



Lambda calculus 2

Štruktúra prednášok:

- úvod do syntaxe (gramatika + konvencie)
- sémantika (redukčné pravidlá)
- programovací jazyk nad λ -kalkulom

domáca úloha: interpretér λ -kalkulu, ...

- opakované aplikovanie β -redukcie

Dnes: (alternatíva: https://en.wikipedia.org/wiki/Lambda_calculus)

- vlastnosti β -redukcie
- rekurzia a pevný bod
- de Bruijn indexy (miesto mien premenných)

Rekapitulácia

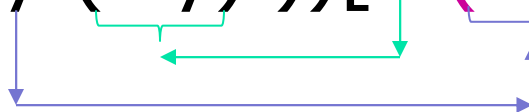
β -redukcia – substitúcia

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

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

$$(\lambda x (\lambda y (x y))) (x y) \rightarrow_{\beta}$$

$$(\lambda y (x y)))[x:(x y)] \rightarrow_{\beta}^{\text{ZLE}} (\lambda y ((x y) y)))$$



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

$$(\lambda y (x y)))[x:(x y)] \rightarrow_{\beta} \text{DOBRE}$$

$$(\lambda w (x w)))[x:(x y)] \rightarrow_{\beta}$$

$$(\lambda w ((x y) w)))$$

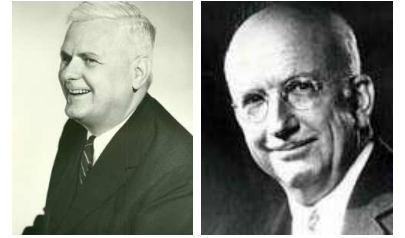


Emmy
Noether

Terminácia – Noetherovská

- výpočet = opakované aplikácie β -redukcie nemusia skončiť
- jediná šanca, aby výpočtový model λ -kalkul mohol byť ekvivalentný Turingovmu stroju, a nie konečnému automatu
- nekonečná sekvencia
 - $\Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \dots$
- neobmedzene puchnúca sekvencia
 - $\omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \omega_3$

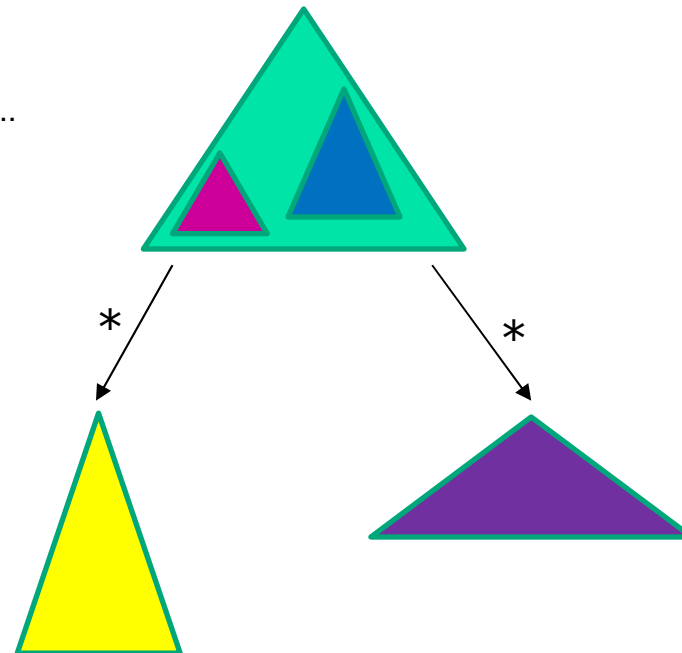
$$\omega = \lambda x.xx = \lambda x.(x x)$$
$$\Omega = \omega\omega$$



Konfluencia-Church-Roser

jednoznačnosť - problém Janka a Marienky

- normálna forma (vzhľadom na β -redukciu) je, keď sa už nedá použiť
- či existujú dve rôzne normálne formy (vzhľadom na β -redukciu) ?
- nejednoznačný "výsledok" pre dva rôzne výpočty ($K = \lambda xy.x$, $I = \lambda x.x$, $\Omega = \omega\omega$)
 - $KI\Omega \rightarrow_{\beta} I$ ale aj
 - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$





Syntaktická analýza λ -termu

(bolo na cvičení)

Pri práci s vašim interpretrom vám (ne)bude chýbať:

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

```
parser "\"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

- použili sme "techniku" rekurzívneho zostupu (Recursive descent), ale o analyzátoroch-parseroch v tejto prednáške bude reč neskôr
- **výstup λ termu** – funkcia `show :: 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 == (APP 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)
```

čo by Vás mohlo inšpirovať



Fold na termoch

```
foldLambda fcie@(lambda,var,apl,con,cn) lterm
  | lterm == (LAMBDA str exp) = lambda str (foldLambda fcie exp)
  | lterm == (VAR str)       = var str
  | lterm == (APP exp1 exp2) = apl  (foldLambda fcie exp1)
                                (foldLambda fcie 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))
```



η -redukcia

- β -redukcia: $(\lambda x.B) E \rightarrow_{\beta} B[x:E]$
- η -redukcia: $\lambda x (B x) \rightarrow_{\eta} B$ ak $x \notin \text{Free}(B)$

Príklad: $\lambda x (f x) \rightarrow_{\eta} x$, resp. v úvode minulej prednášky bolo

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

anti _{η}

$\eta \leftarrow$

let length = $(\lambda f. \lambda ys.(\text{if } (\text{null } xs) \ 0 \ ((+) \ 1 \ (f \ (\text{tail } ys)))) \ \text{length}$ in (length ...

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

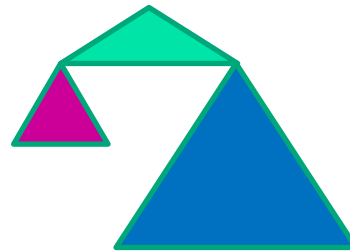


β a η -redukcia

- $\rightarrow_{\beta\eta}$ je **uzáver** $\rightarrow_{\beta} \cup \rightarrow_{\eta}$ vzhľadom na podtermy, čo znamená: matematická definícia:
 - ak $M \rightarrow_{\beta} N$ alebo $M \rightarrow_{\eta} N$, potom $M \rightarrow_{\beta\eta} N$,
 - ak $M \rightarrow_{\beta\eta} N$, potom $(P M) \rightarrow_{\beta\eta} (P N)$ aj $(M Q) \rightarrow_{\beta\eta} (N Q)$,
 - ak $M \rightarrow_{\beta\eta} N$, potom $\lambda x.M \rightarrow_{\beta\eta} \lambda x.N$.

Ako to chápe informatik:

- rekurzívne prejdí celý λ -term
- aplikuj \rightarrow_{β} resp. \rightarrow_{η} kde sa len dá



Otázku, ktorú by si mal položiť:

- v akom poradí, preorder, inorder, ...
- a záleží na poradí ?

$$\omega = \lambda x. xx$$

$$\Omega = \omega \omega$$

$$\omega_3 = \lambda x. xxx$$

Vlastnosti β -redukcie

existuje nekonečná sekvencia

$$\Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \dots$$

existuje neobmedzene puchnúca sekvencia

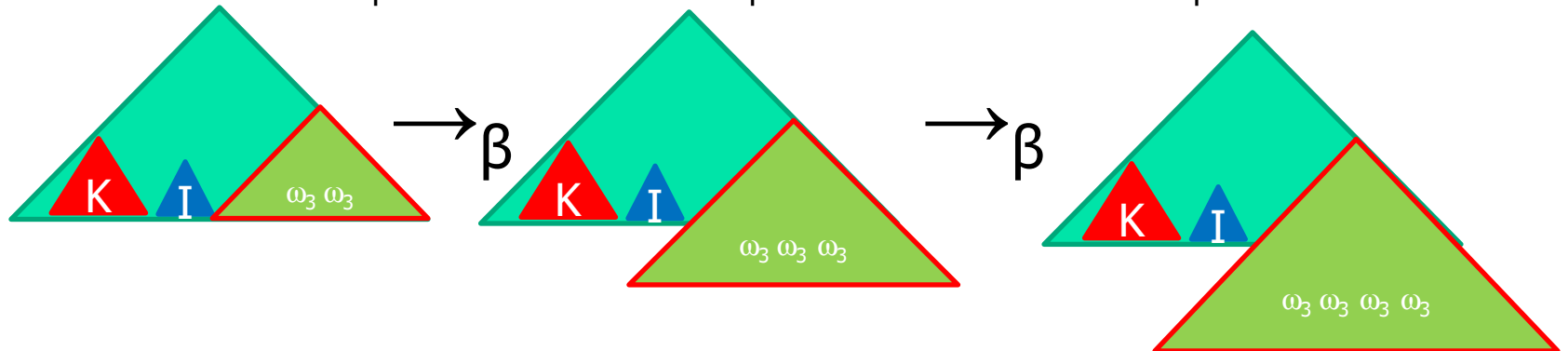
$$\omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \omega_3$$

nejednoznačný výsledok – existuje term s konečným a nekonečným odvodením

$$KI\Omega \rightarrow_{\beta} I \text{ ale aj}$$

$$KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$$

$$KI(\omega_3 \omega_3) \rightarrow_{\beta} KI(\omega_3 \omega_3 \omega_3) \rightarrow_{\beta} KI(\omega_3 \omega_3 \omega_3 \omega_3) \rightarrow_{\beta} \dots$$





