INF340
λ-calcul non typé

J. Leneutre

Introduction (1)

- Le *λ-calcul a été introduit en 1936 par A. Church*

- <u>Rappel</u> : en théorie des combinateurs,
  on transforme une définition implicite d' une fonction f ($f\, x_1 \ldots x_n = E(x_1, \ldots, x_n)$)
  en une définition explicite f^e utilisant des combinateurs ($f^e = \lambda^+ x_1 \ldots \lambda^+ x_1.E(x_1, \ldots, x_n)$),
  en réalisant l' abstraction des variables en paramètre.

- <u>Problème</u> : pour définir en logique combinatoire la fonction qui à x associe la dérivée de la fonction (notée $\underline{D}$) qui à y associe x*y, on aimerait calculer directement:

$$\lambda^+ x.(\underline{D}(\lambda^+ y.\ x \underline{*} y))$$

mais il est impossible d' imbriquer les abstractions

Introduction (2)

- En théorie des combinateurs on est obligé d'écrire :

  $(\lambda^{+}f\ \lambda^{+}\ x.\underline{D}\ (f\ x))\ \ (\lambda^{+}z\ \lambda^{+}\ y.z\underline{*}y)$

  $:= \lambda^{+}\ x.\underline{D}\ ((\lambda^{+}z\ \lambda^{+}\ y.z\underline{*}y)\ x)$

  $:= \lambda^{+}\ x.\underline{D}\ (\lambda^{+}\ y.x\underline{*}y)$

- En λ-calcul on systématise le principe d'abstraction:
  - on utilise des variables liées : $f(x)$ sera noté $\lambda x.f\ x$
  - Exemple : $f(x)=x\underline{+}1$ sera noté $\lambda x.x\underline{+}1$
  - $\lambda x.(\underline{D}(\lambda y.\ x\underline{*}y))$ sera un terme du calcul
  - et on définira une relation de réduction > telle que

    $(\lambda f\lambda x.\underline{D}\ (f\ x))\ (\lambda z\lambda y.\ z\underline{*}y) > \lambda x.(\underline{D}(\lambda y.\ x\underline{*}y))$

Termes du λ-calcul

- Déf: l'ensemble des *termes du λ-calcul non typé* est le plus petit ensemble vérifiant :
  - les variables x,y,z, … sont des termes
  - si u et v sont des termes (u v) est un terme (*application*)
  - si t est un terme λx.t est un terme (*abstraction*)

- Conventions:
  - t, u, v (éventuellement indicés) désigneront des termes,
  - (t u) sera noté t u,
  - $(\ldots((u_1\ u_2)\ u_3)\ldots\ u_n)$ sera noté $u_1\ u_2\ u_3\ldots\ u_n$
  - λx.(t u) sera noté λx.t u,
  - $\lambda x_1.(\ldots(\lambda x_n.t)\ldots)$ sera noté $\lambda x_1\ldots x_n.t$

- Exemples: x, λxy.xyz, xλxy.x, λx.z sont des termes

- Question : quels sont les termes correspondant à I, K, et S ?

Variables libres

- Un terme du λ-calcul comprend des variables libres et des variables liées

- Déf: l'ens. des *occurrences libres* d'une variable x dans un terme t est :
  - si $t \equiv x$, le singleton contenant l'unique occurrence de x
  - si $t \equiv (u\ v)$, la réunion des occurrences libres de x dans u et dans v
  - si $t \equiv \lambda x.u$, l'ensemble vide (x est *liée* dans u)
  - si $t \equiv \lambda y.u$, avec $x \neq y$ l'ensemble des occurrences libres de x dans u

- Une *variable libre* est une variable ayant au moins une occurrence libre

- Un *terme clos* est un terme sans variable libre

Variables libres et substitution simple

- La notion de *substitution simple* introduite en théorie des combinateurs ($[t/x]u$ dans la procédure d'abstraction), correspond en λ-calcul au remplacement de chaque occurrence libre de x dans u par t.

- Cette notion ne convient plus en λ-calcul:

$$[\lambda y.yz/x]\ \lambda z.xz > \lambda z.(\lambda y.yz)z$$

  - l'occurrence de z initialement libre devient liée, elle est «capturée»

- La substitution en λ-calcul nécessite éventuellement un renommage de variables liées.

