



# Logika kombinátorov<sup>(SK)</sup>

Už dávnejšie sme sa stretli s tromi dôležitými príkladmi:

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

inými slovami:

$S$	$f$	$g$	$x$	$\rightarrow$	$(f\ x)$	$(g\ x)$
$K$	$c$	$x$		$\rightarrow$	$c$	
$I$	$x$			$\rightarrow$	$x$	

pričom vieme, že platí (veta o zbytočnosti identity I):

- $((S\ K)\ K) = I$
- $((S\ K)\ S) = I$  -- toto sme možno netušili, tak si to dokážme...

$$S\ K\ K\ x = ((S\ K)\ K)\ x = (K\ x)\ (K\ x) = x$$

$$S\ K\ S\ x = ((S\ K)\ S)\ x = (K\ x)\ (S\ x) = x$$

[https://en.wikipedia.org/wiki/Combinatory\\_logic](https://en.wikipedia.org/wiki/Combinatory_logic)

[http://people.cs.uchicago.edu/~odonnell/Teacher/Lectures/Formal\\_Organization\\_of\\_Knowledge/Examples/combinator\\_calculus/](http://people.cs.uchicago.edu/~odonnell/Teacher/Lectures/Formal_Organization_of_Knowledge/Examples/combinator_calculus/)

1958, Haskell Curry: Combinatory Logic



# Logika kombinátorov<sup>(SK)</sup>

Ukážeme, že ľub. **uzavretý  $\lambda$ -výraz** (neobsahujúci voľné premenné) vieme prepísať pomocou kombinátorov S, K, (I) tak, že zápis neobsahuje abstrakciu (t.j. neobsahuje premenné - pozn. ktoré robia problémy v interprete).

data Ski = S | K | I | APL Ski Ski

Intuitívne smerujeme k transformácii LExp  $\rightarrow$  Ski (pre **uzavreté  $\lambda$ -termy**).

Kým v  $\lambda$ -teórii sme mali hlavné pravidlo  $\beta$ -redukcie, v Ski-teórii máme tri nové redukčné/výpočtové pravidlá (vlastne sú to definície operátorov):

- S-redukcia       $S f g x \rightarrow ((f x) (g x))$
- K-redukcia       $K c x \rightarrow c$
- I-redukcia       $I x \rightarrow x$

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

$$K = \lambda xy.x$$

$$I = \lambda x.x$$

# S, K, (I) transformácie

## Transformácia do SK(I)

- $\lambda x.x \quad \rightarrow_I \quad I = ((S\ K)\ K)$
- $\lambda x.c \quad \rightarrow_K \quad K\ c \quad \text{ak } x \notin \text{Free}(c)$
- $\lambda x.(M\ N) \quad \rightarrow_S \quad S\ (\lambda x.M)\ (\lambda x.N)$

Prvé dve sú evidentné, ale poslednú rovnosť si overme:

- $\lambda x.(M\ N)\ y \rightarrow_\beta (M[x:y]\ N[x:y])$
- $S\ (\lambda x.M)\ (\lambda x.N)\ y \rightarrow_\beta (\lambda x.M\ y)\ (\lambda x.N\ y) \rightarrow_\beta (M[x:y]\ N[x:y])$   
platí...☺

Zmyslom S, K, (I) transformácií je eliminácia abstrakcie, premenných t.j. dostaneme výraz neobsahujúci abstrakciu, viazané premenné.