Stratégia redukcie

(na výbere záležití)

- **$\beta\eta$ -redex**
je podterm λ -termu, ktorý môžeme prepísať β alebo η redukciou
- **normálna forma** λ -termu nemá redex
- **reducibilný** λ -term nie je v normálnej forme
- **Stratégia redukcie** μ je čiastočné zobrazenie λ -termov, že $M \rightarrow_{\beta\eta} \mu(M)$
- μ výpočet je postupnosť $M, \mu(M), \dots, \mu^i(M), \dots$ a môže byť (ne)konečná

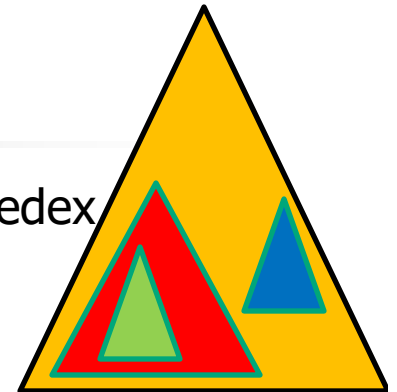
Najznámejšie stratégie

- **leftmost-innermost** – najľavejší redex neobsahuje iný redex

$$(\lambda x.((\lambda y.y) x)) (\lambda z.z) \rightarrow_{\beta} (\lambda x.x) (\lambda z.z) \rightarrow_{\beta} (\lambda z.z)$$

hint: $(\lambda y.y) x \rightarrow_{\beta} x$

redex redukovaný stratégiou je podčiarknutý



- **leftmost-outermost** – najľavejší redex neobsiahnutý v inom redexe

$$(\lambda x.((\lambda y.y) x)) (\lambda z.z) \rightarrow_{\beta} ((\lambda y.y) (\lambda z.z)) \rightarrow_{\beta} (\lambda z.z)$$

hint: $(\lambda x.((\lambda y.y) x)) (\lambda z.z) \rightarrow_{\beta} ((\lambda y.y) x)[x:\lambda z.z] = ((\lambda y.y) (\lambda z.z))$

- **leftmost stratégia** – vyhodnotí funkciu skôr ako argumenty

$$(\lambda x.x)(\lambda y.((\lambda z.z) y)) \rightarrow_{\beta} (\lambda y.((\lambda z.z) y)) \rightarrow_{\beta} (\lambda y.y)$$

- **rightmost stratégia** – vyhodnotí argumenty skôr ako funkciu

$$(\lambda x.x)(\lambda y.((\lambda z.z) y)) \rightarrow_{\beta} (\lambda x.x)(\lambda y.y) \rightarrow_{\beta} (\lambda y.y)$$



Ako to programujeme

- Extrémne drahé riešenie (prečo) ?

```
nf      :: LExp -> LExp      -- test na normálnu formu
nf t    = if t == t' then t else nf t' where t' = oneStepβ t
```

(λx.M)N

```
oneStepβ (APP (LAMBDA x m) n) = substitute m x n
```

(M N)

```
oneStepβ (APP m n)    = if m == m' then
                        (APP m (oneStepβ n))
                        else
                        (APP m' n)
                        where m' = oneStepβ m
```

(λx.M)

```
oneStepβ (LAMBDA x m) = (LAMBDA x (oneStepβ m))
```

- je to innermost či outermost ? Ako vyzerá to druhé ??



Stratégie β -redukcie

Normalizujúca stratégia je taká, ktorá nájde normálnu formu, ak existuje

- **leftmost-innermost** (nie je normalizujúca stratégia)
 - argumenty funkcie sú zredukované skôr ako telo funkcie
 - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$ zacyklí sa pri leftmost-innermost
 - $(\lambda x.x x) (\underline{\lambda x.x} y) \rightarrow_{\beta} (\underline{\lambda x.x x}) y \rightarrow_{\beta} (y y)$
- **leftmost-outermost** (je normalizujúca stratégia, toto nie dôkaz !!!)
 - ak je možné dosadiť argumenty do tela funkcie, urobí sa tak ešte pred ich vyhodnotením, ale tým aj kopíruje redexy ☹
 - $KI\Omega \rightarrow_{\beta} I$
 - $(\underline{\lambda x.x x}) (\lambda x.x y) \rightarrow_{\beta} (\underline{\lambda x.x} y) (\lambda x.x y) \rightarrow_{\beta} y (\underline{\lambda x.x} y) \rightarrow_{\beta} y y$
Call by need (lazy)
 - pri aplikácii funkcie sa do jej tela nedosadzuje argument, ale pointer na hodnotu argumentu, ktorý sa časom event. vyhodnotí



Churchove čísla

- $\underline{0} := \lambda f. \lambda x. x$
- $\underline{1} := \lambda f. \lambda x. f x$
- $\underline{2} := \lambda f. \lambda x. f (f x)$
- ...
- $\underline{n} := \lambda f. \lambda x. f^n x$
- $\text{succ} := \lambda n. \lambda f. \lambda x. f(n f x)$
- $\text{plus} := \lambda m. \lambda n. \lambda f. \lambda x. m f (n f x)$

definujte mult

$\text{mult} := \lambda m. \lambda n. \lambda f. \lambda x. n (m f) x$

lebo $(m f) = \lambda x. (f^m x)$, potom $(n (m f)) = \lambda x. ((f^m)^n x) = \lambda x. (f^{m*n} x)$

•definujte m^n

$\text{exp} := \lambda m. \lambda n. n m$

$\text{exp } m \ n \ f = n \ m \ f = ((n \ m) f) = (m^n f)$

$\text{exp } m \ n \ f \ x = (m^n f) x = f^{(m^n)} x$

•definujte n-1 (na rozmýšľanie)

Churchove čísla

pomalšie

- $\underline{0} := \lambda f. \lambda x. x$
- $\underline{1} := \lambda f. \lambda x. f x$
- $\underline{2} := \lambda f. \lambda x. f (f x)$
- ...
- $\underline{n} := \lambda f. \lambda x. f^n x$
- $\text{succ} := \lambda n. \lambda f. \lambda x. f(n f x)$
- $\text{plus} := \lambda m. \lambda n. \lambda f. \lambda x. m f (n f x)$

$$(\text{succ } \underline{2}) = (\lambda n. \lambda f. \lambda x. f(n f x) \lambda f. \lambda x. f (f x)) = \lambda f. \lambda x. f((\lambda f'. \lambda x'. f' (f' x')) f x) = \lambda f. \lambda x. f (f (f x)) = \underline{3}$$

$$\begin{aligned} ((\text{plus } \underline{2}) \underline{2}) &= ((\lambda m. \lambda n. \lambda f. \lambda x. (m f) (n f x) \underline{2}) \underline{2}) = \\ &= ((\lambda n. \lambda f. \lambda x. (\underline{2} f) (n f x)) \underline{2}) = \\ &= (\lambda f. \lambda x. (\underline{2} f) (\underline{2} f x)) = \\ &= (\lambda f. \lambda x. ((\lambda f'. \lambda x'. f' (f' x')) f) (\underline{2} f x)) = \\ &= (\lambda f. \lambda x. (\lambda x'. f (f x')) (\underline{2} f x)) = \\ &= (\lambda f. \lambda x. (f (f (\underline{2} f x)))) = \\ &= (\lambda f. \lambda x. (f (f (\lambda f. \lambda x. f (f x) f x)))) = (\lambda f. \lambda x. (f (f (f (f x))))) = \underline{4} \end{aligned}$$

