

# Lambda kalkul $\lambda$ -kalkul



Dušan Kolář

# Reference

---

- ▶ Češka, M., Motyčková, L., Hruška, T.: Vyčísitelnost a složitost, 1992, VUT v Brně, ISBN 80-214-0441-8
- ▶ Foundations of functional programming, Matthew Parkinson, 2 Lectures (Lent 2009)
- ▶ Wikipedia
- ▶ Google
- ▶ A mnoho dalších zdrojů

# Úvod

---

- ▶ **Alonso Church**

- ▶ 20. léta 20. století

- ▶ **Haskell Curry**

- ▶ V roce 1930 ukázal, že  $\lambda$ -kalkul je ekvivalentní teorii kombinátorů (Moses Schönfinkel, teorie funkcí)

- ▶ **Kleene**

- ▶ 1930,  $\lambda$ -kalkul je univerzální výpočetní systém

- ▶ **McCarthy**

- ▶ 1950, inspirován při tvorbě jazyka LISP

- ▶ **Dnes**

- ▶ Haskell, ML, Scala apod., ale i Java a C++ 😊

# Syntaxe

---

► Necht'  $E$  je výraz  $\lambda$ -kalkulu

►  $E :=$

$\vee$   
 $(E_1 E_2)$   
 $(\lambda V. E)$

*proměnná*  
*aplikace*  
*abstrakce*

► hlavička

► tělo

# Konvence

---

- ▶  $(( \dots ((E_1 E_2) E_3) \dots ) E_n) \sim E_1 E_2 E_3 \dots E_n$
- ▶  $(\lambda V.(E_1 \dots E_n)) \sim \lambda V.E_1 \dots E_n$
- ▶  $(\lambda V_1.( \dots (\lambda V_n.E) \dots )) \sim \lambda V_1 \dots V_n.E$

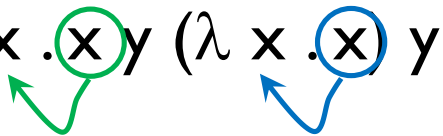
# Volné a vázané proměnné

---

- ▶  $\lambda$  vázaná . volná vázaná



- ▶  $\lambda x . \textcircled{x} y$  ( $\lambda x . \textcircled{x}$ )  $y$



# Substituce

---

- ▶  $E[E'/V]$

- ▶ Volné výskyty  $V$  jsou nahrazeny  $E'$  v  $E$ .

- ▶ Platnost substituce

- ▶ Žádná volná proměnná v  $E'$  se nesmí stát vázanou v  $E[E'/V]$

# $\alpha$ - konverze

---

▶  $\lambda V . E \rightarrow_{\alpha} \lambda V' . E[V'/V]$

▶  ~~$\lambda x . x y \rightarrow_{\alpha} \lambda y . y y$~~

▶  $\lambda x . x y \rightarrow_{\alpha} \lambda z . z y$

▶  ~~$\lambda x . x y \rightarrow_{\alpha} \lambda x . x z$~~

# $\beta$ - konverze

---

▶  $(\lambda V . E_1) E_2 \rightarrow_{\beta} E_1 [E_2/V]$

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

↑
↑  
 volná                      vázaná

▶  $(\lambda x y . x y)(x y) \rightarrow_{\alpha} (\lambda x z . x z) (x y) \rightarrow_{\beta} (\lambda z . (x y) z)$   
 $\rightarrow_{\text{konvence}} \lambda z . x y z$

▶  ~~$(\lambda x y . x y) (x y) \rightarrow_{\alpha} (\lambda x z . x z) (u v) \rightarrow_{\beta} \lambda y . u v y$~~

# $\eta$ - konverze

---

▶  $\lambda V . (EV) \rightarrow_{\eta} E$

▶  $V$  není volné v  $E$

▶  ~~$\lambda x . (x y) x \rightarrow_{\eta} (x y)$~~

▶  $\lambda x . (u v) x \rightarrow_{\eta} u v$

▶  ~~$\lambda x . x y \leftarrow_{\eta} \lambda f x . (x y) f$~~

▶  $\lambda f x . (x y) f = \lambda f . (\lambda x . ((x y) f))$ ; ale!  $\lambda f . (\lambda x . (x y)) f$

## ▶ Haskell

▶ `sqrlist l = map sqr l`

▶ `sqrlist' = map sqr`

# Identita, rovnost, relace $\rightarrow$

---

- ▶ Identita:  $E_1 \equiv E_2$
- ▶ Rovnost:  $E_1 = E_2$
- ▶ Relace  $\rightarrow$ :  $E_1 \rightarrow E_2$

