FLP

Lambda calcul

- sistem care permite manipularea funcțiilor ca expresii

ex. f - fct. $x \mapsto x^2$. $A = f(5)$ ⇒ în lambda calcul scriem doar

$$A = (\lambda x . x^2)(5)$$

funcția care îl duce pe x în x^2

x - locală / legată în termenul $\lambda x . x^2$

- funcții de nivel înalt = fct. ale căror intrări/ieșiri sunt tot fct.

ex. $f \circ f \Leftrightarrow \lambda x . f(f(x))$

$f \mapsto f \circ f \Leftrightarrow \lambda f . \lambda x . f(f(x))$

$((\lambda f . \lambda x . f(f(x)))(\lambda y . y^2))(5) = 625$

x îl primește pe 5

funcția primită de f este $\lambda y . y^2$

- poate fi
  - fără tipuri
    - nu specificăm tipul expresiei / domeniul sau codomeniul funcțiilor
    - flexibilitate maximă, dar riscant
  - cu tipuri simple
    - specificăm mereu tipul expresiei
    - scăpăm de expresiile de forma $f(f)$
  - cu tipuri polimorfice
    - o situație intermediară celor 2
    - specificăm că o expresie are tipul $X \mapsto X$, fără a specifica cine e X

- funcție calculabilă
  - dacă e recursivă
  - dacă se poate scrie ca lambda termen
  - dacă poate fi calculată de o mașină Turing

Lambda termen = variabilă | aplicare | abstractizare

$$M, N ::= x \mid (M\,N) \mid (\lambda x.M)$$

ex. x, y, z, $(x\,y)$, $(\lambda x.x)$

V = mulțimea infinită a variabilelor
A = alfabetul format din elem. din V și simbolurile $(\ )\ \lambda\ .$
A^* = mulțimea cuvintelor infinite pt alfabetul A

Mulțimea lambda termenilor e cea mai mică submulțime $\Lambda \subseteq A^*$ a.î.:
variabilă $V \subseteq \Lambda$
aplicare dacă $M, N \in \Lambda \Rightarrow (M\,N) \in \Lambda$
abstractizare dacă $x \in V$ și $M \in \Lambda \Rightarrow (\lambda x.M) \in \Lambda$

Convenții:
- se elimină parantezele exterioare
- aplicarea e asociativă la stânga
  ex. $MNP \leadsto (M\,N)\,P$
  $f\,x\,y\,z \leadsto ((f\,x)\,y)\,z$
- partea de după punct se extinde la dreapta
  $\lambda x.MN \leadsto \lambda x.(M\,N)$
- λ pot fi comprimați
  $\lambda xyz.M \leadsto \lambda x.\lambda y.\lambda z.M$

Variabile libere și legate

- $\lambda - . -$ e operator de legare (binder)
- x din $\lambda x. -$ e variabilă de legare (binding)
- N din $\lambda x.N$ e domeniul (scopa) de legare a lui x
- toate aparițiile din N ale lui x sunt legate
- O apariție care nu e legată $\Rightarrow$ e liberă
- Termen fără variabile libere se mai numește închis sau combinator

$FV(M)$ = mulțimea variabilelor libere din termenul M

$FV(x) = \{x\}$ $\quad FV(MN) = FV(M) \cup FV(N)$ $\quad FV(\lambda x.M) = FV(M) \setminus \{x\}$

x, y - variabile, M - termen

$M\langle y/x\rangle$ = redenumirea lui x cu y în M

$$x\langle y/x\rangle \equiv y$$
$$z\langle y/x\rangle \equiv z \quad , x \neq z$$
$$(MN)\langle y/x\rangle = (M\langle y/x\rangle)(N\langle y/x\rangle)$$
$$(\lambda x.M)\langle y/x\rangle = \lambda y.(M\langle y/x\rangle)$$
$$(\lambda z.M)\langle y/x\rangle = \lambda z.(M\langle y/x\rangle) \quad , x \neq z$$

α-echivalență $\Leftrightarrow$ $\forall$ termen M și $\forall$ variabilă y care NU apare în M

$$\lambda x.M \equiv_\alpha \lambda y.(M\langle y/x\rangle)$$

Convenție: variabilele legate sunt redenumite pentru a fi distincte

Substituție

Vrem să substituim variabile cu lambda termeni

⚠ Vrem să înlocuim doar variabile libere

⚠ Nu vrem să legăm variabile libere neintenționat $\Rightarrow$ redenumim variabilele legate înainte de substituție (dacă avem de ex. o apariție a lui x legată și altă apariție a lui x liberă)

β-reducții

Convenție: 2 termeni sunt egali, dacă sunt α-echivalenți

β-reducție = procesul de a evalua lambda termeni prin "pasarea de argumente funcțiilor"

β-redex = un termen de forma $(\lambda x.M)N$

redusul unui redex $(\lambda x.M)N$ este $M[N/x]$

- reducem lambda termeni prin găsirea unui subtermen care e redex, și apoi înlocuirea acelui redex cu redusul său
- repetăm până nu mai sunt redexuri

formă normală = lambda termen fără redexuri

Un pas de β-reducție ($\to_\beta$) satisface:

1) $(\lambda x.M)N \to_\beta M[N/x]$
2) $M \to_\beta M' \Rightarrow MN \to_\beta M'N$
3) $N \to_\beta N' \Rightarrow MN \to_\beta MN'$
4) $M \to_\beta M' \Rightarrow \lambda x.M \to_\beta \lambda x.M'$

ex.

