Réécriture

Unification dans des théories

version du 22 novembre 2004 – 14 h 29

Compléments sur l'unification

Ensemble complet d'unificateurs

Il existe des théories pour lesquels les problèmes n'ont pas un unificateur plus général, mais plusieurs (ou pas du tout !).

Un ensemble S de substitutions est un ensemble complet d'unificateurs d'un problème P si

1. $S \subseteq \mathcal{U}(P)$,
2. $(\forall \tau \in \mathcal{U}(P))\ (\exists \sigma \in S)\ (\exists \rho \in Sub(T(\Sigma, V))\ \tau = \rho\,\sigma$.

Combinaison d’algorithmes d’unification

Hétérogénéité

Soit $F = F_0 \uplus F_1$.

On suppose que l'on a deux théories

– E_0 sur $T(F_0, X)$

– et E_1 sur $T(F_1, X)$.

Un terme $t \in T(F, X)$ est pur si $t \in T(F_0, X) \cup T(F_1, X)$.

Si t n'est pas pur, il est hétérogène.

Hétérogénéité (suite)

Un sous-terme pur $t_{|p}$ est maximal

si $(\forall q \in Pos(t))\ q$ ***prefix*** $p \quad \Rightarrow \quad$ «$t_{|q}$ est hétérogène».

Une équation $s \stackrel{?}{=} t$ est pure

– si $\{s, t\} \subseteq T(F_0, X)$

– ou $\{s, t\} \subseteq T(F_1, X)$

Une équation $s \stackrel{?}{=} t$ est propre si ni s, ni t ne sont des variables.

Propriété des théories

Une théorie est sans collapsus si elle ne contient aucune identité de la forme $x = t$.

Une théorie est régulière si dans chaque identité $s = t$ chaque variable qui apparaît dans s apparaît dans t et vice-versa.

Organisation d'un problème hétérogène

Un problème d'unification hétérogène s'écrit

$$P = P_0 \cup P_1 \cup P_V \cup P_H$$

où P_0 est formé des équations propres et pures dans $T(F_0, X)$,

P_1 est formé des équations propres et pures dans $T(F_1, X)$,

P_V est formé des équations de la forme $x \stackrel{?}{=} y$,

P_H est formé des équations hétérogènes.

Dans ce qui suit on travaille sur des problèmes en forme triangulaire.

Donc on n'appelle pas Elimine.

L'unification dans des théories hétérogènes

Abstract $\quad P \cup \{s \stackrel{?}{=} t\} \quad \Rightarrow \quad P \cup \{s[x]_p \stackrel{?}{=} t, x \stackrel{?}{=} s_{|p}\}$

$s_{|p}$ est un sous-terme maximal pur de s.

$p \neq \epsilon$ et x est une variable fraiche

Solve$_0$ $\quad P \cup P_0 \quad \Rightarrow \quad P \cup P_0'$

P_0' est une forme résolue pure de P_0

dans E_0.

Solve$_1$ $\quad P \cup P_1 \quad \Rightarrow \quad P \cup P_1'$

P_1' est une forme résolue pure de P_1

dans E_1.

L'unification dans des théories hétérogènes (suite)

Coalesce $\quad P \cup \{x \stackrel{?}{=} y\} \quad \Rrightarrow \quad P\{x \mapsto y\} \cup \{x \stackrel{?}{=} y\}$ $\quad x \neq y$

Merge $\quad P \cup \{x \stackrel{?}{=} f(\vec{s}), x \stackrel{?}{=} g(\vec{t})\} \quad \Rrightarrow \quad P\{x \stackrel{?}{=} f(\vec{s}), f(\vec{s}) \stackrel{?}{=} g(\vec{t})\}$

Quelques explications

Dans Abstract, la variable x introduite est une variable d'interface entre des sous termes qui appartiennent à deux théories différentes.

Coalesce «prend acte» que la frontière ou l'interface a disparu.

Merge est une forme de la règle Elimine.

Solve_0 et Solve_1 font appel à la résolution dans les théories E_0 et E_1 respectivement.

