Expressivité

Codage des entiers : entiers de Church

$n \in \mathbb{N} \quad \mapsto \quad \overline{n} = \lambda f.\lambda x.(\underbrace{f \cdots (f}_{n}\ x)\cdots)$ (fonction associant $f^n(x)$ à f et x)

Fonction successeur $succ$ t. q. pour tout $n \in \mathbb{N}, (succ\ \overline{n}) \to_\beta^\star \overline{n+1}$:

Ainsi $(((succ\ \overline{n})\ f)\ x) \to_\beta^\star (\underbrace{f \cdots (f}_{n+1}\ x)) =_\beta (f\ ((\overline{n}\ f)\ x))$

D'où $succ = \lambda n.\lambda f.\lambda x.(f\ ((n\ f)\ x))$

Exo. : add, mul, etc.

Expressivité

Projection : $\overline{\pi_{k,i}} = \lambda x_1.\lambda x_2.\cdots.\lambda x_k.x_i$

Composition : $\overline{f(\ldots, g_j(\ldots x_i \ldots), \ldots)}$ = ?

Expressivité

Projection : $\overline{\pi_{k,i}} = \lambda x_1.\lambda x_2.\cdots.\lambda x_k.x_i$

Composition : $\overline{f(\ldots, g_j(\ldots x_i \ldots), \ldots)}$ = ? $\qquad g_1 \cdots g_m$

Forcer évaluation des paramètres :

$E_m = \lambda x_1.\lambda x_2.\cdots.\lambda x_m.\lambda f.\underbrace{(\cdots(x_1(x_2(\cdots(x_m(\lambda x.x)\cdots)\ f)x_1)x_2)\cdots)x_m)}_{\text{itération identité}}$

D'où : $\lambda x_1.\lambda x_2.\cdots.\lambda x_k.(\cdots((E_m(\overline{g_1}\,\vec{x}))(\overline{g_2}\,\vec{x}))\cdots)\overline{f})$

Exemple : $(\overline{h(g)}\ \overline{0})$ pour $\overline{g}$ non déf. en $\overline{0}$ réduction β-normale

$\to_\beta ((E_1(\overline{g}\ \overline{0}))\ \overline{h})$

$= ((\lambda x_1.\lambda f.(((x_1\ (\lambda x.x))\ f)\ x_1)\ (\overline{g}\ \overline{0}))\ \overline{h})$

$\to_\beta (\lambda f.((((\overline{g}\ \overline{0})\ (\lambda x.x))\ f)\ (\overline{g}\ \overline{0}))\ \overline{h})$

$\to_\beta ((((\overline{g}\ \overline{0})\ (\lambda x.x))\ \overline{h})\ (\overline{g}\ \overline{0}))\ \to_\beta \cdots$

Expressivité

Théorème. (Point fixe)

Il existe Y t. q. pour tout H : $(Y\ H) =_\beta (H\ (Y\ H))$

Exemple : opérateur de Curry $\lambda h.(\lambda x.(h\ (x\ x))\ \lambda x.(h\ (x\ x)))$

Corollaire.

Pour tout H, $(HX) = X$ admet (au moins) une solution

Expressivité

Test à 0 : $T_0 = \lambda x.((\overline{n}\ \lambda y.\overline{\pi_{2,2}})\ \overline{\pi_{2,1}})$ on vérifie...

Récursion primitive = ?

On cherche $\overline{f}$ tel que :

$$\overline{f} =_\beta \lambda m.\lambda \overrightarrow{n}.(((T_0\ m)\ (\overline{b}\ \overrightarrow{n}))\ (((\overline{h}\ (pred\ m))\ \overrightarrow{n})\ \underbrace{((\overline{f}\ (pred\ m))\ \overrightarrow{n})}_{\text{appel réc.}}))$$

$$\overline{f} = (Y\ H_f)$$

Expressivité

Minimisation : ? $f(\overrightarrow{n}) = \min\{m \mid g(m, \overrightarrow{n}) = 0\}$

