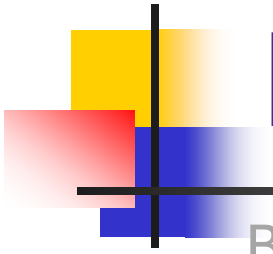


Lambda calculus 3

Bolo:

- vlastnosti β-redukcie
- λ-kalkul ako programovací jazyk (Churchove čísla, dátové štruktúry)
- rekurzia a operátor pevného bodu

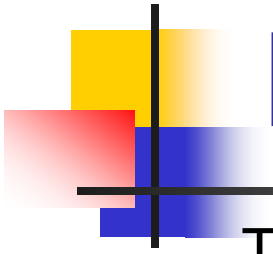
Dnes:

- viac o rekurzii – trochu technickejšie rozprávanie bez kódenia ☹
- de Bruijnove indexy (odstránenie mien premenných) – slajd 28..
- logika kombinátorov SKi (odstránenie mien premenných inak)

Cvičenie:

- de Bruijn indexy – cesta tam a späť a β-redukcia v teórii dB
- logika kombinátorov – cesta tam a späť a β-redukcia v teórii SKi

Rekurzia



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

Príklad:
FAC := λn.(if (= n 0) 1 (* n (FAC (- n 1))))

FAC := λn.if (n = 0) then 1 else (n * FAC (n - 1))
... trik: η-redukcia (λx.M x) = M, ak x nie je Free(M)
FAC := (λfac.(λn.(if (= n 0) 1 (* n (fac (- n 1))))) FAC)

H := λfac.(λn.(if (= n 0) 1 (* n (fac (- n 1)))))

hľadáme funkciu FAC, ktorá má túto vlastnosť:
FAC := (H FAC) f x = x
hľadaná funkcia FAC je *pevný bod* funkcie H

- Aký je typ Y ??

Pevný bod

Potrebujeme 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 = λf.(λx. (f(x x))) (λx. f(x x))
potom
(Y F) je pevný bod F, t.j. (Y F) = F (Y F).

Skúsme to (aspoň) overiť:
Y F = (λf.(λx. f(x x)) (λx. f(x x))) F $\rightarrow_\beta$ (λx. F(x x)) (λx. F(x x)) $\rightarrow_\beta$

- (λx′. F(x′ x′)) (λx. F(x x)) $\rightarrow_\beta$ F(x′ x′)[x′:λx. F(x x)] $\rightarrow_\beta$
- F(λx.F(x x) λx.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:=λfac.(λn.(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 (λfac.(λn.(if (= n 0) 1 (* n (fac (- n 1))))))
(λf.(λx. f(x x)) (λx. f(x x))) (λfac.(λn.(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 Y H=H (Y H)
= (H (Y H)) 1
= λfac.(λn.(if (= n 0) 1 (* n (fac (- n 1))))) (Y H) 1
= λn.(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 = λs.λn.if (= n 0) 0 (+ n (s (- n 1)))

(Y SUM) 2 =

- Y (λs.λn.if (= n 0) 0 (+ n (s (- n 1)))) 2
- (λs.λn.if (= n 0) 0 (+ n (s (- n 1)))) (Y SUM) 2
- (λn.if (= n 0) 0 (+ n ((Y SUM) (- n 1)))) 2
- if (= 2 0) 0 (+ 2 ((Y SUM) (- 2 1)))
- (+ 2 ((Y SUM) 1))
- (+ 2 ((λs.λn.if (= n 0) 0 (+ n (s (- n 1)))) (Y SUM) 1))
- (+ 2 ((λn.if (= n 0) 0 (+ n ((Y SUM) (- n 1)))) 1))
- (+ 2 ((if (= 1 0) 0 (+ n ((Y SUM) (- 1 1))))))
- (+ 2 (+ 1 ((Y SUM) 0)))
- (+ 2 (+ 1 ((λs.λn.if (= n 0) 0 (+ n (s (- n 1)))) (Y SUM) 0)))
- (+ 2 (+ 1 ((λn.if (= n 0) 0 (+ n ((Y SUM) (- n 1)))) 0)))
- (+ 2 (+ 1 ((if (= 0 0) 0 (+ 0 ((Y SUM) (- 0 1)))))))
- (+ 2 (+ 1 0)) = 3