- ▶  $E_1 \equiv E^0 * E^1 * \dots * E^n \equiv E_2$

- ▶ Rovnost:  $* \sim \rightarrow_\alpha, \rightarrow_\beta, \rightarrow_\eta, \leftarrow_\alpha, \leftarrow_\beta, \leftarrow_\eta$

- ▶ Relace  $\rightarrow$ :  $* \sim \rightarrow_\alpha, \rightarrow_\beta, \rightarrow_\eta$

# Pojmenování výrazu

---

- ▶ **LET True** =  $\lambda x y. x$   
**LET False** =  $\lambda x y. y$   
**LET Not** =  $\lambda t. t \text{ False True}$ 
  - ▶ *Jedná se o textovou zkratku, nikoliv rovnost z matematického pohledu, LET FF = ... FF ... je tedy nesmysl*
- ▶ **Příklad**
- ▶ **Not True** =  $(\lambda t. t \text{ False True}) \text{ True}$   
= **True** False True  
=  $(\lambda x y. x) \text{ False True}$   
=  $(\lambda y. \text{False}) \text{ True}$   
= **False**

# Další logické spojky

---

## ▶ Analýza

- ▶ True: vezmi dva argumenty, vrat' první
- ▶ False: vezmi dva argumenty, vrat' druhý

## ▶ AND – short evaluation

- ▶ Je-li první False, výsledek je False
- ▶ Je-li první True, je výsledek druhý výraz
- ▶ LET AND =  $\lambda u v. u v$  False

## ▶ OR

- ▶ LET OR =  $\lambda u v. u$  True v

# Domácí úkol 1

---

- ▶ **Definujte**

- ▶ XOR
- ▶ Implikaci

- ▶ **Ukažte**

- ▶ Pravdivost definice dle pravdivostní tabulky

- ▶ **Pro zdatné**

- ▶ Definujte True a False jiným způsobem (ovšem nikoliv prohozením vlastností True a False) a k tomu logické spojky, ukažte vlastnosti

# Peanova čísla/aritmetika

---

- ▶  $0 = \lambda f n. n$
- ▶  $1 = \text{succ } 0$
- ▶  $2 = \text{succ } 1 = \text{succ } (\text{succ } 0)$
- ▶ ...
  
- ▶ Další funkce: iszero, add, prev, sub, mul, div
- ▶ Pomocné funkce: if-then-else/( ? : ), pár, ...

# Mocniny v aplikaci

---

- ▶  $\text{LET } 0 = \lambda f n. n$
- ▶  $\text{LET } 1 = \lambda f n. f n$
- ▶  $\text{LET } 2 = \lambda f n. f (f n) \qquad = \lambda f n. f^2 n$
- ▶  $\text{LET } 3 = \lambda f n. f (f (f n)) \qquad = \lambda f n. f^3 n$
  
- ▶  $\text{LET } n \qquad \qquad \qquad = \lambda f n. f^n n$

# Následník – succ

---

▶ LET succ =  $\lambda x g m. x g (g m)$

▶ *Příklad*

▶ succ 0 =  $(\lambda x g m. x g (g m)) 0$   
=  $(\lambda g m. 0 g (g m))$   
=  $(\lambda g m. (\lambda f n. n) g (g m))$   
=  $(\lambda g m. (\lambda n. n) (g m))$   
=  $\lambda g m. g m$

▶ succ 2 =  $\lambda g m. 2 g (g m)$   
=  $\lambda g m. (\lambda f n. f (f n)) g (g m)$   
=  $\lambda g m. (\lambda n. g (g n)) (g m)$   
=  $\lambda g m. g (g (g m))$

# Test na 0

---

▶ LET iszero =  $\lambda m. m (\lambda v. \text{False}) \text{True}$

▶ *Příklad*

▶ iszero 0 =  $(\lambda m. m (\lambda v. \text{False}) \text{True}) 0$   
=  $0 (\lambda v. \text{False}) \text{True}$   
=  $(\lambda f n. n) (\lambda v. \text{False}) \text{True}$   
= **True**

▶ iszero 2 =  $2 (\lambda v. \text{False}) \text{True}$   
=  $(\lambda f n. f (f n)) (\lambda v. \text{False}) \text{True}$   
=  $(\lambda v. \text{False}) ((\lambda v. \text{False}) \text{True})$   
= **False**