Skôr, než si to sami naprogramujete, vyskúšajte

<http://tromp.github.io/cl/cl.html>

<http://ski.aditsu.net/>

# Príklad transformácie

Transformácia do SK(I)

$\lambda x.x \rightarrow I$

$\lambda x.c \rightarrow K c$

$\lambda x.(M N) \rightarrow S (\lambda x.M) (\lambda x.N)$

<http://ski.aditsu.net/>

- $\lambda x.\lambda y.(y x) \rightarrow_S$
- $\lambda x.(S (\lambda y.y) (\lambda y.x)) \rightarrow_I$
- $\lambda x.(S I (\lambda y.x)) \rightarrow_K$
- $\lambda x.(S I (K x)) \rightarrow_S$
- $S (\lambda x.(S I)) (\lambda x.(K x)) \rightarrow_K$
- $S (K (S I)) (\lambda x.(K x)) \rightarrow_S$
- $S (K (S I)) (S (\lambda x.K) (\lambda x.x)) \rightarrow_K$
- $S (K (S I)) (S (K K) (\lambda x.x)) \rightarrow_I$
- $S (K (S I)) (S (K K) I). = ((S (K (S ((S K) K)))) ((S (K K)) ((S K) K)))$

Väčší príklad:

- $Y \rightarrow$

$((S ((S ((S (K S)) ((S (K K)) I))) (K ((S I) I)))) ((S ((S (K S)) ((S (K K)) I))) (K ((S I) I))))$



# Príklad

---

Transformácia do SK(I)

$\lambda x.x \rightarrow I$

$\lambda x.c \rightarrow K\ c$

$\lambda x.(M\ N) \rightarrow S\ (\lambda x.M)\ (\lambda x.N)$

■  $\lambda x.\lambda y.(y\ x) = \dots$

# Výpočet v SK(I) teórii

S-redukcia

K-redukcia

I-redukcia

$((S f) g) x \rightarrow (f x) (g x)$

$(K c) x \rightarrow c$

$I x \rightarrow x$

$((\lambda x. \lambda y. (y x) \text{ 5}) \text{ (+1)})$

– pri výpočte nemáme viazané premenné,  
t.j. nepotrebuje pojem substitúcie

$S (K (S I)) (S (K K) I) \text{ 5 (+1)}$

$\rightarrow_S$

$((K (S I)) \text{ 5}) ((S (K K) I) \text{ 5 (+1)})$

$\rightarrow_K$

$(S I) ((S (K K) I) \text{ 5 (+1)})$

$\rightarrow_S$

$(I \text{ (+1)}) ((S (K K) I) \text{ 5 (+1)})$

$\rightarrow_I$

$\text{(+1)} ((S (K K) I) \text{ 5 (+1)})$

$\rightarrow_S$

$\text{(+1)} (((K K) \text{ 5}) (I \text{ 5 (+1)}))$

$\rightarrow_I$

$\text{(+1)} (((K K) \text{ 5}) \text{ 5 (+1)})$

$\rightarrow_K$

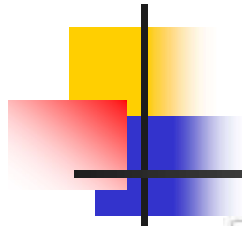
$\text{(+1)} (K \text{ 5 (+1)})$

$\rightarrow_K$



















$\text{(+1)} \text{ 5}$

$\rightarrow_+$

6



ski.aditsu.net



## SKI (combinatory logic) interpreter

**Code:**

```
S (K (S I)) (S (K K) I) a b
```

**Run**

**Output:**

```
b(a)
```



# Domáca úloha

V module `SKI` definujte:

- funkciu `toSki :: LExp -> Ski`
- funkciu `fromSki :: Ski -> LExp`
- funkciu `oneStep :: Ski -> Ski` a `nf :: Ski -> Ski`

Naprogramujte konvertor do/z SK(I)

Naprogramujte SK(I) redukčný stroj, teda normálnu formu ak existuje

Implementačná poznámka:

Funkcia  $LExp \rightarrow Ski$  sa zle píše, lebo proces transformácie do SKI má medzistavy, keď výraz už nie je `LExp` a ešte nie je `Ski`.

```
data LExpSki =  
    LAMBDA String LExp |  
    ID String |  
    APL LExp LExp |  
    CON String |  
    CN Integer |  
    S | K | I  
    deriving(Show, Read, Eq)  
toSki      :: LExpSki -> LExpSki
```





# Vlastnosti SKI teórie

S-redukcia

K-redukcia

I-redukcia

$((S f) g) x \rightarrow (f x) (g x)$

$(K c) x \rightarrow c$

$I x \rightarrow x$

SKI redukcie spĺňajú Church-Rosserovu vlastnosť, t.j.