Cvičenie

- (na zamyslenie) nájdite príklady funkcií
  s nekonečným počtom pevných bodov
  s práve jedným pevným bodom, fix-point : f x = x

- realizujte interpreter λkalkulu, pokračujte v kóde z minulého cvičenia tak, aby počítal hodnoty rekurzívnych funkcii

--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

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

```
-- plati Y f = f(Y f)
y =  LAMBDA "f"
       (APP (LAMBDA "x" (APP (ID "f") (APP (ID "x") (ID "x"))))
              (LAMBDA "x" (APP (ID "f") (APP (ID "x") (ID "x")))))
```

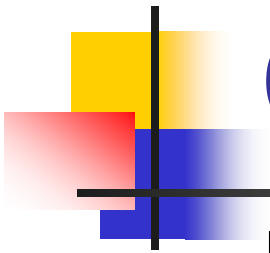
Vyhodnoťte (APP (APP y sucet) CN 4)

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.číslami a interpreter sa vám bude ťažšie ľadiť.



Operátor pevného bodu

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.

$(Y\ F) = F\ (Y\ F)$.

Viacnásobná rekurzia
(idea)

foo x y = … (goo x′ y′) … (foo x″ y″) …

goo x y = … (foo x′ y′) … (foo x″ y″) …

organizujem to vektorovo:

foogoo (x,y) = (

… snd $ (foogoo (x′,y′)) … fst $ (foogoo (x″, y″)) …

,

… fst $ (foogoo (x′, y′)) … snd $ (foogoo (x″, y″)) …

)

X = (x,y) = λ**z**.(foo (fst **z**, snd **z**), goo (fst **z**, snd **z**)) **X**

preto

X = Y (λ**z**.(foo (fst **z**, snd **z**), goo (fst **z**, snd **z**)) **)**

Viacnásobná rekurzia

Veta o pevnom bode:

Pre ľubovoľné $F_1, F_2, \ldots, F_n$ existujú $X_1, X_2, \ldots, X_n$, že

$X_1 = F_1\, X_1\, X_2 \ldots X_n$

$X_2 = F_2\, X_1\, X_2 \ldots X_n$

… …

$X_n = F_n\, X_1\, X_2 \ldots X_n.$

vektorovo:

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

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

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

p_i = i-ta projekcia vektora.

preto

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

Primitívna rekurzia

(ideovo)

Rózsa Peter, founding mother of recursion theory

Due to the effects of the Great Depression, many university graduates could not find work and Péter began private tutoring. At this time, she also began her graduate studies.

https://en.wikipedia.org/wiki/R%C3%B3zsa_P%C3%A9ter

Primitívne rekurzívna funkcia je:

- 0, resp. nulová funkcia $N^n \to N$,
- +1, resp. succ: $N \to N$,
- $p_i\, x_1\, x_2 \ldots x_n = x_i$ resp. projekcia p_i: $N^n \to N$, $p_i\, x_1\, x_2 \ldots x_n = x_i$
- f . g resp. kompozícia

  $f\, x_1\, x_2 \ldots x_n = g(h_1(x_1\, x_2 \ldots x_n) \ldots h_m(x_1\, x_2 \ldots x_n))$
- číselná rekurzia z n+1 na n:

  primitívna rekurzia $g : N^n \to N$, $h : N^{n+2} \to N$, potom $f : N^{n+1} \to N$

operátor primitívnej rekurzie

$f\, \mathbf{0}\quad x_1\, x_2 \ldots x_n = g(x_1\, x_2 \ldots x_n)$

$f\, \mathbf{(n+1)}\, x_1\, x_2 \ldots x_n = h(\, f(\mathbf{n}\, x_1\, x_2 \ldots x_n)\, \mathbf{n}\, x_1\, x_2 \ldots x_n)$

Primitívna rekurzia

Wilhelm Ackermann

Primitívne rekurzívne funkcia je totálna, resp. všade definovaná

- je akákoľvek číselná totálna funkcia (N->N) primitívne rekurzívna ?
- je akákoľvek číselná "haskellovská" totálna funkcia primitívne rekurzívna ? (je predpoklad n+1 -> n vážne obmedzujúci ?)
- je, Ackermannova funkcia, je jednoduchým príkladom funkcie, ktorá nie je primitívne rekurzívna (1935):

$$
\begin{aligned}
\mathrm{A}(0, n) &= n + 1 \\
\mathrm{A}(m + 1, 0) &= \mathrm{A}(m, 1) \\
\mathrm{A}(m + 1, n + 1) &= \mathrm{A}(m, A(m + 1, n))
\end{aligned}
$$

