

Lambda calculus 2

Štruktúra prednášok:

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

domáca úloha: interpretér λ -kalkulu, ...

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



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)
```



β a η -redukcia

- β -redukcia: $(\lambda x.B) E \rightarrow_{\beta} B[x:E]$
- η -redukcia: $\lambda x.(B x) \rightarrow_{\eta} B$ ak $x \notin \text{Free}(B)$
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 podtermíny, čo znamená:
 - 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 $\lambda x.M \rightarrow_{\beta_{\eta}} \lambda x.N$.



Vlastnosti β -redukcie

- známe λ -termy
 - $\omega = \lambda x.xx$
 - $\Omega = \omega\omega$
 - $\omega_3 = \lambda x.xxx$
- 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
 - $KI\Omega \rightarrow_{\beta} I$ ale aj
 - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$
- existuje term s dvomi rôznymi normálnymi formami ?



Stratégia redukcie

(na výber záleží)

- $\beta\eta$ -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

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

- leftmost-outermost – najľavejší redex neobsiahnutý v inom redexe

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

- leftmost – vyhodnotí funkciu skôr ako argumenty

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

- rightmost – vyhodnotí argumenty skôr ako funkciu

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



- $$\text{nf} \quad :: \text{LExp} \rightarrow \text{LExp}$$

$$\begin{aligned} \text{oneStep}_{\beta}(\text{APL } m \ n) &= \text{if } m == m' \text{ then} \\ &\quad (\text{APL } m \ (\text{oneStep}_{\beta} n)) \\ &\quad \text{else} \\ &\quad (\text{APL } m' \ n) \\ &\quad \text{where } m' = \text{oneStep}_{\beta} m \end{aligned}$$

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



Stratégie β -redukcie

- kdekoľvek, až kým nie je v n.f.
 - **leftmost-innermost** (nie je normalizujúca stratégia)
 - argumenty funkcie sú zredukované skôr ako telo funkcie
 - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$
 - $(\lambda x.x x) (\lambda x.x y) \rightarrow_{\beta} (\lambda x.x x)y \rightarrow_{\beta} yy$
 - **leftmost-outermost** (je normalizujúca stratégia)
 - ak je možné dosadiť argumenty do tela funkcie, urobí sa tak ešte pred ich vyhodnotením, ale tým aj kopíruje redexy ☹
 - $KI\Omega \rightarrow_{\beta} I$
 - $(\lambda x.x x) (\lambda x.x y) \rightarrow_{\beta} (\lambda x.x y) (\lambda x.x y) \rightarrow_{\beta} y (\lambda x.x y) \rightarrow_{\beta} y y$
- 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

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

Domáca úloha
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

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

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

•definujte n-1 (na rozmýšľanie)

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

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

Testovanie domácej úlohy

Potrebuje prirodzené 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"  
    (APL (ID "f") (APL (APL (ID "n") (ID "f")) (ID "x"))))))
```

potom vypočítame:

```
one = (APL succ zero) = LAMBDA "f" (LAMBDA "x" (APL (ID "f") (ID "x")))  
two = (APL succ one) = LAMBDA "f" (LAMBDA "x" (APL (ID "f") (APL (ID "f") (ID "x"))))  
three = (APL succ two)  
    = LAMBDA "f" (LAMBDA "x" (APL (ID "f") (APL (ID "f") (APL (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	$:= \lambda x. \lambda y. x \ y \ \text{FALSE}$	$:= \lambda xy. x \ y \ \text{FALSE}$
OR	$:= \lambda x. \lambda y. x \ \text{TRUE} \ y$	$:= \lambda xy. x \ \text{TRUE} \ y$
NOT	$:= \lambda x. x \ \text{FALSE} \ \text{TRUE}$	
IFTHENELSE	$:= \lambda c. \lambda x. \lambda y. (c \ x \ y)$	

Príklad:

AND TRUE FALSE

$$\begin{aligned} &\equiv (\lambda x \ y. x \ y \ \text{FALSE}) \ \text{TRUE} \ \text{FALSE} \quad \beta \quad \text{TRUE} \ \text{FALSE} \ \text{FALSE} \\ &\equiv (\lambda x \ y. x) \ \text{FALSE} \ \text{FALSE} \quad \beta \quad \text{FALSE} \end{aligned}$$

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) β 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:

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

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

PAIR $:= \lambda x. \lambda y. (\lambda c. c \times y) := \lambda x y c. c \times y$



Súčet typov (disjunkcia)

A+B reprezentujeme ako dvojicu Bool x (A|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 ?

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



Zoznamy

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

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

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

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

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

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

Odvodíme, napr.:

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

Domáca úloha (vaším interpretrom):

"null? (cons a Nil) β^*

"head (cons a Nil) β^*

"tail (cons a Nil) β^*

"head (tail (cons a (cons b Nil))) β



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.:

```
root (Node a Empty Empty)      β
  (λt.t (λu.λx.λy.λz.x)) (Node a Empty Empty)      β
    ((Node a Empty Empty) (λu.λx.λy.λz.x))      β
      (((λx.λy.λz.λg.g FALSE x y z) a Empty Empty) (λu.λx.λy.λz.x))      β
        ((λg.g FALSE a Empty Empty) (λu.λx.λy.λz.x))      β
          ((λu.λx.λy.λz.x) FALSE a Empty Empty) (λu.λx.λy.λz.x))      β
            ((λu.λx.λy.λz.x) FALSE a Empty Empty))      β
              a
```



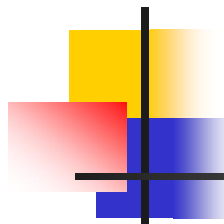
where

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

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

zložený where

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



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:

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:

$$Y F = (\lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))) F = (\lambda x. F(x x)) (\lambda x. F(x x)) \quad \beta$$

- $F(x x)[x:\lambda x. F(x x)] \quad \beta$
- $F(\lambda x. F(x x) \lambda x. F(x x)) =$
- $F (Y F)$

preto $(Y F)$ je naozaj pevný bod F



Operátor Y

$FAC := (H\ FAC)$

$FAC := Y\ H$

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

Platí

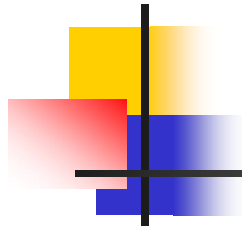
$Y\ H = H\ (Y\ H)$

Presvedčíme sa, že Y nám pomôže definovať rekurzívnu funkciu:

$FAC = Y\ H = Y\ (\lambda fac. (\lambda n. (if\ (= n\ 0)\ 1\ (*\ n\ (fac\ (-\ n\ 1)))))$
 $(\lambda f. (\lambda x. f(x\ x)))\ (\lambda x. f(x\ x)))\ (\lambda fac. (\lambda n. (if\ (= n\ 0)\ 1\ (*\ n\ (fac\ (-\ n\ 1)))))$

– toto je faktoriál – verzia nevhodná pre slabšie povahy

$FAC\ 1 = (Y\ H)\ 1$... z vlastnosti pevného bodu
 $= H\ (Y\ H)\ 1$
 $= \lambda fac. (\lambda n. (if\ (= n\ 0)\ 1\ (*\ n\ (fac\ (-\ n\ 1)))))\ (Y\ H)\ 1$
 $= \lambda n. (if\ (= n\ 0)\ 1\ (*\ n\ ((Y\ H)\ (-\ n\ 1))))\ 1$
 $= if\ (= 1\ 0)\ 1\ (*\ 1\ ((Y\ H)\ (-\ 1\ 1)))$
 $= (*\ 1\ ((Y\ H)\ (-\ 1\ 1)))$
 $= (*\ 1\ ((Y\ H)\ 0))$
 $= (*\ 1\ (H\ (Y\ H)\ 0))$... trochu zrýchlene
 $= (*\ 1\ 1)$
 $= 1$



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



Cvičenie

```
-- plati  $Y f = f(Y f)$   
y = LAMBDA "h"  
    (APL (LAMBDA "x" (APL (ID "h") (APL (ID "x") (ID "x")))))  
    (LAMBDA "x" (APL (ID "h") (APL (ID "x") (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.čísлами a interpreter sa vám bude ťažšie ľadiť.

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)

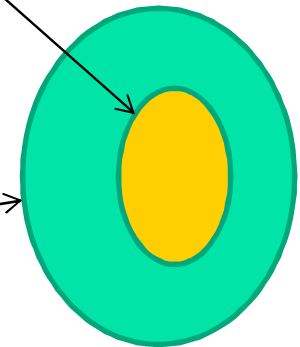
- $v M_1 M_2 \dots M_n$
- v je premenná alebo λ -abstrakcia (resp. konštanta),
- pre ľubovoľné $r \leq n$, $(\dots((v M_1) M_2) \dots M_r)$ nie je redex .

