

λ -račun

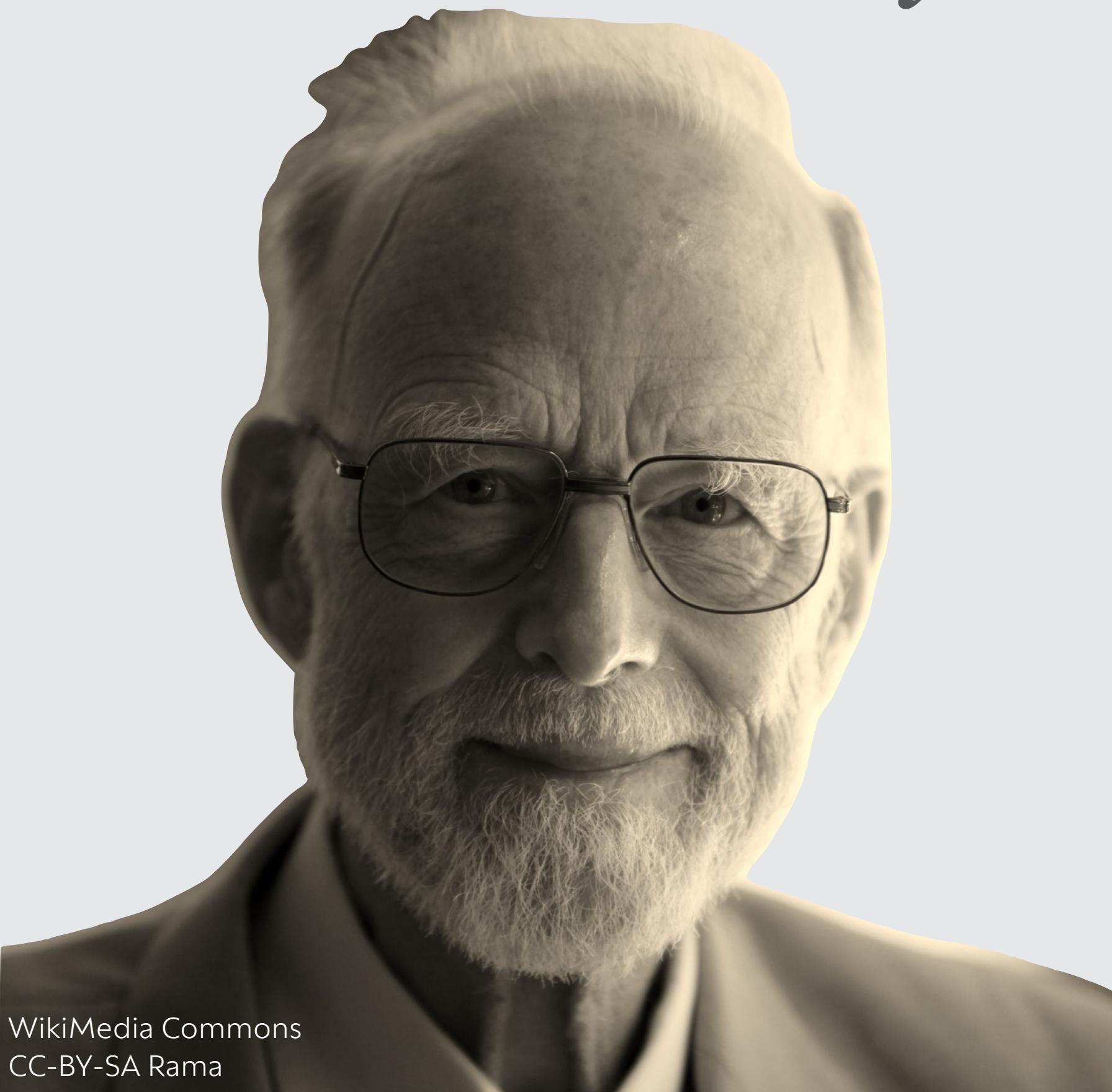
prejšnjič...

