Institut National
Universitaire
Champollion

Types de données, preuves

L3 Info - Semestre 6

Chapitre 4 - Algorithme d'Unification

Qu'est-ce qu'un problème d'unification ?
L'algorithme d'Unification.

Laura Brillon - laura.brillon[at]univ-jfc.fr

Sommaire

Quel est le problème ?

Termes

Notation. On se donne

- Un ensemble de *constantes* - *Exemple :* $2, true, c, d \dots$
  (Les constantes peuvent être des fonctions !)
- Un ensemble de *variables* - *Exemple :* $x, y, z, \dots$

Définition

Les **termes** sont définis *récursivement* de la façon suivante :

- Les constantes et les variables sont des termes.
- Si t_1 et t_2 sont des termes, alors $(t_1\ \ t_2)$ est un terme *(application)*.

Rappel - L'application associe à gauche, $p\ x\ 2 = ((p\ x)\ 2)$.

Notation. Si t est un terme, on note $t[x_1, \dots, x_n]$ si $x_1, \dots, x_n$ sont toutes les variables de t.

Quel est le problème ?

Substitution

Définition

Une **substitution** σ est une fonction d'un ensemble fini de variables ($\subset \mathcal{V}$) dans $\mathcal{T}$.

On peut appliquer une substitution sur un terme.

Notation. Si $t[x_1, \ldots, x_n]$ est un terme, on note

$$t[\sigma] \text{ ou bien } t[x_1 \leftarrow \theta_1, \ldots, x_n \leftarrow \theta_n]$$

(où $\theta_i = \sigma(x_i)$) la substitution obtenue à partir de t en remplaçant chaque occurrence de x_i par le terme θ_i.

Quel est le problème ?

Question !

Considérons

- le terme $t = f\ x\ (h\ y)$
- la substitution $\sigma = [x \leftarrow (i\ y), y \leftarrow 42]$

Quel est le terme $t[\sigma]$?

A) $f\ (i\ y)\ 42$

B) $f\ (i\ y)\ (h\ 42)$

C) $f\ (i\ 42)\ (h\ 42)$

D) $f\ (i\ 42)\ 42$

Quel est le problème ?

Unificateur

Définition

Un **unificateur** des termes $t_1,\ t_2$ est une substitution σ telle que

$$\sigma(t_1) = \sigma(t_2)$$

Le Problème d'Unification

Soient deux termes $t_1,\ t_2$.

Un problème d'unification est de la forme $t_1 \stackrel{?}{=} t_2$.

Il s'agit de trouver une substitution σ telle que

$$\sigma(t_1) = \sigma(t_2)$$

Sommaire

- Filtrage
- Inférence de types

À quoi ça sert ?

Filtrage

Étant donné le code :

```
match Noeud(3, Feuille 1, Feuille 2) with
| Feuille x -> x
| Noeud(y, a , Feuille x) -> x
```

Quelle valeur est renvoyée ?

A) 1

B) 2

C) 3

D) Erreur !

À quoi ça sert ?

Filtrage

Étant donné le code :

```
match Noeud(3, Feuille 1, Feuille 2) with
| Feuille x -> x
| Noeud(y, a , Feuille x) -> x
```

Quelle valeur est renvoyée ?

C'est un problème d'unification :

- `Feuille x` $\stackrel{?}{=}$ `Noeud(3, Feuille 1 , Feuille 2)` **échec**
- `Noeud(y, a, F x)` $\stackrel{?}{=}$ `Noeud(3, F 1, F 2)`

Unificateur : $[y \leftarrow 3,\ a \leftarrow \mathit{Feuille}\ 1,\ x \leftarrow 2]$
Valeur renvoyée : 2

À quoi ça sert ?

Inférence de types

Est-ce que la fonction

$$\texttt{List.rev : 'a list -> 'a list}$$

est applicable à $[2;3]$: `int list` ?

C'est un problème d'unification :

$$\texttt{'a list} \stackrel{?}{=} \texttt{int list}$$

Unificateur : $[\texttt{'a} \leftarrow \texttt{int}]$
Réponse : Oui, `List.rev [2;3] : int list`

À quoi ça sert ?

