



Úvod do λ -kalkulu

Peter Borovanský

I-18

<http://dai.fmph.uniba.sk/courses/FPRO/>



Lambda calculus

Štruktúra prednášok:

- úvod do syntaxe, netypovaný λ -kalkul (gramatika + konvencie)
 - sémantika (redukčné pravidlá)
 - programovací jazyk nad λ -kalkulom
- domáca úloha: interpreter λ -kalkulu, ...

dnes nebude:

- rekurzia (pevný bod)
- vlastnosti teórie
- de Bruijn-ova notácia
- typovaný λ -kalkul

domáca úloha: typovač λ -kalkulu, ...



Trochu z histórie FP

- 1930, Alonso Church, lambda calculus
 - teoretický základ FP
 - kalkul funkcií: abstrakcia, aplikácia, kompozícia
 - Princeton: A.Church, A.Turing, J. von Neumann, K.Gödel - skúmajú formálne modely výpočtov
 - éra: WWII, prvý von Neumanovský počítač: Mark I (IBM), balistické tabuľky
- 1958, Haskell B.Curry, logika kombinátorov
 - alternatívny pohľad na funkcie, menej známy a populárny
 - „premenné vôbec nepotrebujeme“
- 1958, LISP, John McCarthy
 - implementácia lambda kalkulu na „von Neumanovskom HW“
- 1960, SECD (**S**tack-**E**nvironment-**C**ontrol-**D**ump) Machine, Landin
 - predchodca p-code, rôznych stack-orientovaných bajt-kódov, virtuálnych mašín.
 - SECD použili pri implementácii
 - Algol 60, PL/1 – predchodcu Pascalu
 - LISP – prvého funkcionálneho jazyka založenom na -kalkule



Od Haskellu k λ -kalkulu

- `length []` = 0
 - `length (x:xs)` = 1+length xs
- > `length [1,2,3,4,5]`

let `length xs` = if (null xs) then 0 else (1+length (tail xs)) in `length [1,2,3,4,5]`

`let length xs` = (if (null xs) 0 ((+) 1 (length (tail xs))))
in (length ((:) 1 ((:) 2 ((:) 3 ((:) 4 ((:) 5 []))))))

`let length` = $\lambda ys.$ (if (null xs) 0 ((+) 1 (length (tail ys)))) in (length ...

`let length` = ($\lambda f.$ $\lambda ys.$ (if (null xs) 0 ((+) 1 (f (tail ys))))) **length** in (length ...

`let length` = **Y**($\lambda f.$ $\lambda ys.$ (if (null xs) 0 ((+) 1 (f (tail ys))))) in (length ...



Syntax

Celý program v jazyku pozostáva z jedného λ -termu.

L je λ -term:

- x je premenná (spočítateľná množina premenných)

$$L ::= x \mid (L L) \mid (\lambda x L)$$
$$L ::= x \mid L L \mid \lambda x.L$$

- $(L L)$ je aplikácia (funkcie)
- $(\lambda x L)$ je λ -abstrakcia definujúca funkciu s argumentom x a telom L

Cvičenie (na zamyslenie): syntax jazyka je veľmi jednoduchá, neporovnateľná napr. s Pascalom. Zamyslite sa nad tým, či existuje viac programov v Pascale alebo λ -termov.



Príklady λ -termov

- $(\lambda x x)$
- $(\lambda x y)$
- $(\lambda x (x x))$
- $((\lambda x (x x)) (\lambda x (x x)))$
- $(\lambda y (\lambda x (x x)))$

z týchto príkladov zatiaľ nie je evidentné, že to bude programovací jazyk.