Spoznali ste **Hoareovo logiko** za dokazovanje lastnosti programov

$$\{x = m \wedge y = n\}$$

if $x < y$ then ($z := x$) else ($z := y$) end

$$\{z = \min(m, n)\}$$



WikiMedia Commons
CC-BY-SA Rama

Charles Anthony Richard Hoare, 1934–



WikiMedia Commons
CC-BY-SA Andrej Bauer

Funkcijski predpis

Navajeni smo, da ima **vsaka funkcija ime**

$$\sin \frac{\pi}{2} \quad \log_2 1024$$

$$d(x, y) = \sqrt{x^2 + y^2}$$

$$p(x) = x^2 + 3x + 7$$

Nerodno je, če bi morali poimenovati vsako vrednost

$$a = 1$$

$$b = 5$$

$$c = 3$$

$$(1 + 5) \cdot (3 + 4)$$

$$d = 4$$

$$e = a + b$$

$$f = c + d$$

$$e \cdot f$$

Anonimne funkcije podajamo s funkcijskimi predpisi

$$p(x) = x^2 + 3x + 7$$

$$p(3) = 25$$

$$x \mapsto x^2 + 3x + 7$$

$$(x \mapsto x^2 + 3x + 7)(3) = 25$$

Za funkcijске предпise познамо **разлиčне записи**

$$x \mapsto x^2 + 3x + 7$$

$$\lambda x . x^2 + 3x + 7$$

Python `lambda x: x ** 2 + 3 * x + 7`

Haskell `\x -> x ** 2 + 3 * x + 7`

OCaml `fun x -> x * x + 3 * x + 7`

Java `(x) -> x * x + 3 * x + 7`

C++ `[](int x) { return x * x + 3 * x + 7; }`

Spremenljivke v funkcijskih predpisih so lahko **proste** ali **vezane**

prištej a^2

$$\underline{x} \mapsto \underline{x} + \underline{a^2}$$

$$\underline{a} \mapsto \underline{x} + \underline{a^2}$$

kvadriraj in prištej x

Proste in vezane spremenljivke drugje v matematiki

$$\int \underline{x}^2 + \underline{a} \cdot \underline{x} dx$$

$$\sum_{i=1}^n i \cdot (i+k)$$

$$\lim_{x \rightarrow a} \frac{\underline{x} - \underline{a}}{\underline{x} + \underline{a}}$$

$$\exists x \in \mathbb{R}. \underline{x}^3 = \underline{y}$$

Proste in vezane spremenljivke drugje v računalništvu

```
for (int i = 0; i < 10; i++) {  
    s += i;  
}
```

```
if (false) {  
    int s = 0 ;  
    for (int i = 0; i < 10; i++) {  
        s += i;  
    }  
}
```

Vezane spremenljivke lahko **vedno preimenujemo**

$$x \mapsto x + a^2 \quad \text{prištej } a^2$$

=

$$y \mapsto y + a^2 \quad \text{prištej } a^2$$

Paziti moramo, **da ponesreči ne ujamemo** proste spremenljivke

$$x \mapsto x + a^2 \quad \text{prištej } a^2$$

=

$$y \mapsto y + a^2 \quad \text{prištej } a^2$$

≠

$$a \mapsto a + a^2$$

Vsako prosto spremenljivko lahko **substituiramo** oz. **zamenjamo**

$$e[e'/x]$$

ve vse pojavitve x zamenjaj z e'

$$(x^2 + 3 \cdot x + 7)[3/x] = 3^2 + 3 \cdot 3 + 7$$

$$f(a + b)[(b + 1)/a] = f((b + 1) + b)$$

$$f(a + b)[(x \mapsto x^2)/f] = (x \mapsto x^2)(a + b)$$

Pri substituciji moramo paziti, da česa **ponesreči** ne ujamemo

$$\left(\int_0^1 kx \, dx \right) [x/k] \neq \int_0^1 x^2 \, dx$$

Dogovorjeni smo, da **po potrebi** vezane spremenljivke **preimenujemo**

$$\left(\int_0^1 kx \, dx \right) [x/k] = \left(\int_0^1 ky \, dy \right) [x/k]$$
$$= \int_0^1 xy \, dy$$

Funkcijske predpise apliciramo z β -redukcijo

$$(x \mapsto e_1)(e_2) = e_1[e_2/x]$$

$$(x \mapsto x^2 + 3 \cdot x + 7)(3) =$$

$$(x^2 + 3 \cdot x + 7)[3/x] =$$

$$3^2 + 3 \cdot 3 + 7 = 25$$

$$(x \mapsto (y \mapsto x \cdot x + y))(42)$$
$$((x \mapsto (y \mapsto x \cdot x + y))(42))(1)$$
$$(f \mapsto f(f(3))) (n \mapsto n + 1)$$