Už ste to raz videli

(prvá prednáška)

```
succ := λn.λf.λx.f(n f x)
plus := λm.λn.λf.λx. m f (n f x)
mult := λm.λn.λf.λx. n (m f) x
```

```
true  x y = x
false x y = y
```

```
ifte c t e = c t e
```

```
two  f x = f (f x)
one  f x = f x
zero f x = x
```

```
incr n f x = f (n f x)
incr = \n->\f->\x->(f (n f x))
add  m n f x = m f (n f x)
add  = \m->\n->\f->\x->((m f) (n f x))
mul  m n f x = m (n f) x
```

```
isZero n = n (\_ -> false) true
```

```
decr n = n (\m f x -> f (m incr zero))
        zero
        (\x -> x)
        zero
```

```
fact :: (forall a. (a->a)->a->a) -> (a->a) -> a -> a
fact n =
    ifte (isZero n)
        one
        (mul n (fact (decr n)))
```

```
main =
    -- print $ (decr (add (mul two two) one)) (+1) 0
    -- print $ (fact (add (mul two two) one)) (+1) 0
    print $ (fact (add two
                    (add (mul two two) (mul two two))))
            (+1) 0

-- 3628800
-- (4.75 secs, 2,598,673,208 bytes)
```

Aký je rozdiel medzi týmto interpretrom a tým, čo máte nakódiť na domácu úlohu ?

- Haskell realizuje β -redukciu
- Haskell je striktno typovaný

Pokúste sa dokázať $\text{decr } n = n-1$.

Súbor: Church.hs

$$n(+1) 0 = n$$

$$n f x = f^n x$$

Testovanie domácej úlohy

Potrebuje prirozené čísla, použijeme konštrukciu podľa A.Churcha:

- $0 := \lambda f. \lambda x. x$
- $1 := \lambda f. \lambda x. f x$
- $2 := \lambda f. \lambda x. f (f x)$
- $\text{succ} := \lambda n. \lambda f. \lambda x. f (n f x) = \lambda n. \lambda f. \lambda x. (f ((n f) x))$
- $\text{plus} := \lambda m. \lambda n. \lambda f. \lambda x. m f (n f x) = \lambda m. \lambda n. \lambda f. \lambda x. ((m f) ((n f) x))$
-- idea: $f^m(f^n x) = f^{m+n} x$

Zadáme tieto dve konštrukcie:

```
zero = (LAMBDA "f" (LAMBDA "x" (ID "x")))
succ = (LAMBDA "n" (LAMBDA "f" (LAMBDA "x"
  (APP (ID "f") (APP (APP (ID "n") (ID "f")) (ID "x"))))))
```

potom vypočítame:

```
one = (APP succ zero) = LAMBDA "f" (LAMBDA "x" (APP (ID "f") (ID "x")))
two = (APP succ one) = LAMBDA "f" (LAMBDA "x" (APP (ID "f") (APP (ID "f") (ID "x"))))
three = (APP succ two)
  = LAMBDA "f" (LAMBDA "x" (APP (ID "f") (APP (ID "f") (APP (ID "f") (ID "x")))))
```



Logika a predikáty

| | | | |
|-------|------------------------------|--------------------|--------------------|
| TRUE | $:= \lambda x. \lambda y. x$ | $:= \lambda xy. x$ | (vráti 1.argument) |
| FALSE | $:= \lambda x. \lambda y. y$ | $:= \lambda xy. y$ | (vráti 2.argument) |

| | | |
|---------|---|---------------------------------------|
| AND | $:= \lambda x. \lambda y. x \ y \ \text{FALSE}$ | $:= \lambda xy. x \ y \ \text{FALSE}$ |
| AND x y | $= x \ y \ \text{FALSE}$ | |
| OR | $:= \lambda x. \lambda y. x \ \text{TRUE} \ y$ | $:= \lambda xy. x \ \text{TRUE} \ y$ |
| OR x y | $= x \ \text{TRUE} \ y$ | |

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

| | |
|------------|---|
| IFTHENELSE | $:= \lambda c. \lambda x. \lambda y. (c \ x \ y)$ |
|------------|---|

Príklad:

AND TRUE FALSE
 $\equiv (\lambda x \ y. x \ y \ \text{FALSE}) \ \text{TRUE} \ \text{FALSE} \rightarrow_{\beta} \text{TRUE} \ \text{FALSE} \ \text{FALSE}$
 $\equiv (\lambda x \ y. x) \ \text{FALSE} \ \text{FALSE} \rightarrow_{\beta} \text{FALSE}$

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

PAIR A B = $(\lambda c. c \ A \ B)$

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 $((\lambda x y c. c \ x \ y) \ A \ B) \rightarrow_{\beta}$

LEFT $(\lambda c. c \ A \ B) \rightarrow_{\beta}$

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

$(\lambda c. c \ A \ B) \ (\lambda x y. x) \rightarrow_{\beta}$

$((\lambda x y. x) \ A \ B) \rightarrow_{\beta} A$

definujte 3-ticu.

Konštrukcia n-tice nás oprávňuje písať n-árne funkcie, t.j. funkcie, ktorých argumentom je n-tica – tzv. currying, na počesť pána Haskell Curry:

curry $:: ((a,b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

uncurry $:: (a \rightarrow b \rightarrow c) \rightarrow (a,b) \rightarrow c$

$\lambda(x,y).M$ vs. $(\lambda x. \lambda y. M)$

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



Bolo na prvom cvičení

```
dvojica a b = pair
  where pair f = f a b
```

```
prvy p = p (\ a-> \b -> a)
```

```
druhy p = p (\ a-> \b -> b)
```

```
def cons(a, b):
  def pair(f):
    return f(a, b)
  return pair
```

```
def head(p):
  return p(lambda a,b :a)
```

```
def tail(p):
  return p(lambda a,b:b)
```

```
print(head(cons(4,5)))
print(tail(cons(4,5)))
```

```
def pair(a,b):
  return lambda f: f(a,b)
```

```
def fst(p):
  return p(lambda a,b: a)
```

```
def snd(p):
  return p(lambda a,b: b)
```

```
print(fst(pair(4,5)))
```

$\text{PAIR} \quad := \lambda x. \lambda y. (\lambda c. c \times y) := \lambda x y c. c \times y$



Súčet typov (disjunkcia)

$A+B$ reprezentujeme ako dvojicu $\text{Bool} \times (A|B)$
teda $(\text{TRUE } A)$ alebo $(\text{FALSE } B)$

| | | |
|-----------------------|--------------------------------------|---------------------|
| 1^{st} | $:= \lambda x. \text{PAIR TRUE } x$ | konštruktor pre A |
| 2^{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 ? |

$1^{\text{st}^{-1}} 1^{\text{st}} A \equiv$
 $(\lambda z. \text{RIGHT } z) (\lambda x. \text{PAIR TRUE } x) A \rightarrow_{\beta}$
 $\text{RIGHT } (\text{PAIR TRUE } A) \rightarrow_{\beta} A$



Zoznamy

Domáca úloha (vaším interpretrom):

- $\text{null? (cons a Nil)} \rightarrow_{\beta}^*$
- $\text{head (cons a Nil)} \rightarrow_{\beta}^*$
- $\text{tail (cons a Nil)} \rightarrow_{\beta}^*$
- $\text{head (tail (cons a (cons b Nil)))}$

■ List $t = \text{Nil} \mid \text{Cons } t \text{ (List } t)$

$\text{Nil} = \lambda z.z \text{ TRUE FALSE FALSE}$

$\text{Cons} = \lambda x.\lambda y.\lambda z.z \text{ FALSE } x \ y$

$\text{head} = \lambda p.p (\lambda x.\lambda y.\lambda z.y)$

$\text{tail} = \lambda p.p (\lambda x.\lambda y.\lambda z.z)$

$\text{isNil} = \lambda p.p (\lambda x.\lambda y.\lambda z.x)$

Odvod'me, napr.:

$$\begin{aligned} \text{isNil Nil} &= (\lambda p.p (\lambda x.\lambda y.\lambda z.x)) (\lambda z.z \text{ TRUE FALSE FALSE}) \rightarrow_{\beta} \\ & ((\lambda z.z \text{ TRUE FALSE FALSE}) (\lambda x.\lambda y.\lambda z.x)) \rightarrow_{\beta} \\ & ((\lambda x.\lambda y.\lambda z.x) \text{ TRUE FALSE FALSE}) \rightarrow_{\beta} \\ & \text{TRUE} \end{aligned}$$



