NOM : ROUHLING **Prénom :** Damien **Jury :**

Algèbre ← Entourez l'épreuve → **Analyse**

Sujet choisi : 919 - Unification : algorithmes et applications

Autre sujet : Réf : All That, DNR, Cormen, Stern, Fondements mathématiques de l'informatique

I Définitions préliminaires

1- Termes

Déf 1 : Une signature est un ensemble de symboles dits de fonctions, associés à un entier positif appelé arité. Les symboles d'arité 0 sont les constantes.

Ex 1 : $\Sigma_1 = \{c/0, f/1, g/2\}$, $\Sigma_2 = \{0/0, S/1, +/2\}$

Déf 2 : Si Σ est une signature et X un ensemble de variables (avec $\Sigma \cap X = \emptyset$), on note $T(\Sigma, X)$ l'ensemble des Σ-termes sur X défini inductivement par :

- $X \subseteq T(\Sigma, X)$
- $\forall n \geq 0, \forall f \in \Sigma$ d'arité n, $\forall t_1 \dots t_n \in T(\Sigma, X)$, $f(t_1 \dots t_n) \in T(\Sigma, X)$

Ex 2 : $g(f(x), c) \in T(\Sigma_1, \{x\})$, $S(x) + 0 \in T(\Sigma_2, \{x\})$

Rq 1 : les termes ont une représentation arborescente naturelle qui sera raffinée dans la suite (Figure 7)

Dans la suite, on fixe Σ une signature et X un ensemble de variables.

Déf 3 : On définit inductivement l'ensemble $Pos(t)$ des positions d'un terme $t \in T(\Sigma, X)$:

- $\forall x \in X \quad Pos(x) = \{\varepsilon\}$ (mot vide)
- Si $t = f(t_1 \dots t_n)$ $Pos(t) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{ip, p \in Pos(t_i)\}$

le sous-terme à la position p est noté $t|_p$:

- $t|_\varepsilon = t$
- $f(t_1 \dots t_n)|_{ip} = t_i|_p$

le terme obtenu en remplaçant $t|_p$ par t' est noté $t[t']_p$:

- $t[t']_\varepsilon = t'$
- $f(t_1 \dots t_n)[t']_{ip} = f(t_1 \dots t_{i-1}, t_i[t']_p, t_{i+1} \dots t_n)$

Ex 3 : Figures 1 et 2, $g(f(x), c)|_{11} = x$, $S(0)[S(0)]_1 = S(S(0))$

2- Substitutions

Déf 4 : Une substitution est une application $\sigma : X \to T(\Sigma, X)$ telle que $\{x \in X, \sigma(x) \neq x\}$ est fini. On note alors $Dom(\sigma)$ cet ensemble, $\sigma = \{x_1 \mapsto \sigma(x_1), \dots, x_n \mapsto \sigma(x_n)\}$ où $Dom(\sigma) = \{x_1 \dots x_n\}$

$Im(\sigma) = \{\sigma(x), x \in Dom(\sigma)\}$

$VIm(\sigma) = \bigcup_{t \in Im(\sigma)} Var(t)$ où $Var(t)$ est l'ensemble des variables de t.

Rq 2 : On étend $\sigma : X \to T(\Sigma, X)$ en $\hat{\sigma} : T(\Sigma, X) \to T(\Sigma, X)$ par :

- $\forall x \in X \quad \hat{\sigma}(x) = \sigma(x)$
- $\forall n \geq 0, \forall f \in \Sigma$ d'arité n, $\forall t_1 \dots t_n \in T(\Sigma, X)$, $\hat{\sigma}(f(t_1 \dots t_n)) = f(\hat{\sigma}(t_1) \dots \hat{\sigma}(t_n))$

On utilisera la représentation en Figure 3

Déf 5 : Une substitution σ est idempotente ssi $\sigma\sigma = \sigma$

Ex 4 : $\{x \mapsto y, y \mapsto c\}$ n'est pas idempotente

$\{x \mapsto f(y)\}$ est idempotente

Lemme 1 : Une substitution σ est idempotente ssi $Dom(\sigma) \cap VIm(\sigma) = \emptyset$

Déf 6 : Une substitution σ est plus générale qu'une substitution σ', noté $\sigma \leq \sigma'$, s'il existe une substitution δ telle que $\sigma' = \delta\sigma$

Ex 5 : $\{x \mapsto f(y)\} \leq \{x \mapsto f(c), y \mapsto c\}$

$\{x \mapsto f(y)\}$ et $\{y \mapsto f(x)\}$ sont incomparables

Lemme 2 : $\leq$ est un préordre

II Unification syntaxique

1_ Définition et existence

Déf 7: Un problème d'unification est un ensemble fini d'équations $S = \{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\}$ où $\forall i \in [\![1,n]\!]$ $(s_i \in T(\Sigma, X)$, $t_i \in T(\Sigma, X))$.
Une substitution σ est un unificateur pour S si $\forall i \in [\![1,n]\!]$ $\sigma(s_i) = \sigma(t_i)$. On note alors $\sigma \in U(S)$. (Figure 4)
$\sigma \in U(S)$ est un unificateur le plus général si $\forall \sigma' \in U(S)$ $\sigma \leq \sigma'$. On note alors $\sigma = mgu(S)$

Ex 6: $\{x \mapsto f(y)\} = mgu(x \stackrel{?}{=} f(y))$, $\{x \mapsto f(c), y \mapsto c\} \in U(x \stackrel{?}{=} f(y))$

Thm 1: Si un problème d'unification admet une solution, alors il admet un unificateur le plus général idempotent.