Termes *α-équivalents*

- <u>Déf</u>: la relation d'*α-équivalence* ou *α-conversion* notée $=_\alpha$ est la plus petite relation t.q. :
  - si $t \equiv x$, alors $t =_\alpha t'$ ssi $t' \equiv x$
  - si $t \equiv uv$, alors $t =_\alpha t'$ ssi $t' \equiv u'v'$ avec $u =_\alpha u'$ et $v =_\alpha v'$
  - si $t \equiv \lambda x.u$, alors $t =_\alpha t'$ ssi $t' \equiv \lambda y.u'$ avec $[z/x]u =_\alpha [z/y]u'$ pour $z \notin t, t'$

  ($t \equiv_\alpha t'$ si t' est obtenue à partir de t en renommant certaines variables liées)

- La substitution simple ne satisfait pas la propriété suivante:

$$\text{si } t =_\alpha u \quad \text{alors} \quad [v/x]t =_\alpha [v/x]u$$

contre-exemple

$t \equiv \lambda z.xz$, $u \equiv \underline{\lambda z'.xz'}$, $v \equiv \lambda y.yz$

$[v/x]t = \lambda z.(\lambda y.yz)\, z$

$[v/x]u = \underline{\lambda z'.(\lambda y.yz)\, z'}$

Substitution en λ-calcul

- En λ-calcul le résultat de la substitution de la variable x dans le terme u par le terme t, doit être un terme $\langle t/x\rangle v$ avec $v =_\alpha u$ et tel qu'aucune variable liée de v ne soit libre dans t, plus précisément

- <u>Déf</u> : la substitution en λ-calcul est notée $\langle t/x\rangle u$ et définie par (ci-dessous $y \neq x$) :
  - $\langle t/x\rangle x \equiv t$ et $\langle t/x\rangle y \equiv y$
  - $\langle t/x\rangle uv \equiv \langle t/x\rangle u\ \langle t/x\rangle v$
  - $\langle t/x\rangle \lambda x.u \equiv \lambda x.u$ et $\langle t/x\rangle \lambda y.u \equiv \lambda y.u$ si x non libre dans u
  - $\langle t/x\rangle \lambda y.u \equiv \lambda y.\langle t/x\rangle u$ si x libre dans u et y non libre dans t
  - $\langle t/x\rangle \lambda y.u \equiv \lambda z.\langle t/x\rangle\langle z/y\rangle u$ si x libre dans u et y libre dans t, avec z nouvelle variable

- <u>Prop</u> : nous avons $\lambda x.t =_\alpha \lambda y.\langle y/x\rangle t$,

β-contraction

- <u>Déf</u> : On appelle *β-rédex* un terme de la forme $(\lambda x.t)u$, et son *β-contracté* associé est le terme $<u/x>t$.

- <u>Exemple</u> : le *β-rédex* $(\lambda x.xy(xy))\ z$ a pour *β-contracté* $zy(zy)$

- <u>Déf</u> (*β-contraction*): Si v est un terme contenant un rédex r, et si v' est le terme obtenu en remplaçant r par son contracté dans v, on dit que v' est obtenu par *β-contraction* de v, et on note $v \rhd_{1\beta} v'$.

- <u>Exemple</u> : $\lambda x.(\lambda y.xy(xy))\ z \rhd_{1\beta} \lambda x.xz(xz)$

β-réduction et β-équivalence

- Déf : une *β-réduction* d'un terme v est une suite finie ou infinie de β-contraction et d'α-conversion. On note $v \rhd_\beta v'$ s'il existe une β-réduction (finie) allant de v à v'.

- Exemples : $(\lambda xz.xz)\ (\lambda y.yz)\ x =_\alpha (\lambda xz'.xz')\ (\lambda y.yz)\ x$

$$\rhd_{1\beta}\ (\lambda z'.\ ((\lambda y.yz)z'))\ x$$
$$\rhd_{1\beta}\ (\lambda y.yz)x$$
$$\rhd_{1\beta}\ xz$$

- Exercice : réduire $(\lambda xy.yx)\ y\ (\lambda x.xy)$

- Déf : la *β-équivalence* ou *β-conversion*, est la plus petite relation d'équivalence contenant $\rhd_\beta$ et est notée $=_\beta$.

- Exercice : montrer que $(\lambda xy.y)\ z =_\beta (\lambda xy.y)\ (\lambda z.z)$

β-normalisation

- Déf (termes *β-normaux*) : un terme est en forme *β-normale* s'il ne contient aucun rédex.

