Cheatsheet FLP

Litere grecești:

$A\alpha$ *alfa*	$B\beta$ *beta*	$\Gamma\gamma$ *gamma*	$\Delta\delta$ *delta*	$E\epsilon$ *epsilon*	$Z\zeta$ *zeta*	$H\eta$ *eta*	$\Theta\theta$ *theta*	$I\iota$ *iota*	$K\kappa$ *kappa*	$\Lambda\lambda$ *lambda*	$M\mu$ *mu*
$N\nu$ *nu*	$\Xi\xi$ *xi*	Oo *omicron*	$\Pi\pi$ *pi*	$P\rho$ *rho*	$\Sigma\sigma$ *sigma*	$T\tau$ *tau*	$\Upsilon\upsilon$ *upsilon*	$\Phi\phi$ *phi*	$X\chi$ *chi*	$\Psi\psi$ *psi*	$\Omega\omega$ *omega*

Lambda termen = variabilă | aplicare | abstractizare

$$M ::= x|(MM)|(\lambda x.M)$$

Convenții:

- $MNP = (MN)P$, $fxyz = ((fx)y)z$ (asociativitate la stânga a aplicării)
- $\lambda x.MN = \lambda x.(MN)$, deci $\lambda x.MN \neq (\lambda x.M)N$ (extindere la dreapta a corpului abstractizării)
- $\lambda xyz.M = \lambda x.\lambda y.\lambda z.M$ (comprimarea abstractizărilor)

Pentru $\lambda x.M$ avem:

- λ = operator de legare (*binder*)
- x = variabilă de legare (*binding*)
- N = domeniul (*scope*) de legare al lui x
- Aparițiile lui x în N sunt *legate*
- O apariție ce nu este legată se numește *liberă*
- Un termen fără variabile libere se numește *închis*, iar un termen închis se mai numește și *combinator*

$FV(M)$ **- mulțimea variabilelor libere**

$$FV(x) = \{x\}$$
$$FV(M) = FV(M) \cup FV(N)$$
$$FV(\lambda x.M) = FV(M) \backslash \{x\}$$

$M < y/x >$ reprezintă rezultatul obținut după redenumirea variabilei x cu y în lambda termenul M

α-echivalență = egalitate între doi termeni (modulo redenumire de variabile legate), adică

$$\lambda x.M =_\alpha \lambda y.(M < y/x >)$$

$M[N/x]$ este rezultatul obținut după înlocuirea lui x cu N în M și are formula generală:

$x[N/x]$	$\equiv N$	
$y[N/x]$	$\equiv y$	dacă $x \neq y$
$(MP)[N/x]$	$\equiv (M[N/x])(P[N/x])$	
$(\lambda x.M)[N/x]$	$\equiv \lambda x.M$	
$(\lambda y.M)[N/x]$	$\equiv \lambda y.(M[N/x])$	dacă $x \neq y$ și $y \notin FV(N)$
$(\lambda y.M)[N/x]$	$\equiv \lambda y'.(M < y'/y > [N/x])$	dacă $x \neq y, y \in FV(N)$ și y' variabilă nouă

Convenție: $M = N \Leftrightarrow M =_\alpha N$ (2 termeni sunt egali dacă sunt α echivalenți)

β-reducție = procesul de a evalua lambda termeni prin *pasarea de argumente funcțiilor*
β-redex = termen de forma $(\lambda x.M)N$
Redusul unui redex $(\lambda x.M)N$ este $M[N/x]$
Lambda termenii se reduc prin găsirea unui redex căruia i se aplică β-reducția
Forma normală = lambda termen fără redex-uri

Un pas de β-reducție($\rightarrow_\beta$) = cea mai mică relație de lambda termeni ce satisface regulile:

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

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

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

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

Exemplu β**-reducție**:

