Bulova logika

$T = (\lambda\, x\, .\lambda\, y\, .x)$

$F = (\lambda\, x\, .\lambda\, y\, .y)$

- Nije obavezno da tačno i netačno budu definisani na ovaj način

$and = \ (\lambda\, a\, .\lambda\, b\, .a\ b\ F)$

- $and\ T\ T$
- $\big(\lambda\, a.\lambda\, b\, .a\ b\ (\lambda\, x\, .\lambda\, y\, .y)\big)\ T\ T$

and T T

$$\left(\lambda\, a.\lambda\, b\,.a\; b\; (\lambda\, x\,.\lambda\, y\,.y)\right)\; (\lambda\, x\,.\lambda\, y\,.x)\; (\lambda\, x\,.\lambda\, y\,.x)$$

$$\left(\lambda\, b\,.a\; b\; (\lambda\, x\,.\lambda\, y\,.y)\right)\; [a\; \rightarrow (\lambda\, x\,.\lambda\, y\,.x)]\; (\lambda\, x\,.\lambda\, y\,.x)$$

$$\left(\lambda\, b\,.(\lambda\, x\,.\lambda\, y\,.x)\; b\; (\lambda\, x\,.\lambda\, y\,.y)\right)\; (\lambda\, x\,.\lambda\, y\,.x)$$

$$\left((\lambda\, x\,.\lambda\, y\,.x)\; b\; (\lambda\, x\,.\lambda\, y\,.y)\right)\; [b\; \rightarrow (\lambda\, x\,.\lambda\, y\,.x)]$$

$$(\lambda\, x\,.\lambda\, y\,.x)(\lambda\, x\,.\lambda\, y\,.\; x)\; (\lambda\, x\,.\lambda\, y\,.y)$$

$$(\lambda\, y\,.x)[x\; \rightarrow (\lambda\, x\,.\lambda\, y\,.\; x)]\; (\lambda\, x\,.\lambda\, y\,.y)$$

$$\left(\lambda\, y\,.(\lambda\, x\,.\lambda\, y\,.\; x)\right)\; (\lambda\, x\,.\lambda\, y\,.y)$$

$$(\lambda\, x\,.\lambda\, y\,.\; x)\; [y \rightarrow\; (\lambda\, x\,.\lambda\, y\,.y)]$$

$$(\lambda\, x\,.\lambda\, y\,.\; x)$$

$$T$$

and T F

$(\lambda\, a.\lambda\, b\, .a\; b\; F)\; T\; F$

$(\lambda\, b\, .a\; b\; F)\; [a\; \rightarrow T]\; F$

$(\lambda\, b\, .T\; b\; F)\; F$

$(T\; b\; F)[b\; \rightarrow F]$

$(T\; F\; F)$

$(\lambda\, x\, .\lambda\, y\, .x)\; F\; F$

$(\lambda\, y\, .x)\; [x\; \rightarrow F]\; F$

$(\lambda\, y\, .F)\;\; F$

$F\; [y \rightarrow F]$

F

$and\ F\ T$

$(\lambda\ a.\lambda\ b\ .a\ b\ F)\ F\ T$

$(F\ T\ F)$

F

$and\ F\ F$

$(\lambda\ a.\lambda\ b\ .a\ b\ F)\ F\ F$

$(F\ F\ F)$

F

not

- $not\ T = F$
- $not\ F = T$

$not = (\lambda\ a\ .\ a\ F\ T)$

$not\ T$

- $(\lambda\ a\ .\ a\ F\ T)\ T$
- $T\ F\ T$
- F

$not\ F$

- $(\lambda\ a\ .\ a\ F\ T)\ F$
- $F\ F\ T$
- T

IF grananje

```
if c then
    a
else
    b
```

$if\ c\ a\ b$

$if\ T\ a\ b = a$

$if\ F\ a\ b = b$

$if = (\lambda\, a\, .\, a)$

Primeri

$if\ T\ a\ b$

- $(\lambda\ a\ .a)\ T\ a\ b$
- $T\ a\ b$
- a

$if\ F\ a\ b$

- $(\lambda\ a\ .a)\ F\ a\ b$
- $F\ a\ b$
- b

Čerčovi numerali

$0 = \lambda\, f\, .\lambda\, x\, .\, x$

$1 = \lambda\, f\, .\lambda\, x\, .\, f\ x$

$2 = \lambda\, f\, .\lambda\, x\, .\ f\ f\ x$

$3 = \lambda\, f\, .\lambda\, x\, .\, f\ f\ f\ x$

$4 = \lambda\, f\, .\lambda\, x\, .\, f\ f\ f\ f\ x$

- $\lambda\, f\, .\lambda\, x\, .\left(f\ \left(f\ \left(f\ \left(f\ x\right)\right)\right)\right)$