Binárne stromy

- $\text{BinTree } t = \text{Empty} \mid \text{Node } t \ (\text{BinTree } t) \ (\text{BinTree } t)$

$\text{Empty} = \lambda g.g \ \text{TRUE} \ (\lambda x.x) \ (\lambda x.x) \ (\lambda x.x)$

$\text{Node} = \lambda x.\lambda y.\lambda z.\lambda g.g \ \text{FALSE} \ x \ y \ z$

$\text{isEmpty} = \lambda t.t \ (\lambda u.\lambda x.\lambda y.\lambda z.u)$

$\text{root} = \lambda t.t \ (\lambda u.\lambda x.\lambda y.\lambda z.x)$

$\text{left} = \lambda t.t \ (\lambda u.\lambda x.\lambda y.\lambda z.y)$

$\text{right} = \lambda t.t \ (\lambda u.\lambda x.\lambda y.\lambda z.z)$



Binárne stromy

Odvodíme, napr.:

$$\begin{aligned} & \text{root (Node a Empty Empty)} \rightarrow_{\beta} \\ & (\lambda t.t (\lambda u.\lambda x.\lambda y.\lambda z.x)) (\text{Node a Empty Empty}) \rightarrow_{\beta} \\ & ((\text{Node a Empty Empty}) (\lambda u.\lambda x.\lambda y.\lambda z.x)) \rightarrow_{\beta} \\ & (((\lambda x.\lambda y.\lambda z.\lambda g.g \text{ FALSE } x y z) a \text{ Empty Empty}) (\lambda u.\lambda x.\lambda y.\lambda z.x)) \rightarrow_{\beta} \\ & ((\lambda g.g \text{ FALSE } a \text{ Empty Empty}) (\lambda u.\lambda x.\lambda y.\lambda z.x)) \rightarrow_{\beta} \\ & ((\lambda u.\lambda x.\lambda y.\lambda z.x) \text{ FALSE } a \text{ Empty Empty}) (\lambda u.\lambda x.\lambda y.\lambda z.x)) \rightarrow_{\beta} \\ & ((\lambda u.\lambda x.\lambda y.\lambda z.x) \text{ FALSE } a \text{ Empty Empty})) \rightarrow_{\beta} \\ & a \end{aligned}$$



where/let

let $v = N$ in M

vlastne to isté ako

M where $v = N$

$\rightarrow (\lambda v.M) N$

M where $v_1 = N_1$

$v_2 = N_2 \dots$

$v_n = N_n$

$\rightarrow (\lambda(v_1, v_2, \dots, v_n).M) (N_1, \dots, N_n)$

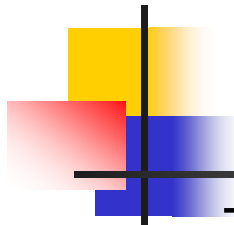
zložený where

$n^*(x+n)$ where

$n = 3$

$x = 4*n+1$

$\rightarrow (\lambda n. (\lambda x.n^*(x+n)) (4*n+1)) 3$



Rekurzia

To, čo stále nevieme, je definovať rekurzívnu funkciu, resp. cyklus.
Na to sa používa konštrukcia pomocou operátora pevného bodu.

Príklad:

$$\text{FAC} := \lambda n. (\text{if } (= n 0) 1 (* n (\text{FAC } (- n 1))))$$

$$\text{FAC} := \lambda n. \text{if } (n = 0) \text{ then } 1 \text{ else } (n * \text{FAC } (n - 1))$$

... trik: η -redukcia $(\lambda x. M x) = M$, ak x nie je $\text{Free}(M)$

$$\text{FAC} := (\lambda \text{fac}. (\lambda n. (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1))))) \text{FAC})$$

$$H := \lambda \text{fac}. (\lambda n. (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1)))))$$

hľadáme funkciu FAC, ktorá má túto vlastnosť:

$$\text{FAC} := (H \text{FAC})$$

hľadaná funkcia FAC je *pevný bod* funkcie H

■ Aký je typ Y ??

Pevný bod

Potrebuje trochu teórie:

Veta:

Pre ľubovoľný λ -term F existuje pevný bod, t.j. X také, že $X = F X$.

Dar nebies (operátor pevného bodu):

$$Y = \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$$

potom

$(Y F)$ je pevný bod F , t.j. $(Y F) = F (Y F)$.

Skúsme to (aspoň) overiť:

$$Y F = (\lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))) F = (\lambda x. F(x x)) (\lambda x. F(x x)) \rightarrow_{\beta}$$

- $F(x x)[x:\lambda x. F(x x)] \rightarrow_{\beta}$
- $F(\lambda x. F(x x) \lambda x. F(x x)) =$
- $F(Y F)$

preto $(Y F)$ je naozaj pevný bod
a je jediný ?



Operátor Y

FAC := (H FAC)

FAC := Y H

H := $\lambda \text{fac} . (\lambda n . (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1)))))$

Platí

$Y H = H (Y H)$

Presvedčíme sa, že Y nám pomôže definovať rekurzívnu funkciu:

$$\text{FAC} = Y H = Y (\lambda \text{fac} . (\lambda n . (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1)))))$$

$$(\lambda f . (\lambda x . f(x x)) (\lambda x . f(x x))) (\lambda \text{fac} . (\lambda n . (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1)))))$$

– toto je faktoriál – verzia nevhodná pre slabšie povahy

$$\begin{aligned} \text{FAC } 1 &= (Y H) 1 && \dots \text{ z vlastnosti pevného bodu} \\ &= H (Y H) 1 \\ &= \lambda \text{fac} . (\lambda n . (\text{if } (= n 0) 1 (* n (\text{fac } (- n 1))))) (Y H) 1 \\ &= \lambda n . (\text{if } (= n 0) 1 (* n ((Y H) (- n 1)))) 1 \\ &= \text{if } (= 1 0) 1 (* 1 ((Y H) (- 1 1))) \\ &= (* 1 ((Y H) (- 1 1))) \\ &= (* 1 ((Y H) 0)) \\ &= (* 1 (H (Y H) 0)) && \dots \text{ trochu zrýchlene} \\ &= (* 1 1) \\ &= 1 \end{aligned}$$



1+2+3+...+n

SUM = $\lambda s. \lambda n. \text{if } (= n 0) 0 (+ n (s (- n 1)))$

DON'T DRINK
AND DERIVE

(Y SUM) 2 =

- $Y (\lambda s. \lambda n. \text{if } (= n 0) 0 (+ n (s (- n 1)))) 2$
- $(\lambda s. \lambda n. \text{if } (= n 0) 0 (+ n (s (- n 1)))) (Y \text{ SUM}) 2$
- $(\lambda n. \text{if } (= n 0) 0 (+ n ((Y \text{ SUM}) (- n 1)))) 2$
- $\text{if } (= 2 0) 0 (+ 2 ((Y \text{ SUM}) (- 2 1)))$
- $(+ 2 ((Y \text{ SUM}) 1))$
- $(+ 2 ((\lambda s. \lambda n. \text{if } (= n 0) 0 (+ n (s (- n 1)))) (Y \text{ SUM}) 1))$
- $(+ 2 ((\lambda n. \text{if } (= n 0) 0 (+ n ((Y \text{ SUM}) (- n 1)))) 1))$
- $(+ 2 ((\text{if } (= 1 0) 0 (+ n ((Y \text{ SUM}) (- 1 1)))))$
- $(+ 2 (+ 1 ((Y \text{ SUM}) 0)))$
- $(+ 2 (+ 1 ((\lambda s. \lambda n. \text{if } (= n 0) 0 (+ n (s (- n 1)))) (Y \text{ SUM}) 0)))$
- $(+ 2 (+ 1 ((\lambda n. \text{if } (= n 0) 0 (+ n ((Y \text{ SUM}) (- n 1)))) 0)))$
- $(+ 2 (+ 1 ((\text{if } (= 0 0) 0 (+ 0 ((Y \text{ SUM}) (- 0 1)))))$
- $(+ 2 (+ 1 0)) = 3$