Développement 1

Démonstration: algorithme de Martelli-Montanari

Ex 7: Soit $(x_n)_{n \in \mathbb{N}} \in X^{\mathbb{N}}$. On définit :
- $u_0 = v_0 = x_0$
- $\forall n \in \mathbb{N}$ $u_{n+1} = f(u_n, x_n)$, $v_{n+1} = f(x_n, v_n)$

L'algorithme de Martelli-Montanari est exponentiel sur le problème $f(u_n, u_n) \stackrel{?}{=} f(v_n, v_n)$

2_ Rappels sur les graphes

Thm 2: L'acyclicité d'un graphe orienté est vérifiable en temps linéaire en la taille du graphe

Déf 8: Une structure d'union-find est une structure de données munie de deux opérateurs, représentant une partition d'un ensemble :
- Union réalise l'union de deux ensembles
- Find retourne le représentant de l'ensemble qui contient l'argument

Thm 3: Il existe une structure d'union-find basée sur les arbres telle qu'en utilisant la compression de chemins et l'union par rang (Figures 5 et 6), une séquence de m opérations pour n éléments est réalisée en temps $O(m\,\alpha(n))$, où α est une fonction croissante vérifiant $\alpha(2^{2^{\cdot^{\cdot^{2}}}}\}n) = n+1$

3_ Algorithme pseudo-linéaire

Idée:
- représenter les termes comme des arbres enracinés avec partage des sous-termes (Figure 7)
- construire des pointeurs des variables vers leurs instantiation
- vérifier l'acyclicité

Algorithmes:
- UnionVar (x, t) = faire pointer x vers t
- Unifier (s, t) =
  - $t_1 = \text{Find}(s)$, $t_2 = \text{Find}(t)$
  - Si $t_1 = t_2$ alors retourner vrai
  - Sinon, par cas sur $(t_1 \in X, t_2 \in X)$:
    - (vrai, vrai) $\to$ Union(t_1, t_2), retourner vrai
    - (faux, vrai) $\to$ UnionVar(t_2, t_1), retourner vrai
    - (vrai, faux) $\to$ UnionVar(t_1, t_2), retourner vrai
    - (faux, faux) $\to$ Si t_1 et t_2 ont même symbole à la racine alors noter $t_1 = f(u_1 \dots u_n)$, $t_2 = f(v_1 \dots v_n)$
      - retourner UnifierListes $([u_1 \dots u_n], [v_1 \dots v_n])$
      - sinon retourner faux
- UnifierListes (l_1, l_2) =
  - par cas sur $(l_1 = [\,], l_2 = [\,])$:
    - (vrai, vrai) $\to$ retourner vrai
    - (vrai, faux) ou (faux, vrai) $\to$ retourner faux
    - (faux, faux) $\to$ noter $l_1 = s :: l_1'$, $l_2 = t :: l_2'$
      - si Unifier (s, t) alors retourner UnifierListes (l_1', l_2')
      - sinon retourner faux

- Unification (t_1, t_2) =
  Si Unifier (t_1, t_2) alors retourner Acyclique (t_1)
  sinon retourner faux

Thm 4 : Si s et t sont unifiables, Unification (s,t) vaut vrai, et s et t sont unifiés en temps $O(m\,\alpha(n))$, où
- m est le nombre d'arcs dans la représentation de s et t
- n est le nombre de noeuds

Ex 8 : Figure 8 : le résultat de l'unification de $f(x, g(x,x))$ et $f(h(y), g(z, h(a)))$.

III Applications

1 - Réécriture

Déf 9 : • Une règle de réécriture est un couple $g \to d$ où g et d sont des termes et $Var(d) \subseteq Var(g)$
• Un système de réécritures de termes est un ensemble de règles de réécritures.
• Une relation $\to$ sur $T(\Sigma, X)$ est une relation de réécriture si elle est compatible avec le contexte et les substitutions, ie : si $s \to t$ alors $\begin{cases} f(t_1 \ldots s \ldots t_n) \to f(t_1 \ldots t \ldots t_n) \\ \sigma(s) \to \sigma(t) \end{cases}$

Rq 3 : Un système de réécriture de termes induit une relation de réécriture.

