# Lambda kalkulus, cvičení ; v. 9. prosince 2013

## 1. cvičení

1. (DC) Dokažte:
a) $\lambda \vdash SKK = I$
b) $\lambda \vdash KI = K_*$

1.1 (DC) a) Rozpište do lambda abstrakcí (pozor na závorky, tečky ...), plně uzávorkujte, co nejméně uzávorkujte a zredukujte: a.1) $KII$ ; a.2) $K(IK_*)I$
b) vysvětlete: $KA \neq \lambda xy.xA$
c) vysvětlete rozdíl mezi $=$ a $\equiv$ ; c.1) pokud $F$ je def. jako $C[x, y]$ a konkrétně $\lambda x.yx$, pro které dvojice termů platí $\equiv$ a pro které $=$: $FI$, $F[y := I]$, $C[x, I]$, $C[I, I]$, $\lambda x.Ix$, $\lambda x.x$, $(\lambda x.yx)I$?

2. Dokažte (indukcí) pro libovolné $\lambda$-termy $s, t, u \in \Lambda$ a (různé) proměnné $x, y$:
a) pokud $s = t$, potom $s[x := u] = t[x := u]$
b) pokud $s = t$, potom $u[x := s] = u[x := t]$
c) $(s[x := u])[y := t] = s[y := t]([x := u[y := t]])$, pokud $x \notin freevar(t)$

3. Najděte uzavřený $\lambda$-term $F$:
a) $F \in \Lambda$, tž. $FGHX = G(HX)(HX)$.
b) $F \in \Lambda$, tž. $FX = XXX$.

4. Dokažte:
a) $\exists F\ \forall X\ FX = FF$
b) $\exists F\ \forall X\ FX = XF$
c) $\exists F\ \forall X\ FX = F$, někdy označován $K_\infty$
d) $\exists F\ \forall X\ FX = SFX$ (a $\eta$ redukovaný výraz)
e) $\exists F\ \forall X\ FX = FXX$
(q) neexistuje $F$, tž. $\forall X, Y\ F(XY) = X$
z) (Najdete řešení a), d) bez použití kombinátoru pevného bodu?)
Návod: Napište rovnici ve tvaru $F = C[f, x]F$ a použijte větu o pevném bodě pro nalezení $F$.

Poznámka: V $\lambda$-kalkulu můžeme redukovat "pod" $\lambda$-abstrakcí, což (ve funkcích) v programovacích jazycích nedokážeme. Použitelné např. při optimalizaci a částečném vyhodnocování.

4.1 (DC) Jak dosadit do (například) druhého argumentu funkce? Ve výrazu $F = \lambda xy.C[x, y]$ chceme dosadit za parametr $y$ term $T$. Napište term $Z$, aby $ZFT = \lambda x.C[x, T] = \lambda x.(C[x, y][y := T])$
Pozn: částečné vyhodnocování, vztah k lambda-liftingu (a superkombinátorům)

5. Zjednodušte:
a) $SIIx$ , $(SII(SII))$
(b) $SIIt$ pro $t = S(Ku)(SII)$
c) $S(KK)I$

6. Dokažte: (1. + 3. cv)
a) pokud $Z$ je kombinátor pevného bodu, potom $Z(SI)$ je kombinátor pevného bodu.
b) kombinátor $Z = VVVV$, kde $V = \lambda ehlo.o(hello)$, je kombinátor pevného bodu.
b1) otázka: platí pro $Z$: $ZF \rightarrow_\beta^* F(ZF)$ ?
c) (Turingův k.p.b.) $\Theta = AA$, kde $A = \lambda xy.y(xxy)$ je kombinátor pevného bodu.
c1) platí $\Theta F \rightarrow_\beta^* F(\Theta F)$
c2) platí $Y(SI) \rightarrow_\beta^* \Theta$ (pro $Y$ z přednášky)
c3) Rozhodněte, zda platí $Z(SI)F \rightarrow_\beta^* F(Z(SI)F)$
pozn. (s. 79): Turingův kombinátor $\Theta$ je ve tvaru superkombinátoru, $Y$ nikoli. ($F$ se po redukci $\Theta F$ dostane na místo argumentu)

d) Turingův $\Theta$, call by value: $\Theta_v = A'A'$, kde $A' = \lambda xy.y(\lambda z.xxyz)$ je kombinátor pevného bodu.
d.2) $Y$, call by value: $Y_v = \lambda f.A'A'$, kde $A' = \lambda x.f(\lambda v.((xx)v))$ je kombinátor pevného bodu.
d.3) Vysvětlete vztah $\Theta$, resp. $Y$, k $\Theta_v$, resp. $Y_v$, pomocí $\eta$-expanze.
e) Může se výpočet $YF$ vůbec někdy zastavit? Uveďte případně příklad.