Présolutions et solutions

Une présolution est un problème dont toutes les équations sont de la forme $x \stackrel{?}{=} t$.

Une présolution est une solution

si elle n'admet pas de cycle de la forme

$$x_1 \stackrel{?}{=} s_1[x_2] \ldots x_i \stackrel{?}{=} s_i[x_{i+1}] \ldots x_{2n} \stackrel{?}{=} s_i[x_1]$$

où les symboles à la racine d'au moins deux s_i et s_j sont dans des théories différentes.

Du problème aux solutions

La relation $\mapsto\!\!\!\!\Rightarrow$ est clairement indéterministe.

En particulier, elle a un branchement multiple pour $Solve_i$ qui peut fournir plus qu'une solution ou pas du tout.

On suppose que $Solve_i$ fournit un ensemble complet d'unificateurs.

$Solve_i$

$Solve_i$

- transforme P en Q (soit $P \mapsto\!\!\!\!\Rightarrow Q$)
- et prend une équation dans P_i et retourne (de façon indéterministe mais complète)
  - des équations de la forme $x_j \stackrel{?}{=} s_j$ dans P_i
  - et des équations de la forme $y_k \stackrel{?}{=} y_l$

    que l'on va mettre dans P_V,
  - une transition existe pour chaque unificateur de l'ensemble complet,

Les échecs

L'algorithme échoue dans trois cas, c'est-à-dire qu'on est en présence d'un problème équationnel pour lequel aucune transition n'existe et pourtant le problème n'est pas lui-même une solution.

1. Seule une transition $Solve_i$ est applicable, mais elle ne retourne aucun unifi cateur,
2. Aucune transition n'est applicable, mais il reste une équation de la forme $s \stackrel{?}{=} t$ avec ni s, ni t qui ne soit une variable. Cela signifi e que s et t sont des termes purs dans deux théories différentes.
3. Le problème est une présolution qui n'est pas une solution (présence d'un cycle).

Du problème aux solutions

Si E_0 et E_1 sont des théories régulières et sans collapsus

alors l'ensemble $\{Q \text{ solution} \mid P \overset{*}{\Rrightarrow} Q\}$

forme un ensemble complet d'unificateurs pour P.

L’unification associative et commutative restreinte

L'unification associative et commutative générale

Dans le cas de l'unification associative et commutative générale, seuls certains opérateurs sont AC (c'est-à-dire associatifs et commutatifs[a]).

Il faudra donc combiner des algorithmes d'unification où seul un opérateur est AC.

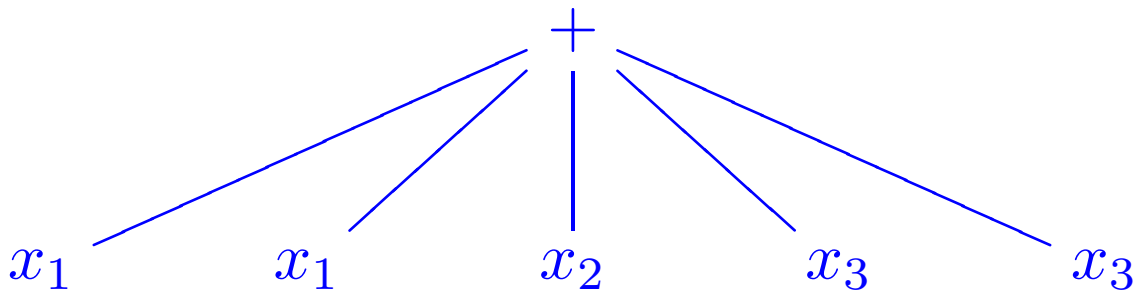
Si $+$ est un opérateur associatif et commutatif, la résolution d'équations dans la théorie $T(\{+\}, X)$ s'appelle l'unification AC restreinte.

[a] et pas automate cellulaire comme le croient certains.

La représentation des termes

A priori un terme associatif et commutatif dans la théorie $T(\{+\}, X)$ est un arbre dont les noeuds internes sont des $+$ et les feuilles des variables.