Inférence de types

Autres exemples.

- Est-il possible d'unifier `'a -> bool` et `int -> 'b` ?
- Est-il possible d'unifier `int -> (int -> 'a)` et `'b -> (int * bool)` ?

Attention. Il n'est pas toujours possible d'unifier deux termes !

Idée de l'algorithme.

· Décomposer les termes tant que les fonctions sont égales.

· *échec* si les fonctions sont différentes.

· Mémoriser la substitution si l'un des termes est une variable.

Sommaire

À quoi ça sert ?

Réécriture

Étant donné la règle de réécriture

`longueur(l1 @ l2) = longueur(l1) + longueur(l2)`

Comment montrer que
`longueur(l1 @ l2) = longueur(l2 @ l1)` ?

→ Automatiser des preuves.

Sommaire

- Unifier hypothèses et conclusion
- La méthode de résolution

À quoi ça sert ?

Unifier hypothèses et conclusion

Est-ce que la conclusion est bien une conséquence des hypothèses ?

$$Q(x), P(f(y)) \vdash P(f(a))$$

C'est un problème d'unification :

- $Q(x) \stackrel{?}{=} P(f(a))$ **échec**
- $P(f(x)) \stackrel{?}{=} P(f(a))$
  Unificateur : $[x \leftarrow a]$

À quoi ça sert ?

La méthode de résolution

Rappel : Une *clause* $\{A, B\}$ représente la disjonction $A \vee B$.

Résolution des clauses :

$$\frac{\{P(a)\}\ \{\neg P(x), Q(f(x))\}}{\{Q(f(a))\}}\ \sigma{=}[x{\leftarrow}a]$$

La résolution est un mécanisme essentiel du langage *Prolog*.

Sommaire

Substitution plus générale

Définition

La substitution σ est plus générale que σ_0 s'il existe σ' telle que

$$\sigma_0 = \sigma' \circ \sigma$$

Nous considérons les deux substitutions suivantes :

$$\sigma_1 = [x \leftarrow y] \text{ et } \sigma_2 = [x \leftarrow 5, y \leftarrow 5]$$

A) σ_1 est plus générale que σ_2

B) σ_2 est plus générale que σ_1

C) Les réponses 1. et 2. sont fausses

D) Je ne sais pas

Unificateur le plus général

Rappel. Un unificateur des termes t_1 et t_2 est une substitution σ telle que $\sigma(t_1) = \sigma(t_2)$

Définition

On appelle *unificateur le plus général* (noté *mgu*) de deux termes t_1 et t_2 l'unificateur qui est plus général que tout autre unificateur de t_1 et t_2.

Exemple de *mgu* - $g\ x\ (f\ y) \stackrel{?}{=} g\ x\ (f\ (f\ z))$
$\rightsquigarrow$ Activité Unificateur le plus général

Variables libres

Définition

L'ensemble des *variables libres* d'un terme t, noté $FV(t)$, est l'ensemble des variables qui apparaissent dans t.

Quel est l'ensemble des variables libres du terme $t_1 = f\ (h\ x\ y)\ (g\ c)$?

A) $FV(t_1) = \{c, f, g, h, x, y\}$

B) $FV(t_1) = \{f, g, h, x, y\}$

C) $FV(t_1) = \{c, x, y\}$

D) $FV(t_1) = \{x, y\}$

L'algorithme - enfin !

L'idée

- L'algorithme manipule des paires (E, S) avec
  - E est un multi-ensemble d'équations à résoudre
  - S est un ensemble de solutions
- Les règles de simplifications sont de la forme $(E, S) \Rightarrow (E', S')$.
- Un ensemble de solutions S est de la forme $\{x_1 = s_1, \ldots, x_n = s_n\}$ tel que
  - tous les x_i sont différents
  - aucun des x_i n'apparaît dans l'un des s_j.

L'idée. Étant donnés deux termes t_1, t_2 à unifier,
· L'algorithme simplifie $(t_1 \stackrel{?}{=} t_2, \{\ \})$ jusqu'à $(\{\ \}, \{x_1 = s_1, \ldots, x_n = s_n\})$ ou termine avec un échec.
· En cas de non-échec, le *mgu* est $[x_1 \leftarrow s_1, \ldots, x_n \leftarrow s_n]$.

