



# Úvod do $\lambda$ -kalkulu

---

Peter Borovanský

I-18

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

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair is located to the left of the title.

# Lambda calculus

---

Štruktúra prednášok:

- pár slov z histórie (niečo už bolo 1.týždeň)
- úvod do syntaxe, netypovaný  $\lambda$ -kalkul (gramatika + konvencie)
- sémantika (redukčné pravidlá)
- programovací jazyk nad  $\lambda$ -kalkulom

domáca úloha: malý interpreter  $\lambda$ -kalkulu, a veci s tým súvisiace

Určite vystačíte s [https://en.wikipedia.org/wiki/Lambda\\_calculus](https://en.wikipedia.org/wiki/Lambda_calculus)

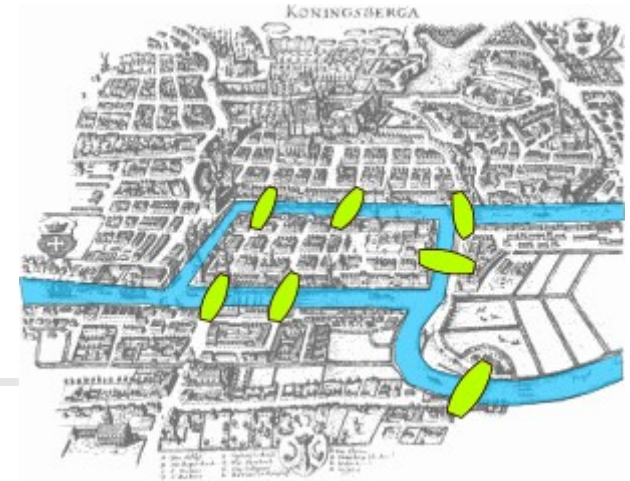
resp. [http://dev.stephendiehl.com/fun/003\\_lambda\\_calculus.html](http://dev.stephendiehl.com/fun/003_lambda_calculus.html)

dnes nebude (ale raz to príde):

- rekurzia (operátor pevného bodu)
- vlastnosti teórie
- de Bruijn-ova notácia (?)
- typovaný  $\lambda$ -kalkul

domáca úloha: typovač pre  $\lambda$ -kalkulu, ...

# Trochu z histórie



- 1900, David Hilbert, formalizácia matematiky

formuloval 23 vtedy neriešiteľných problémov v matematike

[https://en.wikipedia.org/wiki/Hilbert%27s\\_problems#Table\\_of\\_problems](https://en.wikipedia.org/wiki/Hilbert%27s_problems#Table_of_problems)

- 10. problém: algoritmus na riešenie polynomiálnych Diofantických rovníc
  - existencia rozhodovacieho algoritmu, či existuje celočíselné riešenie celočíselnej DR Matiyasevich, 1970

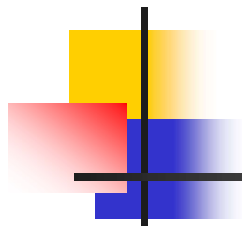
**Entscheidung** der Losbarkeit einer diophantischen Gleichung.

**Rozhodnutel'nost'**

**Decidability**

Diofantov epitaf:

Diofantova mladosť trvala  $\frac{1}{6}$  jeho života. Fúzy mu narástly o ďalšiu  $\frac{1}{12}$  jeho života. O nasledujúcu  $\frac{1}{7}$  života sa Diofantos oženil. Po piatich rokoch sa mu narodil syn. Syn žil presne  $\frac{1}{2}$  dĺžky života svojho otca. Diofantos zomrel 4 roky po smrti svojho syna.  
Koľko rokov sa dožil ?