SKI term má najviac jednu normálnu formu vzhľadom na redukcie S, K, I

Vážny problém: dve **rôzne** SKI-normálne formy predstavujú rovnaký  $\lambda$ -term

$SKK = I = SKS$

Existuje nekonečné odvodenie,  $\Omega = ((S I I) (S I I)) \rightarrow_S ((I (S I I)) (I (S I I)))$   
 $\rightarrow_I ((S I I) (I (S I I))) \rightarrow_I ((S I I) (S I I))$

Cvičenie:

$\lambda x.(x x) \rightarrow_S S (\lambda x.x) (\lambda x.x) \rightarrow_{I,I} S I I$



# Je systém SK minimálny ?

Či existuje verzia kombinátorovej logiky aj s jedným kombinátorom ?  
Je to čisto teoretická otázka, v praxi potrebujeme opak...

Nech  $X = \lambda x.(x K S K)$

- potom vieme ukázať, že  $K = X X X = (X X) X$
- A tiež, že  $S = X . X X = X (X X)$

Skúste si to ako cvičenie...

Iná možnosť je, ak  $X = \lambda x.((x S) K)$

- potom  $K = X (X (X X))$
- a  $S = X (X (X (X X)))$

Toto je na rozcvičke

Môže sa zdať, že ide o čisto teoretický výsledok, ale existuje programovací jazyk (Iota - pokročilé čítanie pre extrémisticky ladených nadšencov) používajúci X ako jedinú jazykovú konštrukciu.

[https://en.wikipedia.org/wiki/Iota\\_and\\_Jot](https://en.wikipedia.org/wiki/Iota_and_Jot)  
<https://esolangs.org/wiki/Iota>  
<http://www.nyu.edu/projects/barker/Iota/>


$$X = \lambda x. (x K S K)$$

---

Nech  $X = \lambda x. (x K S K)$   
potom vieme ukázať, že  $K = X X X = (X X) X$

A tiež, že  $S = X . X X = X (X X)$  ... Rozcvička 8

- $K = X X X = (X X) X$

$$(X X) X =$$

- $S = X . X X = X (X X)$

$$X (X X) = \dots = S$$



$$X = \lambda x. (x K S K)$$

Nech  $X = \lambda x. (x K S K)$   
 potom vieme ukázať, že  $K = X X X = (X X) X$

A tiež, že  $S = X . X X = X (X X)$  ... Rozcvička 6

■  $K = X X X = (X X) X$

$$\begin{aligned} (X X) X &= (\lambda x. (x K S K) X) X \\ &= (X K S K) X \\ &= (\lambda x. (x K S K) K S K) X \\ &= ((\underline{K K S K}) S K) X \\ &= ((\underline{K K}) \underline{S K}) X = \underline{K K X} \\ &= K \end{aligned}$$

■  $S = X . X X = X (X X)$

$$\begin{aligned} X (X X) &= \lambda x. (x K S K) (X X) \\ &= (X X) K S K \\ &= (\lambda x. (x K S K) X) K S K \\ &= (X K S K) K S K \\ &= (\lambda x. (x K S K) K S K) K S K \\ &= ((\underline{K K S K}) S K) K S K \\ &= ((\underline{K K}) \underline{S K}) K S K \\ &= (\underline{K K}) \underline{K S K} \\ &= \underline{K S K} \\ &= S \end{aligned}$$

# Vlastnosti operátorov

## (príklady SKI výrazov)

Komutatívnosť operátora f:

Ak  $C f x y = f y x$ , potom musí platiť, že  $C f = f$  (aby f bola komutatívna oper.)