Cvičenie

- (na zamyslenie) nájdite príklady funkcií
s nekonečným počtom pevných bodov
s práve jedným pevným bodom
- realizujte interpreter λ kalkulu, pokračujte v kóde z minulého cvičenia
tak, aby počítal hodnoty rekurzívnych funkcií

--sucet = \s -> \n -> (if (= n 0) 0 (+ n (s (- n 1))))

```
sucet =  LAMBDA "s"
        (LAMBDA "n"
          (APP
            (APP
              (APP
                (CON "IF")
                (APP (APP (CON "=") (ID "n")) (CN 0)) -- condition
              )
              (CN 0) -- then
            )
            (APP (APP (CON "+") -- else
                      (ID "n"))
                  (APP (ID "s")
                        (APP (APP (CON "-") (ID "n")) (CN 1))
                      )
                )
          )
        )
      )
    )
```



Cvičenie

-- plati $Y f = f(Y f)$

$y = \text{LAMBDA "h"}$

$(\text{APP} (\text{LAMBDA "x"} (\text{APP} (\text{ID "h"}) (\text{APP} (\text{ID "x"}) (\text{ID "x"}))))$
 $(\text{LAMBDA "x"} (\text{APP} (\text{ID "h"}) (\text{APP} (\text{ID "x"}) (\text{ID "x"}))))$

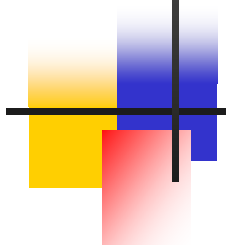
Vyhodnoťte LExp [LExp [y, sucet], CN 4]

$1+2+3+4 = 10$?

A čo faktorial ?

Poznámka:

Obohaťte Váš interpreter o vstavané celé čísla so základnými operáciami (+1, -1, +, *), plus test (napr. na nulu). V opačnom prípade budete bojovať s Church.čísly a interpreter sa vám bude ťažšie ľadiť.





Viacnásobná rekurzia

Veta o pevnom bode: Pre ľubovoľné F_1, F_2, \dots, F_n existujú X_1, X_2, \dots, X_n , že

$$X_1 = F_1 X_1 X_2 \dots X_n$$

$$X_2 = F_2 X_1 X_2 \dots X_n$$

$$\dots$$

$$X_n = F_n X_1 X_2 \dots X_n.$$

vektorovo:

$$(X_1, X_2, \dots, X_n) = (F_1 X_1 X_2 \dots X_n, F_2 X_1 X_2 \dots X_n, \dots, F_n X_1 X_2 \dots X_n)$$

$$\underline{\mathbf{X}} = (F_1 (p_1 \underline{\mathbf{X}}) (p_2 \underline{\mathbf{X}}) \dots (p_n \underline{\mathbf{X}}), \dots, F_n (p_1 \underline{\mathbf{X}}) (p_2 \underline{\mathbf{X}}) \dots (p_n \underline{\mathbf{X}}))$$

$$\underline{\mathbf{X}} = \lambda \underline{\mathbf{z}}. (F_1 (p_1 \underline{\mathbf{z}}) (p_2 \underline{\mathbf{z}}) \dots (p_n \underline{\mathbf{z}}), \dots, F_n (p_1 \underline{\mathbf{z}}) (p_2 \underline{\mathbf{z}}) \dots (p_n \underline{\mathbf{z}})) \underline{\mathbf{X}}$$

p_i = i-ta projekcia vektora.

preto

$$\underline{\mathbf{X}} = \mathbf{Y} (\lambda \underline{\mathbf{z}}. (F_1 (p_1 \underline{\mathbf{z}}) (p_2 \underline{\mathbf{z}}) \dots (p_n \underline{\mathbf{z}}), \dots, F_n (p_1 \underline{\mathbf{z}}) (p_2 \underline{\mathbf{z}}) \dots (p_n \underline{\mathbf{z}}))))$$



Primitívna rekurzia

Primitívne rekurzívna funkcia je:

- nulová funkcia $N^n \rightarrow N$,
- $\text{succ}: N \rightarrow N$,
- projekcia $p_i: N^n \rightarrow N$, $p_i x_1 x_2 \dots x_n = x_i$
- kompozícia $f x_1 x_2 \dots x_n = g(h_1(x_1 x_2 \dots x_n) \dots h_m(x_1 x_2 \dots x_n))$
- primitívna rekurzia $g: N^n \rightarrow N$, $h: N^{n+2} \rightarrow N$, potom $f: N^{n+1} \rightarrow N$
 $f 0 \quad x_1 x_2 \dots x_n = g(x_1 x_2 \dots x_n)$
 $f (n+1) x_1 x_2 \dots x_n = h(f(n x_1 x_2 \dots x_n) n x_1 x_2 \dots x_n)$

Parciálne/Čiastočne vyčísliteľná (nemusí byť totálna):

- μ -rekurzia $r: N^{n+1} \rightarrow N$, potom $f: N^{n+1} \rightarrow N$
 $f y \quad x_1 x_2 \dots x_n = \min_z.(r(z x_1 x_2 \dots x_n) = y)$

λ-vypočítateľná funkcia

Parciálna funkcia $f : N^n \rightarrow N$ je λ-vypočítateľná, ak existuje λ-term F taký, že $F \underline{x_1} \underline{x_2} \dots \underline{x_n}$ sa zredukuje na $\underline{f \ x_1 \ x_2 \dots \ x_n}$, ak n -tica $x_1 \ x_2 \dots \ x_n$ patrí do def.oboru f
 $F \underline{x_1} \underline{x_2} \dots \underline{x_n}$ nemá normálnu, ak n -tica $x_1 \ x_2 \dots \ x_n$ nepatrí do def.oboru f

Veta: Každá parciálne vyčísliteľná funkcia je λ-vypočítateľná.

Dôkaz:

- nulová fcia, succ, projekcie p_i , kompozícia priamočiaro
- primitívna rekurzia $g : N^n \rightarrow N$, $h : N^{n+2} \rightarrow N$, potom $f : N^{n+1} \rightarrow N$

$$f \ 0 \quad x_1 \ x_2 \dots x_n = g(x_1 \ x_2 \dots x_n)$$

$$f \ (n+1) \ x_1 \ x_2 \dots x_n = h(f(n \ x_1 \ x_2 \dots x_n) \ n \ x_1 \ x_2 \dots x_n)$$

$$F = \mathbf{Y} (\lambda f. \lambda y. \lambda x_1. \lambda x_2 \dots \lambda x_n. (\text{if } (\text{isZero } y) \ G(x_1 \ x_2 \dots x_n) \text{ then } \\ \text{else } H(f((\text{pred } y) \ x_1 \ x_2 \dots x_n) \ (\text{pred } y) \ x_1 \ x_2 \dots x_n))))$$
- μ -rekurzia $r : N^{n+1} \rightarrow N$

$$F = \lambda y \lambda x_1 \lambda x_2 \dots \lambda x_n. \mathbf{Y} (\lambda h. \lambda z. (\text{if } (\text{eq } y \ G(z \ x_1 \ x_2 \dots x_n)) \text{ then } z \text{ else } h \ (\text{succ } z)))$$

Veta: Každá λ-vypočítateľná je parciálne vyčísliteľná funkcia.

Weak head normal form

(slabo hlavová normálna forma)

Head normal form (h.n.f)

- $(\lambda x_1. \lambda x_2. \dots \lambda x_k. v) M_1 M_2 \dots M_n$
- v je premenná (resp. konštanta),
- pre ľubovoľné $r \leq n$, $(\dots((v M_1) M_2) \dots M_r)$ nie je redex .

Ak $k=0$, konštanta či premenná s málo argumentami

Ak $k>0$, λ -abstrakcia s nereducibilným telom

Weak head normal form (w.h.n.f)

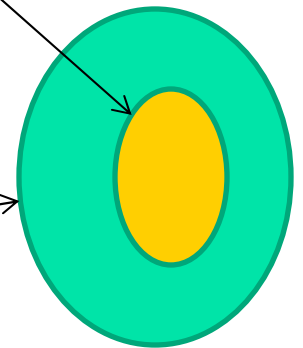
- $v M_1 M_2 \dots M_n$
- v je premenná alebo λ -abstrakcia (resp. konštanta),
- pre ľubovoľné $r \leq n$, $(\dots((v M_1) M_2) \dots M_r)$ nie je redex .