Par exemple $(x_1 + x_2) + (x_3 + (x_3 + x_1))$, se représente mieux par $x_1 + x_1 + x_2 + x_3 + x_3$ ou par l'arbre



ou par le terme $2x_1 + x_2 + 2x_3$.

La représentation des termes et des équations

Plus généralement un terme s non réduit à une varibale est représenté par

$$s = \sum_{i=1}^{n} s_i x_i$$

avec

$$\sum_{i=1}^{n} s_i \geq 2$$

La représentation des termes et des équations

– Un terme réduit à une variable correspond au cas où

$$\sum_{i=1}^{n} s_i = 1$$

– Le cas

$$\sum_{i=1}^{n} s_i = 0$$

ne correspond à aucun terme AC.

Il doit éliminer absolument.

La représentation des termes et des équations

Une équation est représentée par

$$\sum_{i=1}^{n} s_i x_i \stackrel{?}{=} \sum_{i=1}^{n} t_i x_i \qquad (eq)$$

La représentation des substitutions

Clairement une substitution σ idempotente applique à chaque variable x_i un terme AC a_i tel que $\{x_1, \ldots, x_n\} \cap Var(a_i) = \emptyset$ donc

$$\sigma(x_i) = \sum_{j=1}^{q} a_{ij} y_j$$

où $a_{ij} \in \mathbb{N}$, $x_i \notin \{y_1, \ldots, y_q\}$ et la condition (que l'on appelle $\mathcal{C}$) est satisfaite :

$$\sum_{j=1}^{q} a_{ij} \geq 1 \qquad (\mathcal{C})$$

L'unification AC restreinte (1/4)

$\{..., x_i \mapsto \sum_{j=1}^{q} a_{ij} y_j, ...\}$ est un unificateur idempotent de $s \stackrel{?}{=} t$ dans le problème P si

$$\sum_{i=1}^{n} s_i \sum_{j=1}^{q} a_{ij} y_j = \sum_{i=1}^{n} t_i \sum_{j=1}^{q} a_{ij} y_j$$

où $\{y_1, \ldots, y_q\} \cap Var(P) = \emptyset$

L'unification AC restreinte (1/4)

$\{..., x_i \mapsto \sum_{j=1}^{q} a_{ij} y_j, ...\}$ est un unificateur idempotent de $s \stackrel{?}{=} t$ dans le problème P si

$$\sum_{i=1}^{n} s_i \sum_{j=1}^{q} a_{ij} y_j = \sum_{i=1}^{n} t_i \sum_{j=1}^{q} a_{ij} y_j$$

où $\{y_1, \ldots, y_q\} \cap Var(P) = \emptyset$

Soit encore

$$\sum_{j=1}^{q} y_j \sum_{i=1}^{n} s_i a_{ij} = \sum_{j=1}^{q} y_j \sum_{i=1}^{n} t_i a_{ij}$$

L'unification AC restreinte (2/4)

Clairement $\{..., x_i \mapsto a_{ij}, ...\}$ est une solution de (eq), car

$$\sum_{i=1}^{n} s_i a_{ij} = \sum_{i=1}^{n} t_i a_{ij}.$$

Elle engendre l'ensemble des solutions de (eq).

$\{..., x_i \mapsto a_{ij}, ...\}$ est un unificateur si

$$\sum_{j=1}^{q} a_{ij} \geq 1$$

L'unification AC restreinte (2/4)

Donc si $(a_{1j} \ldots, a_{ij}, \ldots, a_{nj})$ (pour un j fixé) est une solution génératrice de l'équation diophantienne

– linéaire (tous les monômes de degré 1),

– homogène (pas de termes constants),

$$\sum_{i=1}^{n} s_i x_i \stackrel{?}{=} \sum_{i=1}^{n} t_i x_i$$

.

L'unification AC restreinte (2/4)

Donc si $(a_{1j} \ldots, a_{ij}, \ldots, a_{nj})$ (pour un j fixé) est une solution génératrice de l'équation diophantienne

– linéaire (tous les monômes de degré 1),

– homogène (pas de termes constants),

$$\sum_{i=1}^{n} s_i x_i \stackrel{?}{=} \sum_{i=1}^{n} t_i x_i$$

.