On cherche A tel que :

$$\begin{array}{lll} ((A\ \overrightarrow{n})\ \overline{m}) & \to_\beta^\star\ \overline{m} & \text{si } g(m,\overrightarrow{n}) = 0 \\ ((A\ \overrightarrow{n})\ \overline{m}) & \to_\beta^\star\ ((A\ \overrightarrow{n})\ (succ\ \overline{m})) & \text{sinon} \end{array}$$

Et donc $\overline{f} = \lambda \overrightarrow{x}.((A\ \overrightarrow{x})\ \overline{0})$

Pour A c'est une équ. de point fixe...

$$H_A = \lambda ha.\lambda g.\lambda \overrightarrow{n}.\lambda m.(((T_0\ ((g\ m)\ \overrightarrow{n}))\ m)\ (((ha\ g)\ \overrightarrow{n})\ (succ\ m)))$$

Expressivité

Théorème.

Pour toute fonction récursive partielle $F : \mathbb{N}^k \to \mathbb{N}$,

il existe un λ-terme $\overline{F}$ t. q. :

- Si F est définie en $(n_1, \ldots, n_k)$ alors $(\cdots((\overline{F}\ \overline{n}_1)\ \overline{n}_2)\cdots\overline{n}_k)$ admet $\overline{F(n_1, \ldots, n_k)}$ comme forme normale
- Si F n'est pas définie en $(n_1, \ldots, n_k)$ alors $(\cdots((\overline{F}\ \overline{n}_1)\ \overline{n}_2)\cdots\overline{n}_k)$ n'a pas de forme normale

Atteint grâce à réduction normale

Expressivité

Corollaire.

F λ-représentable $\Leftrightarrow$ F récursive partielle $\Leftrightarrow$ F Turing-calculable

Corollaire.

$=_\beta$ indécidable

Types

- Pur : trop riche
- Avantage : base de langage de programmation typé (**`Ocaml`**)
- Autre avantage : applications en logique

Dans un premier temps : simplement typé

Types simples

Ensemble de types de base : B

Ensemble des types construits sur B :

- $t \in B$ alors t type
- t_1 type et t_2 type alors $t_1 \to t_2$ type

Environnement de typage : liste de couples (variable, type)

Jugement de typage : formule logique $\Gamma \vdash E : t$

$[n : \text{int}] \vdash (\lambda x.x\ n) : \text{int}$ valide

$[] \vdash \lambda x.x : \text{bool}$ non valide

Types simples

Validité : axiomes + règles d'inférence

Variable $$\frac{x : t \in \Gamma}{\Gamma \vdash x : t}$$ (première occurrence)

Abstraction $$\frac{(x : t_1) \cup \Gamma \vdash E : t_2}{\Gamma \vdash \lambda x.E : t_1 \to t_2}$$

Application $$\frac{\Gamma \vdash E_1 : t_1 \to t_2 \quad \Gamma \vdash E_2 : t_1}{\Gamma \vdash (E_1\ E_2) : t_2}$$

Types simples

Validité : axiomes + règles d'inférence

Typable : il existe une dérivation

$[f : int \to bool; n : int] \vdash (f\ n) : bool$ valide

$[] \vdash \lambda x.x : int \to int$

$[] \vdash \lambda x.x : bool \to bool$

$[] \vdash \lambda x.x : (int \to bool) \to (int \to bool)$

$[] \vdash \lambda x.x : t \to t$ pour tout t

En particulier : pas unique $\rightsquigarrow$ variables de type

Types simples

Exo.

- Ω
- $(\lambda x.x\ \lambda x.x)$
- Entiers de Church

Variables de type

Ensemble A t. q. $A \cap B = \varnothing$ variables de type

Application (substitution) de A vers ensemble des types, étendue sur celui-ci :

- $b\sigma = b$ pour $b \in B$
- $(t_1 \to t_2)\sigma = t_1\sigma \to t_2\sigma$

Type principal t pour E si

- E de type t
- Pour tout type u de E il existe σ t. q. $t\sigma = u$

Variables de type

Théorème.

Le type principal est unique à renommage des variables près

Décidabilité ?