Diofantova mladost' trvala  $\frac{1}{6}$  jeho života. Fúzy mu narástly o ďalšiu  $\frac{1}{12}$  jeho života. O nasledujúcu  $\frac{1}{7}$  života sa Diofantos oženil. Po piatich rokoch sa mu narodil syn. Syn žil presne  $\frac{1}{2}$  dĺžky života svojho otca. Diofantos zomrel 4 roky po smrti svojho syna. Koľko rokov sa dožil ?

$$\frac{1}{6}D + \frac{1}{12}D + \frac{1}{7}D + 5 + \frac{1}{2}D + 4 = D$$

<https://www.wolframalpha.com/input/?i=1%2F6D%2B1%2F12D%2B1%2F7D%2B5%2B1%2F2D%2B4%3D%3DD>

$\frac{1}{6}D + \frac{1}{12}D + \frac{1}{7}D + 5 + \frac{1}{2}D + 4 = D$

Extended Keyboard Upload

Input:

$$\frac{1}{6}D + \frac{1}{12}D + \frac{1}{7}D + 5 + \frac{1}{2}D + 4 = D$$

Result:

$$\frac{25D}{28} + 9 = D$$

Alternate forms:

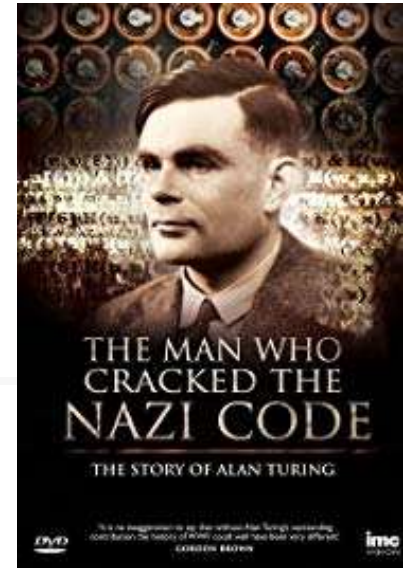
$$9 - \frac{3D}{28} = 0$$
$$\frac{1}{28}(25D + 252) = D$$

Number line:

Solution:

$$D = 84$$

# Trochu z histórie



- 1936, Alan Turing, On Computable Numbers with an Application to **Entscheidungsproblems**

formuloval pojem počítania a čísla, ktoré vieme 'mechanicky' vypočítať

prišiel k záveru, že 'Turingov stroj' nevie vypočítať ľubovoľné reálne číslo lebo Cantorov dôkaz. Turing sa zaujímal o čísla, ktoré by sme vedeli vypočítať

- 1936, Alonzo Church: A note on the **Entscheidungsproblems**
- 1936, Alonzo Church: An Unsolvable Problem of Elementary Number Theory
- základ lambda calculus (effective calculability)
  - teoretický základ FP
  - kalkul funkcií je vlastne: abstrakcia, aplikácia, kompozícia
- Princeton: A.Church, A.Turing, J. von Neumann, K.Gödel - skúmajú formálne modely výpočtov



# Trochu z histórie FP

---

- éra: WWII, prvý von Neumanovský počítač: Mark I (IBM), balistické tabuľky
- 1958, Haskell B. Curry, logika kombinátorov
  - alternatívny pohľad na funkcie, menej známy a populárny
  - „premenné vôbec nepotrebujeme“
- 1958, jazyk LISP, John McCarthy, MIT
  - implementácia lambda kalkulu na „von Neumanovskom HW“
  - otec AI, jazyk Lisp bol prvý, ktorý mal/potreboval Garbage Collector
- 1960, SECD (**S**tack-**E**nvironment-**C**ontrol-**D**ump) Machine, Landin
  - predchodca p-code, rôznych stack-orientovaných bajt-kódov, virtuálnych mašín.
  - SECD použili pri implementácii
    - Algol 60, PL/1 – predchodcu Pascalu
    - LISP – prvého funkcionálneho jazyka založenom na  $\lambda$ -kalkule

# Od Haskellu k $\lambda$ -kalkulu

rýchly nadhl'ad

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

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

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

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

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

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



# Syntax

---

Celý program v jazyku pozostáva z jedného  $\lambda$ -termu.

$L$  je  $\lambda$ -term:

- $x$  je premenná (spočítateľná množina premenných)

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

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

**Cvičenie (na zamyslenie):** syntax jazyka je veľmi jednoduchá, neporovnateľná napr. s Javou, C++. Zamyslite sa nad tým, či existuje viac programov v Jave alebo  $\lambda$ -termov.