- Ackermannova funkcia rastie rýchlejšie ako akákoľvek primitívne rekurzívna

https://en.wikipedia.org/wiki/Primitive_recursive_function
https://en.wikipedia.org/wiki/Fast-growing_hierarchy

Viac než primitívna rekurzia

Primitívne rekurzívna funkcia je:

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

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

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

[Parciálne/Čiastočne] [vyčíslitelná/rekurzívna] (nemusí byť totálna):

nech $r : N^{n+1} \to N$ je primitívne rekurzívna funkcia

operátor μ-rekurzie

- μ-rekurzia definuje $f : N^{n+1} \to N$

  $f\ y \quad x_1\, x_2 \ldots x_n = \min_z.(r(z\, x_1\, x_2 \ldots x_n) == y)$

pohľad programátora:

```
f y x1 x2 ... xn =
        for (int z = 0; ; z++)
            if ( r(z x1 x2 ... xn) == y ) return z;
```

μ-rekurzia

operátor μ-rekurzie

[Parciálne/Čiastočne] [vyčíslitelná/rekurzívna] (nemusí byť totálna):

nech r : $N^{n+1} \rightarrow N$ je primitívne rekurzívna funkcia

- μ-rekurzia definuje f : $N^{n+1} \rightarrow N$

  f y $x_1\ x_2 \ldots x_n = \min_z.(r(z\ x_1\ x_2 \ldots x_n) == y)$

pohľad informatika:

Nech r(z x_1 x_2)

- je funkcia, ktorá ku kódu TM x_1, kódu vstupnej pásky x_2 kódu povie, v ktorom stave stojí TM po z krokoch
- chce to prax a chvíľku by to trvalo, ale zistili by sme, že to je primitívne rekurzívna funkcia...
- --- r ... povie, či TM stoj stojí v koncovom stave, alias, akceptuje slovo x_2

Operátor μ-rekurzia nás privedie k čiastočne rekurzívnej funkcii, ktorá

- vie zistiť, po koľkých krokoch TM zastaví, ak zastaví
- ak nie, nevie nič

Asi tušíte, že toto z princípu veci nemôže byť totálne funkcia, ani primitívne rekurzívna

- sme v inej galaxii…

λ-vypočítateľná funkcia

(technické – na dlhé letné večery)

Parciálna funkcia $f : N^n \to N$ je **λ-vypočítateľná**, ak existuje λ-term F taký, že

F $\underline{x_1}\ \underline{x_2} \ldots \underline{x_n}$ sa zredukuje na $\underline{f\ x_1\ x_2 \ldots x_n}$, ak n-tica $x_1\ x_2 \ldots x_n$ patrí do def.oboru f

F $\underline{x_1}\ \underline{x_2} \ldots \underline{x_n}$ nemá normálnu, ak n-tica $x_1\ x_2 \ldots x_n$ nepatrí do def.oboru f

Veta: Každá parciálne vyčíslitelná funkcia je λ-vypočítateľná.

Dôkaz:

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

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

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

  $F = \mathbf{Y}\ (\lambda f.\lambda y.\lambda x_1.\lambda x_2 \ldots \lambda x_n .$ (if (isZero y) $G(x_1\ x_2 \ldots x_n)$ then

  else $H(f((\text{pred } y)\ x_1\ x_2 \ldots x_n)\ (\text{pred } y)\ x_1\ x_2 \ldots x_n)))$
- μ-rekurzia $r : N^{n+1} \to N$

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

Veta: Každá λ-vypočítateľná je parcialne vyčíslitelná funkcia.