Syntaktické konvencie

- malé písmená označujú premenné: x, y, x_1, x_2, \dots
- veľké písmená označujú λ -termy: M, N, \dots
- vonkajšie zátvorky nepíšeme
- symbol $.$ nahradzuje $(, \text{ zodpovedajúca })$ chýba
 - $(\lambda x x) \rightarrow \lambda x.x$
 - $(\lambda x (x x)) \rightarrow \lambda x.xx$ ale nie $(\lambda x.x)x$
 - $((\lambda x (x x)) (\lambda x (x x))) \rightarrow (\lambda x.xx)(\lambda x.xx)$
- vnorené abstrakcie majú asociativitu vpravo
 - $(\lambda y (\lambda x (x x))) \rightarrow \lambda y.\lambda x.xx \rightarrow \lambda yx.xx$
- vnorené aplikácie majú asociativitu vľavo
 - $((((\lambda xyz.yz) a) b) c) \rightarrow (\lambda xyz.yz)abc$

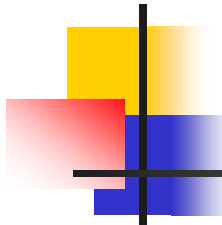


Dôležité príklady termov

O ich dôležitosti sa dozvieme neskôr, keď budeme vedieť, ako sa v tejto teórii počíta

- $K = \lambda xy.x = \lambda x.\lambda y.x$ (funkcia s dvomi argumentami, výsledkom je 1.)
- $I = \lambda x.x$ (identita)
- $S = \lambda xyz.xz(yz) = \lambda xyz.((x\ z)\ (y\ z))$ ($S\ a\ b\ c = (a\ c)\ (b\ c)$)

- $\omega = \lambda x.xx$
- $\Omega = \omega\omega = (\lambda x.x\ x)(\lambda x.x\ x)$
- $\omega_3 = \lambda x.xxx = (\lambda x((x\ x)\ x))$



Výpočet

- na to, aby sme vedeli počítať v tejto teórii, potrebujeme definovať redukčné pravidlo(á), ktoré simuluje krok výpočtu,
- redukčné pravidlo je založené na pojme substitúcie, ktorú, ak pochopíme len intuitívne, dostaneme intuitívne zlé výsledky,
- preto sa vybudovaniu substitúcie treba chvíľku venovať s pomocnými pojmami, ako je voľná premenná, ...
- musíme sa presvedčiť, že redukčné pravidlo má rozumné vlastnosti, zamyslíme sa nad tým, čo je rozumné...
- výpočet je opakované aplikovanie redukčného pravidla, a to kdekoli'vek to v terme ide. Keď to už nejde, máme výsledok (tzv. normálna forma)
- môže sa stať, že rôznym aplikovaním red.pravidla prideme k rôznym výsledkom, resp. rôznym výpočtom ???



Voľná premenná, podterm

- voľná premenná λ -termu

- $\text{Free}(x) = x$
- $\text{Free}(\lambda x.M) = \text{Free}(M) - \{x\}$
- $\text{Free}(M N) = \text{Free}(M) \cup \text{Free}(N)$

- viazaná premenná λ -termu

- $\text{Bound}(x) = \{x\}$
- $\text{Bound}(\lambda x.M) = \text{Bound}(M) \cup \{x\}$
- $\text{Bound}(M N) = \text{Bound}(M) \cup \text{Bound}(N)$

viazaná premenná a voľná premenná:

- $\lambda x.xy$ – y je voľná, x je viazaná

$\text{Bound}(M) \quad \text{Free}(M) = ??? \quad \emptyset$

- podtermy λ -termu

- $\text{Subt}(x) = x$
- $\text{Subt}(\lambda x.M) = \text{Subt}(M) \cup \{\lambda x.M\}$
- $\text{Subt}(M N) = \text{Subt}(M) \cup \text{Subt}(N) \cup \{ (M N) \}$