## 2. cvičení

6.1 Najděte příklady $\lambda$-termů (Š2.2,2.6):
Term je *silně normalizovatelný*, pokud libovolná redukční strategie končí. Teorie je silně normalizovatelná, pokud každý term je silně normalizovatelný.
a) Dvojice termů, které ukazují, že relace $\rightarrow_\beta$, $\rightarrow_\beta^*$, $=_\beta$ jsou různé
b) v $\beta$-normálním tvaru
c) silně normalizovatelné, ale ne v $\beta$-normálním tvaru
d) normalizovatelné, ale ne silně normalizovatelné
e) nejsou normalizovatelné

7. Def. Pravidlo $\eta$. Pro lib. $\lambda$-term $F$ a proměnnou $x$, která se nevyskytuje volně v $F$ platí rovnost

$$\lambda x.Fx = F$$

Dá se ukázat, že pravidlo $\eta$ je bezesporné s axiomy $\lambda$-kalkulu. Kalkul, který vznikne přidáním pravidla $\eta$ označujeme $\lambda\eta$ .

Pravidlo $\eta$ se nazývá pravidlem extenzionality, protože pro libovolné dva $\lambda$-termy $F$ a $G$ a proměnnou $x$, která se nevyskytuje volně v $F$ ani v $G$, dovoluje dokázat a), tj. z rovnice $Fx = Gx$ (1) odvodit rovnici $F = G$. (2)
a) Ukažte, že v $\lambda\eta$-kalkulu z $Fx = Gx$ plyne $F = G$
b) Platí opačná implikace? :-)

Pokud se na $F$ a $G$ díváme jako na funkce jedné proměnné, potom (1) tvrdí, že tyto funkce mají sobě rovné hodnoty pro všechny *argumenty*. Pravidlo $\eta$ dovolí odvodit rovnost (2) *funkcí* $F$ a $G$.
Extenzionalita funkcí vyjadřuje fakt, že funkce, které mají stejné všechny hodnoty, se sobě rovnají.
Def. Pravidlo *ext*: Když $Nx = Mx$ pro $x \notin FV(MN)$, potom $M = N$. (!pouze pro proměnnou $x$)
Def. Nechť $T$ je formální teorie, jejímiž formulemi jsou rovnosti mezi termy. Říkáme, že $T$ je *extenzionální*, když platí:

$$T \vdash ML = NL \text{ pro každé } L, \text{ pak } T \vdash M = N$$

c) Ukažte, že teorie $\lambda$ není extenzionální.
d) Dokažte:
d.1) Teorie $\lambda + ext$ je extenzionální
d.2) Teorie $\lambda\eta$ je extenzionální
d.3) $\lambda + ext \vdash M = N \Leftrightarrow \lambda\eta \vdash M = N$
d.4) Teorie $\lambda + ext$ a $\lambda\eta$ jsou nejmenším extenzionálním rozšířením teorie $\lambda$
Pozn: Pravidlo $\eta$ (eta) opravňuje považovat lib. term $M$ za funkci $\lambda x.Mx$, ale to nemusí být v některém kontextu použití žádoucí. Proto nebylo Churchem zahrnuto do teorie $\lambda$.
Pozn: Pravidlo $\xi$ (ksi), s. 17, $M = M' \Rightarrow (\lambda x.M) = (\lambda x.M')$ je *slabá extenzionalita*: pokud jsou konvertibilní těla funkcí, jsou konvertibilní i funkce.
Další: redukce $\eta$; $\eta$ silně normalizuje, je CR; redukce $\beta\eta$, je CR, vztah s $\lambda\eta$-kalkulem, konzistence $\lambda\eta$, $\beta\eta$-NF $\Leftrightarrow$ $\beta$-NF, "úplnost" $\lambda\eta$: pro $M, N$ s $\beta\eta$-NF: $M = N \vee M\#N$