Weak head normal form

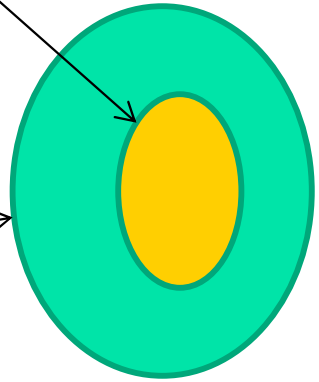
(slabo hlavová normálna forma)

Head normal form (h.n.f)

- $(\lambda x_1.\ \lambda x_2.\ \ldots\ .\ \lambda x_k.\ v)\ M_1\ M_2 \ldots\ M_n$ $(k, n \in N)$
- v je premenná (resp. konštanta),
- pre ľubovoľné $r \leq n$, $(\ldots((v\ M_1)\ M_2) \ldots\ M_r)$ 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)

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

Najznámejšie stratégie

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

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

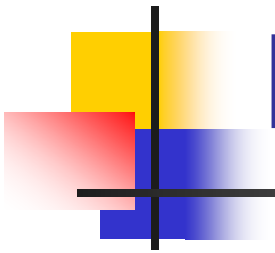
  $\underline{(\lambda x.\ \lambda y.(x\ y))}\ (\lambda z.z) \rightarrow_\beta \lambda y.(\underline{(\lambda z.z)}\ y) \rightarrow_\beta \lambda y.y$ n.f.

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

  normalizuje úplne...

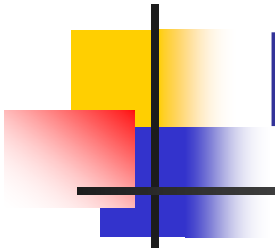
- eager - argumenty najprv (call by value/data driven/strict)

  nenormalizuje...



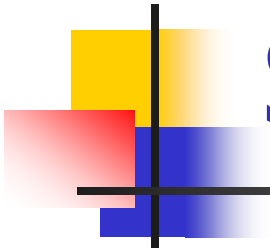
Lazy

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



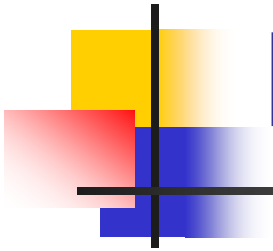
Full lazy

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



Strict

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

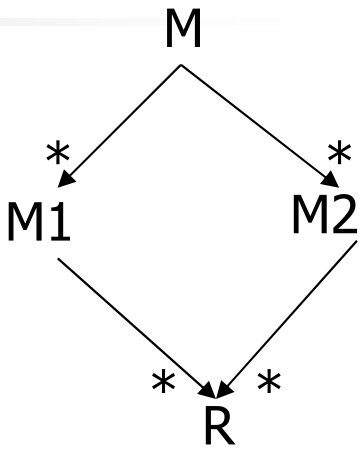


Eager

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

Church-Rosser vlastnosť

(konzistentnosť λ-kaklulu – *Janko/Marienka vlastnosť*)



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 $M1 \leftarrow_\beta^* M \rightarrow_\beta^* M2$, potom existuje R, že $M1 \rightarrow_\beta^* R \leftarrow_\beta^* M2$

Veta: β-redukcia spĺňa Church-Rosserovu vlastnosť

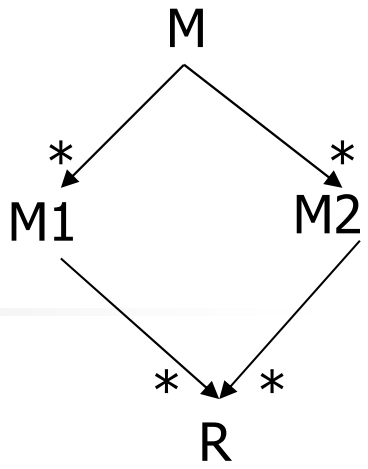
Dôkazy sú technicky náročné:

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

Dôsledok:

ak term má normálnu formu vzhľadom na $\rightarrow_\beta$, potom je jednoznačne určená

Vysvetlenie

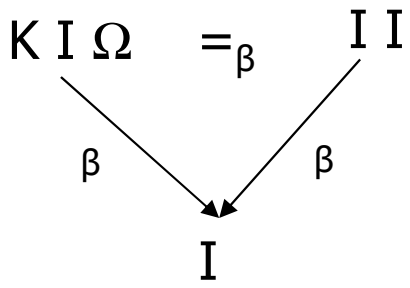


- $(\leftarrow_\beta^* \circ \rightarrow_\beta^*) \subseteq (\rightarrow_\beta^* \circ \leftarrow_\beta^*)$
- $x\ (\leftarrow_\beta^* \circ \rightarrow_\beta^*)\ y \Rightarrow x\ (\rightarrow_\beta^* \circ \leftarrow_\beta^*)\ y$
- $\forall\ m, \exists\ r:$