Príklady

- $\lambda x. \lambda y. (x \ z)$
 - x je viazaná,
 - z voľná,
 - y sa nenachádza v $\text{Subt}(\lambda x. \lambda y. (x \ z))$
- $\lambda x. ((\lambda y. y) \ (x \ (\lambda y. y)))$
 - má dva výskyty podtermu $(\lambda y. y)$
- $(x \ (y \ z)) \in \text{Subt}((w \ (x \ (y \ z))))$
 - ale $(x \ (y \ z)) \notin \text{Subt}(w \ x \ (y \ z)) = \text{Subt}((w \ x) \ (y \ z))$, lebo
 - $\text{Subt}(w \ x \ (y \ z))$ obsahuje tieto podtermy:
 - $((w \ x) \ (y \ z)), (w \ x), (y \ z), w, x, z, y$
 - teda $w \ x \ (y \ z) = (w \ x)(y \ z)$



Substitúcia

- ak sa na to ide naivne (alebo "textovo"):

- $(\lambda x.zx)[z:y] \rightarrow \lambda x.yx$
- $(\lambda y.zy)[z:y] \rightarrow \lambda y.yy$

Problém: rovnaké vstupy, rôzne výsledky – výrazy $(\lambda x.zx)$ a $(\lambda y.zy)$ *intuitívne* predstavujú rovnaké funkcie a po substitúcii $[z:y]$ sú výsledky $\lambda x.yx$ a $\lambda y.yy$ *intuitívne* rôzne funkcie

- substitúcia $N[x:M]$

$$x[x:M] = M$$

$$y[x:M] = y$$

$$(A B)[x:M] = (A[x:M] B[x:M])$$

$$(\lambda x.B)[x:M] = (\lambda x.B)$$

$$(\lambda y.B)[x:M] = \lambda z.(B[y:z][x:M]) \text{ ak } \mathbf{x \in \text{Free}(B)}, \mathbf{y \in \text{Free}(M)}$$

ak z nie je voľné v B alebo M , $z \notin \text{Free}((B M)), x \neq y$

inak $(\lambda y.B)[x:M] = \lambda y.B[x:M] \quad x \neq y$

- správne: $(\lambda y.zy)[z:y] \rightarrow (\lambda w.(zy)[y:w])[z:y] \rightarrow (\lambda w.(z w))[z:y] \rightarrow \lambda w.yw$



Príklady

naivne

$(\lambda x.zx)[z:y] \rightarrow$

$\lambda x.yx$

$(\lambda y.zy)[z:y] \rightarrow$

$\lambda y.yy$

- $(\lambda x.zx)[z:y] =$
 - $\lambda x.((zx)[z:y]) =$
 - $\lambda x.(z[z:y]x[z:y]) =$
 - $\lambda x.(yx)$

- $(\lambda y.zy)[z:y] =$
 - $(\lambda w.(zy)[y:w])[z:y] =$ – treba si vymyslieť novú premennú
 - $(\lambda w.(z[y:w]y[y:w]))[z:y] =$
 - $(\lambda w.(zw))[z:y] =$
 - $\lambda w.(zw)[z:y] =$
 - $\lambda w.(z[z:y]w[z:y]) =$
 - $\lambda w.(yw)$



Vlastnosti substitúcie

Ak premenná x nie je voľná v M , $x \notin \text{Free}(M)$, potom $M[x:N] = M$.

Dôkaz indukciou...

Ak $\text{Free}(M) = \emptyset$, M nazývame uzavretý výraz.

Dosledok: Uzavretý výraz sa aplikáciou substitúcie nezmení.

Lemma:

- $x \neq y$ sú rôzne premenné,
- x nie je voľná v L , $x \notin \text{Free}(L)$,
- ak každá viazaná premenná v M nie je voľná v $(N \ L)$
 $v \in \text{Bound}(M) \Rightarrow v \notin \text{Free}((N \ L))$,

potom

1. $M[x:N] [y:L] = M[y:L][x:N[y:L]]$
2. $M[y:N] [y:L] = M[y:N[y:L]]$



$$1) M[x:N] [y:L] = M[y:L][x:N[y:L]]$$

Indukciou vzhľadom na M:

- M je premenná

- $M = x$, obe strany sú $N[y:L]$
- $M = y$, obe strany sú L , lebo x nie je voľná v L ,
- $M = z$, rôzna premenná od x, y , potom obe strany sú z .