La solution générale est une combinaison $\mathbb{N}$-linéaire des ces solutions.

L'unification AC restreinte (3/4)

Étant donné un problème d'unification P, l'ensemble des solutions du système d'équations diophantiennes associé qui satisfont la condition $\mathcal{C}$ est l'ensemble des unificateurs de P.

L'unification AC restreinte (3/4)

Étant donné un problème d'unification P, l'ensemble des solutions du système d'équations diophantiennes associé qui satisfont la condition $\mathcal{C}$ est l'ensemble des unificateurs de P.

Reste à trouver un ensemble fini de générateurs de l'ensemble des solutions qui satisfont $\mathcal{C}$

pour avoir un ensemble complet et fini d'unificateurs de P.

L'unification AC restreinte (4/4)

Pour cela on cherche d'abord un ensemble de générateurs des solutions du système dans $\mathbb{N}$.

Ensuite on s'en sert pour engendrer un générateur des solutions qui satisfont $\mathcal{C}$.

L’unification en lambda calcul

ou

unification d’ordre supérieur

Objectif

On peut vouloir résoudre des équations comme

$$\lambda x.sin(cos(x)) \stackrel{?}{=} \lambda x.sin(F(x))$$

qui a pour solution $\lambda y.cos(y)$.

Alors que

$$\lambda x.F \stackrel{?}{=} \lambda x.sin(cos(x))$$

n'a pas de solution.

Exemple

Plus généralement, on va résoudre des équations du type

$$\lambda x.F(\lambda y.xya) \stackrel{?}{=} \lambda z.z(zaa)$$

où $a : \tau$ et $\vdash \lambda z.z(zaa) : ?$

Exemple

Plus généralement, on va résoudre des équations du type

$$\lambda x.F(\lambda y.xya) \stackrel{?}{=} \lambda z.z(zaa)$$

où $a : \tau$ et $\vdash \lambda z.z(zaa) : (\tau \to \tau \to \tau) \to \tau \to \tau.$

F est une inconnue à instancier.

Quel est son type ?

Le problème

Les termes que l'on cherche à unifier

– sont simplement typés,

– ils ne contiennent que des variables liées que nous appellerons variables notées $x, x', x_1, x_2, ..., y, y', y_1, ..., z, ...$

– ils contiennent des inconnues que nous cherchons à «résoudre» et que nous noterons F, G, H (l'ensemble de inconnues de t est noté $INC(t)$),

– ils peuvent contenir des constantes notées $g, h, k, c...$

– et l'on travaille modulo β et η.

On considère donc la forme β-normale de chaque terme.

Le problème

Les constituants atomiques d'un terme sont

– soit des constantes,
– soit des variables (liées),
– soit des inconnues.

Le problème

On va présenter une algorithme d'unification dû à Huet revisité par Snyder pour une présentation en règles.

– Gérard Huet. A unification algorithm for typed λ-calculus. Theoretical Computer Science, Vol.1, pp 27–57, 1975.
– Wayne Snyder, A proof theory for general unification, Birkhäuser, (1991), Vol. 11, Progress in computer science and applied logic,
– Christian Prehofer, Solving Higher-Order Equations : From Logic to Programming, Birkhäuser PTCS Series, 1997.

Forme β-normale

Une forme β-normale est un terme de la forme

$$\lambda x_1 ... x_n . v M_1 .. M_p$$

– où v est soit l'une des variables x_i, soit une constante, soit une inconnue,
– où chacun des M_j est une forme normale.

Forme β-normale

En effet tous les sous-termes d'une forme normale sont en forme normale.

Les membres gauches d'une application ne peuvent pas être des abstractions donc ce sont

- soit des applications,
- soit des variables ou des constantes ou des inconnuees.

Rigidité et flexibilité

Si un terme est de forme $\lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})$

où v est soit une variable ou une constante,

le terme est dit rigide.