- Déf : un terme v est *β-normalisable* s'il existe v' en forme *β-normale* tel que $v \rhd_\beta v'$.

- Exemples : (λxy.yx) y (λx.xy) est normalisable, mais pas (λx.xx)(λx.xx)

- Déf : un terme v est *fortement β-normalisable* s'il n'existe pas de β-réduction infinie de v.

- Prop : tout terme fortement β-normalisable est faiblement β-normalisable.

- Exercice : montrer que (λx.y) ((λz.zz) (λz.zz)) est faiblement normalisable mais n'est pas fortement normalisable.

Résultats de normalisation

- Lemme (confluence) :

  Si $t \rhd_\beta u$ et $t \rhd_\beta v$ alors il existe w t.q. $u \rhd_\beta w$ et $v \rhd_\beta w$

- Théorème (Church-Rosser) :

  Si $t =_\beta u$ alors il existe v t.q. $t \rhd_\beta v$ et $u \rhd_\beta v$

- Corrollaire (unicité de la forme normale) :

  la forme β-normale d'un terme est unique si elle existe.

- Théorème :

  un terme a une forme β-normale u ssi la réduction «la plus à gauche» se termine en u

βη-équivalence

- Prop: soient x, x' des variables resp. non libres dans les termes t et t'.

  Si $\lambda x.(tx) =_\alpha \lambda x'.(t'x')$ alors $t =_\alpha t'$

- Déf (η-*rédex*): on appelle η-*rédex* un terme de la forme $\lambda x.tx$ avec $x \notin VL(t)$, son η-*contracté* est t.

- Déf : Une *βη-réduction* d'un terme v est une suite finie ou infinie de β-contraction ou η-contraction. On note $v \triangleright_{\beta\eta} v'$ s'il existe une βη-réduction (finie) allant de v à v'.

- Déf : la *βη-équivalence* ou *βη-conversion*, est la plus petite relation d'équivalence contenant $\triangleright_{\beta\eta}$ et est notée $=_{\beta\eta}$

Stratégies de réduction

- Stratégie interne (appel par valeur) :

$$(\lambda x.\underline{u})\ v \rhd_\beta (\lambda x.u')\ \underline{v} \rhd_\beta \underline{(\lambda x.u')\ v'} \rhd_\beta <v'/x>u'$$

  - interne droite : réduit les rédex de la partie gauche uniquement si il n'y en a plus à droite
  - interne gauche : gauche puis droite

- Stratégie externe (appel par nom) :

$$\underline{(\lambda x.u)\ v} \rhd_\beta <v/x>u \rhd_\beta \ldots$$

Pouvoir d'expression

- Comme pour la théorie des combinateurs on peut représenter les entiers, les booléens, couples, …, et les opérations associées.
- Exemple : entiers de church $\underline{n} \equiv \lambda f.\lambda x.f^n x$
- On peut également représenter les fonctions récursives grâce à un combinateur de point fixe.
- Théo : Toute fonction récursive partielle de N^n dans N est représentable par un terme du λ-calcul.

Exercice

- On considère le terme $T \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
- T est-il en forme normale? Faiblement normalisable ? Fortement normalisable ?
- Vérifier que $T \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ est un combinateur de point fixe
- A-t'on $TF \triangleright_\beta F(TF)$?

- Trouver un terme T' β-équivalent à T , avec T' de la forme

$$(\lambda yf.t)\ (\lambda zx.u)$$

  où t et u sont des termes ne contenant pas d'abstraction

- En utilisant le terme T' de la question précédente, exprimer T sous la forme d'une expression de la théorie des combinateurs construite uniquement avec les combinateurs W, S et B.

λ-calcul et Scheme

- Scheme est un langage de programmation de la famille Lisp reposant sur le λ-calcul simple.

λ-calcul	**Scheme**
Variable x	Symbole `x`
Abstraction λx.corps	`(lambda (x) corps)`
Application (λx.corps) e	`(let ((x e)) corps)`
Entiers de Chuch <u>n</u>	Entier `n`
Booléens <u>t</u> <u>f</u>	`#t #f`
test e1 e2	`(if test e1 e2)`
Opérations sur les listes	`Car`, `cdr`, `cons`, `‘()`, `null?`
Réduction	Evaluation
Stratégie de réduction	Appel par valeur