Konštanta, premenná alebo λ -abstrakcia s málo argumentami.

$\lambda x.((\lambda y.y) z)$ nie je h.n.f. (až po red. $((\lambda y.y) z) \rightarrow_{\beta} z$), ale je w.h.n.f.

$(k, n \in \mathbb{N})$

$(n \in \mathbb{N})$





Najznámejšie stratégie

- weak leftmost outermost (call by need/output driven/lazy/full lazy)

$$\underline{(\lambda x. \lambda y. (x \ y))} \ (\lambda z. z) \rightarrow_{\beta} \lambda y. ((\lambda z. z) \ y) \quad \text{w.h.n.f.}$$

redukuje argumenty funkcie, len ak ich treba

Keďže w.h.n.f. môže obsahovať redex, tak nenormalizuje úplne...

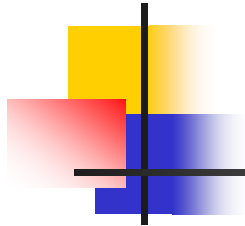
- strong leftmost outermost (call by name/demand driven)

$$\underline{(\lambda x. \lambda y. (x \ y))} \ (\lambda z. z) \rightarrow_{\beta} \lambda y. ((\underline{\lambda z. z}) \ y) \rightarrow_{\beta} \lambda y. y \quad \text{n.f.}$$

redukuje argumenty funkcie, len ak ich treba, ale pokračuje v hľadaní redexov, kým nejaké sú

normalizuje úplne...

- eager - argumenty najprv (call by value/data driven/strict)
nenormalizuje...



Lazy

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x.x)(* 3 4))) (\lambda x.(+2 x) 6) \rightarrow_{\beta}$
- $(\lambda y. (* (+ ((\lambda x.x)(* 3 4)) ((\lambda x.x)(* 3 4))) y) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* (+ ((\lambda x.x)(* 3 4)) ((\lambda x.x)(* 3 4))) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* (+ (* 3 4) ((\lambda x.x)(* 3 4))) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* (+ 12 ((\lambda x.x)(* 3 4))) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* (+ 12 (* 3 4)) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* (+ 12 12) (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* 24 (\lambda x.(+2 x) 6)) \rightarrow_{\beta}$
- $(* 24 (+2 6)) \rightarrow_{\beta}$
- $(* 24 8) \rightarrow_{\beta}$
- 192



Full lazy

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6) \rightarrow_{\beta}$
- $(\lambda y. (* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) y) (\lambda x. (+ 2 x) 6)) \rightarrow_{\beta}$
- $(* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6)) \rightarrow_{\beta}$
- $(* (+ (* 3 4) (* 3 4)) (\lambda x. (+ 2 x) 6)) \rightarrow_{\beta}$
- $(* (+ 12 12) (\lambda x. (+ 2 x) 6)) \rightarrow_{\beta}$
- $(* 24 (\lambda x. (+ 2 x) 6)) \rightarrow_{\beta}$
- $(* 24 (+ 2 6)) \rightarrow_{\beta}$
- $(* 24 8) \rightarrow_{\beta}$
- 192



Strict

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (+ 2 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) 8 \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) (* 3 4)) 8 \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) 12) 8 \rightarrow_{\beta}$
- $(\lambda y. (* (+ 12 12) y)) 8 \rightarrow_{\beta}$
- $(* (+ 12 12) 8) \rightarrow_{\beta}$
- $(* 24 8) \rightarrow_{\beta}$
- 192



Eager

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) 12)) (\lambda x. (+ 2 x) 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) 12) (\lambda x. (+ 2 x) 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) 12) (+ 2 6) \rightarrow_{\beta}$
- $(\lambda x. \lambda y. (* (+ x x) y) 12) 8 \rightarrow_{\beta}$
- $(\lambda y. (* (+ 12 12) y)) 8 \rightarrow_{\beta}$
- $(\lambda y. (* 24 y)) 8 \rightarrow_{\beta}$
- $(* 24 8) \rightarrow_{\beta}$
- 192

Church-Rosser vlastnosť

(konzistentnosť λ -kalkulu)

pre ľubovoľnú trojicu termov M, M_1, M_2 takých, že

$$M \rightarrow_{\beta}^* M_1 \text{ a } \rightarrow_{\beta}^* M_2$$

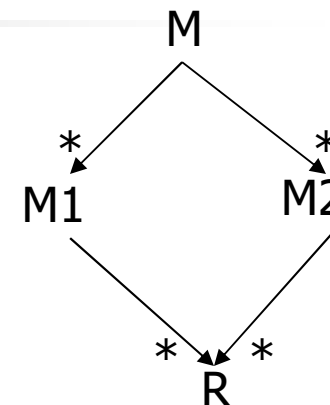
existuje R , že

$$M_1 \rightarrow_{\beta}^* R \text{ a } M_2 \rightarrow_{\beta}^* R$$

Inak:

$$(\leftarrow_{\beta}^* \circ \rightarrow_{\beta}^*) \subseteq (\rightarrow_{\beta}^* \circ \leftarrow_{\beta}^*)$$

teda ak $M_1 \leftarrow_{\beta}^* M \rightarrow_{\beta}^* M_2$, potom existuje R , že $M_1 \rightarrow_{\beta}^* R \leftarrow_{\beta}^* M_2$



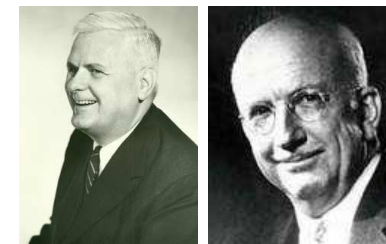
Veta: β -redukcia spĺňa Church-Rosserovu vlastnosť

Dôkazy sú technicky náročné:

- 1936 Church, Rosser: Some properties of conversion
- 1981 Barendregt
- 1981 Löf, Tait

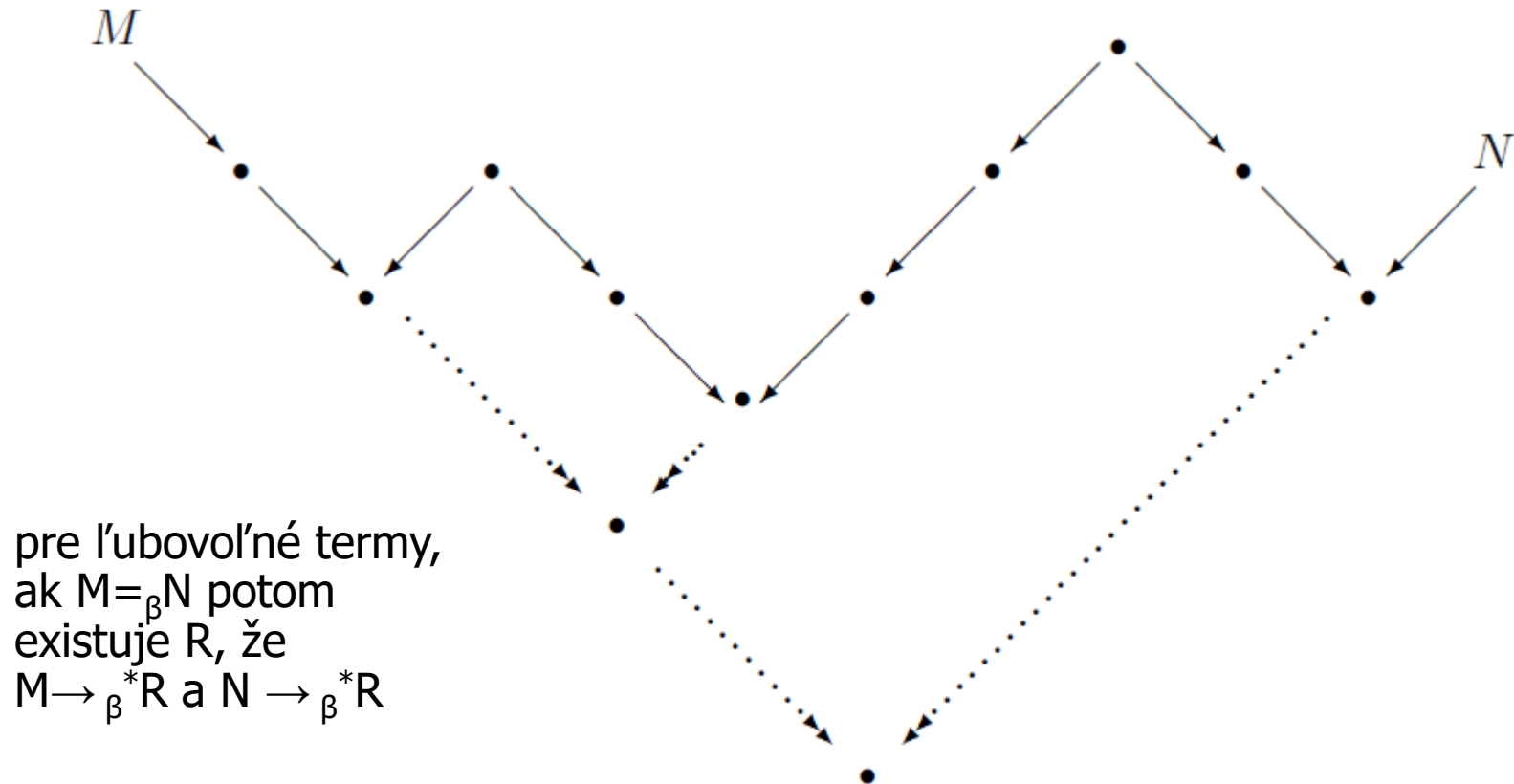
Dôsledok:

ak term má normálnu formu vzhľadom na \rightarrow_{β} , potom je jednoznačne určená



β ekvivalencia

■ $=_{\beta} \equiv (\rightarrow_{\beta} \cup \leftarrow_{\beta})^*$ Príklad: $KI\Omega =_{\beta} I$



Slabá Church-Rosser vlastnosť

pre ľub. trojicu termov M, M_1, M_2 takých, že

$$M \rightarrow M_1 \text{ a } M \rightarrow M_2$$

existuje R , že

$$M_1 \rightarrow^* R \text{ a } M_2 \rightarrow^* R$$

Inak:

$$(\leftarrow \circ \rightarrow) \subseteq (\rightarrow^* \circ \leftarrow^*)$$

teda ak $M_1 \leftarrow M \rightarrow M_2$, potom existuje R , že $M_1 \rightarrow^* R \leftarrow^* M_2$

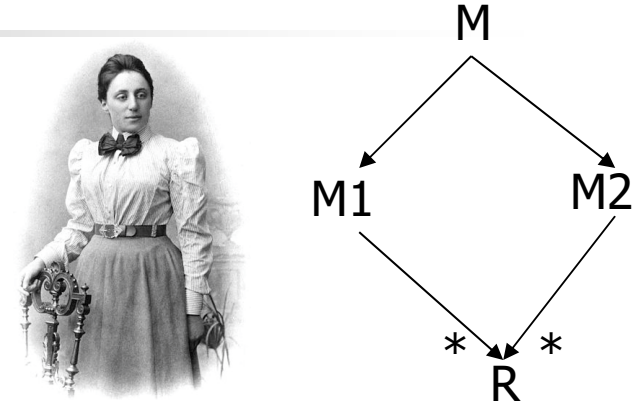
Veta: **Nech \rightarrow je noetherovská/silne normalizujúca/terminujúca relácia.**

- \rightarrow **má Church-Rosser vlastnosť (confluent) je ekvivalentné s**
- \rightarrow **má slabú Church-Rosser vlastnosť (local confluent)**

Dôkaz sporom (SCR implikuje CR, spor: SCR and \neg CR):

- Nech M má dve normálne formy, $M_1 <> M_2$, t.j. $M \rightarrow^* M_1$ a $M \rightarrow^* M_2$.
- M nie je v normálnej forme (ak by bolo, $M = M_1 = M_2$ a pritom $M_1 <> M_2$),
- potom existuje M' , že $M \rightarrow M'$,
- M' má tiež dve normálne formy, ak by nie, spor s lokálnou konfluentnosťou,
- M'', M''', M'''' , a t.d' (noetherovskosť relácie vyrobí spor).

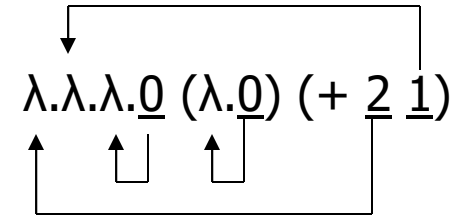
Zamyslenie: je noetherovská podmienka podstatná, neplatí veta aj bez nej ?



de Bruijn index

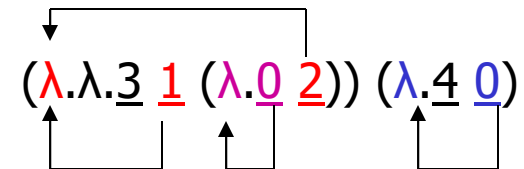
čo robilo problémy pri substitúcii, sú mená premenných
idea (pána de Bruijn): premenné nahradíme indexami

- $\lambda x. (+ x 1)$
 - $\lambda. (+ \underline{0} 1)$
- $\lambda x. \lambda y. \lambda f. f ((\lambda x. x) (+ x y))$
 - $\lambda. \lambda. \lambda. \underline{0} ((\lambda. \underline{0}) (+ \underline{2} \underline{1}))$



index: neform.: cez koľko λ treba vyskákať, aby sme našli λ danej premennej

- α -konverzia neexistuje lebo premenné nemajú mená
 - dôsledok: rôzne premenné môžu mať rovnaký index ($v \neq$ kontextoch)
 - voľné premenné majú index $>$ hĺbkou λ -vnorení
- $(\lambda x. \lambda y. ((z x) (\lambda u. (u x)))) (\lambda x. (w x))$
 - $(\lambda. \lambda. ((\underline{3} \underline{1}) (\lambda. (\underline{0} \underline{2})))) (\lambda. (\underline{4} \underline{0}))$



β -redukcia s de Bruijn-indexami

Príklady:

- $K = \lambda x. \lambda y. x$
 - $\lambda. \lambda. \underline{1}$
- $S = \lambda x. \lambda y. \lambda z. ((x\ z)\ (y\ z))$
 - $\lambda. \lambda. \lambda. ((\underline{2}\ \underline{0})\ (\underline{1}\ \underline{0}))$

- $\lambda x. (+\ x\ 1)\ 5$
 - $\lambda. (+\ \underline{0}\ 1)\ 5 = (+\ 5\ 1)$

hypotéza, ako by to mohlo fungovať
 β -redukcia $(\lambda.M)\ N = M[\underline{0}:N]\ ???$
ale nefunguje...

- $K\ a\ b = (\lambda x. \lambda y. x)\ a\ b$
 - $(\lambda. \lambda. \underline{1}\ a)\ b = \lambda. a\ b = a$

skúsme intuitívne

- $(\lambda x. \lambda y. ((z\ x)\ (\lambda u. (u\ x))))\ (\lambda x. (w\ x))$
 - $(\lambda. \lambda. ((\underline{3}\ \underline{1})\ (\lambda. (\underline{0}\ \underline{2}))))\ (\lambda. (\underline{4}\ \underline{0}))$
 - $(\lambda. \lambda. ((\underline{3}\ \underline{\square})\ (\lambda. (\underline{0}\ \underline{\square}))))\ (\lambda. (\underline{4}\ \underline{0}))$
 - $(\lambda. (\underline{2}\ (\lambda. \underline{5}\ \underline{0}))\ (\lambda. (\underline{0}\ (\lambda. \underline{6}\ \underline{0}))))$

označíme si miesta, kam sa substituuje
nahrať, ale pozor na voľné premenné

$(\lambda y. \underline{z}\ (\lambda x. \underline{w}\ \underline{x})\ (\lambda u. \underline{u}\ (\lambda x. \underline{w'}\ \underline{x})))$

β s de Bruijn.indexami

Substitúcia $[t_0, t_1, \dots, t_n] = [\underline{0}:t_0][\underline{1}:t_1]\dots[\underline{n}:t_n]$

- $\underline{k}[t_0, t_1, \dots, t_n] = t_k, k \leq n$
- $(M N) [t_0, t_1, \dots, t_n] = (M[t_0, t_1, \dots, t_n] N[t_0, t_1, \dots, t_n])$
- $(\lambda M) [t_0, t_1, \dots, t_n] = (\lambda M[\underline{0}, t_0^{\wedge 1}, t_1^{\wedge 1}, \dots, t_n^{\wedge 1}])$