13. Dokažte, že pro $n, m \in N$, $c_n \equiv \lambda fx.f^n(x)$:
a) pro $A_+ \equiv \lambda xypq.xp(ypq)$ platí $A_+c_nc_m = c_{n+m}$
b) pro $A_* \equiv \lambda xyz.x(yz)$ platí $A_*c_nc_m = c_{n*m}$
c) pro $A_{exp} \equiv \lambda xy.yx$ platí $A_{exp}c_nc_m = c_{(n^m)}$, kromě m=0

d) Najděte term $A_{succ}$, pro který platí: $A_{succ}c_n = c_{n+1}$ (dvě možnosti)
(e) Najděte term $A_{pred}$, pro který platí: $A_{pred}c_{n+1} = c_n$
(f) Vyjádřete (a zredukujte) funkce $2 + n$, $n + 2$, $2 * n$, $n^2$, $2^n$, věž($n$,$m$)

14. (DC) a) Sformulujte a dokažte analogickou větu k "Double Fixed Point Theorem" pro $n$ termů v $n$ (vzájemně závislých) rovnicích. (s. 40)
b) Sformulujte a dokažte analogický důsledek (s. 43) pro $n$ vzájemně rekurzivních rovnic (daných $n$ kontexty).
c.1) Najděte kombinátor(y) analogické k Turingovu kombinátoru (s. 80) pro "Double F.P.Th", které se budou *$\beta$-redukovat* zleva doprava
c.2) dtto pro Second F.P.Th

## 3. cvičení

6. c), d) viz výš

6.2 Najděte příklady situací (aneb Co Church-Rosserova vlastnost netvrdí):
a) $M =_\beta N$, $M \not\equiv N$, $M$ má NF a zároveň má nekonečnou redukční posloupnost
a.1) navíc $M \not\rightarrow_\beta N$
a.2) navíc $M$ má nekonečnou redukční posloupnost *různých* termů
b) $M =_\beta N$, $M$ a $N$ mají NF, ale (nějaký) společný term $L$ není v NF
c) $M =_\beta N$, $M \not\equiv N$, $M$ nemá NF
d) $M$ má (aspoň) dvě nekonečné redukční posloupnosti navzájem *různých* termů
z) def: Slabá C-R vlastnost (pro relaci R): pokud $M_1 \leftarrow^1_R M \rightarrow^1_R M_2$, potom existuje $M_3$ tž. $M_1 \rightarrow^*_R M_3 \leftarrow^*_R M_2$
z.1) Slabá C-R je slabší než (silná) C-R; splývají, pokud redukce $R$ vždy končí (tj. pro silně normalizující teorie).

10. Definice. Říkáme, že $\lambda$-termy $s$ a $t$ jsou nekompatibilní (s $\lambda\beta$), jestliže přidáním axiomu $s = t$ k $\lambda\beta$ vznikne sporná teorie. Nekompatibilnost termů značíme $s\#t$.

Ve sporné teorii platí $s = t$ pro každé dva $\lambda$-termy $s$ a $t$.

Ukažte:
a) $S\#K$. Návod: aplikujte obě strany rovnice $S = K$ na vhodné termy $p$, $q$, $r$ a ukažte, že $I = t$ pro každý term $t$.
b) $I\#K$. (Taky přímý důkaz)
c) $I\#S$.
d) $K\#K_*$.
e) $s\#t \leftrightarrow \lambda + (s = t) \vdash K = K_*$
(Pokud $true \equiv K$ a $false \equiv K_*$, pak rovnost vpravo znamená $true = false$.)
(f) Najděte $\lambda$-term $F$, tž. $FI = x$ a $FK = y$.

12. Nechť $G = C[f, n]$ je $\lambda fn.if\ Zero\ n\ then\ 1\ else\ 2 * (f(Pred\ n))$
a) Co počítá (rekurzivní) funkce $F$ s def. $FN = GFN$
b) Najděte explicitní vyjádření $F$
c) Redukujte $F$ 2
d) Jakou funkci počítá konečný počet $k$ aplikací $G$ na nikde nedefinovanou funkci $\perp$ (bottom). Tj. $c_kG\Omega$, vyjádřeno pomocí Churchova numerálu.