# Príklady $\lambda$ -termov

---

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

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

- syntax jazyka je len o tom, čo patrí do jazyka, a čo nepatrí
- čo to znamená, čo je výsledkom, ..., to je sémantika jazyka
  - operačná – popisuje dej výpočtu – „while ... cyklí až kým“
    - je to väčša obsahom Reference manuálu k jazyku
  - denotačná/matematická – popisuje, čo program počíta iným formalizmom, napr. ako limita čiastočne definovaných funkcií, alebo predikátov
    - je to zväčša tvrdá matika, tzv. Scott-Strachey semantics



# Syntaktické konvencie

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

BTW: to súvisí s tým, že v Haskellí je typový operátor `->` vpravo asociatívny, teda `foo :: Int -> (Int -> Int)`

BTW: to súvisí s tým, že v Haskellí je neviditeľný operátor aplikácie vľavo asociatívny, teda `((foo a) b) c`



# Dôležité príklady termov

O ich dôležitosti sa dozvieme neskôr, keď budeme vedieť, ako sa v tejto teórii počíta. Ale sú to termy, ktoré sa oplatí mať napísané na t'aháku, lebo budú často účinkovať na scéne

- $K = \lambda xy.x = \lambda x.\lambda y.x$  (funkcia s dvomi argumentami, výsledkom je 1.)  
inak tú sme už 2x videli ( $\text{true } x \ y = x$ ), a jej dôležitosť nedocenili...
- $I = \lambda x.x$  (identita)
- $S = \lambda xyz.(xz)(yz) = \lambda xyz.((x \ z) (y \ z))$  ( $S \ x \ y \ z = (x \ z) (y \ z)$ )
- $\omega = \lambda x.xx = \lambda x (x \ x)$  (veľmi záľudná, kde sa zjaví  $\omega$ , budú problémy)
- $\Omega = \omega\omega = (\lambda x.x \ x)(\lambda x.x \ x) = ( (\lambda x (x \ x)) (\lambda x (x \ x)) )$  (veľké problémy)
- $\omega_3 = \lambda x.xxx = \lambda x.(xx)x = (\lambda x ((x \ x) \ x))$  (mega-problémy)

Problém: viete si predstaviť funkciu  $f$ , že by  $(f \ f)$  malo zmysel, malo typ ?



# Výpočet

- na to, aby sme vedeli počítať v tejto teórii, potrebujeme definovať redukčné pravidlo(á), ktoré simuluje krok výpočtu,
  - redukčné pravidlo je analógia kroku Turingovho stroja alebo DFA
- redukčné pravidlo je založené na pojme **substitúcie**, ktorú, ak pochopíme len **intuitívne**, dostaneme **intuitívne zlé výsledky**,
  - substitúcia, hoc to tak možno znie, nie je textový replacement
- preto sa vybudovaniu substitúcie treba chvíľku venovať s pomocnými pojmami, ako je voľná/viazaná premenná, ...
  - čo je trochu úvodná nuda, ale treba si cez to prejsť
- musíme sa presvedčiť, že redukčné pravidlo má rozumné vlastnosti, zamyslíme sa nad tým, čo je rozumné...
  - asi by bolo fajn, ak by ak ho používa *Janko* dávalo rovnaké výsledky ako keď *Marienka*
- výpočet je opakované aplikovanie redukčného pravidla, a to kdekoľvek to v terme ide. Keď to už nejde, máme výsledok (tzv. normálna forma)
  - a môže sa to zacykliť ? Kedy ? Nevedeli to vymyslieť tak, aby to vždy skončilo ?
- môže sa stať, že rôznym aplikovaním red.pravidla prideme k rôznym výsledkom, resp. rôznym výpočtom ???