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

redukuje argumenty funkcie, len ak ich treba

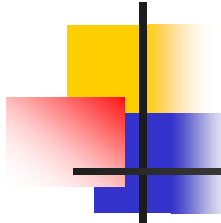
Keďže w.h.n.f. môže obsahovať redex, tak nenormalizuje úplne...

- strong leftmost outermost (call by name/demand driven)
$$\underline{(\lambda x. \lambda y. (x \ y))} \ (\lambda z. z) \quad \beta \quad \lambda y. ((\lambda z. z) \ y) \quad \beta \quad \lambda y. y \quad \text{n.f.}$$

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

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda y. (* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) y) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ (* 3 4) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ 12 ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ 12 (* 3 4)) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ 12 12) (\lambda x. (+ 2 x) 6))$ β
- $(* 24 (\lambda x. (+ 2 x) 6))$ β
- $(* 24 (+ 2 6))$ β
- $(* 24 8)$ β
- 192



Full lazy

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda y. (* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) y) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ ((\lambda x. x) (* 3 4)) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ (* 3 4) (* 3 4)) (\lambda x. (+ 2 x) 6))$ β
- $(* (+ 12 12) (\lambda x. (+ 2 x) 6))$ β
- $(* 24 (\lambda x. (+ 2 x) 6))$ β
- $(* 24 (+ 2 6))$ β
- $(* 24 8)$ β
- 192



Strict

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (+ 2 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) 8$ β
- $(\lambda x. \lambda y. (* (+ x x) y) (* 3 4)) 8$ β
- $(\lambda x. \lambda y. (* (+ x x) y) 12) 8$ β
- $(\lambda y. (* (+ 12 12) y)) 8$ β
- $(* (+ 12 12) 8)$ β
- $(* 24 8)$ β
- 192



Eager

- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) (* 3 4))) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) ((\lambda x. x) 12)) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) 12) (\lambda x. (+ 2 x) 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) 12) (+ 2 6)$ β
- $(\lambda x. \lambda y. (* (+ x x) y) 12) 8$ β
- $(\lambda y. (* (+ 12 12) y)) 8$ β
- $(\lambda y. (* 24 y)) 8$ β
- $(* 24 8)$ β
- 192

Church-Rosser vlastnosť

(konzistentnosť λ -kalkulu)

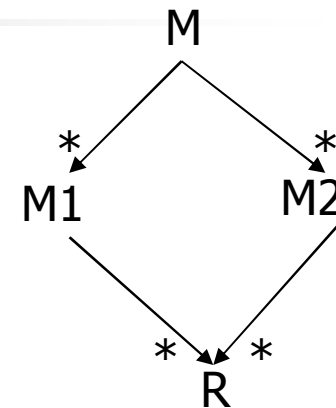
pre ľubovoľnú trojicu termov M, M_1, M_2 takých, že

$M \xrightarrow{\beta^*} M_1$ a $M \xrightarrow{\beta^*} M_2$
existuje R , že
 $M_1 \xrightarrow{\beta^*} R$ a $M_2 \xrightarrow{\beta^*} R$

Inak:

$$(\xrightarrow{\beta^*} \circ \xrightarrow{\beta^*}) \subseteq (\xrightarrow{\beta^*} \circ \xrightarrow{\beta^*})$$

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



Veta: β -redukcia spĺňa Church-Rosserovu vlastnosť

Dôkazy sú technicky náročné:

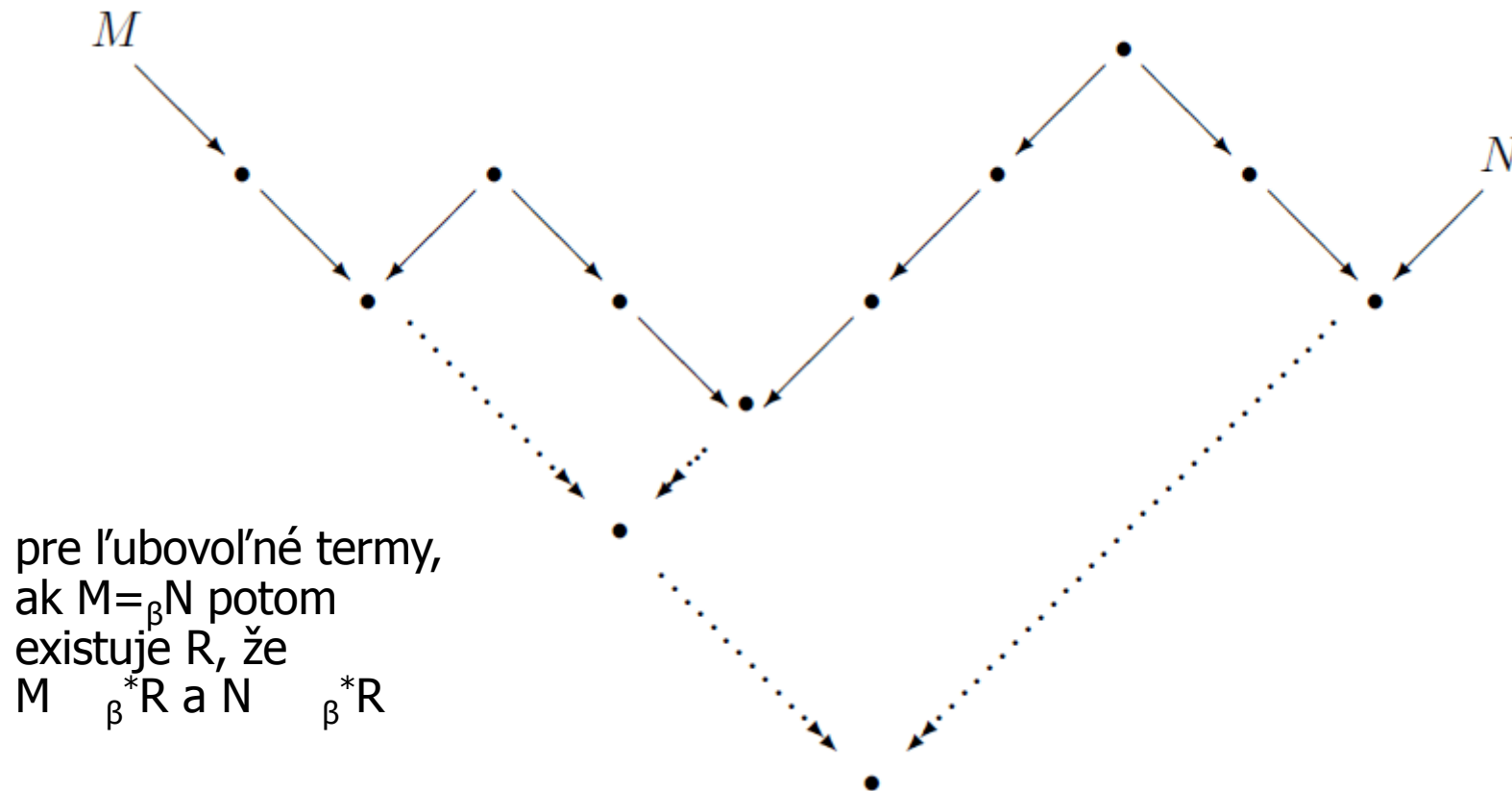
- 1936 Church, Rosser: Some properties of conversion
- 1981 Barendregt
- 1981 Löf, Tait

Dôsledok:

ak term má normálnu formu vzhľadom na β , potom je jednoznačne určená

β ekvivalencia

$\equiv_{\beta} \equiv (\quad_{\beta} \cup \quad_{\beta})^*$ Príklad: $KI\Omega =_{\beta} II$



Slabá Church-Rosser vlastnosť

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

$M \rightarrow M_1$ a $M \rightarrow M_2$
 existuje R , že
 $M_1 \rightarrow^* R$ a $M_2 \rightarrow^* R$

Inak:

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

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

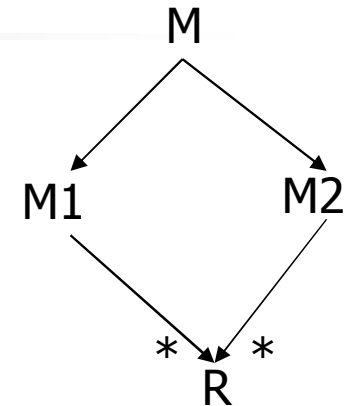
Veta: **Nech \rightarrow je noetherovská/silne normalizujúca/terminujúca relácia.**

- **má Church-Rosser vlastnosť (confluent) je ekvivalentné s**
- **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 konfluentosťou,
- M'', M''', M'''' , a t.d' (noetherovskosť relácie vyrobí spor).

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





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

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

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

•definujte n-1 (na rozmýšľanie)