λ -račun

Funkcijske predpise bomo pisali z λ -abstrakcijami

$\lambda x.e$

Alonzo Church, 1903–1995



λ -račun vsebuje samo tri vrste izrazov

izraz $e ::= x \mid \lambda x . e \mid e_1 e_2$

V gnezdenih abstakcijah bomo **združevali** spremenljivke

$$\lambda x y z . e$$
$$=$$
$$\lambda x . \lambda y . \lambda z . e$$

Aplikacijo bomo pisali **levo asociativno**

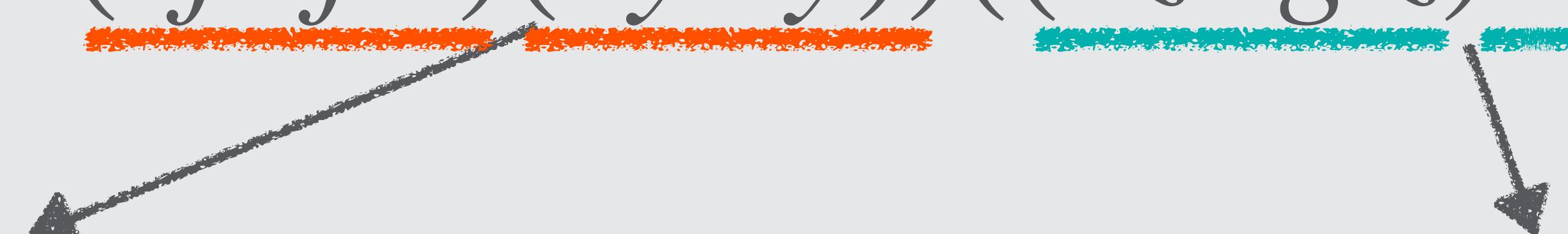
$$\begin{aligned} & e_1 \ e_2 \ e_3 \ e_4 \\ & = \\ & ((e_1 \ e_2) \ e_3) \ e_4 \end{aligned}$$

β -redukcijo lahko izvedemo na različnih mestih

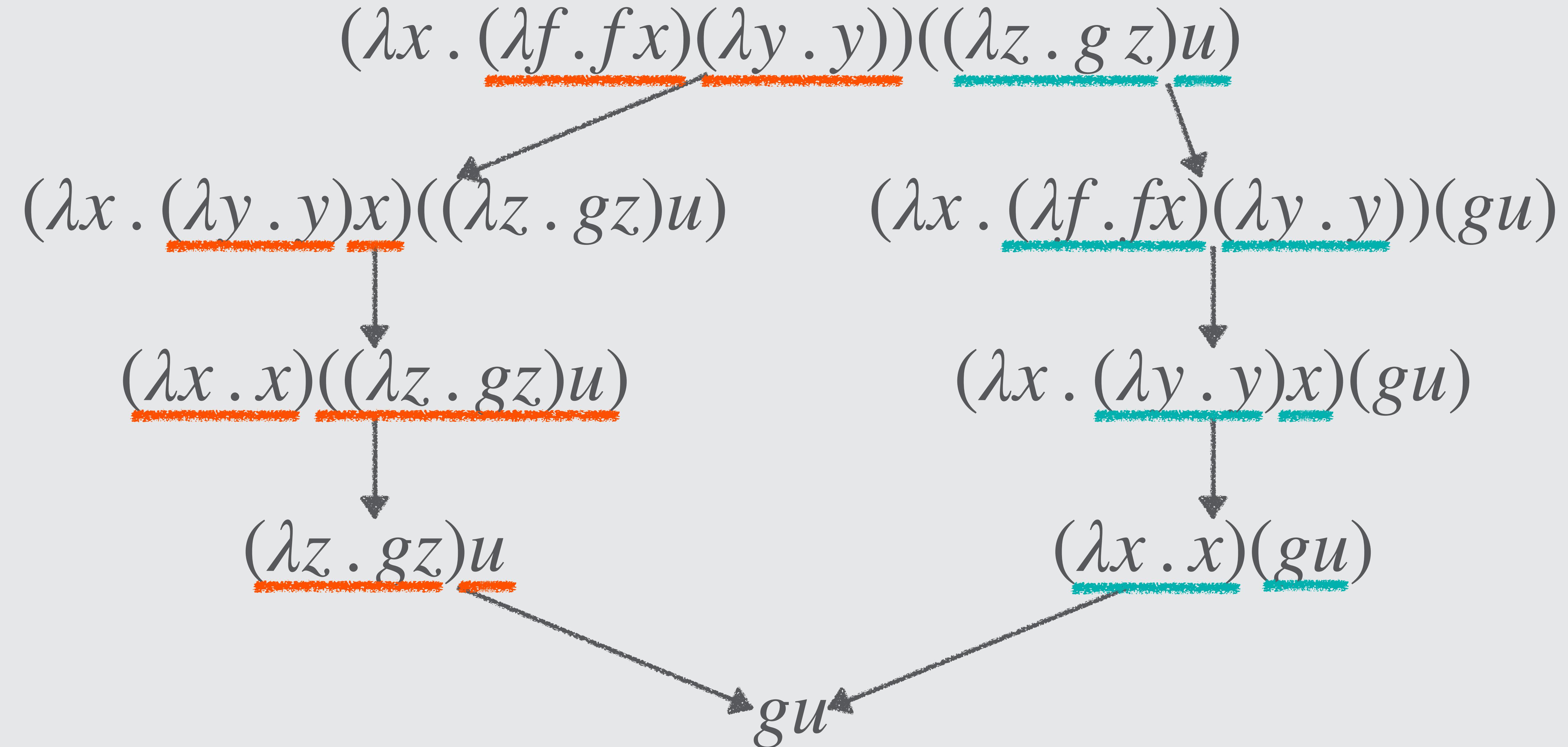
$$(\lambda x . (\lambda f . fx)(\lambda y . y))((\lambda z . g z)u)$$