- v  $\lambda$ -teórii je  $C = \lambda f. \lambda x. \lambda y. (f y) x$
- v SKI-teórii

$$C = ((S ((S (K S)) ((S (K K)) ((S (K S)) ((S ((S (K S)) ((S (K K)) I))) (K I)))))) (K ((S (K K)) I)))$$

naozaj:  $((S ((S (K S)) ((S (K K)) ((S (K S)) ((S ((S (K S)) ((S (K K)) I))) (K I)))))) (K ((S (K K)) I))) f a b \rightarrow ((f b) a)$

Asociatívnosť operátora f:

$$A_1 f = A_2 f$$

$$A_1 f x y z = (f (f x y) z) = \lambda f. \lambda x. \lambda y. \lambda z. (f (f x y) z) =$$

$$((S ((S (K S)) ((S (K (S (K S)))) ((S (K (S (K (S (K S)))))) ((S (K (S (K (S (K K)))))) ((S ((S (K S)) ((S (K K)) ((S (K S)) ((S (K K)) I)))))) ((S ((S (K S)) ((S (K (S (K S)))) ((S (K (S (K K)))) ((S ((S (K S)) ((S (K K)) I))) (K I)))))) (K (K I)))))) (K (K I)))))) (K (K I))))$$

naozaj:  $A_1 f a b c \rightarrow ((f((fa)b))c)$

$$A_2 f x y z = (f x (f y z)) = \lambda f. \lambda x. \lambda y. \lambda z. (f x (f y z)) =$$

$$((S ((S (K S)) ((S (K (S (K S)))) ((S (K (S (K K)))) ((S (K (S (K S)))) ((S (K (S (K K)))) ((S ((S (K S)) ((S (K K)) I))) (K I)))))) ((S (K K)) ((S ((S (K S)) ((S (K (S (K S)))) ((S (K (S (K K)))) ((S ((S (K S)) ((S (K K)) I))) (K I)))))) (K (K I)))))) (K (K I))))$$

naozaj:  $A_2 f a b c \rightarrow ((fa)((fb)c)$

$$\lambda x.(M\ N) \rightarrow S\ (\lambda x.M)\ (\lambda x.N)$$

## B, C kombinátory

Praktický problém pri používaní kombinátorov je v tom, že výsledné SK výrazy sú veľké, napr.  $\lambda x.(+ 1) \rightarrow S\ (K\ +)\ (K\ 1)$  pričom aj  $K\ (+\ 1)$  by stačilo.

Problém je  $\lambda$ -abstrakcia pri S transformácii, ktorá sa množí do M aj N.

Špecializujme S kombinátor na dve verzie:

$$B = \lambda xyz.(x\ (yz))$$

$$\text{B-redukcia } B\ f\ g\ x = (f\ (g\ x))$$

$$C = \lambda xyz.((x\ z)\ y)$$

$$\text{C-redukcia } C\ f\ g\ x = ((f\ x)\ g)$$

Následne potom zavedieme dve nové transformácie:

Transformácia do SK(I)BC

- $\lambda x.x \rightarrow I = S\ K\ K$
- $\lambda x.c \rightarrow K\ c$  ak  $x \notin \text{Free}(c)$
- $\lambda x.(M\ N) \rightarrow S\ (\lambda x.M)\ (\lambda x.N)$  ak  $x \in \text{Free}(M)$  &  $x \in \text{Free}(N)$
- $\lambda x.(M\ N) \rightarrow B\ M\ (\lambda x.N)$  ak  $x \in \text{Free}(N)$  &  $x \notin \text{Free}(M)$
- $\lambda x.(M\ N) \rightarrow C\ (\lambda x.M)\ N$  ak  $x \in \text{Free}(M)$  &  $x \notin \text{Free}(N)$

**Cvičenie: doprogramujte BC transformácie a BC redukcie do interpretera**

$B = \lambda xyz.x (yz)$     B-redukcia  $B f g x = f (g x)$   
 $C = \lambda xyz.(x z) y$     C-redukcia  $C f g x = (f x) g$

# Optimalizácia výsledného kódu

Problém SK termov je ich veľkosť, preto sa sústredíme na to, ako ju zmenšiť pri zachovaní jeho sémantiky.