  $x \leftarrow_\beta^* m \rightarrow_\beta^* y \Rightarrow x \rightarrow_\beta^* r \leftarrow_\beta^* y$

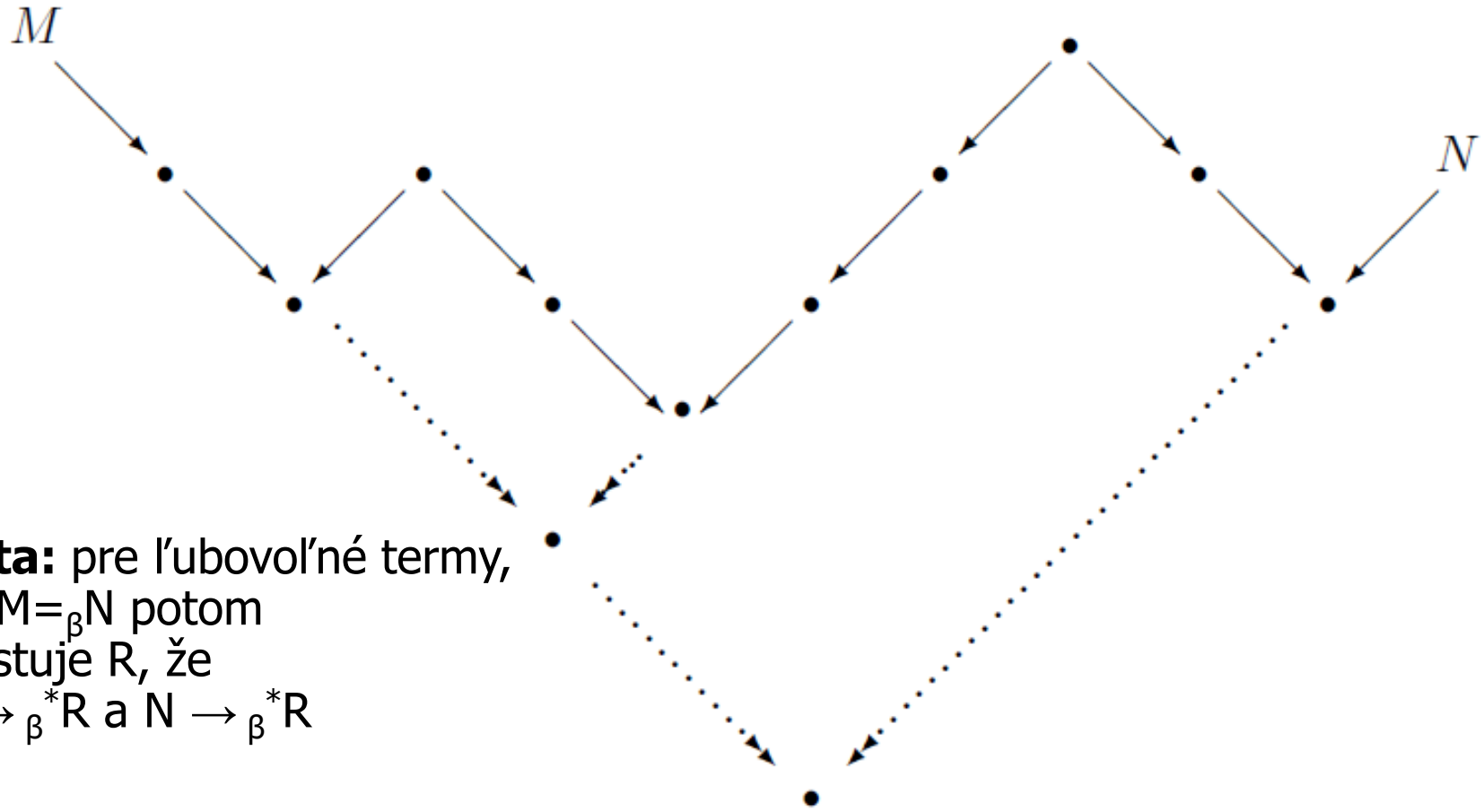
- $\forall\ m\ \exists\ r:$

  $x \leftarrow_\beta^* m \ \wedge\ m \rightarrow_\beta^* y \Rightarrow x \rightarrow_\beta^* r \ \wedge\ r \leftarrow_\beta^* y$