$4\ a\ b$

- $a\ a\ a\ a\ b$

Funkcija naslednika

$succ = \lambda\, n\, .\lambda\, f\, .\lambda\, x\, .f\ (n\ f\ x)$

$0 = \lambda\, f\, .\lambda\, x\, .x$

$succ\ 0$

- $\lambda\, n\, .\lambda\, f\, .\lambda\, x\, .f\ (n\ f\ x)\ 0$
- $\lambda\, f\, .\lambda\, x\, .f\ (0\ f\ x)$
- $\lambda\, f\, .\lambda\, x\, .f\ \big((\lambda\, f\, .\lambda\, x\, .x)\ f\ x\big)$
- $\lambda\, f\, .\lambda\, x\, .f\ x$
- 1

Funkcija naslednika

$succ = \lambda n . \lambda f . \lambda x . f\ (n\ f\ x)$

$1 = \lambda f . \lambda x .\ f\ x$

$succ\ 1$

- $\lambda n . \lambda f . \lambda x . f\ (n\ f\ x)\ 1$
- $\lambda f . \lambda x . f\ (1\ f\ x)$
- $\lambda f . \lambda x . f\ \left((\lambda f . \lambda x .\ f\ x)\ f\ x\right)$
- $\lambda f . \lambda x . f\ f\ x$

$2 = \lambda f . \lambda x . f\ f\ x$

$succ\ 1 = 2$

$succ\ n = n + 1$

Sabiranje

$add\ 0\ 1 = 1$

$add\ 1\ 2 = 3$

$add = \lambda\, n\, .\lambda\, m\, .\lambda\, f\ .\ \lambda\, x\ .\ n\ f\ (m\ f\ x)$

$add\ 0\ 1$

- $(\lambda\, n\, .\lambda\, m\, .\lambda\, f\ .\ \lambda\, x\ .\ n\ f\ (m\ f\ x)\ )\ 0\ 1$
- $(\lambda\, m\, .\lambda\, f\ .\ \lambda\, x\ .\ 0\ f\ (m\ f\ x)\ )\ 1$
- $\lambda\, f\ .\ \lambda\, x\ .\ 0\ f\ (1\ f\ x)$
- $\lambda\, f\ .\ \lambda\, x\ .\ 0\ f\ (f\ x)$
- $\lambda\, f\ .\ \lambda\, x\ .\ f\ x$

Množenje

$mult\ 0\ 1 = 0$

$mult\ 1\ 2 = 2$

$mult\ 2\ 5 = 10$

$mult = \lambda\, n\, .\lambda\, m\, .m\ (add\ n)\ 0$

$mult\ 0\ 1$

- $(\lambda\, n\, .\lambda\, m\, .m\ (add\ n)\ 0)\ 0\ 1$
- $(\lambda\, m\, .m\ (add\ 0)\ 0)\ 1$
- $1\ (add\ 0)\ 0$
- $add\ 0\ 0$
- 0

Množenje

$mult\ 1\ 2$

- $(\lambda\, n\, .\lambda\, m\, .m\ (add\ n)\ 0)\ 1\ 2$
- $(\lambda\, m\, .m\ (add\ 1)\ 0)\ 2$
- $2\ (add\ 0)\ 0$
- $(add\ 1)\ \big((add\ 1)\ 0\big)$
- $(add\ 1)\ (add\ 1\ 0)$
- $(add\ 1)\ (1)$

Faktorijel

$n!$

- fact(0) = 1
- fact(n) = n * fakt(n-1)

$$fact = \left(\lambda\, n\, .if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(fact\ (pred\ n)\right)\right)\right)$$

- Funkciju ne možemo definisati ovako!

```
int fact(int n)
{
    if (n==0)
        return 1;
    return n * fact(n-1);
}
```