# Voľná premenná, podterm

- voľná premenná  $\lambda$ -termu

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

- viazaná premenná  $\lambda$ -termu

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

viazaná premenná a voľná premenná:  $\lambda x.xy$  –  $y$  je voľná,  $x$  je viazaná

Je to dichotómia ?  $\text{Bound}(M) \cap \text{Free}(M) = ??? \emptyset$

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

Asi...keby sme sa bavili o konkrétnom výskyte premennej, tak je to *bud'/alebo*

- podtermy  $\lambda$ -termu

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



# Príklady

---

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



# Substitúcia

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

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

Problém: 'rovnaké' vstupy po aplikovaní rovnakej operácie dali 'rôzne' výsledky – výrazy  $(\lambda x.zx)$  a  $(\lambda y.zy)$  *intuitívne* predstavujú rovnaké funkcie a po substitúcii  $[z:y]$  sú výsledky  $\lambda x.yx$  a  $\lambda y.yy$  *intuitívne* rôzne  
*Intuitívne musíme raz formalizovať*

- substitúcia  $N[x:M]$

$$x[x:M] = M \quad \text{-- } x \text{ za } x$$

$$y[x:M] = y \quad \text{-- } x \text{ za } y$$

$$(A B)[x:M] = (A[x:M] B[x:M]) \quad \text{-- aplikácia}$$

$$(\lambda x.B)[x:M] = (\lambda x.B) \quad \text{-- viazaná } \lambda$$

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

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

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

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

# Príklady

Naivne a zle:

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

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

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

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

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

inak

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

**$x \notin \text{Free}(B) \mid \mid y \notin \text{Free}(M)$**

- $(\lambda y.zy)[z:y] =$ 
  - $(\lambda w.(zy)[y:w])[z:y] =$
  - $(\lambda w.(z[y:w]y[y:w]))[z:y] =$
  - $(\lambda w.(zw))[z:y] =$
  - $\lambda w.(zw)[z:y] =$
  - $\lambda w.(z[z:y]w[z:y]) =$
  - $\lambda w.(yw)$

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

– treba si vymyslieť novú premennú





# Vlastnosti substitúcie 1

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

Dôkaz „*mávaním rukami*“: jasné, ak sa  $x$  v  $M$  nevyskytuje, substitúcia na neho nemá...

Dôkaz indukciou (takmer každý bude M.I. a niečo sporom)...

- $M = x$ , neplatí predpoklad...
- $M = y$ , potom  $y[x:N] = y = M$
- $M = (A \ B)$ , potom z  $x$  nie je voľná v  $(A \ B)$  vyplýva, že  $x$  nie je voľná v  $A$  aj v  $B$ ,  $(A \ B)[x:N] = (A[x:N] \ B[x:N]) = \text{ind.predp. } (A \ B) = M$
- $M = (\lambda x. B)$ , potom  $(\lambda x. B)[x:N] = (\lambda x. B) = M$
- $M = (\lambda y. B)$ , potom  $x$  nie je voľná v  $B$ ,  $(\lambda y. B)[x:N] = (\lambda x. B[x:N]) = M$

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

**Dôsledok:** Uzavretý výraz sa aplikáciou substitúcie nezmení.



# Malá odbočka ku Map-kvízu

- $\text{map } f . \text{reverse} = \text{reverse} . \text{map } f$
- $\text{map } f (\text{reverse } xs) = \text{reverse } (\text{map } f xs)$

Indukcia:

- ak  $xs = []$ , tak L.S. = [] aj P.S. = []
- $xs = y:ys$ , tak dokážte  $\text{map } f (\text{reverse } (y:ys)) \stackrel{?}{=} \text{reverse } (\text{map } f (y:ys))$
- L.S. =  $\text{map } f (\text{reverse } ys ++ [y])$
- =  $\text{map } f (\text{reverse } ys) ++ \text{map } f [y]$
- =  $\text{map } f (\text{reverse } ys) ++ [f y]$
- =  $\text{reverse } (\text{map } f ys) ++ [f y]$
- =  $\text{reverse } (\text{map } f ys) ++ [f y]$
- =  $\text{reverse } ((f y):\text{map } f ys)$
- =  $\text{reverse } (\text{map } f (y:ys))$
- = P.S.
- čbtd.

