913. UNIFICATION : ALGORITHMES ET APPLICATIONS

I | Syntaxe et unification

1) Termes et substitutions

déf 1 Un langage du premier ordre L est la donnée d'un ensemble de symboles de variables $V = \{x_i\}_i$, d'un ensemble de symboles de constantes $C = \{c_i\}_i$, d'un ensemble de symboles de fonctions $\{f_i\}_i = F$ et d'un ensemble de symboles de prédicats $P = \{p_i\}_i$

déf 2 L'ensemble T des termes sur le langage L est le plus petit ensemble tel que $V \subset T$, $C \subset T$ et stable par l'application de symboles de F à des termes de T.

On dit que le terme t est clos si il ne contient aucune variable.

ex 3 $f_1(x_1, c_2, v_3)$ est un terme mais pas un terme clos

$f_1(f_2(c_1, c_3), c_1)$ est un terme clos.

déf 4 Une substitution σ est une application de V dans T

Dans la suite, on supposera que le domaine $D = \{x \in V \mid \sigma(x) \neq x\}$ de σ est fini et égal à $\{x_1, \dots, x_n\}$

notation 5 Si $\forall i \in [\![1,n]\!]$, $\sigma(x_i) = t_i$ on notera $\sigma = [x_1 := t_1, \dots, x_n := t_n]$

déf 6 La substitution σ appliquée au terme t est le terme noté $t[\sigma]$ obtenu en remplaçant (simultanément) chaque occurrence de x_i dans t par $\sigma(x_i)$ pour $i \in [\![1,n]\!]$

ex 7 Pour $t = f_1(x_1, c_2)$ et $\sigma = [x_1 := f_3(c_3)]$ on a $t[\sigma] = f_1(f_3(c_3), c_2)$

Si t est un terme clos, pour toute substitution σ, $t[\sigma] = t$

rq 8 Attention : $t[\sigma][\sigma'] = t[\sigma' \circ \sigma]$

2) Unification

déf 9 Deux termes t et t' sont unifiables s'il existe une substitution σ telle que $t[\sigma] = t'[\sigma]$. La substitution σ est alors appelée "unificateur de t et t'".

ex 10 $f_1(c_2, x_1)$ et $f_1(x_2, c_1)$ sont unifiables via $\sigma = [x_1 := c_1 ; x_2 := c_2]$

déf 11 Si S est un ensemble fini de couples $(t_i, t_i')_{1 \le i \le m}$ de termes (S est un ensemble d'équations), la substitution σ est un unificateur de S si $\forall i \in [\![1,m]\!]$, $t_i[\sigma] = t_i'[\sigma]$.

déf 12 Un unificateur σ de S est principal si pour tout unificateur α de S il existe une substitution β telle que $\alpha = \beta \circ \sigma$

prop 13 Si S admet un unificateur, il admet un unificateur principal

II | Algorithmes

rq : la taille de l'entrée (t_1, t_2) est le nombre de symboles utilisés dans l'écriture de t_1 et t_2

1) Algorithme "Historique" de Robinson

algo 14 Entrée : deux termes t_1 et t_2 en écriture préfixe

Sortie : oui si ils sont unifiables, non sinon

```
Robinson(t1, t2) :
    σ = id
    tant que t1[σ] ≠ t2[σ]
        choisir une paire de symboles (u, v) telle que u est à la position
        i dans t1, v à la position i dans t2  et u ≠ v
        si l'un des symboles est une variable x et que x n'apparaît
        pas dans le terme de racine l'autre symbole (mettons v),
            σ ← [x := v] ∘ σ
        sinon renvoyer non
    renvoyer oui
```

prop 15 cet algorithme n'est pas déterministe et est exponentiel en temps et en espace

2) Algorithme de Martelli et Montanari et améliorations

prop 16 L'algorithme Unifie présenté ci dessus en **algo 17** termine et est correct (DEV 1)

algo 17 Entrée = un ensemble fini E d'équations (= couples de termes)

Sortie = un unificateur principal de E s'il existe, un échec sinon

Unifier (E) =

$\sigma \leftarrow id$

tant que $E \neq \emptyset$ faire

- si $E = E' \cup \{f(u_1, \dots, u_p) \simeq g(v_1, \dots v_q)\}$ alors
  - si $f = g$ alors $E \leftarrow E' \cup \{u_1 \simeq v_1, \dots, u_p \simeq v_p\}$
  - sinon renvoyer échec 1