$t^{\wedge 1}$ – pripočítaj 1 k voľným premenným

β : $(\lambda M) N = M[N, \underline{0}, \underline{1}, \underline{2}, \underline{3}, \dots]$

- $(\lambda. \lambda. \underline{1} a) b = ((\lambda. \underline{1}) [a, \underline{0}, \underline{1}, \underline{2}, \dots]) b = (\lambda. (\underline{1} [\underline{0}, a, \underline{1}, \underline{2}, \dots])) b = \lambda. a \quad b = a$
- a, b sú konštanty neobs. premenné

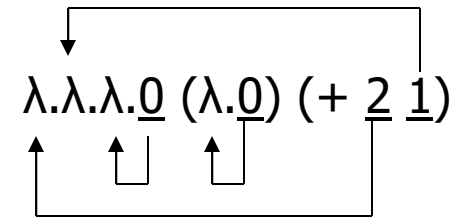
Príklad z predošlého slajdu:

- $(\lambda. \lambda. ((\underline{3} \underline{1}) (\lambda. (\underline{0} \underline{2})))) (\lambda. (\underline{4} \underline{0})) =$
 - $\lambda. ((\underline{3} \underline{1}) (\lambda. (\underline{0} \underline{2}))) [(\lambda. (\underline{4} \underline{0})), \underline{0}, \underline{1}, \underline{2}, \dots] =$
 - $\lambda. (((\underline{3} \underline{1}) [\underline{0}, (\lambda. (\underline{5} \underline{0})), \underline{1}, \underline{2}, \dots]) ((\lambda. (\underline{0} \underline{2})) [\underline{0}, (\lambda. (\underline{5} \underline{0})), \underline{1}, \underline{2}, \dots]))) =$
 - $\lambda. ((\underline{2} (\lambda. (\underline{5} \underline{0}))) (\lambda. (\underline{0} \underline{2})) [\underline{0}, (\lambda. (\underline{5} \underline{0})), \underline{1}, \underline{2}, \dots])) =$
 - $\lambda. ((\underline{2} (\lambda. (\underline{5} \underline{0}))) (\lambda. (\underline{0} \underline{2}) [\underline{0}, \underline{1}, (\lambda. (\underline{6} \underline{0})), \underline{2}, \underline{3}, \underline{4}, \dots])))$
 - $\lambda. ((\underline{2} (\lambda. (\underline{5} \underline{0}))) (\lambda. (\underline{0} (\lambda. (\underline{6} \underline{0}))))))$
- $= (\lambda y. (\underline{z} (\lambda x. (\underline{w} \underline{x}))) (\lambda u. (\underline{u} (\lambda x. (\underline{w'} \underline{x}))))))$

de Bruijn index

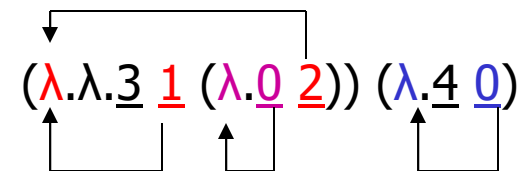
čo robilo problémy pri substitúcii, sú mená premenných
 idea (pána de Bruijn): premenné nahradíme indexami

- $\lambda x. (+ x 1)$
 - $\lambda. (+ \underline{0} 1)$
- $\lambda x. \lambda y. \lambda f. f ((\lambda x. x) (+ x y))$
 - $\lambda. \lambda. \lambda. \underline{0} ((\lambda. \underline{0}) (+ \underline{2} \underline{1}))$



index: neform.: cez koľko λ treba vyskákať, aby sme našli λ danej premennej

- α -konverzia neexistuje lebo premenné nemajú mená
- dôsledok: rôzne premenné môžu mať rovnaký index ($v \neq$ kontextoch)
- voľné premenné majú index $>$ hĺbkou λ -vnorení



β -redukcia s de Bruijn-indexami

Príklady:

- $K = \lambda x. \lambda y. x$
 - $\lambda. \lambda. \underline{1}$
- $S = \lambda x. \lambda y. \lambda z. ((x\ z)\ (y\ z))$
 - $\lambda. \lambda. \lambda. ((\underline{2}\ \underline{0})\ (\underline{1}\ \underline{0}))$

- $\lambda x. (+\ x\ 1)\ 5$
 - $\lambda. (+\ \underline{0}\ 1)\ 5 = (+\ 5\ 1)$

hypotéza, ako by to mohlo fungovať
 β -redukcia $(\lambda.M)\ N = M[\underline{0}:N]\ ???$
ale nefunguje...

- $K\ a\ b = (\lambda x. \lambda y. x)\ a\ b$
 - $(\lambda. \lambda. \underline{1}\ a)\ b = \lambda. a\ b = a$

skúsme intuitívne

- $(\lambda x. \lambda y. ((z\ x)\ (\lambda u. (u\ x))))\ (\lambda x. (w\ x))$
 - $(\lambda. \lambda. ((\underline{3}\ \underline{1})\ (\lambda. (\underline{0}\ \underline{2}))))\ (\lambda. (\underline{4}\ \underline{0}))$
 - $(\lambda. \lambda. ((\underline{3}\ \underline{\square})\ (\lambda. (\underline{0}\ \underline{\square}))))\ (\lambda. (\underline{4}\ \underline{0}))$
 - $(\lambda. (\underline{2}\ (\lambda. \underline{5}\ \underline{0}))\ (\lambda. (\underline{0}\ (\lambda. \underline{6}\ \underline{0}))))$

označíme si miesta, kam sa substituuje
nahrať, ale pozor na voľné premenné
- $(\lambda y. \underline{z}\ (\lambda x. \underline{w}\ \underline{x})\ (\lambda u. \underline{u}\ (\lambda x. \underline{w'}\ \underline{x})))$



- $K = \lambda x. \lambda y. x$
 - $\lambda. \lambda. \underline{1}$
- $S = \lambda x. \lambda y. \lambda z. x \ z \ (y \ z)$
 - $\lambda. \lambda. \lambda. \underline{2} \ \underline{0} \ (\underline{1} \ \underline{0})$

Ďalší testovací příklad

- $S \ K \ K = ((\lambda. \lambda. \lambda. \underline{2} \ \underline{0} \ (\underline{1} \ \underline{0})) \ \lambda. \lambda. \underline{1}) \ \lambda. \lambda. \underline{1} =$
 - $(\lambda. \lambda. \underline{2} \ \underline{0} \ (\underline{1} \ \underline{0})) [\lambda. \lambda. \underline{1}, \underline{0}, \underline{1}, \underline{2}, \dots] \ \lambda. \lambda. \underline{1} =$
 - $(\lambda. \lambda. (\underline{2} \ \underline{0} \ (\underline{1} \ \underline{0})) [\underline{0}, \underline{1}, \lambda. \lambda. \underline{1}, \underline{0}, \underline{1}, \underline{2}, \dots]) \ \lambda. \lambda. \underline{1} =$
 - $(\lambda. \lambda. ((\lambda. \lambda. \underline{1}) \ \underline{0} \ (\underline{1} \ \underline{0}))) \ \lambda. \lambda. \underline{1} =$
 - $(\lambda. ((\lambda. \lambda. \underline{1}) \ \underline{0} \ (\underline{1} \ \underline{0}))) [\lambda. \lambda. \underline{1}, \underline{0}, \underline{1}, \underline{2}, \dots] =$
 - $\lambda. (((\lambda. \lambda. \underline{1}) \ \underline{0} \ (\underline{1} \ \underline{0})) [\underline{0}, \lambda. \lambda. \underline{1}, \underline{0}, \underline{1}, \underline{2}, \dots]) =$
 - $\lambda. ((\lambda. \lambda. \underline{1}) \ \underline{0} \ (\lambda. \lambda. \underline{1} \ \underline{0})) =$
 - $\lambda. \underline{0} = I$



Cvičenie

Prepíšte do de Bruijn notácie

- $\lambda x. \lambda y. y (\lambda z. z x) x$
- $\lambda x. (\lambda x. x x) (\lambda y. y (\lambda z. x))$
- $(\lambda x. + x ((\lambda y. y) (- x (\lambda z. 3)(\lambda y. y y))))$

Definujte funkciu na prevod do de Bruijn notácie.

Implementujte β -redukciu s pomocnými funkciami