Si un terme est de forme $\lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})$

où v est une inconnue

le terme est dit flexible.

Forme $\beta\eta$ longue

D'autre part, on veut que si $\vdash M : \sigma_1 \to \ldots \to \sigma_n \to \tau$

alors le terme étudié soit de la forme

$$\lambda x_1 \ldots x_n . N$$

Forme $\beta\eta$ longue

Si toutes les variables ne sont pas présentes il faut faire une ou plusieurs η-expansions.

En résumé pour obtenir la forme $\beta\eta$ longue d'un terme

$$\vdash M : \sigma_1 \to \ldots \to \sigma_n \to \tau$$

On calcule la forme β-normale $\lambda x_1 \ldots x_k.N$.

Clairement, $k \leq n$ et la forme $\beta\eta$ longue est

$$\lambda x_1 \ldots x_k x_{k+1} \ldots x_n.N x_{k+1} \ldots x_n$$

Forme $\beta\eta$ longue

La forme $\beta\eta$-longue du terme

$$\lambda x.F(\lambda y.xya) \stackrel{?}{=} \lambda z.z(zaa)$$

est

?

Forme $\beta\eta$ longue

La forme $\beta\eta$-longue du terme

$$\lambda x.F(\lambda y.xya) \stackrel{?}{=} \lambda z.z(zaa)$$

est

$$\lambda x_1 x_2.F(\lambda y.x_1 ya)x_2 \stackrel{?}{=} \lambda x_1 x_2.x_1(x_1 aa)x_2$$

Notations

On écrit $v(\vec{t_n})$ pour $v\ t_1 \ldots t_n$

où $v = F$, $v = x$ ou $v = f$, c'est-à-dire v une inconnue ou une variable liée ou une constante.

Trois règles de transformation de problèmes de base

Supprime

$$\{u \stackrel{?}{=} u\} \cup P \quad \Rrightarrow \quad P$$

Decompose

$$\{\lambda\overrightarrow{x_k}.v(\overrightarrow{t_m}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \Rrightarrow \quad \bigcup_{i=1}^{m} \{\lambda\overrightarrow{x_k}.t_i) \stackrel{?}{=} \lambda\overrightarrow{x_k}.t'_i\} \cup P$$

Elimine

$$\{F \stackrel{?}{=} t\} \cup P \quad \Rrightarrow \quad \theta P \qquad \text{si } \theta = \{F \mapsto t\} \text{ et } F \notin INC(t)$$

Les règles d’imitation et de projection

Forme *flexible-rigide*

Si l'équation est de la forme $\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})$

où v n'est pas une inconnue,

on appelle cela un cas flexible-rigide.

Forme *flexible-rigide* : projection

Si l'équation est de la forme $\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})$

où v n'est pas une inconnue.

On peut faire l'hypothèse que v est le symbole de tête de l'un des t_i et que l'affectation que l'on fait à F vise à «atteindre» ce terme, pour créer une meilleure ressemblance.

On appelle cette règle projection.

Forme *flexible-rigide* : projection

Alors posons $\theta := F \mapsto \lambda\overrightarrow{y_n}.y_i(\overrightarrow{H_q(\overrightarrow{y_n})})$.

$$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P$$

$$\Longmapsto$$

$$\{\lambda\overrightarrow{x_k}.\theta(F(\overrightarrow{t_n})) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\theta(v(\overrightarrow{t'_m}))\} \cup \theta P$$

Puisque $\theta(F(\overrightarrow{t_n})) \underset{\beta}{\longrightarrow} \theta(t_i(\overrightarrow{H_q(\overrightarrow{t_n})})) \equiv (\theta t_i)(\overrightarrow{H_q(\overrightarrow{\theta t_n})})$

$$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P$$

$$\Longmapsto$$

$$\{\lambda\overrightarrow{x_k}.\theta t_i(\overrightarrow{H_q(\overrightarrow{\theta t_n})}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{\theta t'_m})\} \cup \theta P$$

Forme *flexible-rigide* : projection, exercice 1

Soit l'équation :

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

Forme *flexible-rigide* : projection, exercice 1

Soit l'équation :

$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$

On a $g : \iota \to \iota \to \iota$ et $F : \iota \to \iota$ où ι est un type de base.