11. (DC) Superkombinátory jsou kombinátory, kde každá lambda abstrakce (všech svých proměnných) je opět (uzavřený) superkombinátor. Teda každý term $S$ tvaru $(\lambda x_1 \lambda x_2 \ldots \lambda x_n.E)$, $n \geq 0$ je superkombinátor (četnosti $n$) pokud platí, že $S$ nemá volné proměnné, $E$ nezačíná $\lambda$-abstrakcí a každá lambda abstrakce v $E$ je opět superkombinátor. Tj. i vnořené funkce (všech svých víc proměnných) jsou uzavřené, tj. nemají nelokální termy. (Pro připomenutí: kombinátory jsou uzavřené $\lambda$-termy.) Konstanta je taky superkombinátor.

Př.: $\lambda x.x(\lambda y.\underline{xy}) \to \lambda x.x((\underline{\lambda z}\lambda y.zy)\underline{x})$
a) převeďte na superkombinátor: $\lambda fgh.f(\lambda y.g(hy))$
b) převeďte $Y$, $Y_v$, $\Theta_v$ na superkombinátor.
c) jsou $S$, $K$, $I$ superkombinátory?
Impl: lambda-lifting, (superkombinátory nepotřebují environment/ohodnocení): funkce s volnými proměnnými dostanou navíc nové argumenty a původně volné proměnné se předají jako nové hodnoty argumentů, vzniká *closure*.

15. Def. Množina $A$ $\lambda$-termů je *uzavřená vzhledem k rovnosti*, pokud pro lib. termy $M$, $N$ platí: pokud $M \in A$ a $M =_\beta N$, potom $N \in A$.
Dokažte zobecnění Scottovy věty (sl. 54) a důsledky:
a) Zl 104: Nech $A$, $B$ jsou neprázdné množiny $\Lambda$-termů uzavřené vzhledem k rovnosti. Potom $A$ a $B$ nejsou rekurzivně oddělitelné. Tj. neexistuje rekurzivní množina $R$, tž. $A \subset R$ a $B \subset -R$.
a.1) jiný způsob použití diagonalizace/věty o pevném bodě: pokud je oddělující funkce totální, pak pevný bod (analogického funkcionálu k sl. 55) nepatří ani do $A$, ani do $B$.
Aplikace/Důsledky:
b) Scott (54): $A$ uzavřená k rovnosti, $A \neq \emptyset$, $A \neq \Lambda$, potom $A$ není rekurzivní
c) Church (60): $\{M|M$ má NF $\}$ není rekurzivní (a je rekurzivně spočetná)
d) Relace konvertibility není rekurzivní. Hint: $A = \{M|M = I\}$
e) Nech $E$ je konzistentní množina rovností. Potom relace $=_E$ $E$-konvertibility v teorii $\Lambda{+}E$ není rekurzivní. (Konzistentní rozšíření $\lambda$-kalkulu jsou nerozhodnutelná.)
f) Problém zastavení není rozhodnutelný. Hint: $A = \{M|M$ má hlavovou normální formu $\}$