Déf 10 : Un système de réécriture est terminant s'il n'existe pas de chaîne infinie $t_1 \to t_2 \to \ldots$

Déf 11 : • Deux termes t_1 et t_2 sont joignables $(t_1 \downarrow t_2)$ s'il existe $t \in T(\Sigma, X)$ tel que $t_1 \to^* t$ et $t_2 \to^* t$
• Un système de réécriture est :
- confluent si $\forall s, t_1, t_2 \quad s \to^* t_1$ et $s \to^* t_2 \Rightarrow t_1 \downarrow t_2$
- localement confluent si $\forall s, t_1, t_2 \quad s \to t_1$ et $s \to t_2 \Rightarrow t_1 \downarrow t_2$

Déf 12 : Si $g_1 \to d_1$ et $g_2 \to d_2$ sont deux règles d'un système de réécriture, s'il existe $p \in Pos(g_1)$ tel que $g_1|_p$ n'est pas une variable et $\sigma = mgu(g_1|_p \stackrel{?}{=} g_2)$, on dit que $(\sigma(d_1), \sigma(g_1)[\sigma(d_2)]_p)$ forme une paire critique

Développement 2

Thm 5 : Un système de réécriture est localement confluent ssi toutes ses paires critiques sont joignables.

Cor 1 : La confluence d'un système de réécriture fini et terminant est décidable

2 - Logique

Déf 13 : Un littéral est une formule atomique ou sa négation. Une clause est un ensemble (disjonction) de littéraux.
La résolution vise à montrer qu'un ensemble (conjonction) de clauses est contradictoire, avec les règles suivantes :

$$\frac{C_1, L_1 \quad C_2, L_2 \quad \sigma = mgu(L_1, \bar{L}_2)}{\sigma(C_1), \sigma(C_2)}$$

$$\frac{C_1, L_1, L_2 \quad \sigma = mgu(L_1, L_2)}{\sigma(C_1), \sigma(L_1)}$$

L'objectif est d'aboutir à la clause vide.

Thm 6 : Une formule est contradictoire ssi la méthode de résolution dérive la clause vide à partir de sa forme clausale.

Rq 4 : Une autre méthode, la méthode des tableaux, ne nécessite pas le passage en forme clausale. L'unification est alors réalisée aux feuilles de l'arbre de preuve.

Figure 1 :

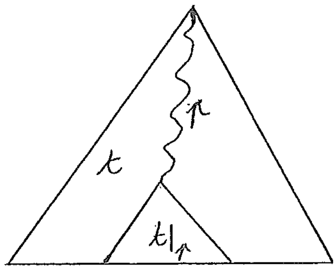


Figure 2 :

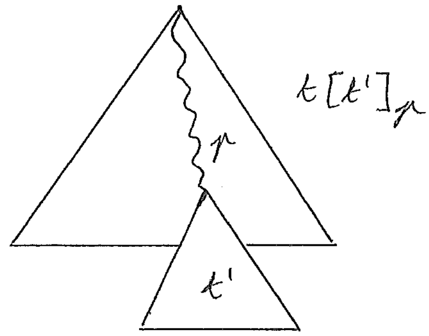


Figure 3 :

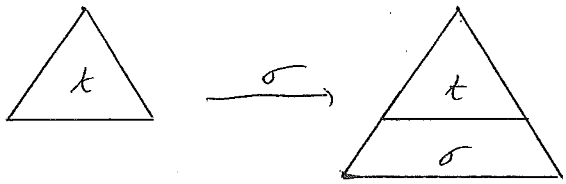
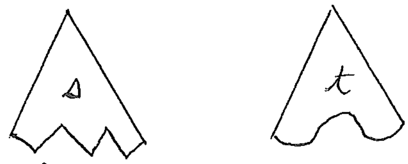


Figure 4 :



$\sigma \in U(s \stackrel{?}{=} t)$:

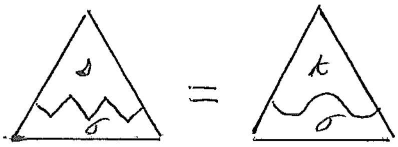


Figure 5 : Compression de chemin

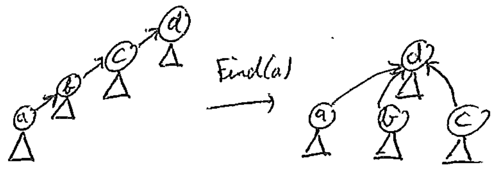


Figure 6 : Union par rang

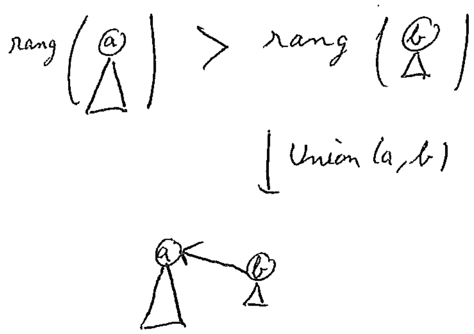


Figure 7 : $f(x, h(y))$ et $g(h(y), z)$

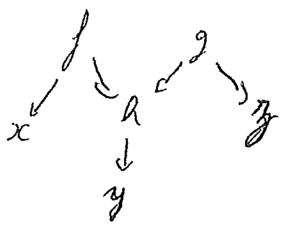


Figure 8 :