$\text{reverse } [] = []$

$\text{reverse } (y:ys) = \text{reverse } ys ++ [y]$



# Vlastnosti substitúcie 2

---

## **Lemma:**

Predpoklady:

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

potom platí:

- 1)  $M[x:N] [y:L] = M[y:L][x:N[y:L]]$     -- superpozícia substitúcií  
-- výmena poradia aplikovania substitúcií
- 2)  $M[y:N] [y:L] = M[y:N[y:L]]$     -- skladanie substitúcií



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

superpozícia substitúcií

Indukciou vzhľadom na M:

- **M je premenná**

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

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

- **z = x**, L.S. =  $(\lambda x.Q)[x:N] [y:L] = (\lambda x.Q)[y:L] = \lambda x.(Q[y:L])$ ,  
P.S. =  $(\lambda x.Q) [y:L][x:N[y:L]] = (\lambda x.Q[y:L])[x:N[y:L]] = \lambda x.(Q[y:L]) = \text{L.S.}$
- **z = y**, obe strany sú  $(\lambda y.(Q[x:N]))$ , lebo  $y$  je viazaná v  $M$ , preto nie je voľná v  $(N \ L)$ , takže ani  $N$
- **z je rôzne od x, y**, potom, podľa predpokladu,  $z$  nie je voľná v  $N$  ani  $L$   
 $(\lambda z.Q)[x:N] [y:L] = \lambda z.(Q[x:N]) [y:L] =$   
 $\lambda z.(Q[x:N][y:L]) = \dots$  podľa indukčného predpokladu  
 $\lambda z.(Q[y:L][x:N[y:L]]) = (\lambda z.Q)[y:L][x:N[y:L]].$

Predpoklady:

$x \neq y$

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

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

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


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

skladanie substitúcií

Predpoklady:

$x \neq y$

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

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

Domáca úloha:

podobne dokážte tvrdenie 2) predchádzajúcej lemmy.

Domáca úloha:

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

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

Cieľom týchto úloh je, aby ste sa zamysleli nad pojmom substitúcie a jej vlastnosťami, nie len ju naprogramovali. Bez pochopenia substitúcie vám nebude správne fungovať interpreter  $\lambda$ -calculu.

# Hra na kontrášov

$$M[x:N] [y:L] \neq M[x:N[y:L]]$$

skladanie substitúcií

Čo ak neplatí a)  $x == y$

$$M[x:N] [x:L] \neq M[x:N[x:L]]$$

... kontrapríklad

Predpoklady:

a)  $x \neq y$

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

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

Čo ak neplatí b)  $x \in \text{Free}(L)$

... kontrapříklad

Čo ak neplatí c)  $\exists v: v \in \text{Bound}(M) \wedge v \in \text{Free}((N L))$

... kontrapříklad



# $\alpha$ -konverzia

---

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

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

$$\lambda x.x =_{\alpha} \lambda y.y$$

$$\lambda x.(x\ y) \neq_{\alpha} \lambda y.(y\ y) \quad \text{ale} \quad \lambda x.(x\ y) =_{\alpha} \lambda z.(z\ y)$$



# β-redukcia

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

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

Príklad:

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





# β-redukcia

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

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

Príklad:

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



# Vlastnosti $\beta$ -redukcie

- $\omega = \lambda x. xx = \lambda x. (x x)$
- $\Omega = \omega \omega$
- $\omega_3 = \lambda x. ((x x) x)$
  
- Explózia času: existuje nekonečná sekvencia, nekončiaci výpočet
  - $\Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \Omega \rightarrow_{\beta} \dots$
- Explózia pamäte: existuje ľubovoľne puchnúca sekvencia
  - $\omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \rightarrow_{\beta} \omega_3 \omega_3 \omega_3 \omega_3$
- nejednoznačný výsledok pre dva rôzne výpočty ( $K = \lambda xy. x$ )
  - $KI\Omega \rightarrow_{\beta} I$  ale aj
  - $KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} KI\Omega \rightarrow_{\beta} \dots$

**Cvičenie:** overte si tieto tvrdenia,

Pokúste sa nájsť  $\lambda$  term, ktorý vedie k rôznym výsledkom 😊

# Cvičenie

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

$$\Omega = \omega \omega$$

$$\omega_3 = \lambda x. ((x x) x)$$

$$K = \lambda xy. x$$

$$I = \lambda x. x$$

$$S = \lambda xyz. xz(yz)$$

- $\Omega = \omega \omega = ((\lambda x. xx) (\lambda x. xx)) \rightarrow_{\alpha} (\lambda z. zz) (\lambda x. xx) \rightarrow_{\beta} ((\lambda x. xx) (\lambda x. xx)) \rightarrow_{\beta} \dots$
  