Uvedieme niekoľko príkladov a z toho odvodených pravidiel:

$\lambda x.(+ 1) \rightarrow S (K +) (K 1) \rightarrow K (+ 1)$

■  **$S (K p) (K q) = K (p q)$**

$\lambda x.- x \rightarrow S (K -) I \rightarrow B - I$

■  **$S (K p) q = B p q$**

■  **$S (K p) I = B p I = p$**

■  **$S p (K q) = C p q$**

$S p (K q) x \rightarrow C p q x \rightarrow$   
 $(p x) (K q x) \rightarrow (p x) q$

Simon Peyton Jones, 1987,  
**The Implementation of Functional Programming Languages**

<http://research.microsoft.com/en-us/um/people/simonpj/papers/slpj-book-1987/>



# S,K,I,B,C kombinátory

---

- $\lambda x.x \rightarrow I$
- $\lambda x.C \rightarrow K C$   $x \notin \text{Free}(C)$
- $\lambda x.(M N) \rightarrow S (\lambda x.M) (\lambda x.N)$
- $\lambda x.(M x) \rightarrow M$   $x \notin \text{Free}(M)$
- $S (K M) N \rightarrow B M N$
- $S M (K N) \rightarrow C M N$
- $S (K M) (K N) \rightarrow K (M N)$
- $S (K M) I \rightarrow M$



$$p^1 = \text{toSki } \lambda x_1.p, q^1 = \text{toSki } \lambda x_1.q$$

$$p^2 = \text{toSki } \lambda x_2.\lambda x_1.p, q^2 = \text{toSki } \lambda x_2.\lambda x_1.q$$

...

## S' kombinátor

- $\lambda x_n \dots \lambda x_3.\lambda x_2.\lambda x_1.(p \ q)$  →
  - $\lambda x_n \dots \lambda x_3.\lambda x_2.(S \ p^1 \ q^1)$  →
  - $\lambda x_n \dots \lambda x_3.(S \ (B \ S \ p^2) \ q^2)$  →
  - $\lambda x_n \dots \lambda x_4.(S \ (B \ S \ (B \ (B \ S) \ p^3)) \ q^3)$  →
  - $\lambda x_n \dots \lambda x_5.(S \ (B \ S \ (B \ (B \ S) \ (B \ (B \ (B \ S)) \ p^4))) \ q^4)$  → rastie kvadraticky od n
- často vyskytujúci sa vzor  $S \ (B \ x \ y)$  z nahradíme novým kombinátorom  $S' \ x \ y \ z$   
teda S B nahradíme  $S'$

→  $\lambda x_n \dots \lambda x_4.(S \ (B \ S \ (B \ (B \ S) \ p^3)) \ q^3)$  →

$\lambda x_n \dots \lambda x_4.(S' \ (S \ (B \ S) \ p^3)) \ q^3)$  →

$\lambda x_n \dots \lambda x_4.(S' \ (S' \ (B \ S) \ p^3) \ q^3)$  →

$\lambda x_n \dots \lambda x_4.(S' \ (S' \ S) \ p^3 \ q^3)$

$$B \ f \ g \ x = f \ (g \ x)$$

- $\lambda x_n \dots \lambda x_3.\lambda x_2.\lambda x_1.(p \ q)$  →
- $\lambda x_n \dots \lambda x_3.\lambda x_2.(S \ p^1 \ q^1)$  →
- $\lambda x_n \dots \lambda x_3.(S' \ S \ p^2 \ q^2)$  →
- $\lambda x_n \dots \lambda x_4.(S' \ (S' \ S) \ p^3 \ q^3)$  →
- $\lambda x_n \dots \lambda x_5.(S' \ (S' \ (S' \ S)) \ p^4 \ q^4)$  → rastie už len lineárne...