- $M = (\lambda z.Q)$

- $z = x$, obe strany sú $(\lambda x.(Q[y:L]))$
- $z = y$, obe strany sú $(\lambda y.(Q[x:N]))$, lebo y nie je voľná v $(N \ L)$, takže ani N
- z je rôzne od x, y , potom, podľa predpokladu, z nie je voľná v N ani L

$$(\lambda z.Q)[x:N] [y:L] =$$

$$\lambda z.(Q[x:N]) [y:L] =$$

$$\lambda z.(Q[x:N][y:L]) = \dots \text{ indukcia}$$

$$\lambda z.(Q[y:L][x:N[y:L]]) =$$

$$(\lambda z.Q)[y:L][x:N[y:L]].$$

- $M = (Q \ R)$, no problem, indukciou na Q a R ...

Predpoklady:

$$x \neq y$$

$$x \notin \text{Free}(L),$$

$$v \in \text{Bound}(M) \Rightarrow v \notin \text{Free}((N \ L))$$


$$2) M[y:N] [y:L] = M[y:N[y:L]]$$

Domáca úloha:

podobne dokážte tvrdenie 2) predchádzajúcej lemmy.

Domáca úloha:

za akých podmienok (najslabších) platí, navrhните a zdôvodните, dokážte...

- $M[x:N] [y:L] = M[y:L][x:N]$
- $M[x:y] [y:N] = M[x:N]$
- $M[x:y] [y:x] = M$



α -konverzia

$$\lambda x.M =_{\alpha} \lambda y.M[x:y]$$

- $\lambda x.M$ je premenovaním viazanej premennej $\lambda y.M[x:y]$, ak y nie je voľná v M
- $=_{\alpha}$ je relácia ekvivalencie
- $=_{\alpha}$ kongruencia na λ termoch
- intuícia: výrazy, ktoré sa odlišujú menom viazanej premennej predstavujú rovnaké funkcie



β -redukcia

$K = \lambda xy.x$
 $I = \lambda x.x$
 $S = \lambda xyz.xz(yz)$

$$(\lambda x.B) E \rightarrow_{\beta} B[x:E]$$

Príklad:

- $I M = x$
 - $(\lambda x.x) M \rightarrow_{\beta} x[x:M] = M$
- $K M N = M$
 - $(\lambda xy.x)MN \rightarrow_{\beta} (\lambda y.M)N \rightarrow_{\beta} M$
- $S M N P = M P (N P)$
 - $\lambda xyz.xz(yz) MNP \rightarrow_{\beta}^3 MP(NP)$
- $S K K = I$
 - $\lambda xyz. ((xz)(yz)) (\lambda xy.x) (\lambda xy.x) \rightarrow_{\beta}$
 - $\lambda yz. (((\lambda xy.x)z) (yz)) (\lambda xy.x) \rightarrow_{\beta}$
 - $\lambda z. ((\lambda xy.x)z((\lambda xy.x)z)) \rightarrow_{\beta}$
 - $\lambda z. ((\lambda y.z)((\lambda xy.x)z)) \rightarrow_{\beta}$
 - $\lambda z. ((\lambda y.z)(\lambda y.z)) \rightarrow_{\beta}$
 - $\lambda z.z = I$



Vlastnosti β -redukcie

- $\omega = \lambda x. xx = \lambda x. (x x)$
- $\Omega = \omega \omega$
- $\omega_3 = \lambda x. ((x x) x)$
- nekonečná sekvencia
 - $\Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \dots$
- puchnúca sekvencia
 - $\omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \omega_3$
- nejednoznačný výsledok pre dva rôzne výpočty
 - $KI\Omega \rightarrow_{\beta} I$ ale aj
 - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$