$k = 2, n = 1, i = 1, m = 2, q = 0$

$F \mapsto \lambda y_1 . y_1$.

Forme *flexible-rigide* : projection, exercice 1

Soit l'équation :

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

$$k = 2, n = 1, i = 1, m = 2, q = 0$$

plus précisément $\lambda x_1 x_2 . v(x_1\ x_2) = \lambda x_1 x_2 . g\ x_1\ x_2$

et $t_1 = g\ x_1\ x_2$

$F \mapsto \lambda y_1 . y_1$.

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

$$\stackrel{Projette}{\Longmapsto}$$

$$\lambda x_1 x_2 . g\ x_1\ x_2 \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

Forme *flexible-rigide* : imitation (1/3)

Si l'équation est de la forme $\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})$

où v n'est pas une inconnue.

On peut faire l'hypothèse que v est précisément le symbole par lequel on souhaite que F commence.

Dans ce cas, v n'est pas une variable liée.

Forme *flexible-rigide* : imitation (2/3)

Alors posons $\theta := F \mapsto \lambda\overrightarrow{y_n}.v(\overrightarrow{H_q(\overrightarrow{y_n})})$.

$$\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}v(\overrightarrow{t'_m}) \cup P$$

$$\Longmapsto$$

$$\{\lambda\overrightarrow{x_k}.\theta(F(\overrightarrow{t_n})) \stackrel{?}{=} \lambda\overrightarrow{x_k}\theta(v(\overrightarrow{t'_m}))\} \cup \theta P$$

Forme *flexible-rigide* : imitation, exercice 2

Soit la même équation :

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1 x_2$$

$$F \mapsto \lambda y_1 . g(H_1\ y_1)(H_2\ y_1).$$

$$\lambda x_1 x_2 . F(g\ x_1 x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

$$\stackrel{Imite}{\Longmapsto}$$

$$\lambda x_1 x_2 . g\ (H_1\ (g\ x_1\ x_2))(H_2\ (g\ x_1\ x_2)) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

Forme *flexible-rigide* : imitation, exercice 2

Soit la même équation :

$$\lambda x_1 x_2.F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2$$

$$F \mapsto \lambda y_1 y_2.g(H_1\ y_1)(H_2\ y_1).$$

$$\lambda x_1 x_2.F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2$$

$$\stackrel{Imite}{\Longmapsto}$$

$$\lambda x_1 x_2.g\ (H_1\ (g\ x_1\ x_2))(H_2\ (g\ x_1\ x_2)) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2$$

$$\stackrel{Decompose}{\Longmapsto}$$

$$\{\lambda x_1 x_2.H_1\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.x_1, \lambda x_1 x_2.H_2\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.x_2\}$$

Forme *flexible-rigide* : imitation, exercice 2

On ne peut pas appliquer Projette sur

$$\lambda x_1 x_2 . H_1\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . x_1$$

mais par Imite, elle donne $H_1 \mapsto \lambda z_1 z_2 . x_1$

$$\lambda x_1 x_2 . x_1 \stackrel{?}{=} \lambda x_1 x_2 . x_1$$

Par symétrie on voit que cette branche conduit à la solution

$F \mapsto \lambda y_1 y_2 . g(H_1\ y_1)(H_2\ y_1)$,

$H_1 \mapsto \lambda z_1 z_2 . x_1, \quad H_2 \mapsto \lambda z_1 z_2 . x_2.$

Donc $F \mapsto \lambda y_1 y_2 . g x_1 x_2$.