# Předchůdce – prev

---

- ▶  $\text{prev } 0 = 0$   
 $\text{prev } 1 = 0$   
 $\text{prev } 2 = 1$   
...
- ▶  $\text{LET } (? : ) = \lambda c t f. c t f$
- ▶  $\text{LET } ( \_ , \_ ) = \lambda f s e. e f s$
- ▶  $\text{LET } \text{fst} = \lambda p. p \text{ True}$
- ▶  $\text{LET } \text{snd} = \lambda p. p \text{ False}$

# Domácí úkol 2

---

▶ Ukažte, že:

- ▶  $(\text{True} ? 1 : 0) = 1$
- ▶  $(\text{False} ? 0 : 3) = 3$
- ▶  $\text{fst}(1, 2) = 1$
- ▶  $\text{snd}(\text{True}, \text{False}) = \text{False}$

# Definice prev

---

- ▶  $\text{LET pref n} = \lambda f p. (\text{fst } p ?$   
 $(\text{False, snd } p) :$   
 $(\text{False, } f (\text{snd } p)) )$
- ▶  $\text{LET prev} = \lambda x g m. \text{snd } (x (\text{prefn } g) (\text{True, } m))$
- ▶  $\text{prev } 0 = (\lambda x g m. \text{snd } (x (\text{prefn } g) (\text{True, } m))) 0$   
 $= \lambda g m. \text{snd } (0 (\text{prefn } g) (\text{True, } m))$   
 $= \lambda g m. \text{snd } ((\lambda f n. n) (\text{prefn } g) (\text{True, } m))$   
 $= \lambda g m. \text{snd } (\text{True, } m)$   
 $= \lambda g m. m$

# Užití prev

---

- ▶  $\text{prev } 2 = (\lambda x g m. \text{snd } (x (\text{prefn } g) (\text{True}, m))) 2$ 
  - $= \lambda g m. \text{snd } (2 (\text{prefn } g) (\text{True}, m))$
  - $= \lambda g m. \text{snd } ((\lambda f n.f (f n)) (\text{prefn } g) (\text{True}, m))$
  - $= \lambda g m. \text{snd } (\text{prefn } g (\text{prefn } g (\text{True}, m)))$
  - $= \lambda g m. \text{snd } (\text{prefn } g ((\lambda f p. (\text{fst } p ?$ 
    - $(\text{False}, \text{snd } p) :$
    - $(\text{False}, f (\text{snd } p)))) g (\text{True}, m)) )$
    - $= \lambda g m. \text{snd } (\text{prefn } g (\text{False}, m))$
    - $= \lambda g m. \text{snd } ((\lambda p. (\text{fst } p ? (\text{False}, \text{snd } p) :$ 
      - $(\text{False}, g (\text{snd } p)))) (\text{False}, m))$
      - $= \lambda g m. \text{snd } (\text{False}, g m)$
      - $= \lambda g m. g m$

# Součet – add

---

▶ LET add =  $\lambda a b g m. a g (b g m)$

▶ *Příklad*

▶ add 2 3 =  $(\lambda a b g m. a g (b g m)) 2 3$   
=  $\lambda g m. 2 g (3 g m)$   
=  $\lambda g m. (\lambda f n. f(f n)) g ((\lambda f n. f(f(f n)))) g m$   
=  $\lambda g m. (\lambda n. g (g n)) (g (g (g m)))$   
=  $\lambda g m. g (g (g (g (g m))))$

# Rozdíl – sub

---

▶ LET sub =  $\lambda a b. b$  prev a

▶ *Příklad*

▶ sub 3 2 =  $(\lambda a b. b$  prev a) 3 2  
=  $(\lambda f n. f (f n))$  prev 3  
= prev (prev 3)  
= prev 2  
= 1

# Domácí úkol 3

---

- ▶ Ukažte (detailně) správnost
  - ▶ iszero (succ 0)
  - ▶ sub 2 4
  - ▶ iszero (add 0 0)
- ▶ Definujte
  - ▶ eq tak, že eq bere dvě čísla a vrací True, pokud jsou si rovna, jinak False
  - ▶ Ukažte správnou funkci eq

# Násobení – mult

---

## ▶ Rozbor:

- ▶ Je-li jedno z čísel 0, potom výsledek je 0
- ▶ Jinak je to, pro čísla  $m$  a  $n$ ,  $m$ -krát sečteno číslo  $n$

