FLP Cheat Sheet

Lambda calcul - elemente de bază

Funcții

Fie $f,\ g : X \to Y$, $f(x) = x^2 - 1$, $g(x) = (x-1)(x+1)$

extensional egale: $f(x) = g(x),\ \forall x \in X$. (DA)

intensional egale: sunt definite de aceeași formulă. (NU)

Termeni lambda

$M, N ::= x$ $(variabilă)$

$\mid (MN)$ $(aplicare)$

$\mid (\lambda x.M)$ $(abstractizare)$

Convenții

1. aplicarea e asociativă la stânga:
$$f\ \ x\ \ y\ \ z = ((f\ \ x)\ y)\ \ z$$
2. corpul abstractizării se extinde la dreapta:
$$\lambda x.M\ N = \lambda x.(M\ N)$$
3. mai mulți λ se comprimă:
$$\lambda xyz.M = \lambda x.\lambda y.\lambda z.M$$

Variabile

Exemplu: $\lambda x.N$

1. operator de legare(binder): $\lambda\_.\_$
2. variabilă de legare(binding): x
3. domeniu de legare a lui x: N
4. termen fără variabile libere: închis/combinator

Exemplu: $M \equiv (\lambda x.xy)(\lambda y.yz)$

1. Mulțimea variabilelor legate: $\{x,\ y\}$
2. Mulțimea variabilelor libere *(FV(M))*: $\{y,\ z\}$

Obs. $FV(x) = x$

$FV(M\ N) = FV(M) \cup FV(N)$

$FV(\lambda x.M)\ = FV(M) \setminus x$

α-echivalența

$$(refl)\ \frac{}{M = M} \qquad (cong)\ \frac{M = M' \qquad N = N'}{MN = M'N'}$$

$$(symm)\ \frac{M = N}{N = M} \qquad (\xi)\ \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

$$(trans)\ \frac{M = N \qquad N = P}{M = P} \qquad (\alpha)\ \frac{y \notin M}{\lambda x.M = \lambda y.(M\{y/x\})}$$

Substituții

$[Variabilă]$ $(x[u/x] = u)$ sau $(y[u/x] = y$, dacă $x \neq y)$

$[Aplicare]$ $(M\ N)[u/x] = (M[u/x]\ N[u/x])$

$[Abstractizare]$

$$\lambda y.t[u/x] = \begin{cases} \lambda y.(t[u/x]),\ y \neq x,\ y \notin FV(u) \\ \lambda y'.(t\langle y'/y\rangle[u/x]),\ y \neq x,\ y \in FV(u) \end{cases}$$

Lambda calcul - β-reducții

- **β-redex**: termen de forma $(\lambda x.M)\ N$
- **Redusul** redex-ului $(\lambda x.M)N : M[N/x]$
- **Forma normală**: un lambda termen fără redex-uri
- Daca $\exists$ N în forma normală a.î. $M \twoheadrightarrow_\beta N \Rightarrow M$ e **slab normalizabil**.
- Dacă $\nexists$ reduceri infinite care încep din $M \Rightarrow M$ e **puternic normalizabil**.
- Algoritm: reducem lambda termeni prin găsirea unui subtermen care e redex, îl înlocuim cu redusul său și ciclăm până nu mai sunt redex-uri.

β-reducții

$$(\beta)\ \frac{}{(\lambda x.M)N \to_\beta M[N/x]}$$

$$(cong_1)\ \frac{M \to_\beta M'}{MN \to_\beta M'N}$$

$$(cong_2)\ \frac{N \to_\beta N'}{MN \to_\beta MN'}$$

$$(\xi)\ \frac{M \to_\beta M'}{\lambda x.M \to_\beta \lambda x.M'}$$

Exemplu:

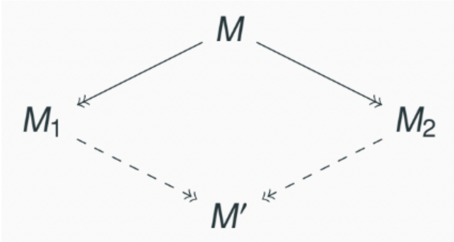
$$\begin{aligned} (\lambda x.y)\,(\underline{(\lambda z.zz)\,(\lambda w.w)}) &\to_\beta (\lambda x.y)\,((z\,z)[\lambda w.w/z]) \\ &\equiv (\lambda x.y)\,((z[\lambda w.w/z])\,(z[\lambda w.w/z])) \\ &\equiv (\lambda x.y)\,(\underline{(\lambda w.w)\,(\lambda w.w)}) \\ &\to_\beta \underline{(\lambda x.y)\,(\lambda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

Observații:

- reducerea unui redex poate crea/șterge redex-uri
- găsirea unei forme normale depinde de ordinea reducerii redex-urilor

Teorema Church-Rosser

Dacă $M \twoheadrightarrow_\beta M1$ și $M \twoheadrightarrow_\beta M2$ atunci există M' astfel încât $M1 \twoheadrightarrow_\beta M'$ și $M2 \twoheadrightarrow_\beta M'$.



