lambda termenii $\rightarrow$ cea mai mică mulțime $\Lambda \subseteq A^*$ a.î.

[**Variabilă**] $V \subseteq \Lambda$

[**Aplicare**] daca $M, N \in \Lambda$, $(MN) \in \Lambda$

[**Abstractizare**] daca $x \in V$ și $M \in \Lambda$ atunci $(\lambda x. M) \in \Lambda$

- $MNP \equiv (MN)P$
- $f\,x\,y\,z \equiv ((f\,x)\,y)\,z$
- $\lambda x. MN \equiv \lambda x. (MN)$
- $\lambda xyz. M \equiv \lambda x. \lambda y. \lambda z. M$

$x\,x\,x\,x \rightarrow (((x\,x)\,x)\,x)$

$\lambda x. x\ \lambda y. y \rightarrow (\lambda x. (x\,(\lambda y. y)))$

$FV(x) = \{x\}$

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

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

- $\lambda \_ . \_$ **operator de legare (binder)**

x din $\lambda x. \_$ **variabilă de legare (binding)**

N din $\lambda x. N$ **domeniul de legare a lui x**

apariție liberă = nu este legată

termen închis = FĂRĂ variabile libere
= combinator

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

x - legată, y - liberă

y - legată, z - liberă

α **echivalență** : $\lambda x. M =_\alpha \lambda y. (M \langle y / x \rangle)$

- refl
- sym
- trans
- cong
- $(\xi)\ \dfrac{M = M'}{\lambda x. M = \lambda x. M'}$
- $(\alpha)\ \dfrac{y \notin M}{\lambda x. M = \lambda y. (M\{y/x\})}$

Substituții

1. Vrem să înlocuim var. libere $\Rightarrow x\,(\lambda xy. x)[N/x] \equiv N(\lambda xy. x)$
2. $M \equiv \lambda x. y\,x \quad N \equiv \lambda z. x\,z$

$M[N/y] \equiv (\lambda x'. y\,x')[N/y] = \lambda x'. N\,x' = \lambda x'. (\lambda z. x\,z)\,x'$

- $x[N/x] \equiv N$
- $y[N/x] \equiv y$, dacă $x \neq y$
- $(MP)[N/x] \equiv (M[N/x])(P[N/x])$
- $(\lambda x. M)[N/x]$
- $(\lambda y. M)[N/x] \equiv \lambda y. (M[N/x])$, $x \neq y$, $y \notin FV(N)$
- $(\lambda y. M)[N/x] \equiv \lambda y'. (M\langle y'/y \rangle [N/x]$
$x \neq y$, $y \in FV(N)$

β-reducții : M = N => sunt α-echivalenți

· β-redex => $(\lambda x.M)N$

· redusul lui $(\lambda x.M)N$ este $M[N/x]$

M weakly normalising, dacă ∃ N în f.m. a.î. $M \twoheadrightarrow_\beta N$.

M strong normalising, dacă ∄ reduceri finite care încep din M.

- Normal order = leftmost - outermost (CBN) - Haskell
- Applicative order = leftmost - innermost (CBV)

formă normală = λ termen FĂRĂ redex-uri

$T \triangleq \lambda xy.x$ and $\triangleq \lambda b_1 b_2.$ if $b_1 b_2 F$

$F \triangleq \lambda xy.y$ or $\triangleq \lambda b_1 b_2$ if $b_1 T b_2$

if $\triangleq \lambda btf.\, btf$ not $\triangleq \lambda b_1.$ if $b_1 F T$

Numerali Church

$\bar{n} \triangleq \lambda fx.f^n x$ succ $\triangleq \lambda nfx.f(nfx)$

add $\triangleq \lambda mnfx.mf(nfx)$

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

exp $\triangleq \lambda mn.m(\text{mul } m)\bar{1}$

isZero $\triangleq \lambda nxy.n(\lambda z.y)x$

Puncte fixe

F, M λ-term., M pct fix al lui F dacă $FM =_\beta M$

În λ-calc. FĂRĂ TIPURI, ∀ termen are un PUNCT FIX

Combinatori de puncte fixe = termeni închiși care "construiesc" un termen fix pentru un termen arbitrar

< Curry
< Turing

Tipuri simple

$V = \{\alpha, \beta, \gamma \ldots\}$ — tipuri variabila

$T = V \mid T \to T$ — {tipuri simple T}

- Tipul variabilă. Dacă $\alpha \in V$, atunci $\alpha \in T$ (tipuri bază ex. Nat, Bool)
- Tipul săgeată - Dacă $\sigma, \tau \in T$, atunci $(\sigma \to \tau) \in T$.
  (tipuri pt. funcții)
  — asociativ la dreapta (parantexe)

- $\alpha_1 \to \alpha_2 \to \alpha_3 \to \alpha_3 \equiv (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_4)))$
- $x_1 x_2 x_3 x_4 \equiv (((x_1 x_2) x_3) x_4)$

$x : \sigma$ și $x : \tau \Rightarrow \sigma \equiv \tau$ (Variabilă)

Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $MN : \tau$. (Aplicare)

Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x\, M : \sigma \to \tau$ (Abstractizare)

M are tip, daca $\exists$ un tip σ a.î. $M : \sigma$

ex: $x : \sigma \Rightarrow \lambda x\, x : \sigma \to \sigma$

$x\, x$ nu poate avea tip!

Sistem deducție pentru Church $\lambda\to$

- λ-termeni cu pre-tipuri Λ_T $\quad \Lambda_T = x \mid \Lambda_T \Lambda_T \mid \lambda x : T.\, \Lambda_T$
- afirmație = expr. de forma $M : \sigma$ (subiect, tip)
- declarație = afirmație în care subiectul este o var. ($x : \sigma$)
- context = listă de declarații cu subiecți diferiți
- judecată = expresie de forma $\Gamma : M : \sigma$, unde Γ context, $M : \sigma$ afirmație

Un termen M în calculul $\lambda\to$ este legal dacă $\exists$ un context Γ și un tip ρ a.î. $\Gamma \vdash M : \rho$

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

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau} \text{(app)} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma . M) : \sigma \to \tau} \text{(abs)}$$

- Type Checking: se reduce la a putea găsi o derivare pentru: context $\vdash$ term : type.
- Well-typedness: ? $\vdash$ term : ?
- Type Assignment: context $\vdash$ term : ?
- Term Finding (Inhabilitation): context $\vdash$? : type
  → decidabile în λ→

- Nu avem recursie nelimitată! (comb. de pct fix nu sunt typeable)

- recursie primitivă

tipuri $T = V \mid T \to T \mid Unit$

λ-termeni cu pre-tipuri ΛT : $\Lambda_T = x \mid \Lambda_T\, \Lambda_T \mid \lambda x : T . \Lambda_T \mid unit$

$$\frac{}{\Gamma \vdash unit : Unit} \text{(unit)}$$

- Tipul void - fără inhabitant.

$T = V \mid T \to T \mid Unit \mid Void$

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

- Tipul produs și constructorul pereche.

$T = V \mid T \to T \mid Unit \mid Void \mid T \times T$

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

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau} (\times_I) \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash fst\ M : \sigma} (\times_{E_1})$$

$$\frac{\Gamma \vdash M\ \sigma \times \tau}{\Gamma \vdash snd\ M ; \tau} (\times_{E_2})$$