## ▶ Tedy, pracovně:

- ▶ LET mult =  $\lambda m n. (\text{OR } (\text{iszero } m) (\text{iszero } n) ? 0 : \text{multfn } m n)$
- ▶ LET multfn =  $\lambda m n. m (\text{add } n) 0$

## ▶ A vidíme, že

- ▶  $\text{multfn } 2 \ 3 = (\lambda m n. m (\text{add } n) 0) \ 2 \ 3 = 2 (\text{add } 3) 0$   
 $= (\lambda f n. f (f n)) (\text{add } 3) 0 = (\text{add } 3) ((\text{add } 3) 0)$   
 $= \text{add } 3 (\text{add } 3 0) = \text{add } 3 \ 3 = 6$
- ▶  $\text{multfn } 0 \ 2 = (\lambda m n. m (\text{add } n) 0) \ 0 \ 2 = 0 (\text{add } 2) 0$   
 $= (\lambda f n. n) (\text{add } 2) 0 = 0$

# Násobení – mult (pokračování)

---

▶ No a také

▶  $\text{multfn } 2 \ 0 = (\lambda m \ n. m \ (\text{add } n) \ 0) \ 2 \ 0 = 2 \ (\text{add } 0) \ 0$   
 $= (\lambda f \ n. f \ (f \ n)) \ (\text{add } 0) \ 0$   
 $= \text{add } 0 \ (\text{add } 0 \ 0) = \text{add } 0 \ 0$   
 $= 0$

▶ Takže:

▶  $\text{LET mult} = \lambda m \ n. m \ (\text{add } n) \ 0$

# Domácí úkol 4

---

- ▶ Definujte funkci `sqr`, která vrací druhou mocninu parametru a ukažte její správnou funkci
- ▶ Definujte funkci `gt`, která bere dvě čísla jako parametr a vrací `True`, pokud je první ostře větší, jak druhé, jinak vrací `False`
- ▶ Ukažte, že `gt (sqr 3) (sqr 2) = True`

# Celočíselné dělení – div

---

- ▶ Uvažme  $\text{div } m \ n$ 
  - ▶ Je-li  $n$  rovno 0, potom výsledek není definován
  - ▶ Jinak odčítáme od  $m$  tak dlouho  $n$ , dokud výsledek není menší, jak  $n$ , u toho počítáme počet odečtení
- ▶ Co to znamená, že není definován v rovině  $\lambda$ -kalkulu?
  - ▶ Z pohledu programátora
    - ▶ Výjimka
    - ▶ **Nekonečná smyčka**
  - ▶ Definujme výraz, který bere neomezené množství argumentů do nekonečna...

# Bottom

---

- ▶ Výraz, který bude neustále na výstup produkovat sebe nazveme bottom (anglicky, nepřekládáme)
  - ▶ Tento modeluje do jisté míry nekonečnou smyčku v programu.
- ▶ Využijeme pro jeho definici **operátor pevného bodu**
  - ▶ Pevný bod výrazu  $E$ :  $Y E$ 
    - ▶ Operátor je  $Y$
  - ▶ Necht' pro výraz  $E$  je pevný bod  $k_E = Y E$
  - ▶ Z definice vlastnosti:  $E k_E = k_E = Y E = E (Y E)$
- ▶ Tedy pro bottom definujeme
  - ▶  $LET \perp = Y (\lambda f x. f)$

# Příklad

---

▶  $\perp$  I True AND      =  $(Y (\lambda f x. f))$  I True AND  
=  $((\lambda f x. f) (Y (\lambda f x. f)))$  I True AND  
=  $((\lambda f x. f) \perp)$  I True AND  
=  $(\lambda x. \perp)$  I True AND  
=  $\perp$  True AND  
=  $(Y (\lambda f x. f))$  True AND  
=  $((\lambda f x. f) \perp)$  True AND  
=  $(\lambda x. \perp)$  True AND  
=  $\perp$  AND  
=  $(Y (\lambda f x. f))$  AND  
=  $((\lambda f x. f) \perp)$  AND  
=  $\perp$

