{x_1 \stackrel{?}{=} f(x_0, x_0), \dots, x_n \stackrel{?}{=} f(x_{n-1}, x_{n-1})\}$

a comme unificateur le plus général $x_i \mapsto t_i$ où t_i est un arbre binaire complet de feuilles, de hauteur i.

Il est donc de taille exponentielle par rapport au problème en entrée.

Notation 14 : On représente un problème d'unification en utilisant un graphe orienté sans cycle.

Exemple : $f(x,y) \stackrel{?}{=} f(y,y)$ représenté par



D2

Proposition 15 : En utilisant cette structure, une implémentation de l'algo de Martelli-Montanari existe en $O(m^2)$, où m est le nombre d'arêtes du graphe

Note : On peut améliorer l'algorithme précédent afin de le rendre quasi-linéaire (complexité $O(m.G(m))$ avec $G(n) \leq 5 \ \forall n \leq 2^{65536}$).

D1

II] Applications

a) Inférence de type en Caml

Des contraintes (ou équations) sont déduites à partir de règles de typage. Elles peuvent être vue comme un problème d'unification syntaxique.

Exemple 16 Typer :

let f = fun g b → if b then g 2 else (g b) + 2.

D3

b) Test de confluence basé sur les paires critiques

On considère ici un système de réécriture de termes T.

Def 17 : 2 termes u et v sont joignables s'il existe w tel que $u \to^* w \leftarrow^* v$.

Def 18 : R est localement confluent si $\forall t, u, v$, $u \leftarrow t \to v \Rightarrow u$ et v sont joignables.

Def 19 : R est confluent si $\forall t, u, v$ $u \leftarrow^* t \to^* v \Rightarrow u$ et v sont joignables.

Exemple 20 de système de réécriture Abstrait :

est localement confluent mais pas confluent.

Lemme 21 (Newman) : Une relation localement confluente et terminante est confluente.

Def 22 : On définit par induction sur la structure de $t \in T(\Sigma, V)$ l'ens des positions de t, noté $pos(t)$ qui sont des mots d'entiers :

$pos(x) = \{\varepsilon\}$

$pos(f(t_1,\dots t_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{iw ;\ w \in pos(t_i)\}$

D4

Exemple 23 $t = f(g(a), f(x,z))$
$pos(t) = \{\varepsilon, 1, 2, 11, 21, 22\}$.

Déf 24 Par induction sur t, on définit les sous-termes de t à partir des positions par :
- $t|_\varepsilon = t$
- $t = f(t_1 \dots t_m) \Rightarrow t|_{ij} = t_i|_j$

Ex 25 Dans l'ex 23, $t|_1 = g(a)$, $t|_{22} = x$

Déf 26 : On définit, pour $p \in pos(t)$, $u \in T(\Sigma, V)$, $t[u]_p$ le terme obtenu à partir de t en insérant u à la position p à la place de $t|_p$ par induction :
- $t[u]_\varepsilon = u$
- $f(t_1 \dots t_n)[u]_{ij} = t_i[u]_j$

Déf 27 On considère $l_1 \to r_1$ et $l_2 \to r_2$ deux règles de R renommées de telle sorte que $Var(l_1, r_1) \cap Var(l_2, r_2) = \emptyset$. Soit $p \in pos(l_1)$ tel que $l_2|_p$ n'est pas une variable. Soit σ un unificateur le plus général de $\{l_1|_p =^? l_2\}$. Alors, la paire : $(\sigma(r_1), \sigma(l_1)[\sigma(r_2)]_p)$ est une paire critique de R

Théorème 28 : Un système de réécriture de termes est confluent si et seulement si ses paires critiques sont jointables.

Références :
- Term Rewriting and All That, Baader, Nipkow. (disponible à la bibliothèque)
- Cours de théorie de la programmation de L3.