Cvičenie: overte si tieto tvrdenia,

Pokúste sa nájsť λ term, ktorý vedie k rôznym výsledkom 😊



η -redukcia

- $\lambda x.(B\ x) \rightarrow_{\eta} B$ ak $x \notin \text{Free}(B)$

podmienka je podstatná, lebo ak napr. $B=x$, teda $x \in \text{Free}(B)$, $\lambda x.(x\ x) \neq x$

- β_{η} je uzáver $\beta \cup \eta$ vzhľadom na podtermy, čo znamená
 - ak $M \beta N$ alebo $M \eta N$, potom $M \beta_{\eta} N$,
 - ak $M \beta_{\eta} N$, potom $(P\ M) \beta_{\eta} (P\ N)$ aj $(M\ Q) \beta_{\eta} (N\ Q)$,
 - ak $M \beta_{\eta} N$, potom $\lambda x.M \beta_{\eta} \lambda x.N$.



Domáca úloha

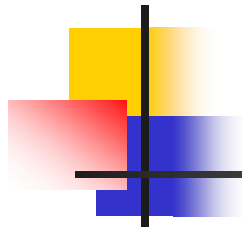
Definujte základné funkcie pre interpreter λ -kalkulu:

- `free` - zistí, či premenná je voľná
- `subterm` - vráti zoznam podtermov
- `substitute` - korektne implementuje substitúciu
- `oneStepBetaReduce`
- `normalForm` - opakuje redukciu, kým sa dá

navrhovaná reprezentácia (kľudne si zvolíte inú):

```
data LExp = LAMBDA String LExp |  
           ID String |  
           LExp [LExp] |  
           App LExp LExp |  
           CON String |  
           CN Integer  
deriving(Show, Read, Eq)
```

- **abstrakcia**
- **premenná**
- aplikácia, zovšeobecnená
- **aplikácia**
- **konštanta, built-in fcia**
- **int.konštanta**



Cvičenie (použite váš tool)

1) určite voľné a viazané premenné:

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

2) redukujte:

- $(\lambda x. \lambda y. x (\lambda z. y \ z)) (((\lambda x. \lambda y. y) \ 8) (\lambda x. (\lambda y. y) \ x))$
- $(\lambda h. (\lambda x. h \ (x \ x)) (\lambda x. h \ (x \ x))) ((\lambda a. \lambda b. a) (+ \ 1 \ 5))$

3) Nech $F = (\lambda t. t \ t) (\lambda f. \lambda x. f \ (f \ x))$.

Vyhodnoťte $F \text{ succ } 0$, $\text{succ} = \lambda x. (+ \ x \ 1)$



Riešenie (zle)

- $(\lambda x. \lambda y. x (\lambda z. y z)) (((\lambda x. \lambda y. y) 8) (\lambda x. (\lambda y. y) x)) \rightarrow_{\beta}$
 - $(\lambda x. \lambda y. x (\lambda z. y z)) ((\lambda y. y) (\lambda x. (\lambda y. y) x)) \rightarrow_{\beta}$
 - $(\lambda x. \lambda y. x (\lambda z. y z)) ((\lambda y. y) (\lambda y. y)) \rightarrow_{\beta}$
 - **$(\lambda x. \lambda y. x (\lambda z. y z)) (\lambda y. y) \rightarrow_{\beta}$**
 - **$\dots(\lambda y. x)[x:(\lambda z. y z)] \dots$ $y \in \mathbf{Free}(\lambda z. y z)$, $x: \mathbf{Free}(x)$**
 - **$\lambda y. (\lambda z. y z) (\lambda y. y) \rightarrow_{\beta}$**
 - $(\lambda z. (\lambda y. y) z) \rightarrow_{\beta}$
 - $(\lambda z. z) \rightarrow_{\beta}$
 - **I**



Riešenie (dobre)