$(\lambda x.y)(\underline{(\lambda z.zz)(\lambda w.w)}) \to_\beta (\lambda x.y)((zz)[\lambda w.w/z])$
$\equiv (\lambda x.y)((z[\lambda w.w/z])(z[\lambda w.w/z]))$
$\equiv (\lambda x.y)(\underline{(\lambda w.w)(\lambda w.w)})$
$\to_\beta \underline{(\lambda x.y)(\lambda w.w)}$
$\to_\beta y$

Există lambda termeni ce nu pot fi reduși la o β-formă normală (evaluarea nu se termină).

Există lambda termeni care deși pot fi reduși la o formă normală pot să nu o atingă niciodată.

⚠ Contează strategia de evaluare

Notăm cu $M \to\!\!\!\!\to_\beta M'$ faptul că M poate fi β-redus la M' în 0 sau mai mulți pași.

M e slab normalizabil dacă $\exists N$ în formă normală a.î. $M \to\!\!\!\!\to_\beta N$

M e puternic normalizabil dacă nu există reduceri infinite care încep din M.

ex. $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$ e puternic normalizabil

$(\lambda xy.y)((\lambda x.xx)(\lambda x.xx))(\lambda z.z)$ e slab normalizabil, dar nu puternic normalizabil

Teoremă

Dacă $M \to_\beta M_1$ și $M \to_\beta M_2$, atunci $\exists M'$ a.î. $M_1 \to_\beta M'$ și $M_2 \to_\beta M'$

Consecință

Un lambda termen are cel mult o β-formă normală.

Strategii de evaluare

Strategia normală = leftmost - outermost
- alegem redex-ul cel mai din stânga și apoi cel mai din exterior
- dacă M_1 și M_2 sunt redex-uri și M_1 e un subtermen al lui M_2 atunci M_1 nu va fi următorul redex ales

ex. $(\lambda x y.y)((\lambda x.xx)(\lambda x.xx))(\lambda z.z) \to_\beta (\lambda y.y)(\lambda x.x)$
$\to_\beta \lambda x.x$

Strategia aplicativă = leftmost - innermost
- alegem redex-ul cel mai din stânga și apoi cel mai din interior
- dacă M_1 și M_2 sunt redex-uri și M_1 e un subtermen al lui M_2, atunci M_2 nu va fi următorul ales

ex. $(\lambda x y.y)((\lambda x.xx)(\lambda x.xx))(\lambda z.z) \to_\beta (\lambda x y.y)((\lambda x.xx)(\lambda x.xx))(\lambda z.z)$

Strategia call-by-name (CBN) = strategia normală fără a face reduceri în corpul unei λ-abstractizări

Strategia call-by-value (CBV) = strategia aplicativă fără a face reduceri în corpul unei λ-abstractizări

O valoare e un λ-termen pentru care nu există β-reducții date de strategia de evaluare considerată.
De exemplu, $\lambda x.x$ e mereu o valoare, dar $(\lambda x.x)$ I nu este.

ex.

CBV

$(\lambda x.succ\,x)((\lambda y.succ\,y)\,3) \to_\beta (\lambda x.succ\,x)(succ\,3)$
$\to (\lambda x.succ\,x)\,4$
$\to\!\!\!\!\to_\beta succ\,4$
$\to 5$

CBN

$(\lambda x.succ\,x)((\lambda y.succ\,y)\,3) \to\!\!\!\!\to_\beta succ\,((\lambda y.succ\,y)\,3)$
$\to\!\!\!\!\to_\beta succ\,(succ\,3)$
$\to succ\,4$
$\to 5.$

Booleeni

$T \triangleq \lambda xy.x$ (din cele 2, alegem primul)
$F \triangleq \lambda xy.y$ (——— u ——— al doilea)

Avem $if = \lambda btf.$ $\begin{cases} t, & \text{if } b = true \\ f, & \text{if } b = false \end{cases}$. $Ttf \to\!\!\!\!\to_\beta t$ și $Ftf \to\!\!\!\!\to_\beta f$

$\Rightarrow$ $if \triangleq \lambda btf.\ btf$
$and \triangleq \lambda b_1 b_2.\ if\ b_1\ b_2\ F$
$or \triangleq \lambda b_1 b_2.\ if\ b_1\ T\ b_2$
$not \triangleq \lambda b_1.\ if\ b_1\ F\ T$

Numere naturale

Numeralul Church pentru $n \in \mathbb{N}$ este $\bar{n}$.
f^n = compunerea lui f de n ori

$\bar{0} \triangleq \lambda fx.f^0x = \lambda fx.x$
$\bar{1} \triangleq \lambda fx.f^1x = \lambda fx.fx$
$\bar{n} \triangleq \lambda fx.f^nx = \lambda fx.\underbrace{f(f(\dots(f}_{n\text{ ori}}x)\dots))$

$Succ \triangleq \lambda nfx.f(nfx)$
$add \triangleq \lambda uvfx.uf(vfx)$

$mul \triangleq \lambda m\, n\, .\, m\,(add\ n)\ \bar{0}$

$exp \triangleq \lambda m\, n\, .\, m\,(mul\ n)\ \bar{1}$

$iszero \triangleq \lambda n\, x\, y\, .\, n\,(\lambda z\, .\, y)\, x$

Puncte fixe

Dacă F și M sunt λ-termeni, spunem că M e pct fix al lui F dacă $F\,M =_\beta M$.