$B = \lambda xyz.x (yz)$     B-redukcia  $B f g x = f (g x)$   
 $C = \lambda xyz.(x z) y$     C-redukcia  $C f g x = (f x) g$



## $S', B', C'$ kombinátory

---

často vyskytujúci sa vzor  $(S (B x y) z)$  nahradíme novým kombinátorom  $S' x y z$

Keď dosadíme  $S B$ , dostaneme  $S'$  kombinátor ako

$$S' c f g x = S (B c f) g x = (B c f x) (g x) = (c (f x)) (g x)$$

$$S' c f g x = c (f x) (g x)$$

Analogicky potom zavedieme dva obmedzené kombinátory podľa vzoru  $B', C'$

$$B' c f g x = c f (g x)$$

$$C' c f g x = c (f x) g$$

Cvičenie:

Pridajte špeciálne kombinátory pre IF, =, MOD, DIV a definujte rekurzívne NSD.  
Vypočítajte NSD 18 24.

$B = \lambda xyz.x (yz)$     B-redukcia  $B f g x = f (g x)$   
 $C = \lambda xyz.(x z) y$     C-redukcia  $C f g x = (f x) g$

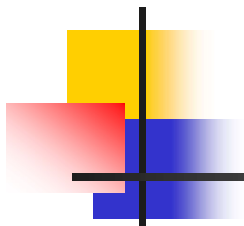
[http://en.wikipedia.org/wiki/Combinatory\\_logic](http://en.wikipedia.org/wiki/Combinatory_logic))

(**N** x) => True, if x ak má normálnu formu [True = K]  
F, inak. [False = (K I)]

$$(S\ I\ I\ Z) \rightarrow_S$$
$$((I \ Z) \ (I \ Z)) \rightarrow_I$$
$$(Z(IZ)) \rightarrow I$$
$$(Z\ Z) \rightarrow_Z$$
$$(C \text{ (C (B N (S I I)) } \Omega) \text{ I Z}) \rightarrow_C$$
$$(((C \text{ (B N (S I I)) } \Omega) \text{ Z) I}) \rightarrow_C$$
$$(((B \text{ N } (S \text{ I } I) Z) \Omega) I) \rightarrow_B$$
$$((N(S I I Z)) \Omega I)$$

Ak  $N(S I I Z) = \text{True}$  (**má n.f.**)  
tak výsledok je True  $\Omega I \rightarrow \Omega \rightarrow \Omega \rightarrow \Omega$   
**teda nemá n.f.**, preto SPOR.

Ak  $N(S I I Z) = \text{False}$  (**nemá n.f.**)  
tak výsledok je  $(K I) \Omega I \rightarrow I$ ,  
**teda. nemá n.f.**, preto SPOR





# Super-kombinátor

---

Uzavretý term  $\lambda x_1 \lambda x_2 \dots \lambda x_n. E$  je super-kombinátor, ak

- $E$  nie je  $\lambda$ -abstrakcia
- všetky  $\lambda$ -abstrakcie v  $E$  sú super-kombinátory

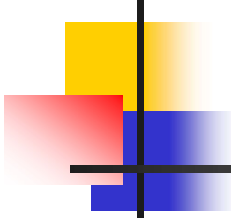
Príklady:

- $\lambda x. x$
- $\lambda x. \lambda y. (- x y)$
- $\lambda f. (f \lambda x. (* x x))$

Príklady termov, ktoré nie sú super-kombinátory:

- $\lambda x. y$  - nie je uzavretý
- $\lambda f. (f \lambda x. (f x x))$  - nie je  $\lambda x. (f x x)$  super-kombinátor
- $\lambda x. (x (\lambda y. y (\lambda z. z y)))$  - nie je  $(\lambda z. z y)$  super-kombinátor

# Transformácia termu do super-kombinátorov



- $\lambda x. (\lambda y. (+ y x) x) 4$

$\lambda y. (+ y x)$  nie je super-kombinátor, ale (po  $\eta$ -konverzii) dostaneme