Nájdite pomocou vášho nástroja pre vyhodnocovanie λ -výrazov

- $(\lambda x. \lambda y. (x ((\lambda z. y) z))) (((\lambda x. \lambda v. v) 8) (\lambda x. (\lambda w. w) x)) \rightarrow_{\beta}$
 - $(\lambda x. \lambda y. (x ((\lambda z. y) z))) ((\lambda v. v) (\lambda x. (\lambda w. w) x)) \rightarrow_{\beta}$
 - $(\lambda x. \lambda y. (x ((\lambda z. y) z))) ((\lambda v. v) (\lambda w. w)) \rightarrow_{\beta}$
 - $(\lambda x. \lambda y. (x ((\lambda z. y) z))) (\lambda v. v) \rightarrow_{\beta}$
 - $\lambda y. ((\lambda v. v) ((\lambda z. y) z)) \rightarrow_{\beta}$
 - $\lambda y. (((\lambda z. y) z)) \rightarrow_{\beta}$
 - $\lambda y. y$



Riešenie

- $(\lambda h.(\lambda x.h (x x)) (\lambda x.h (x x))) ((\lambda a.\lambda b.a) (+ 1 5)) \rightarrow_{\beta}$
 - $(\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x)) (\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x)) \rightarrow_{\beta}$
 - $((\lambda a.\lambda b.a) (+ 1 5)) ($(\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x)) (\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x))) \rightarrow_{\beta}$$
 - $(\lambda b.(+ 1 5)) ($(\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x)) (\lambda x.((\lambda a.\lambda b.a) (+ 1 5)) (x x))) \rightarrow_{\beta}$$
 - $(+ 1 5) \rightarrow_{\beta}$
 - 6



Domáca úloha (nepovinná)

Pri práci s vašim interpretrom vám bude chýbať:

- vstup λ termu – funkcia `fromString :: String -> LExp`, ktorá vám vytvorí vnútornú reprezentáciu z textového reťazca, príklad:

```
fromString "\x.xx" = (LAMBDA "x" (LExp [(Id "x"), (Id "x")]))
```

takejto funkcii sa hovorí syntaktický analyzátor a musíte sa vysporiadať s problémom, keď je vstupný reťazec nekorektný

- výstup λ termu – funkcia `toString :: LExp -> String`, ktorá vám vytvorí textovú (čitateľnú) reprezentáciu pre λ term.

čo by Vás mohlo inšpirovať



Fold na termoch

```
foldLambda lambda var apl con cn lterm
| lterm == (LAMBDA str exp) =
    lambda str (foldLambda lambda var apl con cn exp)
| lterm == (VAR str) = var str
| lterm == (APL exp1 exp2) =
    apl      (foldLambda lambda var apl con cn exp1)
             (foldLambda lambda var apl con cn exp2)
| lterm == (CON str) = con str
| lterm == (CN int) = cn int
```

```
vars = foldLambda (\x y->y) (\x->[x]) (++) (\_->[]) (\_->[])
```

```
show :: LExp -> String
```

```
show = foldLambda (\x y->"(\\"++x++"->"++y++")")
    (\x->x) (\x y->"("++x++" "++y++")") (\x->x) (\x->x)
```



Od λ -termu k programu

Na to, aby sme vedeli v tomto jazyku programovať, potrebujeme:

- mať v ňom nejaké hodnoty, napr. aspoň int, bool, ...
- základné dátové typy, záznam (record, record-case), zoznam, ...
- if-then-else
- let, letrec, či where
- rekurziu

V ďalšom obohatíme λ -kalkul syntaktickými cukrovinkami tak, aby sme sa presvedčili, že sa v tom programovať naozaj dá.

Rekurzia pomocou operátora pevného bodu bude najnáročnejším klincom v tejto línii.