β ekvivalenica

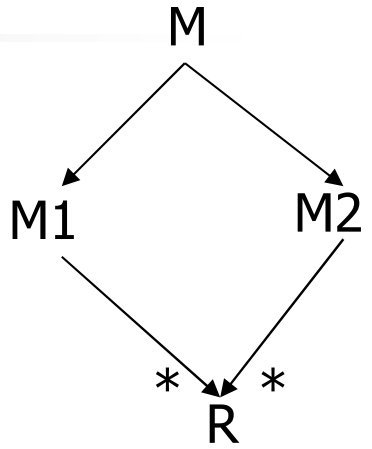
- $=_\beta$ definujeme ako $(\rightarrow_\beta \cup \leftarrow_\beta)^*$ Príklad: $KI\Omega =_\beta II$



Veta: pre ľubovoľné termy,
ak $M=_\beta N$ potom
existuje R, že
$M \rightarrow_\beta^* R$ a $N \rightarrow_\beta^* R$

Slabá Church-Rosser vlastnosť

(*slabá Janko/Marienka 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:

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

teda ak M1←M →M2, potom existuje R, že M1 $\rightarrow^{*}$R$\leftarrow^{*}$ M2

- $\forall\, m\ \exists\, r: x \leftarrow_{\beta} m \ \wedge\ m \rightarrow_{\beta} y \ \Rightarrow\ x \rightarrow_{\beta}^{*} r \ \wedge\ r \leftarrow_{\beta}^{*} y$

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

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

Dôkaz: CR => SCR, to je jasné...
preto zostáva SCR =>?? CR

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

Slabá Church-Rosser vlastnosť

Veta: **Nech → je noetherovská/silne normalizujúca/terminujúca relácia**

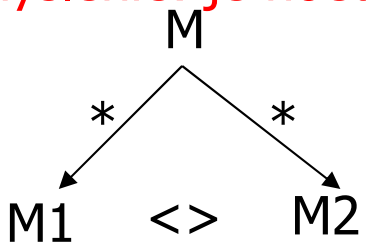
- **→ má Church-Rosser vlastnosť (confluent) je ekvivalentné s**
- **→ má slabú Church-Rosser vlastnosť (local confluent)**

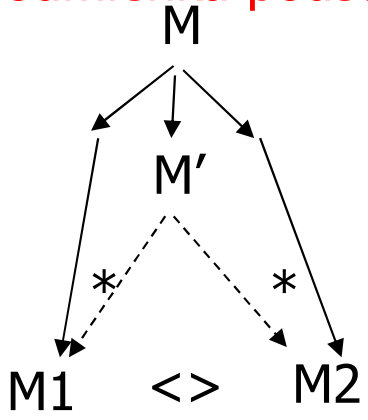
Dôkaz sporom:

NOE & SCR implikuje CR, ukážeme **spor**: NOE & SCR & ¬CR:

- (¬CR): Nech M má dve normálne formy, M1<>M2, t.j. $M \rightarrow^* M_1$ a $M \rightarrow^* M_2$.
- M nie je v normálnej forme (ak by bolo, M=M1=M2 a pritom M1<>M2),
- potom existuje M', že M → M',
- M' má tiež dve normálne formy, ak by nie, spor s lokálnou konfluentosťou,
- M'', M''', M'''', a t.ď (noetherovskosť relácie vyrobí spor).

Zamyslenie: je noetherovská podmienka podstatná, neplatí veta ajbez nej ?



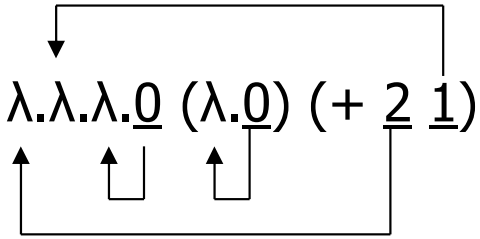


de Bruijn index

Nicolaas Govert de Bruijn

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

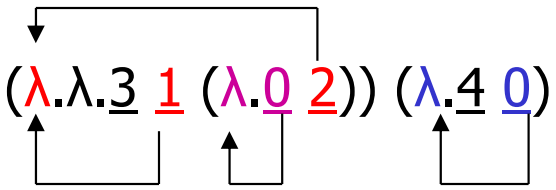
- λx.(+ x 1)
  - λ.(+ 0 1)
- λx.λy.λf.f ((λx.x) (+ x y))
  - λ.λ.λ.0 ((λ.0) (+ 2 1))



de Bruijn index: neformálne:
cez koľko λ treba preskákať hore, aby sme našli λ-príslušnej/danej premennej