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

Observație: Există lambda termeni care nu pot fi reduși la o β-formă normală deoarece evaluarea este infinită (ex: $(\lambda x.xx)(\lambda x.xx)$) sau care pot fi reduși, dar pot să nu o atingă niciodată (alegerea β-redexurilor duce la un lambda termen ireductibil)

$M \twoheadrightarrow_\beta M' \Leftrightarrow M$ poate fi β-redus până în M' în 0 sau mai mulți pași
M-slab normalizabil $\Leftrightarrow (\exists)N$-formă normală a.i. $M \twoheadrightarrow_\beta N$
M-puternic normalizabil $\Leftrightarrow (\nexists)$ reduceri infinite care încep din M
Orice termen puternic normalizabil este și slab normalizabil.

Teorema Church-Rosser: Dacă $M \twoheadrightarrow_\beta M_1$ și $M \twoheadrightarrow_\beta M_2 \Rightarrow \exists M'$ a.i. $M_1 \twoheadrightarrow_\beta M'$ și $M_2 \twoheadrightarrow_\beta M'$.
Consecință: Un lambda termen are cel mult o β-formă normală (modulo α-echivalență).

Strategii de evaluare:

1. Strategia normală (*leftmost-outermost*)
   Dacă M_1 M_2 sunt redex-uri și M_1-subtermen al lui $M_2 \Rightarrow M_1$ nu va fi următorul redex ales.
   $$\underline{(\lambda xy.x)((\lambda x.xx)(\lambda x.xx))}(\lambda z.z)$$
2. Strategia aplicativă (*leftmost-innermost*)
   Dacă M_1 M_2 sunt redex-uri și M_1-subtermen al lui $M_2 \Rightarrow M_2$ nu va fi următorul redex ales.
   $$(\lambda xy.x)(\underline{(\lambda x.xx)(\lambda x.xx)})(\lambda z.z)$$

Strategii în programarea funcțională:

1. Call-by-name (CBN) = strategia **normală** fără a face reduceri în corpul unei λ-abstractizări
2. Call-by-value (CBV) = strategia **aplicativă** fără a face reduceri în corpul unei λ-abstractizări (strategie folosită în Haskell = *lazy evaluation*)

Valoare = λ-termen pentru care nu există β-reducții pe strategia de evaluare considerată

$\lambda x.x$ - mereu valoare
$(\lambda x.x)1$ - nu mereu valoare

Valori boolene:

b - valoare booleană
x, y - λ-termeni oarecare

$$T \triangleq \lambda xy.x \qquad F \triangleq \lambda xy.y$$

$$\textbf{if} = \lambda bxy. \begin{cases} x, \textbf{if}\ b = true \\ y, \textbf{if}\ b = false \end{cases}$$

$$\textbf{and} \triangleq \lambda b_1 b_2.\textbf{if}\ b_1 b_2 F$$
$$\textbf{or} \triangleq \lambda b_1 b_2.\textbf{if}\ b_1 T b_2$$
$$\textbf{not} \triangleq \lambda b_1 b_2.\textbf{if}\ b_1 F T$$

Numerali Church:

$$\begin{array}{lclcl} \overline{0} & \triangleq & \lambda fx.f^0 x & = & \lambda fx.x \\ \overline{1} & \triangleq & \lambda fx.f^1 x & = & \lambda fx.fx \\ \overline{2} & \triangleq & \lambda fx.f^2 x & = & \lambda fx.f(fx) \\ \overline{3} & \triangleq & \lambda fx.f^3 x & = & \lambda fx.f(f(fx)) \\ & \vdots & & & \\ \overline{n} & \triangleq & \lambda fx.f^n x & = & \lambda fx.\underbrace{f(f(\ldots(f}_{n}x)\ldots)) \end{array}$$

$\textbf{Succ} \triangleq \lambda nfx.f(nfx)$
$\textbf{add} \triangleq \lambda mnfx.mf(nfx) \triangleq \lambda mn.m\textbf{Succ}\ n$