Churchove čísla

- $0 := \lambda f. \lambda x. x$
- $1 := \lambda f. \lambda x. f\ x$
- $2 := \lambda f. \lambda x. f\ (f\ x)$
- $3 := \lambda f. \lambda x. f\ (f\ (f\ x))$

.....

$$C(+1)\ 0 = c$$

- $\text{succ} := \lambda n. \lambda f. \lambda x. f(n\ f\ x)$
- $\text{plus} := \lambda m. \lambda n. \lambda f. \lambda x. m\ f\ (n\ f\ x)$

Domáca úloha (povinná):

- definujte mult ,
- definujte 2^n , m^n ,
- definujte $n-1$



Logické hodnoty a operátory

TRUE := $\lambda x. \lambda y. x := \lambda xy. x$

FALSE := $\lambda x. \lambda y. y := \lambda xy. y$

AND := $\lambda x. \lambda y. x y$ FALSE := $\lambda xy. x y$ FALSE

OR := $\lambda x. \lambda y. x \text{ TRUE } y := \lambda xy. x \text{ TRUE } y$

NOT := $\lambda x. x \text{ FALSE TRUE}$

IFTHENELSE := $\lambda pxy. p x y$

AND TRUE FALSE

$\equiv (\lambda p q. p q \text{ FALSE}) \text{ TRUE FALSE} \quad \beta \text{ TRUE FALSE FALSE}$

$\equiv (\lambda x y. x) \text{ FALSE FALSE} \quad \beta \text{ FALSE}$

Cvičenie: definujte XOR



Kartézsky súčin typov (pár)

PAIR $:= \lambda x. \lambda y. \lambda c. c \ x \ y$ $:= \lambda x y c. c \ x \ y$

LEFT $:= \lambda x. x \ \text{TRUE}$

RIGHT $:= \lambda x. x \ \text{FALSE}$

TRUE $:= \lambda x. \lambda y. x$ $:= \lambda x y. x$

FALSE $:= \lambda x. \lambda y. y$ $:= \lambda x y. y$

LEFT (PAIR A B) \equiv

LEFT (($\lambda x y c. c \ x \ y$) A B) β

LEFT ($\lambda c. c \ A \ B$) β

($\lambda x. x \ \text{TRUE}$) ($\lambda c. c \ A \ B$) β

($\lambda c. c \ A \ B$) ($\lambda x y. x$) β

(($\lambda x y. x$) A B) $\beta \ A$

Cvičenie: definujte n-ticu

Curry

$\lambda(x,y).M \rightarrow \lambda p. (\lambda x \lambda y. M) (\text{LEFT } p) (\text{RIGHT } p)$



Súčet typov (disjunkcia)

$A+B$ reprezentujeme ako pár $[\text{Bool} \times (A|B)]$

$1^{\text{st}} := \lambda x. \text{PAIR TRUE } x$ konštruktor pre A

$2^{\text{nd}} := \lambda y. \text{PAIR FALSE } y$ B

$1^{\text{st}^{-1}} := \lambda z. \text{RIGHT } z$ deštruktor pre A

$2^{\text{nd}^{-1}} := \lambda z. \text{RIGHT } z$ B

$?1^{\text{st}^{-1}} := \lambda z. \text{LEFT } z$ test, či A

$1^{\text{st}^{-1}} 1^{\text{st}} A \equiv$

$(\lambda z. \text{RIGHT } z) (\lambda x. \text{PAIR TRUE } x) A \quad \beta$

$\text{RIGHT } (\text{PAIR TRUE } A) \quad \beta A$

Cvičenie: reprezentujte
zoznam s konštruktormi Nil,
Cons a funkciami isEmpty,
head a tail



where (let, letrec)

$M \text{ where } v = N \quad \rightarrow (\lambda v.M) N$

$M \text{ where } \begin{array}{l} v_1 = N_1 \\ v_2 = N_2 \dots \\ v_n = N_n \end{array} \quad \rightarrow (\lambda(v_1, v_2, \dots, v_n).M) (N_1, \dots, N_n)$

zložený where

$n^*(x+n) \text{ where } \begin{array}{l} n = 3 \\ x = 4*n+1 \end{array} \quad \rightarrow (\lambda n. (\lambda x.n^*(x+n)) (4*n+1)) 3$