Y kombinator

$$Y = \left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)$$

$Y\ foo$

- $\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\ foo$
- $\left(\lambda\, y\, .y\ \left(\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\ y\right)\right)\ foo$
- $foo\ \left(\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\left(\lambda\, x\, .\lambda\, y\, .y\ (x\ x\ y)\right)\ foo\right)$
- $foo\ (Y\ foo)$
- $foo\ \left(foo\ (Y\ foo)\right)$
- $foo\ \left(foo\ \left(foo\ (Y\ foo)\right)\right)$
- …

Rekurzija

- $fact = \left(\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(fact\ (pred\ n)\right)\right)\right)$

$fact = Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)$

$fact\ 1$

- $Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\ 1$
- $\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\left(Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)\ 1$
- $\left(\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ n)\right)\right)\right)\ 1$
- $if\ (iszero\ 1)\ (1)\ \left(mult\ 1\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 1)\right)\right)$
- $if\ F\ (1)\ \left(mult\ 1\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 1)\right)\right)$
- $mult\ 1\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.\, if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 1)\right)$

Rekurzija

$fact\ 1$

- $mult\ 1\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 1)\right)$
- $mult\ 1\ \left(\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 1)\right)$
- $mult\ 1\ \left(\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ n)\right)\right)\right)(pred\ 1)$
- $mult\ 1\ \left(\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ n)\right)\right)\right)\ 0$
- $mult\ 1\ \left(if\ (iszero\ 0)\ (1)\ \left(mult\ 0\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 0)\right)\right)\right)$
- $mult\ 1\ \left(if\ T\ (1)\ \left(mult\ 0\ \left(\left(Y\left(\lambda\, f.\lambda\, n\,.if\ (iszero\ n)\ (1)\ \left(mult\ n\ \left(f\ (pred\ n)\right)\right)\right)\right)(pred\ 0)\right)\right)\right)$
- $mult\ 1\ 1$
- 1

Tjuring-kompletan sistem

Bulova logika

Aritmetika

Petlje

Haskell

GRAHAM HUTTON, **PROGRAMMING IN HASKELL**, CAMBRIDGE UNIVERSITY PRESS (2007)

Haskell kao kalkulator

```
Prelude> 2+3*4
14
Prelude> (2+3)*4
20
Prelude> sqrt(3^2 + 4^2)
5.0
Prelude> 2^1000
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682
518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439
460770629145711964776865421676604298316526243868372056680693 76
Prelude>
```

```
Prelude> (+) 2 3
5
Prelude> (^) 2 5
32
```

Standardne funkcije

Prvi element liste

```
Prelude> head [1, 2, 3, 4]
1
```

Svi elementi sem prvog

```
Prelude> tail [1, 2, 3, 4]
[2,3,4]
```

N-ti element liste

```
Prelude> [1, 2, 3, 4] !! 2
3
```

Prvih n elemenata liste

```
Prelude> take 3 [1, 2, 3, 4]
[1,2,3]
```

Standardne funkcije

Uklanjanja prvih n elemenata liste

```
Prelude> drop 3 [1, 2, 3, 4]
[4]
```

```
xs = take n xs ++ drop n xs
```

Dužina liste

```
Prelude> length [1, 2, 3, 4]
4
```

Suma elemenata liste

```
Prelude> sum [1, 2, 3, 4]
10
```

Standardne funkcije

Proizvod elemenata liste

```
Prelude> product [1, 2, 3, 4]
24
```

Nadovezivanje lista

```
Prelude> [1, 2, 3] ++ [4, 5]
[1,2,3,4,5]
```

Inverzna lista

```
Prelude> reverse [1, 2, 3, 4]
[4,3,2,1]
```

Funkcijsko vs. objektno programiranje

drop 3 [1, 2, 3, 4, 5] vs. [1, 2, 3, 4, 5].drop(3)

method receiver a b c vs. receiver.method(a,b,c)

OO programiranje lakše „teče“ – metode objekta navode na način rešavanja

vs.

U funkcijskom programiranju programer preuzima inicijativu

Svi argumenti su jednko važni

Matematika vs. funkcijsko programiranje

$f(a, b) + c\ d$ vs. `f a b + c * d`

- Primeniti funkciju f na argumente a i b i dodati proizvod $c\ d$
- Najčešće korišćen simbol zauzima najmanje mesta -- proizvod vs. aplikacija
- Prioritet operacija -- proizvod vs. aplikacija

```
ghci> let f x = x + 3
ghci> f 5 * 7
56
```

Elegentna sintaksa!