- $\omega_3 \omega_3 \rightarrow_{\beta} (\lambda x. ((x x) x) \lambda x. ((x x) x)) \rightarrow_{\alpha} (\lambda z. ((z z) z) \lambda x. ((x x) x)) \rightarrow_{\beta} ((\omega_3 \omega_3) \omega_3) \rightarrow_{\beta} \dots (((\omega_3 \omega_3) \omega_3) \omega_3) \rightarrow_{\beta} \dots$
  
- $KI\Omega$ 

$$(\lambda xy. x) I \Omega \rightarrow_{\beta} I$$

$$(\lambda xy. x) I \Omega \rightarrow_{\beta} (\lambda xy. x) I \Omega \rightarrow_{\beta} (\lambda xy. x) I \Omega \rightarrow_{\beta} (\lambda xy. x) I \Omega \rightarrow_{\beta} \dots$$



# $\eta$ -redukcia

---

- $\lambda x.(B x) \rightarrow_{\eta} B$  ak  $x \notin \text{Free}(B)$ 
  - Dôvod:
  - L.S. =  $(\lambda x.(B x))M \rightarrow_{\beta} (B[x:M] M) \rightarrow_{\beta} (B M)$
  - P.S. =  $(B M)$

**Kontráži:**

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

- $\rightarrow_{\beta\eta}$  je uzáver  $\rightarrow_{\beta} \cup \rightarrow_{\eta}$  vzhľadom na podtermy, 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  $\lambda x.M \rightarrow_{\beta\eta} \lambda x.N$ .



# Domáca úloha

---

Definujte základné funkcie pre interpreter  $\lambda$ -kalkulu:

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

navrhovaná reprezentácia v Terms.hs:

```
module Terms where
```

```
type Var = String
```

```
data LExp = LAMBDA Var LExp |
```

```
  ID Var |
```

```
  App LExp LExp |
```

```
  CON String |
```

```
  CN Integer
```

```
  deriving(Eq)
```

```
  deriving(Show, Read, Eq)
```

-- identifikator premennej je String

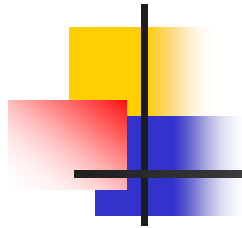
– abstrakcia

– premenná

– aplikácia

– časom konštanta, built-in fcia

– časom int.konštanta



# Cvičenie (použite váš tool)

---

1) určite voľné a viazané premenné:

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

2) redukujte:

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

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

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



## Riešenie (zle)

---

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



# Riešenie (dobre)

---

Nájdite pomocou vášho nástroja pre vyhodnocovanie  $\lambda$ -výrazov

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





# Riešenie

---

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



# Domáca úloha (nepovinná)

---

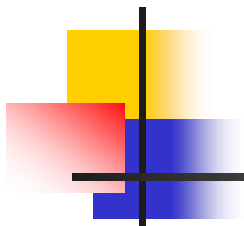
Pri práci s vašim interpretrom vám bude chýbať:

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

`fromString "\x.xx" = (LAMBDA "x" (APP (Id "x") (Id "x")))`

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

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





# Fold na termoch

---

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

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

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

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



# Od $\lambda$ -termu k programu

---

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

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

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

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



# Churchove čísla

---

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

.....

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

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

Domáca úloha (povinná):

- definujte  $\text{mult}$ ,
- definujte  $2^n$ ,  $m^n$ ,
- definujte  $n-1$



# Logické hodnoty a operátory

---

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

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

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

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

NOT :=  $\lambda x. x$  FALSE TRUE

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

AND TRUE FALSE

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

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

Cvičenie: definujte XOR



# Kartézsky súčin typov (pár)

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

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

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

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

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

LEFT (PAIR A B)  $\equiv$

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

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

Cvičenie: definujte n-ticu

Curry

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





# Súčet typov (disjunkcia)

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

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

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

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

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

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

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

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

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

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



# where (let, letrec)

---

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

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

zložený where

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