  } CAS 1
- si $E = E' \cup \{x \simeq x\}$ alors $E \leftarrow E'$ } CAS 2
- si $E = E' \cup \{x \simeq u\}$ ou $E = E' \cup \{u \simeq x\}$ avec x variable et $x \neq u$
  - si $x \notin u$ alors $\sigma \leftarrow [x := u] \circ \sigma$ et $E \leftarrow E'[x := u]$
  - sinon renvoyer échec 2

  } CAS 3

renvoyer σ

prop 18 Soit $(x_m)_{m \in \mathbb{N}}$ une suite de variables et $(u_m)_{m \in \mathbb{N}}$, $(v_m)_{m \in \mathbb{N}}$ les suites de termes définis par : $u_0 = v_0 = x_0$ et $\forall m \geq 0$, $u_{m+1} = f(u_m, x_m)$

$v_{m+1} = f(x_m, v_m)$

Unifier $f(u_m, u_m)$ et $f(v_m, v_m)$ se fait en temps et en espace exponentiel en la taille de ces termes.

rq 19 Pour améliorer la complexité en espace, on considère que les termes sont sous forme de graphes tels que présenté en ANNEXE A

Puis on utilise l'algorithme Unifier2 suivant

algo 20 Entrée : 2 termes t_1 et t_2 sous forme de graphes

Sortie : oui si t_1 et t_2 sont unifiables, non sinon

Unifier2 (t_1, t_2) =

$t_1 \leftarrow$ trouve(t_1) et $t_2 \leftarrow$ trouve(t_2)

si $t_1 = t_2$, renvoyer oui

sinon
- si $(t_1, t_2) = (x, y)$ où x et y sont des variables
  faire pointer x vers y et renvoyer oui
- si $(t_1, t_2) = (x, t)$ ou (t, x) où x = variable et t = terme
  si x apparaît dans t, renvoyer faux
  sinon faire pointer x vers t et renvoyer oui
- si $(t_1, t_2) = (f(L_1), g(L_2))$ où f, g = fonctions et L_1, L_2 = listes de termes
  si $f = g$ renvoyer unifierlistes (L_1, L_2)
  sinon renvoyer non

où "trouve" prend en entrée un terme et renvoie le terme en bout de chaîne de t et "apparaît dans" est implémenté naïvement

ex 21 ☞ ANNEXE B

prop 22 cet algorithme est linéaire en espace mais reste exponentiel en temps car les tests d'appartenance de la variable x au terme t se font sur l'arbre "déplié" de t

prop 23 En utilisant un parcours en profondeur pour tester l'appartenance de x à t, la complexité temporelle de Unifier2 devient quadratique.

rq 24 En utilisant la structure Union-find dans Unifier2, on peut même aboutir à une complexité pseudo-linéaire en temps

II/ Application à la résolution

1) Clauses

def 25 L'ensemble A des formules atomiques sur L est le plus petit ensemble tel que $\forall p \in P$, $\forall t \in T$, $p(t) \in A$

def 26 Une clause est une disjonction de formules atomiques et de négations de formules atomiques. La clause vide est notée $\perp$

ex 27 $\neg p_1(x_1) \vee p_2(f(x_1, c_2), c_3)$ est une clause.

rq 28 On peut transformer une formule de L en un ensemble de clauses

ex 29 Pour $F = \forall x_1 \exists x_2\, p_1(x_1, x_2) \wedge \forall x_3\, p_3(x_3)$

cet ensemble est $\{p_1(x_1, f_{x_2}(x_1)),\ p_3(x_3)\}$

2) Résolution

def 30 La résolution est un système de déduction composé des règles suivantes : $\frac{C_1 \vee A \quad C_2 \vee \neg A'}{(C_1 \vee C_2)[\sigma]}$ res, et $\frac{C_1 \vee A \quad C_2 \vee A'}{(A \vee C_1 \vee C_2)[\sigma]}$ contr, où σ = unificateur principal de A et A'

def 31 Une preuve par résolution d'une clause C à partir de S est une suite $C_1, \dots, C_n$ telle que $C_n = C$ et $\forall i \in [\![1,n]\!]$, $C_i \in S$ ou $\exists j,k < i$, $\frac{C_j \quad C_k}{C_i}$

def 32 Une preuve de C par réfutation est une preuve de la clause vide à partir de l'ensemble $S \cup \{\neg C\}$

Th 33 (Correction) S'il existe une réfutation de C à partir de S, il n'y a pas de modèle de S qui soit modèle de C

def 34 Un modèle de Herbrand est un modèle dont le domaine D est l'ensemble des termes clos et dans lequel l'interprétation de $f \in F$ est $f_i^H : \begin{cases} D^k \longrightarrow D \\ (t_1, \dots t_k) \longmapsto f(t_1, \dots t_k) \end{cases}$ l'interprétation des prédicats est libre.