Pour résumer

L'équation

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ x_1\ x_2$$

a donc deux solutions

$$F \mapsto \lambda y_1 . y_1$$

et

$$F \mapsto \lambda y_1 y_2 . g x_1 x_2 .$$

Forme *flexible-rigide* : imitation, exercice 3

Soit l'équation :

$$\lambda x_1 x_2.F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ (g\ x_1\ x_2)\ (g\ x_1\ x_2)$$

$$F \mapsto \lambda y_1 y_2.g(H_1\ y_1)(H_2\ y_1).$$

$$\lambda x_1 x_2.F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ (g\ x_1\ x_2)\ (g(\ x_1\ x_2)$$

$$\stackrel{Imite}{\Longmapsto}$$

$$\lambda x_1 x_2.g\ (H_1\ (g\ x_1\ x_2))(H_2\ (g\ x_1\ x_2)) \stackrel{?}{=} \lambda x_1 x_2.g\ (g\ x_1\ x_2)\ (g(\ x_1\ x_2)$$

$$\stackrel{Decompose}{\Longmapsto}$$

$$\{\lambda x_1 x_2.H_1\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2, \lambda x_1 x_2.H_2\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2\}$$

Forme *flexible-rigide* : imitation, exercice 3

Soit donc

$$\{\lambda x_1 x_2.H_1\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2, \lambda x_1 x_2.H_2\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2\}$$

Par projection, sur le modèle de l'exercice précédent, en prenant

$H_1 \mapsto \lambda y.y$ et $H_2 \mapsto \lambda y.y$, on trouve

$$\{\lambda x_1 x_2.H_1\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2, \lambda x_1 x_2.H_2\ (g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2\}$$

$$\stackrel{2,Projette}{\Longmapsto}$$

$$\{\lambda x_1 x_2.(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2, \lambda x_1 x_2.(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2.g\ x_1\ x_2\}$$

Le système PT0 de transformation de problèmes

Supprime

$$\{u \stackrel{?}{=} u\} \cup P \quad \mapsto \quad P$$

Decompose

$$\{\lambda\overrightarrow{x_k}.v(\overrightarrow{t_m}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \bigcup_{i=1}^{m}\{\lambda\overrightarrow{x_k}.\overrightarrow{t_i}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{t'_i}\} \cup P$$

Elimine

$$\{F \stackrel{?}{=} t\} \cup P \quad \mapsto \quad \theta P \qquad \text{si } \theta = \{F \mapsto t\} \text{ et } F \notin INC(t)$$

Projette

$$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \{\lambda\overrightarrow{x_k}.\theta(F(\overrightarrow{t_n})) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\theta(v(\overrightarrow{t'_m}))\} \cup \theta P$$

où $\theta = \{F \mapsto \lambda\overrightarrow{y_n}.y_i(\overrightarrow{H_q(\overrightarrow{y_n})})\}$

Imite

$$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \{\lambda\overrightarrow{x_k}.\theta(F(\overrightarrow{t_n})) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\theta(v(\overrightarrow{t'_m}))\} \cup \theta P$$

où $\theta := \{F \mapsto \lambda\overrightarrow{y_n}.v(\overrightarrow{H_m(\overrightarrow{y_n})})\}$

Le système PT1 de transformation de problèmes

Forme *flexible-rigide* : imitation (3/3)

Soit $\theta := F \mapsto \lambda\overrightarrow{y_n}.v(\overrightarrow{H_m(\overrightarrow{y_n})})$.

$$\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}v(\overrightarrow{t'_m}) \cup P$$

$$\Longmapsto_{Imite-PT0}$$

$$\lambda\overrightarrow{x_k}.\theta(F(\overrightarrow{t_n})) \stackrel{?}{=} \lambda\overrightarrow{x_k}\theta(v(\overrightarrow{t'_m}))\} \cup \theta P$$

$$\Longmapsto_{Decompose}$$

$$\{\lambda\overrightarrow{x_k}.H_1(\overrightarrow{\theta t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{\theta t'_1}, \ldots, \lambda\overrightarrow{x_k}.H_m(\overrightarrow{\theta t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{\theta t'_m}\} \cup \theta P$$

donc

$$\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}v(\overrightarrow{t'_m}) \cup P$$

$$\Longmapsto_{Imite-PT1}$$

$$\{\lambda\overrightarrow{x_k}.H_1(\overrightarrow{\theta t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{\theta t'_1}, \ldots, \lambda\overrightarrow{x_k}.H_m(\overrightarrow{\theta t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{\theta t'_m}\} \cup \theta P$$