Teoremă

În lambda calculul fără tipuri, orice termen are un pct. fix.

Combinatorii de puncte fixe sunt termeni închiși care "construiesc" un punct fix pentru un termen arbitrar.

ex.

Combinatorul de punct fix al lui Curry

$Y \triangleq \lambda y\, .\, (\lambda x\, .\, y\,(x\,x))\,(\lambda x\, .\, y\,(x\,x))$

$\forall$ termen F, YF e punct fix al lui F, deoarece

$YF \twoheadrightarrow_\beta F\,(YF)$

Combinatorul de punct fix al lui Turing

$\Theta \triangleq (\lambda x\, y\, .\, y\,(x\,x\,y))\,(\lambda x\, y\, .\, y\,(x\,x\,y))$

Lambda calcul cu tipuri simple

Tip simplu = V | $T \to T$

variabilă — tip săgeată

tipuri pentru funcții

- Parantezele în tipul săgeată sunt asociative la dreapta

ex.

$\alpha_1 \to \alpha_2 \to \alpha_3 \to \alpha_4 = (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_4)))$

$x_1\, x_2\, x_3\, x_4 = (((x_1\, x_2)\, x_3)\, x_4)$

$M : \tau$, M are tip τ

Variabilă: variabila x are tipul τ : $x : \tau$

Aplicare: $M\ N$, M are ~~intrare~~ tip funcție, iar N are tip adecvat pt această fct;

$M : \tau \to \sigma$ și $N : \tau$, atunci $MN : \sigma$

Abstractizare: $M : \sigma$ și $x : \tau$, atunci $\lambda x . M : \tau \to \sigma$

Asociere explicită = Church-typing
- tipurile variabilelor sunt explicit stabilite
- tipurile termenilor complecși se obțin natural

Asocierea implicită = Curry-typing
- lăsăm variabilele deschise, fără tip
- termenii typeable vor fi căutați, putând presupune "ghicirea" lor

Λ_T = mulțimea λ-termenilor cu pre-tipuri Λ_T

$\Lambda_T = x \mid \Lambda_T\ \Lambda_T \mid \lambda x : T . \Lambda_T$

Afirmație: $M : \tau$, $M \in \Lambda_T$ (subiect) și $\tau \in T$ (tip)

Declarație: Afirmație unde subiectul e variabilă ($x : \tau$)

Context: Listă de declarații cu subiecți diferiți

Judecată: $\Gamma \vdash M : \tau$, unde Γ context și $M : \tau$ e afirmație

Reguli:

$$\frac{}{\Gamma \vdash x : \tau} \text{ dacă } x : \tau \in \Gamma \ (\text{var}) \qquad \frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash (\lambda x : \tau . M) : \tau \to \sigma} \ (\text{abs})$$

$$\frac{\Gamma \vdash M : \tau \to \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \ (\text{app})$$

Un termen M e legal dacă ∃ un context Γ și un tip τ a.î. $\Gamma \vdash M : \tau$

Ce probleme putem rezolva?

- Type checking: putem găsi o derivare pt: context ⊢ term : type
- Typability: dacă un termen e legal
- Term Finding: dându-se un context și un tip, să stabilim dacă ∃ un termen cu acel tip în contextul dat

⚠ Toate acestea sunt decidabile pt. calculul Church $\lambda\!\rightarrow$.

Nu mai avem recursie nelimitată, dar avem recursie primitivă (nr. de iterații e cunoscut dinainte).

Alte tipuri:

$T = V \mid T \to T \mid Unit \mid Void \mid T \times T \mid T + T$ (Void: nu are înhabitant)

$\Lambda_T = x \mid \Lambda_T\ \Lambda_T \mid \lambda x{:}T.\ \Lambda_T \mid unit \mid \langle \Lambda_T, \Lambda_T \rangle \mid fst\ \Lambda_T \mid snd\ \Lambda_T$

$\mid Left\ \Lambda_T \mid Right\ \Lambda_T \mid case\ \Lambda_T\ of\ \Lambda_T\ ;\ \Lambda_T$

Corespondența Curry - Howard

Teoria tipurilor	Logică
tipuri	formule
termeni	demonstrații
înhabitarea tipului τ	demonstrație a lui τ
tip produs	conjuncție
tip funcție	implicație
tip sumă	disjuncție
tip void	false
tip unit	true