## 4. cvičení

8. Kombinatorická logika CL, (SK kalkul), eliminace abstrakce
V $\lambda$-kalkulu platí:
a.1) $\lambda x.x = SKK$ $(= I)$
a.2) $\lambda x.M = KM$, pro $x \notin FV(M)$
a.3) $\lambda x.MN = S(\lambda x.M)(\lambda x.N)$
Tvrzení dávají (až) algoritmus překladu do aplikací $S$ a $K$, pokrývají všechny ;-) struktury termů a vylučují se. Termy vpravo značíme $\lambda^* x.x$, $\lambda^* x.M$ a $\lambda^* x.MN$.
Def. *Kombinatorická logika* (CL) má axiomy pro $K$ a $S$: $KMN = M$, $SMNL = ML(NL)$. Dále schemata pro rovnost a dosazovaní v aplikaci (s. 17, kromě $\xi$ (ksí), které v CL neplatí: $M =_{CL} N \not\Rightarrow \lambda x.M =_{CL} \lambda x.N$). (CL nemá abstrakci, vázané proměnné a $\alpha$-konverzi.) *Termy kombinatorické logiky* $\mathcal{C}$ jsou $S$, $K$, (volné) proměnné $x$ a uzavřené na aplikaci. (`T::=V|<prim>|TT`, v jiných systémech se primitivní funkce mohou lišit)
Redukce v CL (analogicky s.65): osnova slabé $w$-redukce (i.1): $KMN \to_w M$ a $SMNL \to_w ML(NL)$, dále jednokroková redukce, redukce, konvertibilita
Platí:
b.1) $SKK \neq_{CL} I$, pro $I$ s $IM = M$
b.2) $SKKx =_{CL} x$ $(= Ix)$
b.3) $CL \not\vdash SKK = SKS$, přitom $SKK =_\beta SKS$
Pozn: z b.3 vidíme, že dokazatelnost (a slabá $w$-konvertibilita) v CL nesplývá s dokazatelností v $\lambda$ (a $\beta$-konvertibilitou). ("Neshodu" lze odstranit přidáním axiomů – za cenu ztráty jednoduchosti. Např. $K = \lambda^* xy.Kxy$, tj. popis chování při nedostatku argumentů. (*?Kritické páry.*))
Př: Vyjadřování pomocí kombinátorů: $F$ je komutativní pomocí kombinátoru $C$: místo $\forall x, y : Fxy = Fyx$ požadujeme pro $C$ rovnost $Cfxy = fyx$ a komutativita $F$ je vyjádřena $CF = F$.
Př: $F$ je asociativní: $\forall x, y, z : Fx(Fyz) = F(Fxy)z$, zavedeme $A_1, A_2$ splňující $A_1 fxyz = fx(fyz)$ a $A_2 fxyz = f(fxy)z$ a požadujeme pro $F$ rovnost $A_1 F = A_2 F$
(c) Převod $\lambda$-termů do kombinátorů $S$ a $K$ ; úplnost

d) kódování $\lambda$-termů pomocí $\lambda^*$ transformace/notace do termů kombinatorické logiky (podle b). !Zavedení $\lambda^* : \Lambda \to \mathcal{C}$ neznamená zavedení lambda abstrakce, ale pouze syntaktický cukr (synt. pozlátko).
(e) Definice $B, C$ (a $W, K$) ; úplnost. $Bfgx = f(gx)$, $Cfxy = fyx$. Jsou podobné $Sxyz$, ale argument $z$ nepropagují do $x$, resp. $y$. ($Wfx = fxx$)
(e.1) $\mathrm{CL}_I$ (vs. $\mathrm{CL}_K$) pro striktní funkce ($\lambda_I$: v $\lambda x.E$ má prom. $x$ aspoň jeden výskyt v $E$): používá $S$, $B$, $C$, $I$ (bez $K$)
(f) "Extrémní programování": stačí jeden kombinátor $X \equiv \lambda x.xKSK$ a platí $XXX = K$ a $X.XX = S$. (pozn. C-H izo: typ X obs. K a S v předpokladech)
Pozn: pravidlo $M = N \wedge N = L \Rightarrow \underline{L = M}$ zahrnuje/nahrazuje symetrii a tranzitivitu.

9. Převeďte do CL
a) $\lambda x.xx$, tj. napište $\lambda^* x.xx$
b) $(K=)\lambda xy.x$, b2) $(K_*=)\lambda xy.y$
c) $\lambda xy.yx$
d) d1) $\lambda xy.xx$, d2) $\lambda xy.yy$, d3) $\lambda xy.xy$
e) Převeďte do S, I, B, C: e1) $\lambda xy.yx$, e2) $\lambda xy.xy$, e3) $\lambda x(\lambda y.y)x$

10.-15. viz výš

16. a) Jak vypadá $[D \to D]$ pro jednoprvkové $D$?
b) Platí v jednoprvkovém modelu $S \neq K$?
c) Důsledek?

## 5. cvičení

Typovaný l.k.

Errata. Kalkulus 1.