Th 35 (Complétude) Si S est un ensemble de clauses n'admettant pas de modèle alors il existe une réfutation de S au premier ordre (DEV 2)

IV Application à la réécriture

Soit R un système de réécriture noté $\to$.

Def 36 : On note $u \downarrow v$ si $\exists w$, $u \to^* w$ et $v \to^* w$. u et v sont dit joignables.

def 37 : R est dit confluent si : $t \to^* u$, $t \to^* v \Rightarrow u \downarrow v$. R est localement confluent si : $t \to u$, $t \to v \Rightarrow u \downarrow v$.

def 38 : On définit $t|_w$ par $t|_\varepsilon = t$ si $t = f(t_1, \dots, t_n)$ $t|_{w_i \tilde{w}} = t_i|_{\tilde{w}}$. la position w du sous terme de t.

def 39 : La substitution du sous terme de t à la position w par le terme u est notée $t[u]_w$ et définie par $t[u]_\varepsilon = u$ et $f(t_1, \dots, t_n)[u]_{w_i \tilde{w}} = t_i[u]_{\tilde{w}}$.

def 40 Soient $l_1 \to r_1$, $l_2 \to r_2$ deux règles de R sans variables communes. Soit p une position de l_1 telle que $l_1|_p$ ne soit pas une variable. Soit σ un unificateur principal de $l_1|_p$ et l_2.

$\langle r_1[\sigma], l_1[\sigma][r_2[\sigma]]_p \rangle$ est une paire critique associée à $l_1 \to r_1$ et $l_2 \to r_2$.

Exemple 41 : Annexe C est un exemple de paire critique.

Lemme des paire critiques 42 : Si $t \to u$ et $t \to v$ alors $u \downarrow v$ ou il existe une position p telle que $u = t[C_1]_p$, $v = t[C_2]_p$ et $\langle C_1, C_2 \rangle$ instance d'une paire critique de R.

Théorème 43 : R est localement confluent si et seulement si ses paire critiques sont joignables.

Lemme de Newman 44 : Un système de réécriture noetherien est confluent si et seulement si il est localement confluent.

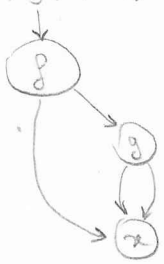
Corollaire 45 : Un système de réécriture noetherien est confluent si et seulement si ses paires critiques sont joignables.

Application : On peut décider la confluence d'un système de réécriture noetherien en calculant toutes ses paire critiques.

Ref : David, Nour, Raffali.
Baader, Nipkow.
Lassaigne, de Rougemont.

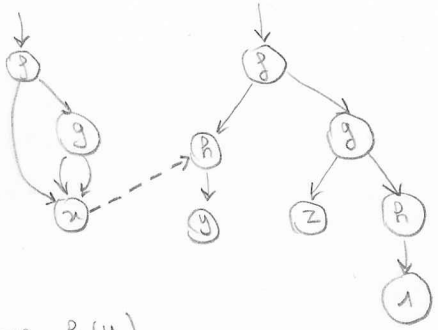
ANNEXE A

Au terme $f(x, g(x,x))$ correspond le graphe orienté



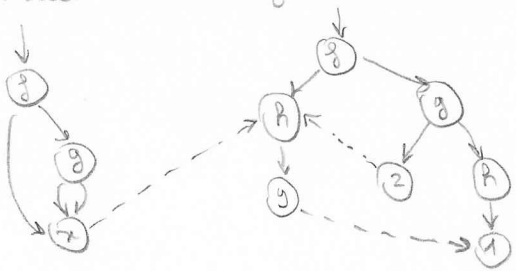
ANNEXE B

* Dans la structure



trouve(x) renvoie le terme $h(y)$

* L'unification des termes $f(x, g(x,x))$ et $f(h(y), g(z, h(1))$ avec l'algorithme Unifier2 aboutit à la structure suivante :
(les flèches pointillées sont celles ajoutées lors de l'exécution de l'algo)



Un unificateur est donc $[x := h(1) ; y := 1 ; z := h(1)]$

ANNEXE C :

Pour les règles $f(f(x,y), z) \to f(x, f(y,z))$

$f(i(x_1), x_1) \to e$

$f(f(i(x_1), x_1), z)$

$f(i(x_1), f(x_1, z))$ $\quad$ $f(e, z)$

est une paire critique.