# Operátor pevného bodu Y

---

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

▶ Ukázka vlastnosti

$$\begin{aligned} \text{▶ } (Y\ E) &= (\lambda\ f.\ (\lambda\ x.\ f\ (x\ x))\ (\lambda\ x.\ f\ (x\ x)))\ E \\ &= (\lambda\ x.\ E\ (x\ x))\ (\lambda\ x.\ E\ (x\ x)) \\ &= E\ ((\lambda\ x.\ E\ (x\ x))\ (\lambda\ x.\ E\ (x\ x))) \\ &= E\ (YE) \end{aligned}$$

# Celočíselné dělení – div (pokračování)

---

## ▶ Tedy

- ▶ LET div =  $\lambda m n. (iszero\ n\ ?\ \perp : divfn\ m\ n\ 0)$
- ▶ *LET divfn =  $\lambda m\ n\ r. (gt\ n\ m\ ?\ r : \underline{divfn}\ (sub\ m\ n)\ n\ (succ\ r))$*

## ▶ Jak z toho ven?

- ▶ Operátorem pevného bodu
- ▶ LET divf =  $(\lambda f\ m\ n\ r. (gt\ n\ m\ ?\ r : f\ (sub\ m\ n)\ n\ (succ\ r)))$
- ▶ LET divfn = Y divf

## ▶ Příklad

- ▶  $div\ 5\ 2 = (\lambda m\ n. (iszero\ n\ ?\ \perp : divfn\ m\ n\ 0))\ 5\ 2$   
=  $divfn\ 5\ 2\ 0 = (Y\ divf)\ 5\ 2\ 0 = divf\ (Y\ divf)\ 5\ 2\ 0$   
=  $(gt\ 2\ 5\ ?\ 0 : divfn\ (sub\ 5\ 2)\ 2\ (succ\ 0))$   
=  $divfn\ 3\ 2\ 1 = divf\ (Y\ divf)\ 3\ 2\ 1$   
=  $(gt\ 2\ 3\ ?\ 1 : divfn\ (sub\ 3\ 2)\ 2\ (succ\ 1)) = divfn\ 1\ 2\ 2 = 2$

# Domácí úkol 5

---

- ▶ Definujte operace `sub` a `mult` pomocí operátoru pevného bodu
- ▶ Ukažte jejich správnou funkčnost

# Seznamy

---

- ▶ Prázdný seznam
  - ▶  $\text{LET } [] = (\text{False}, \text{False})$
- ▶ Konstruktor seznamu
  - ▶  $\text{LET } (:) = \lambda h t. (\text{True}, (h, t))$
- ▶ Hlavička (první prvek) neprázdného seznamu
  - ▶  $\text{LET head} = \lambda l. (\text{fst } l ? \text{fst } (\text{snd } l) : \perp)$
- ▶ Zbytek seznamu
  - ▶  $\text{LET tail} = \lambda l. (\text{fst } l ? \text{snd } (\text{snd } l) : \perp)$
- ▶ Test na prázdný seznam
  - ▶  $\text{LET null} = \lambda l. \text{not } (\text{fst } l)$

# Domácí úkol 6

---

- ▶ Ukažte správnou funkčnost funkcí pro seznamy
  - ▶ head (1:2:3:[])
  - ▶ tail (2:[])
  - ▶ null (1:2:[])
- ▶ Definujte funkci vracející délku seznamu vloženého jako parametr
- ▶ Definujte funkci vracející n-tý prvek seznamu délky alespoň n

- ▶  $E_1 \rightarrow E_2$ 
  - ▶  $E_2$  jsme získali „vyhodnocením“  $E_1$
  - ▶ Pokud  $E_2$  neobsahuje žádný  $\beta$ -,  $\eta$ - redex, je úplně vyhodnocen
- ▶ **Definice NF**
  - ▶ Výraz  $E$  je v **normální formě**, pokud neobsahuje žádné jiné redexy krom  $\alpha$ -redexu.
- ▶ **Definice HNF**
  - ▶ Výraz  $E$  je v **hlavičkové normální formě**, pokud to je proměnná, hodnota, vestavěná funkce aplikovaná na příliš málo argumentů, nebo  $\lambda$ -abstrakce jejíž tělo není redukovatelné (oproti NF může obsahovat redexy na pozicích argumentů)
- ▶ **Definice WHNF**
  - ▶ Výraz  $E$  je ve **slabé hlavičkové normální formě** pokud je ve HNF nebo je to  $\lambda$ -abstrakce