54, dolu. $\#A = \{\#M | M \in A\}$ místo $\#A = \{\#M | M \in \underline{\Lambda}\}$
60, nahoře. $NF = \{M | M$ <u>has</u> a normal form$\}$ místo $NF = \{M | M$ a normal form$\}$
109 dole úplně vpravo: chybí pravá závorka: ... $= \|M\|_{\underline{\rho(x:=\|N\|_\rho)}}$

Kalkulus 2.
30. Ex. $S = (\lambda xyz.x\underline{z}(yz))$ místo $S = (\lambda xyz.xy(yz))$
32. pod čárou. $\lambda uv.\underline{u}$ : místo $\lambda uv$ :
33(,34). $M' - >>_\beta M$, viz doporučený uč. text 5.12
42 ř.2: <u>and</u> and
45 c) $\underline{x : \sigma} \vdash (\lambda y : \tau.x) : (\tau \to \sigma)$
82 b) $\vdash (\lambda xy.y) : (\forall \alpha\beta.\alpha \to \beta \to \underline{\beta})$
99 terminologie: preorder: kvaziuspořádání, předuspořádání
99 (i) ř.7: $\sigma \leq \tau, \sigma \leq \rho \Rightarrow \sigma \leq \underline{\tau \cap \rho}$
103 a) $\vdash (\lambda x.xx) : ((\sigma \to \tau) \cap \underline{\sigma}) \to \tau$
114 (i) $\Gamma \vdash x : \sigma \Rightarrow \exists \sigma' \geq \sigma((x : \underline{\sigma'}) \in \underline{\Gamma})$
182 iii. including

Kalkulus 3.
53 ř.2 conjun<u>c</u>tion

Komentáře:
s 21. Přímá rekurze pomocí komb. pevného bodu, vzájemná rekurze později pomocí dvojitého pevného bodu. (Podobně: pevný bod pro $n$ rovnic.) Impl: Jeden pevný bod pro celý program vs. (stratifikovaná) definice pro komponenty silné souvislosti v grafu závislostí.

116: první varianty kombinatorické logiky a lambda kalkulu byly nekonzistentní. :-(
Lambda 2, 99: kontravariantně v argumentu, kovariantně ve výsledku

Pozn.

A.1 Použití Y a Mu($\mu$) v datatypu (alias `Fix`, `FixF`), zpracování po úrovních; fold, unfold, refold

A.2 Zl 103: rekurzivně neoddělitelné množiny

A.3 Hlavová NF: nelze redukovat funkci (i pod $\lambda$-abstrakcí), ale nestaráme se o redukci argumentů; má tvar $M \equiv \lambda x_1 ... x_n . y L_1 ... L_m$ . Termy bez hlavové NF lze považovat za označení nedefinovaných výrazů.

A.3.1 Zl 96: nelze ztotožnit termy bez NF; ZL 138: pro M, N v NF(!) je v lambda,eta teorii buď $M = N$ nebo $M \# N$

A.4 Přepisovací systémy (rewriting s.), Knuth-Bendixův alg. pro dokazování v rovnostních teoriích, neselhávající varianta KB; rippling; konfluence, slabá konfluence;

A.5 v druhém semestru: Curry-Howardův izomorfizmus (a "proof assistants"), závislé typové systémy (př. vektory s délkou);
Dokazovače jsou killer application pro OCAML, (ale: soutěž dokazovačů vyhrávají ty rychlé)

A.5.1 implicitní typování v Haskellu; při rozšířeních explicitní typ pouze u určitých konstrukcí; (statické) typy taky umožňují optimalizace (v runtime se (nepoužívají, a proto) netestují tagy/typy); Haskell jako typová laboratoř;
typy: (velmi) různé použití (polyTypické, fantomové, multistage, BTA v PE: Binding Time Anotation) ..., statické typy vyvolají typovou chybu při kompilaci (1. v netestovaných částech, 2. v jiné úrovni (generovaný string, on-the-fly, DSEL - ale: kryptické chybové hlášky))

A.5.2 Vztah k teorii (k teorii kategorií); funktory, monády, komonády, Arrows, `Fix`, `FixF` pro datové typy, fold, unfold;
... a nástroje: GADT (Generalized Abstract Data Types), aplikativní funktory, aktivní konstruktory;
... a dřívější práci: s-výrazy $\approx$ XML;
... parametricity; catamorphism, anamorphism, paramorphism (cata/fold se zachováním hodnoty), apomorphism, hylomorphism = cata(f).ana(g) (je případ deforestation), metamorphism ...