Exemplu:

$$\begin{array}{lcl} \textbf{add}\ \overline{m}\ \overline{n} & = & (\lambda mn.m\textbf{Succ}\ n)\overline{m}\ \overline{n} \twoheadrightarrow_\beta \overline{m}\textbf{Succ}\ \overline{n} = (\lambda fx.f^m x)\ \textbf{Succ}\ \overline{n} \twoheadrightarrow_\beta \textbf{Succ}^m\ \overline{n} \\ & = & \underbrace{\textbf{Succ}(\textbf{Succ}(\ldots(\textbf{Succ}}_{m}\ \overline{n})\ldots)) \twoheadrightarrow_\beta \underbrace{\textbf{Succ}(\textbf{Succ}(\ldots(\textbf{Succ}}_{m-1}\ \overline{n+1})\ldots)) \\ \twoheadrightarrow_\beta & & \overline{m+n} \end{array}$$

$\textbf{mul} \triangleq \lambda mn.m(\textbf{add}\ n)\ \overline{0}$
$\textbf{exp} \triangleq \lambda mn.m(\textbf{mul}\ n)\ \overline{1}$
$\textbf{isZero} \triangleq \lambda nxy.n(\lambda z.y)x$

$M =_\beta M'$ reprezintă faptul că M poate fi transformat în M' în 0 sau mai mulți pași de β-reducție, transformare în care pașii pot fi și întorși.

- $\twoheadrightarrow_\beta$ este închiderea reflexivă și tranzitivă a relației $\rightarrow_\beta$
- $=_\beta$ este închiderea reflexivă, simetrică și tranzitivă a relației $\rightarrow_\beta$

Exemplu:

$$(\lambda y.yv)z =_\beta (\lambda x.zx)v$$
$$\text{deoarece}$$
$$(\lambda y.yv)z \rightarrow_\beta zv \leftarrow_\beta (\lambda x.zx)v$$

Punct fix al unui lambda termen F este un lambda termen M care verifică $FM =_\beta M$.

Teoremă: În lambda calcul fără tipuri orice termen are un punct fix.

Combinatori de punct fix:

- Combinatorul de punct fix al lui Curry
$$\mathbf{Y} \triangleq \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$$
- Combinatorul de punct fix al lui Turing
$$\Theta \triangleq (\lambda xy.y(xxy))(\lambda xy.y(xxy))$$

$\textbf{fact} \triangleq \mathbf{Y}\ (\lambda fn.\textbf{if}\ (\textbf{isZero}\ n)(\overline{1})(\textbf{mul}\ n(f(\textbf{pred}\ n))))$

Lambda calcul cu tipuri:

$\mathbb{V} = \{\alpha, \beta, \gamma, \dots\}$ este o mulțime finită de <u>tipuri variabilă</u>.
$\mathbb{T} = \underbrace{\mathbb{V}}_{\text{variabilă}} \mid \underbrace{\mathbb{T} \to \mathbb{T}}_{\text{săgeată}}$ este mulțimea tuturor <u>tipurilor simple</u>.

Exemple de tipuri simple:

- γ - variabilă
- $(\beta \to \gamma)$ - săgeată (funcție)
- $(\gamma \to \gamma) \to (\alpha \to (\beta \to \gamma))$ - săgeată (funcție)

Asociativitate:

- $\alpha_1 \to \alpha_2 \to \alpha_3 \to \alpha_4 \equiv (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_4)))$ (asociativitate la dreapta a tipului săgeată)
- $x_1 x_2 x_3 x_4 \equiv (((x_1 x_2) x_3) x_4)$ (asociativitate la stânga a tipului variabilă)