Math	Haskell
f(x)	`f x`
f(x,y)	`f x y`
f(g(x))	`f (g x)`
f(x, g(y))	`f x (g x)`
f(x)g(x)	`f x * g x`

Zagrade mogu dodatno da se izbegnu korišćenjem $ za definisanje desne asocijativnosti

Skript fajl

```
double x = 2 * x
quadruple x = double (double x)
```

```
double x = 2 * x
quadruple = double . double
```

```
ghci> :load Primeri01_1.hs
[1 of 1] Compiling Main             ( Primeri01_1.hs, interpreted )
Ok, one module loaded.
ghci> quadruple 2
8
ghci> :type quadruple
quadruple :: Integer -> Integer
ghci> take (double 3) [1..15]
[1,2,3,4,5,6]
```

Kako u C++/C# izgleda definicija i testiranje ovakvog koda?

Glavni delovi koda i „šum“

Skript fajl

```
faktor n = product [1 .. n]
```

- U proceduralnim jezicima petlja ili rekurzija
- 1*2*…*n
- [1 .. n]
- [1 ..]
- take n [1 ..]

```
faktor n = product (take n [1 .. ])
```

```
average ns = sum ns `div` length ns
```

- Korišćenje funkcija kao infiksnih operatora

```
average ns = div (sum ns) (length ns)
```

Imenovanje

Nazivi funkcija i argumenata počinju malim slovom

Imena tipova počinju velikim slovom

Konvencija

- n – broj
- xs – lista
- xss – lista lista

Izgled

```
a = 5
b = 10
c = 30
```

```
a = 5
 b = 10
c = 30
```

```
 a = 5
b = 10
  c = 30
```

Izbegava se pisanje dodatnih simbola za ograničavanja blokova

Tipovi i klase

Tipovi

Tip je ime za kolekciju povezanih vrednosti

Osnovni tipovi

- `Bool = False | True`

```
Prelude> 1 + False

<interactive>:16:1: error:
    * No instance for (Num Bool) arising from a use of `+'
    * In the expression: 1 + False
      In an equation for `it': it = 1 + False
```

- `Char = 'a' | 'b' | ... | 'A' | 'B' | ...`
- `String`
- `Int = -2^31 | ... | -1 | 0 | 1 | ... | 2^31-1`
- `Integer` (neograničen tip, proizvoljno velike vrednosti)
- `Float`
- `Double`

Tipovi

```
exp :: Type
```

```
ghci> [1,2,3] :: [Int]
[1,2,3]
ghci> :type [1,2,3]
[1,2,3] :: Num a => [a]
ghci> :type [['a'],['b','c']]
[['a'],['b','c']] :: [[Char]]
ghci> :type ('a', True, 1)
('a', True, 1) :: Num c => (Char, Bool, c)
ghci> :type ('a',(False, "abc"))
('a',(False, "abc")) :: (Char, (Bool, String))
ghci>  :type [1, 2.2, 3]
[1, 2.2, 3] :: Fractional a => [a]
```

Definisanje tipova nije obavezno

Može donekle da ubrza kod

Tipovi lista

Lista je niz elemenata istog tipa

- `[False, True, False] :: [Bool]`
- `['a', 'b', 'c', 'd'] :: [Char]`

`[t]` je lista elemenata tipa `t`

Ista notacija za konstruktor tipa i konstruktor vrednosti

Tip liste ništa ne govori o dužini liste

`[[[Char]]] ?`

Tipovi torki

Torka je niz vrednosti različitog tipa

- `(False, True) :: (Bool, Bool)`
- `(False, 'a', True) :: (Bool, Char, Bool)`
- `(1, True, 'a') :: (Int, Bool, Char)`

`(t1, t2, …, tn)` je n-torka čija i-ta komponenta ima tip ti za svako i u [1..n]

Ista notacija za konstruktor tipa i konstruktor vrednosti

Definica tipa defineše i dužinu/veličinu

Ne postoji ograničenje za tip komponente

- `('a', (False, 'b')) :: (Char, (Bool,Char))`
- `(True, ['a', 'b']) :: (Bool, [Char])`