L'algorithme - enfin !

Règles de simplification

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Rightarrow (E, S)$
- **Decompose :**
  $((s_1 \ s_2) \stackrel{?}{=} (t_1 \ t_2) \cup E, S) \Rightarrow (\{s_1 \stackrel{?}{=} t_1, s_2 \stackrel{?}{=} t_2\} \cup E, S)$
- **Fail :** $((s_1 \ s_2) \stackrel{?}{=} c \cup E, S) \Rightarrow \mathit{fail}$
- **Clash :** $(c_1 \stackrel{?}{=} c_2 \cup E, S) \Rightarrow \mathit{fail}$
  si c_1 et c_2 sont des constantes différentes.
- **Eliminate :** $(x \stackrel{?}{=} t \cup E, S) \Rightarrow (E[x \leftarrow t], S[x \leftarrow t] \cup \{x = t\})$
  si $t \neq x$ et $x \notin FV(t)$
- **Check :** $(x \stackrel{?}{=} t \cup E, S) \Rightarrow \mathit{fail}$
  si $t \neq x$ et $x \in FV(t)$.

Remarque : Algorithme non déterministe
$\rightarrow$ Appliquer les règles dans n'importe quel ordre

L'algorithme - enfin !

Exemple

Exemple

Nous considérons le problème d'unification $p\ x\ 2 \stackrel{?}{=} p\ 2\ x$.

$$\begin{aligned}
&\left(\{(p\ x)\ 2 \stackrel{?}{=} (p\ 2)\ x\};\ \{\ \}\right) \\
\Rightarrow\ &\left(\{(p\ x) \stackrel{?}{=} (p\ 2), 2 \stackrel{?}{=} x\};\ \{\ \}\right) && \text{(Decompose)} \\
\Rightarrow\ &\left(\{p \stackrel{?}{=} p, x \stackrel{?}{=} 2, 2 \stackrel{?}{=} x\};\ \{\ \}\right) && \text{(Decompose)} \\
\Rightarrow\ &\left(\{x \stackrel{?}{=} 2, 2 \stackrel{?}{=} x\};\ \{\ \}\right) && \text{(Delete)} \\
\Rightarrow\ &\left(\{2 \stackrel{?}{=} 2\};\ \{x = 2\}\right) && \text{(Eliminate)} \\
\Rightarrow\ &(\{\ \};\ \{x = 2\}) && \text{(Delete)}
\end{aligned}$$

Donc le *mgu* est $\sigma = [x \leftarrow 2]$.

Sommaire

Différentes unifications

Unification syntaxique : prend en compte la structure des termes

Exemple - Syntaxiquement, on ne peut pas unifier $(x_f\ 5) = 5$ et $x \oplus 2 = y \oplus 3$.

Unification d'ordre supérieur : donner un "sens" aux fonctions

Unification "modulo" théorie : prend en compte les propriétés de certains opérateurs

Exemple - associativité et commutativité de $\oplus$.

Unification d'ordre supérieur

Exemple - Trouver la fonction x_f telle que $(x_f\ 5) = 5$.

- *Imitation :* $x_f =$ `fun y -> 5`
- *Projection :* $x_f =$ `fun y -> y`

Observations : Deux solutions *indépendantes*.

- *Indécidabilité* de l'unification d'ordre supérieur
  Cf. Cours Théorie des langages
- Algorithme qui énumère les solutions

Pour aller plus loin

Unification "modulo" ACU

Exemple - Trouver un unificateur pour $x \oplus 2 = y \oplus 3$, sachant que $\oplus$ est un opérateur commutatif quelconque.

- *Solution* : $[x \leftarrow 3; y \leftarrow 2]$
- *Non-solution* : $[x \leftarrow 1; y \leftarrow 0]$ parce que $\oplus$ n'est pas $+$!

$\rightarrow$ Axiomatisation d'un opérateur $\oplus$ - associatif, commutatif et a une unité e ($x \oplus e = x$) - ACU.

- *Décidabilité* du problème d'unification
- Algorithmes de programmation linéaire
- Produit *mgu* unique