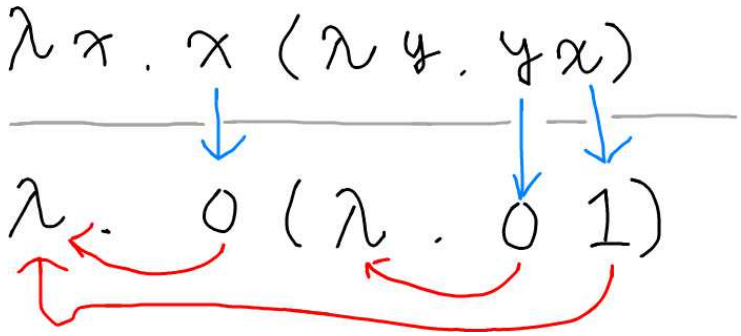
- Dôsledok 1: α-konverzia tu neexistuje, lebo premenné nemajú/stratili mená
- Dôsledok 2: rôzne premenné môžu mať rovnaký index v rôznych kontextoch
- Dôsledok 3: voľné premenné majú index >= hľbku λ-vnorení
- Dôsledok 4: voľné premenné musia mať rôzne indexy

- (λx.λy.((z x) (λu.(u x)))) (λx.(w x))
  - (λ.λ.((3 1) (λ.(0 2))) (λ.(4 0))



1972 Lambda Calculus Notation with Nameless Dummies

https://en.wikipedia.org/wiki/De_Bruijn_index



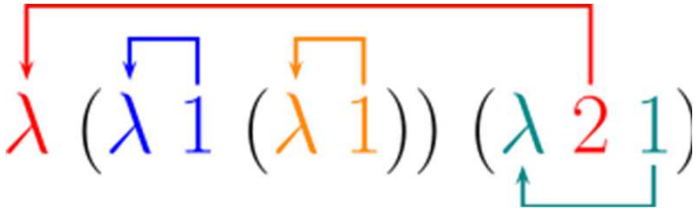
de Bruijn index
(príklady)

- λx.λy.x (K-kombinátor):
  λ λ 1

- λx.λy.λz. ((x z) (y z)) (S-kombinátor):
  λ λ λ ((2 0) (1 0))

- λz.((λy.y (λx.x)) (λx.(z x)))

  λ ((λ 0 (λ 0)) (λ (1 0)))



pekná grafika, ale my máme indexy od 0, preto decr 1

Syntax de Bruijnovej notácie:

$L_{dB} ::= n \mid (L_{dB}\ L_{dB}) \mid \lambda\ L_{dB}$

```
data LExpDB = LAMBDADB LExpDB |
              IDDB Int |
              APPDB LExpDB LExpDB deriving(Eq)
```

https://en.wikipedia.org/wiki/De_Bruijn_index

β-redukcia s de Bruijn-indexami (pointa)

Príklady:

- K=λx.λy.x
  - λ.λ.1
- S=λx.λy.λz.((x z) (y z))
  - λ.λ.λ.((2 0) (1 0))

- λx.(+ x 1) 5 — hypotéza, ako by to mohlo fungovať
  - λ.(+ 0 1) 5 = (+ 5 1) — β-redukcia (λ.M) N = M[0:N] ??? ale nefunguje... ☹
- K a b =(λx.λy.x) a b
  - (λ.λ.1 a) b = λ.a b = a — skúsme intuitívne

- (λx.λy.((z x) (λu.(u x)))) (λx.(w x))
  - (λ.λ.((3 1) (λ.(0 2)))) (λ.(4 0))
  - (λ.λ.((3 □) (λ.(0 □)))) (λ.(4 0)) označíme si miesta □, kam sa substituuje
  - (λ.(2 (λ.5 0)) (λ.(0 (λ.6 0)))) nahraďme, ale pozor na voľné premenné

  (λy.z (λx.w x) (λu.u (λx.w′ x))) tie sa posunú/*shiftujú* s indexom hore

β-redukcia s de Bruijn-indexami (príklad)