Consecință:

Un λ-termen are cel mult o β-formă normală (modulo α-echivalență).

Strategii de evaluare

Strategia normală: leftmost-outermost

$\left.\begin{array}{l} M1,\ M2 \text{ redex-}uri \\ M1 = \text{subtermen } M2 \end{array}\right\} \to$ M1 nu e viitorul redex ales

Strategia aplicativă: leftmost-innermost

$\left.\begin{array}{l} M1,\ M2 \text{ redex-}uri \\ M1 = \text{subtermen } M2 \end{array}\right\} \to$ M2 nu e viitorul redex ales

Strategia call-by-name - strategia normală fără a face reduceri în corpul unei λ – *abstractizări*.

$$\begin{aligned} (\lambda x.succ\ x)\,((\lambda y.succ\ y)\,3) &\twoheadrightarrow_\beta succ\,((\lambda y.succ\ y)\,3) \\ &\twoheadrightarrow_\beta succ\,(succ\,3) \\ &\to succ\,4 \\ &\to 5 \end{aligned}$$

Strategia call-by-value - strategia aplicativă fără a face reduceri în corpul unei λ – *abstractizări*.

$$\begin{aligned} (\lambda x.succ\ x)\,((\lambda y.succ\ y)\,3) &\twoheadrightarrow_\beta (\lambda x.succ\ x)\,(succ\,3) \\ &\to (\lambda x.succ\ x)\,4 \\ &\twoheadrightarrow_\beta succ\,4 \\ &\to 5 \end{aligned}$$

Expresivitatea λ-calculului

Booleeni

$T \triangleq \lambda xy.x$ | if $\triangleq \lambda btf.b\ t\ f$ | or $\triangleq \lambda xy.\text{if } x\ T\ y$

$F \triangleq \lambda xy.y$ | not $\triangleq \lambda x.\text{if } x\ F\ T$ | and $\triangleq \lambda xy.\text{if } x\ y\ F$

Numere naturale

Numeralul Church $\bar{n} \triangleq \lambda fx.f^n x$

Succ $\triangleq \lambda nfx.f\ (n\ f\ x) \implies$ Succ $\bar{n} = \overline{n+1}$

add $\triangleq \lambda mn.m$ Succ n   mul $\triangleq \lambda mn.m$ (add n)

exp $\triangleq \lambda mn.m$ (mul n)   isZero $\triangleq \lambda nxy.n\ (\lambda z.y)\ x$

Puncte fixe

$\left.\begin{array}{l} F,\ M \text{ sunt } \lambda\text{-termeni} \\ F\ M =_\beta M \end{array}\right\} \to$ M este punct fix al lui F

În λ-calcul fără tipuri, orice termen are punct fix.

Combinator de puncte fixe = termen închis care construiește un punct fix pe un termen arbitrar.

- Curry: $Y \triangleq \lambda y.(\lambda x.y\ (x\ x))\ (\lambda x.y\ (x\ x))$
- Turing: $\Theta \triangleq (\lambda xy.y\ (x\ x\ y))\ (\lambda xy.y\ (x\ x\ y))$

fact $\triangleq Y\ F$ [Y F e punct fix pentru F]

fact $\triangleq Y(\lambda fn.\text{if (isZero n)}\ (\bar{1})\ (\text{mul n (f(pred n))}))$

FLP Cheat Sheet

Lambda calcul cu tipuri simple

Mulțimea tuturor tipurilor simple T este definită prin T = V | T → T, unde V = {α, β, γ, ...} este o mulțime infinită de tipuri variabilă.

Exemplu:
- α
- $((\gamma \to \alpha) \to (\alpha \to (\beta \to \gamma)))$

[**Tipul variabilă**] Dacă $\alpha \in$ V, atunci $\alpha \in$ T.
[**Tipul săgeată**] Dacă $\delta, \tau \in$ T, atunci $(\delta \to \tau) \in$ T.
[paranteze asociative la dreapta]

[**Variabilă**] x : δ.
[**Aplicare**] Dacă M : $\delta \to \tau$ și N : $\delta \Rightarrow M\ N : \tau$.
[**Abstractizare**] Dacă x : δ, M : $\tau \Rightarrow \lambda$x.M : $\delta \to \tau$.

Dacă $\exists$ un tip δ a.î. M : $\delta \Rightarrow$ M are **tip** (e **typeable**).

Convenții

1. $y\ x$ poate avea un tip doar dacă y are un tip săgeată de forma $\delta \to \tau$ și tipul lui x se potrivește cu tipul domeniu δ. Astfel, $y\ x : \tau$.
2. Termenul $x\ x$ nu poate avea nici un tip.
   a. apariția I: x : $\delta \to \tau$.
   b. aparația II: x : δ.
   Cum orice variabilă are un unic tip, ar trebui ca $\delta \to \tau \equiv \delta$, ceea ce este imposibil.