Variables de type

Théorème.

Le type principal est unique à renommage des variables près

E typable : « solution » pour tout E' sous-terme de E de

$$\begin{cases} \alpha_{E_1} = \alpha_{E_2} \to \alpha_{E'} & \text{pour} \quad E' = (E_1\ E_2) \\ \alpha_{E'} = \alpha_x \to \alpha_{E_1} & \qquad\quad E' = \lambda x.E_1 \\ \alpha_{E'} = \alpha_x & \qquad\quad E' = x \end{cases}$$

Théorème.

La simple typabilité d'un λ-terme est décidable

Si un λ-terme est typable alors il admet un type principal

$\rightsquigarrow$ unification

β-réduction des λ-termes typables

Théorème.

Si $E \to_\beta^\star E'$ et $E : t$ alors $E' : t$

Corollaire.

Si $E \to_\beta^\star E_1$ et $E \to_\beta^\star E_2$ avec E_1, E_2 bien typés,
alors il existe E_3 bien typé t. q. $E_1 \to_\beta^\star E_3$ et $E_2 \to_\beta^\star E_3$
Dém. par Church-Rosser pur et th.

β-réduction des λ-termes typables

Proposition.

Si $\begin{cases} U & : & t \\ x & : & t' \\ V & : & t' \end{cases}$ alors $U[x \mapsto V] : t$

Dém. par induction sur U.

Normalisation

Théorème. (Normalisation faible)

Tout λ-terme typable est faiblement normalisable

Théorème.

(Normalisation forte) Tout λ-terme typable est fortement normalisable

Corollaire.

Les fonctions récursives partielles ne sont pas représentables en λ-calcul simplement typé.

Extensions du λ-calcul simplement typé

Manque d'expressivité $\rightsquigarrow$ extensions

- Système T (Gödel, $\simeq$ 40) ;
- Système F (Girard, $\simeq$ 70) ;
- Calcul des constructions ($\simeq$ 80) ;
- ...

Inférence ?

Vérification ?

Système T

Ajout de récursion (syntaxe supp.)

Types : $B = \{int, bool\}$

Termes :

- 0
- $S(t)$ où t λ-terme
- *true* et *false*
- rec(n, t, u) où n, t, u λ-termes
- *if*(b, t, u) où b, t, u λ-termes

Système T

Sémantique : rec opérateur de récursion

Ajout à β de la ι-réduction :

$$\begin{array}{lcl} \text{rec}(0, t, u) & \to_\iota & t \\ \text{rec}(S(n), t, u) & \to_\iota & ((u\ \text{rec}(n, t, u))\ n) \end{array}$$

$$\begin{array}{lcl} \textit{if}(\textit{true}, t, u) & \to_\iota & t \\ \textit{if}(\textit{false}, t, u) & \to_\iota & u \end{array}$$

Système T

Nouvelles règles de typage :

$$\frac{}{\Gamma \vdash 0 : int} \qquad \frac{\Gamma \vdash n : int}{\Gamma \vdash S(n) : int}$$

$$\frac{}{\Gamma \vdash \textit{true} : bool} \qquad \frac{}{\Gamma \vdash \textit{false} : bool}$$

$$\frac{\Gamma \vdash n : int \quad \Gamma \vdash t : \tau \quad \Gamma \vdash u : \tau \to (int \to \tau)}{\Gamma \vdash \text{rec}(n, t, u) : \tau}$$

$$\frac{\Gamma \vdash b : bool \quad \Gamma \vdash t : \tau \quad \Gamma \vdash u : \tau}{\Gamma \vdash \textit{if}(b, t, u) : \tau}$$

Système T

Résultats :

- $\to_\beta \cup \to_\iota$ confluente
- $\to_\beta \cup \to_\iota$ fortement normalisante
- Représente des fonctions récursives totales

Exemple :

$$\begin{array}{lcl} ADD & = & \lambda x.\lambda y.\text{rec}(y, x, \lambda z.\lambda t.S(z)) \\ MULT & = & ? \\ FACT & = & ? \end{array}$$