## ▶ Church-Rosserův teorém pro $\lambda$ -kalkul

- ▶ Pokud  $E_1 = E_2$ , pak existuje  $E$  takové,

že  $E_1 \rightarrow E$  a  $E_2 \rightarrow E$

## ▶ Důsledky Church-Rosserova teorému

- ▶ Pokud  $E$  má normální formu, potom  $E \rightarrow E'$  pro nějaké  $E'$  v normální formě
- ▶ Pokud  $E$  má normální formu a  $E = E'$ , potom  $E'$  má normální formu
- ▶ *Pokud  $E = E'$  a  $E$  i  $E'$  jsou oba v normální formě, potom  $E$  a  $E'$  jsou identické až na přejmenování vázaných proměnných ( $\alpha$ -konverzi)*

## ▶ Normalizační teorém

- ▶ Pokud  $E$  má normální formu, potom opakovaná redukce nejlevějšího  $\beta$ -,  $\eta$ - redexu (po možné  $\alpha$ -konverzi pro zabránění neplatné náhradě) bude končit ve výrazu v normální formě.

## ▶ Definice

- ▶ Posloupnost redukcí, v nichž je vždy redukován nejlevější redex, se nazývá **posloupnost redukcí v normálním pořadí**.

# Kombinátory

---

- ▶ Teorie funkcí
- ▶ Lze odvodit z  $\lambda$ -kalkulu
- ▶ Lze zavést jako zcela oddělenou teorii
  
- ▶ V 80. letech 20. století „strojový kód“ pro překlad funkcionálních jazyků – Turner

# Kombinátory S, K, I

---

- ▶ LET K =  $\lambda x y. x$
- ▶ LET S =  $\lambda f g x. (f x) (g x)$
- ▶ LET I =  $\lambda x. x = S K K$
  
- ▶ Úkol:
  - ▶ Ukažte, že  $S K K = I$ .

# Kombinátory B, C, K, W

---

- ▶  $\text{LET } B = \lambda x y z. x (y z)$
- ▶  $\text{LET } C = \lambda x y z. x z y$
- ▶  $\text{LET } K = \lambda x y. x$
- ▶  $\text{LET } W = \lambda x y. x y y$
  
- ▶ Úkol, ukažte, že:
  - ▶  $B = S (K S) K$
  - ▶  $C = S (S (K (S (K S) K)) S) (K K)$
  - ▶  $W = S S (K (S K K))$
  - ▶  $S = B (B (B W) C) (B B)$

# Převod kalkulu na kombinátory

---

## ▶ Definice

- ▶ Pro libovolnou proměnnou  $V$  a kombinatorický výraz  $E$  existuje jiný kombinatorický výraz  $\lambda^*V.E$ , který simuluje  $\lambda V.E$  tak, že  $\lambda^*V.E = \lambda V.E$

## ▶ Induktivně pro $S, K, I$ , když

- ▶  $P ::= S \mid K \mid I \mid P P \mid x$

- ▶  $\lambda^*x.x = I$

- ▶  $\lambda^*x.P = K P$  ,  $x \notin FV(P)$

- ▶  $\lambda^*x.P P' = S (\lambda^*x.P) (\lambda^*x.P')$

# Ukázka převodu

---

- ▶  $\lambda^*x. \lambda^*y. x$ 
  - $= \lambda^*x. K x$
  - $= S (\lambda^*x. K) (\lambda^*x. x)$
  - $= S (\lambda^*x. K) I$
  - $= S (K K) I$
  
- ▶  $(\lambda^* x. \lambda^* y. y x)$ 
  - $= \lambda^* x. S (\lambda^* y. y) (\lambda^* y. x)$
  - $= \lambda^* x. (S I) (K x)$
  - $= S (\lambda^* x. (S I)) (\lambda^* x. K x)$
  - $= S (K (S I)) (S (\lambda^* x. K) (\lambda^* x. x))$
  - $= S (K (S I)) (S (K K) I)$

# Redukce

---

- ▶  $I E \rightarrow E$
- ▶  $K E_1 E_2 \rightarrow E_1$
- ▶  $S E_1 E_2 E_3 \rightarrow (E_1 E_3) (E_2 E_3)$
  
- ▶  $S (K K) I P Q R$ 
  - $\rightarrow ((K K) P) (I P) Q R$
  - $\rightarrow K P Q R$
  - $\rightarrow P R$

# Optimalizace

---

- ▶  $S (K E) I = E$
- ▶  $S (K E) (K E') = K (E E')$
- ▶ ?
- ▶  $S (K K) I = K$
- ▶ Domácí úkol 7
  - ▶ Ukažte!