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

( $\lambda$ -lifting)

a  $R = (\lambda x. \lambda y. (+ y x))$  je super-kombinátor, dosadíme  $R x$ , teda

$$\lambda x. ((R x) x) 4$$

$Q = \lambda x. (R x x)$  je super-kombinátor, takže celý program zredukujeme

$$Q 4$$

pričom

$$Q = \lambda x. (R x x)$$

$$R = (\lambda x. \lambda y. (+ y x))$$



# Rekurzia

---

Opäť ale máme problém s rekurziou:

1) riešenie pomocou Y operátora:

- $f\ x = g\ (f\ (x-1))\ 0$
- $f = \lambda F. \lambda x. (g\ (F\ (x-1))\ 0)\ f$
- $f = Y\ (\lambda F. \lambda x. (g\ (F\ (x-1))\ 0))$

2) riešenie pomocou rekurzívnych super-kombinátorov



# Lifting lambda

---

```
sumInts m      = suma (count 1) where
    count n | n > m  = []    -- lokálna definícia funkcie
    count n | otherwise      = n : count (n+1)
```

```
suma []        = 0
```

```
suma (n:ns)    = n + suma ns
```

```
----- zavedieme let-in
```

```
sumInts' m     =
```

```
    let count' n = if n > m then [] else n : count' (n+1)
```

```
    in  suma' (count' 1)
```

```
suma' ns = if ns == [] then 0 else (head ns) + suma' (tail ns)
```

```
----- prehodíme definíciu suma dovnútra
```





# Lifting lambda / 1

---

```
sumInts'' m =  
  let  
    count'' n = if n > m then [] else n : count'' (n+1)  
    suma'' ns =  
      if ns == [] then 0 else (head ns) + suma'' (tail ns)  
  in  
    suma'' (count'' 1)
```

----- zavedieme  $\lambda$  abstrakcie

```
sumInts''' m =  
  let  
    count''' = \n->if n > m then [] else n : count''' (n+1)  
    suma''' = \ns->if ns == [] then 0  
      else (head ns)+suma''' (tail ns)  
  in suma''' (count''' 1)
```

----- count''' nie je super-kombinátor



# Lifting lambda / 2

-----  $\lambda$ -lifting

```
sumInts'''' m =  
  let  
    count = \c->\m->\n->if n > m then [] else n:c (n+1)  
    count'''' = count count'''' m          -- rekurzia  
    suma'''' = \ns -> if ns == [] then 0  
                  else (head ns) + suma'''' (tail ns)  
  in suma'''' (count'''' 1)
```

----- definícia pomocou super-kombinátorov

```
sumInts'''''' m =  
  let  
    sc1 = \c->\m->\n->if n > m then [] else n : c (n+1)  
    sc2 = \ns->if ns == [] then 0 else (head ns) + sc2 (tail ns)  
    sc3 = sc1 sc3 m -- rekurzia  
  in sc2 (sc3 1)
```



$$X = \lambda x. (x K S K)$$

Nech  $X = \lambda x. (x K S K)$   
 potom vieme ukázať, že  $K = X X X = (X X)$

A tiež, že  $S = X . X X = X (X X)$

■  $K = X X X = (X X) X$

$$\begin{aligned} (X X) X &= (\lambda x. (x K S K) X) X \\ &= (X K S K) X \\ &= (\lambda x. (x K S K) K S K) X \\ &= ((\underline{K K S K}) S K) X \\ &= ((\underline{K K}) \underline{S K}) X = \underline{K K X} \\ &= K \end{aligned}$$

■  $S = X . X X = X (X X)$

$$\begin{aligned} X (X X) &= (\lambda x. (x K S K)) (X X) \\ &= ((X X) K S K) \\ &= (((\lambda x. (x K S K) X) K S K) \\ &= ((X K S K) K S K) \\ &= (((\lambda x. (x K S K)) K S K) K S K) \\ &= (((\underline{K K S K}) S K) K S K) \\ &= (((\underline{K K}) \underline{S K}) K S K) \\ &= ((\underline{K K}) \underline{K S K}) \\ &= (K S K) \\ &= S \end{aligned}$$