Lambda calcul - tipuri

Lambda calcul fără tipuri

nu se specifică tipul niciunei expresii

nu se specifică domeniul/codomeniul funcțiilor

Lambda calcul cu tipuri simple

se specifică mereu tipul oricărei expresii

nu se poate aplica o funcție unui argument care are alt tip față de domeniul funcției

expresiile de forma f(f) sunt eliminate

Lambda calcul cu tipuri polimorfice

se poate specifica că o expresie are tipul X → X, fără a specifica cine de fapt este X

Sistem de deducție pentru Church $\lambda\to$

Mulțimea λ-termenilor cu pre-tipuri Λ_T :

$$\Lambda_T = x \mid \Lambda_T\ \Lambda_T \mid \lambda x : \mathrm{T}.\Lambda_T$$

- O **afirmație** este o expresie de forma M : δ, unde $M \in \Lambda_T$ și $\delta \in$ T.
- În această afirmație, M se numește **subiect**.
- O **declarație** este o afirmație de forma x : δ.
- Un **context** Γ este o listă de declarații cu subiecți diferiți.
- O **judecată** este o expresie de forma $\Gamma \vdash$ M : δ.

Sistem de deducție pentru Church $\lambda\to$

$$\frac{}{\Gamma \vdash x : \sigma}\ \text{dacă } x : \sigma \in \Gamma\ (var)$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (app)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\, M) : \sigma \to \tau}\ (abs)$$

Exemplu:
1) y : $\alpha \to \beta$
2) z : α
$\lambda y.\lambda z.yz$ are tipul $(\alpha \to \beta) \to \alpha \to \beta$ în contextul vid.

1.	$y : \alpha \to \beta, z : \alpha \vdash y : \alpha \to \beta$	(var)
2.	$y : \alpha \to \beta, z : \alpha \vdash z : \alpha$	(var)
3.	$y : \alpha \to \beta, z : \alpha \vdash (yz) : \beta$	(app) cu 1 și 2
4.	$y : \alpha \to \beta \vdash (\lambda z : \alpha.\, yz) : \alpha \to \beta$	(abs) cu 3
5.	$\emptyset \vdash (\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta$	(abs) cu 4

Probleme decidabile

Type-checking: se verifică posibilitatea de găsire a unei derivări pentru [*context* $\vdash$ *term* : *type*] .
Well-typedness (Typability): se verifică dacă un termen este legal [? $\vdash$ *term* : ?] .
Type Assignment: se găsește tipul când contextul este dat [*context* $\vdash$ *term* : ?] .
Term Finding: având un context și un tip anumit, se verifică dacă există un termen cu acel tip, în contextul dat. [*context* $\vdash$? : *type*] .

Alte tipuri

Mulțimea tipurilor

$$\mathrm{T} = \mathrm{V} \mid \mathrm{T} \to \mathrm{T} \mid Unit \mid Void \mid \mathrm{T}\ \mathrm{x}\ \mathrm{T} \mid \mathrm{T} + \mathrm{T}$$

Mulțimea λ-termenilor cu pre-tipuri Λ_T :

$$\Lambda_T = x \mid \Lambda_T\ \Lambda_T \mid \lambda x : \mathrm{T}.\Lambda_T \mid unit \mid <\Lambda_T, \Lambda_T> \mid fst \Lambda_T \mid snd\ \Lambda_T \mid Left\ \Lambda_T \mid Right\ \Lambda_T \mid case\ \Lambda_T\ of\ \Lambda_T;\ \Lambda_T$$

Teoria tipurilor	**Logica**
tipuri	formule
termeni	demonstrații
inhabitation a tipului δ	demonstrație a lui δ
tipul produs	conjucție
tipul funcție	implicație
tipul sumă	disjuncție
tipul void	false
tipul unit	true

Church-typing

Tip unic explicit stabilit pentru fiecare variabilă

Exemplu: Tipul expresiei (λzu.z) (y x) - ?, știind că:
1)x : $\alpha \to \alpha$ 2)y : $(\alpha \to \alpha) \to \beta$ 3)z : β 4)u : γ

- Aplicare (1) și (2) $\Rightarrow$ (5): y x : β.
- Abstractizare (4) și (3) $\Rightarrow$ (6): λu.z : $\gamma \to \beta$.
- Abstractizare (3) și (6) $\Rightarrow$ (7): λzu.z : $\beta \to \gamma \to \beta$.
- Aplicare (7) și (5) $\Rightarrow$ (λzu.z) (y x) : $\gamma \to \beta$.

Curry-typing

Nu se prescrie un tip pentru fiecare variabilă

Exemplu: Tipul expresiei M = (λzu.z) (y x)

- M e o aplicare $\Rightarrow$ λzu.z : A $\to$ B, $y\ x$: A, M : B.
- λzu.z : A $\to$ B $\Rightarrow$ z : A și λu.z : B.
- B e tipul unei abstractizări $\Rightarrow$ u : C și z : D.
- y x e o aplicare $\Rightarrow$ y : E $\to$ F, x : E și y x : F.
- z : A și z : D, y x : A și y x : F $\Rightarrow$ A $\equiv$ F.

Astfel se obține schema generală:
x : E   y : E $\to$ A   z : A   u : C   M : C $\to$ A

Se pot considera și tipuri reale în schema de mai sus:
x : β y : $\beta \to \alpha$ z : α u : δ M : $\delta \to \alpha$