A.5.3 implementační triky: superkombinátory, grafové přepisování, lambda-lifting - přeměna lokálních procedur na globální (viz jiné přednášky) - a lambda dropping, closure conversion; defunkcionalizace (s pomocí apply); full laziness (pro částečně aplikované funkce)

A.5.4 HOAS: higher-order abstract syntax, repr. abstraktních syntaktických stromů s vázanými proměnnými, prom. nemají jména a jejich výskyty ukazují na vázací místo; možná impl. HOAS: de Bruijn indexy; FOAS: first-order abstract syntax;

A.6 pull (FP) vs. push (OO) alg./styl, práce s celými dat. strukturami, "vzory" rekurze: map, fmap, fold, unfold, scan, ...; Filter-Map-Reduce; lambda abstrakce (FP: lambda funkce, closures) umožňuje dynamickou vazbu, hooks, (implementaci TVM, parametrizaci pomocí "výměny" procedur, metaparametry; srv. Lua: metatabulky);
výhody a přínosy Lispu (Paul Graham, viz);
abstraktní interpretace: počítání s jinou, typicky omezenou sémantikou; použitelné pro globální analýzu programu před kompilací; (impl.: ...); př: místo standardní sémantiky regulárních výrazů/BKG nad jazyky počítám first, empty a follow (v konečných doménách);

A.7 Manipulace s programy: odvozování typů, částečné vyhodnocování (partial evaluation, PE), Futamurovy projekce, optimalizační transformace (`map f.map g=map(f.g)`), počítání s programy (fusion laws, P. Wadler: Theorems for free!; pointfree vs. pointwise), (taky run-time code generation), deforestation;
Jiné styly/druhy psaní: (continuation passing style); polytypické programy (pro představu: umožní vygenerovat definici funkce map pro uživatelský datatyp pomocí strukturální rekurze podle struktury typu), generic programming: Generic Haskell; (multi-)staged programming (víceúrovňové programování): MetaOCaml, (MetaML), (pro představu: generuje typově bezpečný kód za běhu (typesafe run-time code generation), např. `.<int->int>.`; nepřesně/zjednodušeně: typově správná makra), splicing - vkládání kódu (a

dat) - typově správně, (operace: `.< >.`, `~` splice, lift, run);
meta-jazyk a objektový jazyk; přístup k zdrojáku (string nebo syntax tree): lispovské eval (A.8b), quasi-quote; Template Haskell `[| |]`;

A.7.1a Type-directed partial evaluation (TDPE);

A.7.1b Normalization by evaluation (NBE): využíva převod do domén a zpátky - fce reflect a reify; reifikace; eta-dlouhá forma;

A.7.2 Dokazování vlastností programu/modulů, zapisování vlastností (např. pro unit testy, `rev(rev(x))=x`); QuickCheck - testování vlastností pomocí náhodných vstupů (srv: redukce třídících sítí na 0-1 vstupy) - později převeden do jiných jazyků; (navíc Hs: (nějaký) generátor náhodných vstupů, i strukturovaných dat (seznamy, stromy, funkce ...) vygenerován automaticky pomocí typových tříd (při trošce opatrnosti))

A.8a pohled zvrchu na hierarchii jazyků (Paul Graham);
A.8b Greenspun's tenth rule: Any sufficiently complicated C or Fortran program contains an ad hoc, informally-specified, bug-ridden, slow implementation of half of Common Lisp.
(Morrisův dodatek: :-) ... včetně Common Lispu)

A.9 větší abstrakce = kratší programy (=méně chyb=lepší udržovatelnost), abstrakce dovolí lepší/obecnější návrh; (návrhové vzory (skoro) jako kód: co je (ve FP) vzor Strategie? apod.; ale: FP má jiné návrhové vzory), znovupoužitelnost a parametrizace, včetně metaparam.;
Pro "uživatele": Tvorba DSEL: Domain Specific (Embedded) Languages, domenové kombinátory (např. parsery), (např. generování správných (DTD valid) XML/HTML dat, SQL dotazů); prototypy, psaní interpretů, fantomové typy;

A.10 Kombinace FP a LP: funkcionální logické programování, např. jazyky Curry, Mercury; obsahuje výpočtový mechanizmus Narrowing (zužování), používá substituce, rewriting; vstup: funkcionální term s logickými proměnnými;
FP: OCAML; FP+OOP: Scala (nad JVM), F# (nad .NET); některé rysy mají skriptovací jazyky (ale: dynamické typování)