Keďže nemáme mená premenných, substituujeme len za indexy

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

```
type SubstDB = [LExpDB]    -- [t0,t1,..] znamená {0/t0,1/t1,..}
```

Aplikácia substitúcie:

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

t^1 – pripočítaj 1 k voľným premenným

Beta redukcia:

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

Príklad: $(K\ a)\ b = (\lambda.\lambda.\underline{1}\ a)\ b = ((\lambda.\underline{1})\ [a,\underline{0},\underline{1},\underline{2},\ldots])b$

$= (\lambda.(\underline{1}\ [\underline{0}, a, \underline{1}, \underline{2},\ldots]))\ b = \lambda.a\ b = \ldots$ zrychlene = a

…ak a, b sú konštanty neobsahujúce premenné

β-redukcia s de Bruijn-indexami

(príklad vhodný do domácej úlohy)

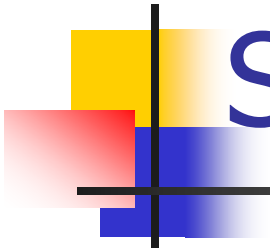
- $(\lambda M)\ [t_0, t_1, \ldots, t_n] = (\lambda\ M[\underline{0}, t_0\text{^}1, t_1\text{^}1, \ldots, t_n\text{^}1])$

  t^1 – pripočítaj 1 k voľným premenným

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

Príklad z predpredošlého slajdu:

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

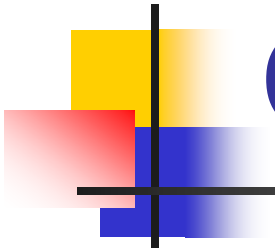


SKK

- K=λx.λy.x
  - λ.λ.1
- S=λx.λy.λz.x z (y z)
  - λ.λ.λ.2 0 (1 0)

Ďalší testovací príklad

- S K K = ((λ.λ.λ.2 0 (1 0)) λ.λ.1) λ.λ.1 =
  - (λ.λ.2 0 (1 0) [λ.λ.1, 0,1,2,…]) λ.λ.1 =
  - (λ.λ.(2 0 (1 0)) [0,1,λ.λ.1, 0,1,2,…]) λ.λ.1 =
  - (λ.λ.((λ.λ.1) 0 (1 0))) λ.λ.1 =
  - (λ.((λ.λ.1) 0 (1 0))) [λ.λ.1,0,1,2,…] =
  - λ.(((λ.λ.1) 0 (1 0)) [0,λ.λ.1,0,1,2,…]) =
  - λ.((λ.λ.1) 0 (λ.λ.1 0)) =
  - λ.0 = I



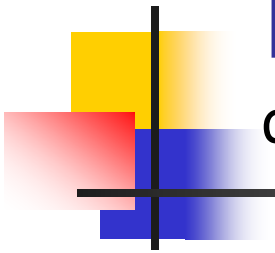
Cvičenie

Prepíšte do de Bruijn notácie

- λx.λy.y (λz.z x) x
- λx.(λx.x x) (λy.y (λz.x))
- (λx. + x ((λy.y) (- x (λz.3)(λy.y y)))

Definujte funkciu na prevod do a z de Bruijn notácie

Implementujte aplikáciu substitúcie a β-redukciu v de Bruijn kalkule s pomocnými funkciami.



Domáca úloha, časť 1

de Bruijnova notácia

V `module DB` definujte:

- funkciu `toDB :: LExp -> LExpDB`, ...ale vstupný λ-term môže obsahovať voľné premenné
- funkciu `fromDB :: LExpDB -> LExp`, ...pri tom si treba vedieť vymýšľať mená premenných
- ak substitúcia v kalkule de Bruijna je definovaná `type SubstDB = [LExpDB]`, tak definujte:
  - aplikáciu substitúcie na λ-term, teda `subst :: LExpDB -> SubstDB -> LExpDB`
  - potom `beta :: LExpDB -> LExpDB -> LExpDB`
  - a potom `oneStep :: LExpDB -> LExpDB` a normálnu formu `nf :: LExpDB -> LExpDB`

  dostávate ako veľkonočný výslužku v priloženom súbore `DB.hs`.

Príklady nájdete v priloženom súbore `DB.hs` a `Terms.hs`