$M : \sigma$ înseamnă că M are tipul σ.
O variabilă x dintr-un termen M are un unic tip. (i.e dacă $x : \sigma$ și $x : \tau$ atunci $\sigma \equiv \tau$)
Dacă $M : \sigma \to \tau$ și $N : \sigma$ atunci $MN : \tau$
Dacă $M : \tau$ și $x : \sigma$ atunci $\lambda x.M : \sigma \to \tau$
M este *typeable* (are tip) dacă $(\exists)$ σ a.i. $M : \sigma$

Metode de asociere de tipuri variabilelor:

- Church-typing (asociere explicită). Fiecare variabilă are un tip la introducerea ei.
- Curry-typing (asociere implicită). Nu se prescrie nici un tip variabilelor la început, iar termenii typeable sunt descoperiți printr-un proces de căutare ce poate presupune o "ghicire"de tipuri.

Tipul variabilelor libere este dat de un *context*. În lambda calcul cu tipuri, termenul $(\lambda z.\lambda u.z)(yx)$ se scrie:

$$(\lambda z : \beta.\lambda u : \gamma.z)(yx)$$

Dacă presupunem un context pentru variabilele libere știute, scriem:

$$x : \alpha \to \alpha, y : (\alpha \to \alpha) \to \beta \vdash (\lambda z : \beta.\lambda u : \gamma.z)(yx)$$

$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}}$ - mulțimea lambda termenilor cu pre-tipuri

→ **Afirmație** - expreesie de forma $M : \sigma$, unde $M \in \Lambda_{\mathbb{T}}$ și $\sigma \in \mathbb{T}$ (M se numește *subiect*, și σ *tip*)

→ **Declarație** - afirmație în care subiectul este variabilă

→ **Context** - listă de declarații cu subiecți diferiți

→ **Judecată** - expresie de forma $\Gamma \vdash M : \sigma$, unde Γ este context și $M : \sigma$ este afirmație

Reguli pentru stabilirea unui tip într-un anumit context:

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

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

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

M este termen *legal* dacă există un context Γ și un tip ρ astfel încât $\Gamma \vdash M : \rho$
Exemplu: Arătăm că termenul $\lambda y : \alpha \to \beta.\lambda z : \alpha.yz$ are tipul $(\alpha \to \beta) \to \alpha \to \beta$ în contextul vid.

$$\emptyset \vdash (\lambda y : \alpha \to \beta.\lambda z : \alpha.yz) : (\alpha \to \beta) \to \alpha \to \beta$$

$$\cfrac{\cfrac{\cfrac{\overline{y{:}\alpha\to\beta, z{:}\alpha \vdash y{:}\alpha\to\beta}\,(var) \quad \overline{y{:}\alpha\to\beta, z{:}\alpha \vdash z{:}\alpha}\,(var)}{y{:}\alpha\to\beta, z{:}\alpha \vdash (yz){:}\beta}\,(app)}{y{:}\alpha\to\beta \vdash (\lambda z{:}\alpha.yz){:}\alpha\to\beta}\,(abs)}{\emptyset \vdash (\lambda y{:}\alpha\to\beta.\lambda z{:}\alpha.yz){:}(\alpha\to\beta)\to\alpha\to\beta}\,(abs)$$

Probleme de rezolvat în teoria tipurilor:

1. **Type Checking**
   Verificarea dacă există o derivare pentru
   $$\text{context} \vdash \text{term} : \text{type}$$
2. (a) **Well-typedness (typability)**
   Verificarea dacă un termen este legal (i.e găsirea unui context și a unui tip dacă termenul e legal, altfel să arătăm de ce nu se poate)
   $$? \vdash \text{term} : ?$$
   (b) **Type Assignment**
   Contextul este dat și trebuie găsit tipul
   $$\text{context} \vdash \text{term} : ?$$
3. **Term Finding (Inhabitation)**
   Se cunosc contextul și tipul, trebuie să stabilim dacă există un termen cu acel tip în contextul dat
   $$\text{context} \vdash \ ? : \text{type}$$

Limitări are λ-calculului cu tipuri simple:

- Nu mai avem recursie nelimitată deoarece combinatorii de punct fix nu sunt *typeable*.
- Tipurile pot fi prea restrictive. $(\lambda f.\mathbf{if}(fT)(f3)(f5))(\lambda x.x))$

Soluții posibile:

- Let-polymorphism - variabilele libere se redenumesc la fiecare folosire
  $$\begin{array}{l}\mathbf{let}\ f = \lambda x.x\ \mathbf{in}\\ \mathbf{if}\ (fT)(f3)(f5)\end{array}$$
- Cuantificatori de tipuri
  $$\lambda x.x : \Pi\alpha.\alpha \to \alpha$$
  Operatorul de legare Π face ca variabila de tip α să nu fie rigidă.

Tipul **Unit**:

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \mathbf{Unit}$$
$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}} \mid \mathbf{unit}$$

$$\overline{\Gamma \vdash \mathbf{unit}{:}\mathbf{Unit}}\,(unit)$$

Tipul **Void**:

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \mathbf{Unit} \mid \mathbf{Void}$$
$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}} \mid \mathbf{unit}$$

Nu există regulă de tipuri deoarece tipul **Void** nu are inhabitant.

Tipul **produs** și **constructorul pereche**:

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \mathbf{Unit} \mid \mathbf{Void} \mid \mathbb{T} \times \mathbb{T}$$
$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}} \mid \mathbf{unit} \mid <\Lambda_{\mathbb{T}},\Lambda_{\mathbb{T}}> \mid \mathit{fst}\ \Lambda_{\mathbb{T}} \mid \mathit{snd}\ \Lambda_{\mathbb{T}}$$

$$\frac{\Gamma \vdash M{:}\sigma \qquad \Gamma \vdash N{:}\tau}{\Gamma \vdash <M,N> : \sigma\times\tau}\,(\times_I)$$

$$\frac{\Gamma \vdash M{:}\sigma\times\tau}{\Gamma \vdash \mathit{fst}\ M{:}\sigma}\,(\times_{E_1}) \qquad \frac{\Gamma \vdash M{:}\sigma\times\tau}{\mathit{snd}\ M{:}\tau}\,(\times_{E_2})$$

Tipul **Sumă** și **constructorii Left/Right**:

$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \textbf{Unit} \mid \textbf{Void} \mid \mathbb{T} \times \mathbb{T} \mid \mathbb{T} + \mathbb{T}$

$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}} \mid \textbf{unit} \mid \ < \Lambda_{\mathbb{T}}, \Lambda_{\mathbb{T}} > \ \mid \textit{fst}\ \Lambda_{\mathbb{T}} \mid \textit{snd}\ \Lambda_{\mathbb{T}} \mid \textit{Left}\ \Lambda_{\mathbb{T}} \mid \textit{Right}\ \Lambda_{\mathbb{T}} \mid \text{case}\ \Lambda_{\mathbb{T}}\ \text{of}\ \Lambda_{\mathbb{T}};\ \Lambda_{\mathbb{T}}$

$$\frac{\Gamma \vdash M{:}\sigma}{\Gamma \vdash \textit{Left}\ M{:}\sigma{+}\tau}(+_{I_1}) \qquad \frac{\Gamma \vdash M{:}\tau}{\Gamma \vdash \textit{Right}\ M{:}\sigma{+}\tau}(+_{I_2})$$

$$\frac{\Gamma \vdash M{:}\sigma{+}\tau \quad \Gamma \vdash M_1{:}\sigma{\to}\gamma \quad \Gamma \vdash M_2{:}\tau{\to}\gamma}{\Gamma \vdash \text{case}\ M\ \text{of}\ M_1;\ M_2 : \gamma}(+_E)$$

Propositions are types!

♡

Faptul că există un termen de tip σ (inhabitant de tip σ) înseamnă că σ este o teoremă și are o demonstrație în ⋆LOGICĂ⋆!

♡