Supprime

$\{u \stackrel{?}{=} u\} \cup P \quad \mapsto \quad P$

Decompose

$\{\lambda\overrightarrow{x_k}.v(\overrightarrow{t_m}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \bigcup_{i=1}^{m}\{\lambda\overrightarrow{x_k}.\overrightarrow{t_i}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{t'_i}\} \cup P$

Elimine

$\{F \stackrel{?}{=} t\} \cup P \quad \mapsto \quad \theta P \qquad \text{si } \theta = \{F \mapsto t\} \text{ et } F \notin INC(t)$

Projette

$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \{\lambda\overrightarrow{x_k}.\theta t_i(\overrightarrow{H_q(\theta t_n)}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{\theta t'_m})\} \cup \theta P$

où $\theta = \{F \mapsto \lambda\overrightarrow{y_n}.y_i(\overrightarrow{H_q(\overrightarrow{y_n})})\}$

Imite

$\{\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.v(\overrightarrow{t'_m})\} \cup P \quad \mapsto \quad \bigcup_{i=1}^{m}\{\lambda\overrightarrow{x_k}.H_i(\overrightarrow{\theta t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.\overrightarrow{\theta t'_i}\} \cup \theta P$

où $\theta := \{F \mapsto \lambda\overrightarrow{y_n}.v(\overrightarrow{H_m(\overrightarrow{y_n})})\}$

Forme *flexible-flexible*

Les équations de la forme $\lambda\overrightarrow{x_k}.F(\overrightarrow{t_n}) \stackrel{?}{=} \lambda\overrightarrow{x_k}.G(\overrightarrow{t'_m})$ sont dites flexibles-flexibles.

On montre

1. qu'elles ont toujours au moins une solution,
2. qu'elles ont une infinité de solutions sans pourtant posséder un ensemble complet fini d'unificateurs.

PT1 est une procédure de préunification.

σ est un préunificateur de P si σP produit par Supprime, Decompose, Elimine un ensemble d'équations flexibles-flexibles.

Forme *flexible-flexible : exercice 4*

Soit l'équation :

$$\lambda x_1 x_2 . F(g\ x_1\ x_2) \stackrel{?}{=} \lambda x_1 x_2 . g\ (G\ x_1\ x_2)\ (G\ x_2\ x_1)$$

Montrer que l'on aboutit à des équations flexibles-flexibles et qu'une solution possible est $F \mapsto \lambda y.y$ et $G \mapsto \lambda y_1 y_2 . y_1$.

Trouver une autre solution pour G.

Cohérence

Si $P \Mapsto P'$ et si θ est un préunificateur de P' alors θ est un préunificateur de P.

Complétude

Si P a une solution θ alors il existe un P' tel que

– $P \mapsto P'$ en accumulant la substitution σ au cours de Elimine,

– P' est un ensemble d'équations flexibles-flexibles ayant entre autres comme solution ρ

– et $\theta = \rho\sigma$.

Indécidabilité

Golfarb (1981) :

Le problème de l'unification du second ordre est indécidable.

La preuve se fait par réduction au $10^{ème}$ problème de Hilbert.

Les motifs ou patterns

Un terme simplement typé en $\beta\eta$-forme normale longue est un motif ou un pattern

1. si toutes les occurrences de ses inconnues ont seulement des variables liées en paramètre
2. si toutes ces variables liées sont différentes les unes des autres.

$\lambda xy.f(F(y,x))$ et $\lambda xy.g(F(y,x),G(y))$ sont des motifs.

$\lambda xy.F(f(y),x)$ et $\lambda xy.f(F(x,x)$ ne sont pas des motifs.

Unification des motifs ou patterns

Si un $P = s \stackrel{?}{=} t$ est un problème d'unification où s et t sont deux motifs et si P a une solution

alors cette solution est unique

et peut être trouvée en temps linéaire.