$$(\lambda x . (\lambda y . y)x)((\lambda z . g z)u)$$

$$(\lambda x . (\lambda f . fx)(\lambda y . y))(gu)$$



Kljub temu je λ -račun **konfluenten**



Obstaja več strategij izvajanja

neučakana (eager/call-by-value/CBV)

pred klicem funkcije argument izračunamo do konca

$$(\lambda x . x \cdot x)(1 + 2) \mapsto (\lambda x . x \cdot x)3 \mapsto 3 \cdot 3 \mapsto 9$$

lena (lazy/call-by-name/CBN)

argument izračunamo po potrebi

$$(\lambda x . x \cdot x)(1 + 2) \mapsto (1 + 2) \cdot (1 + 2) \mapsto 3 \cdot (1 + 2) \mapsto 3 \cdot 3 \mapsto 9$$

jezik
tambda

Pravila za neučakano semantiko malih korakov

$$e \mapsto e'$$

$$\frac{e_1 \mapsto e'_1}{e_1 e_2 \mapsto e'_1 e_2}$$

$$\frac{e_2 \mapsto e'_2}{v e_2 \mapsto v e'_2}$$

$$\frac{}{(\lambda x . e) v \mapsto e[v/x]}$$

Pravila za **leno** semantiko malih korakov

$$e \mapsto e'$$

$$\frac{e_1 \mapsto e'_1}{e_1 e_2 \mapsto e'_1 e_2}$$

$$\overline{(\lambda x . e) e' \mapsto e[e'/x]}$$

Programiranje v λ -računu

Definicije identitete, kompozitura in konstantne preslikave

$$\text{id} = \lambda x . x$$

$$\text{compose} = \lambda f g x . f(g x)$$

$$\text{const} = \lambda c x . c$$

Naravna števila predstavimo s Churchevimi numerali

$$\underline{0} = \lambda f x . x$$

$$\underline{1} = \lambda f x . f x$$

$$\underline{2} = \lambda f x . f(f x)$$

$$\underline{n} = \lambda f x . f(f(\cdots(f x)\cdots))$$

$\underbrace{\hspace{2cm}}$

n

Alonzo Church, 1903–1995



succ + *

$$6 * 7 = 42$$

Definicija predhodnika je bolj zahtevna

$$\lambda n . \lambda f . \lambda x . n(\lambda g . \lambda h . h(gf))(\lambda u . x)(\lambda u . u)$$



Stephen Kleene, 1909–1994

Logične vrednosti ustrezano izbiri eni od dveh možnosti

$$\text{true} = \lambda x y . x$$

$$\text{false} = \lambda x y . y$$

$$\text{if} = \lambda b t e . b t e$$

and or not
is zero

Pari omogočajo dostop do dveh vrednosti

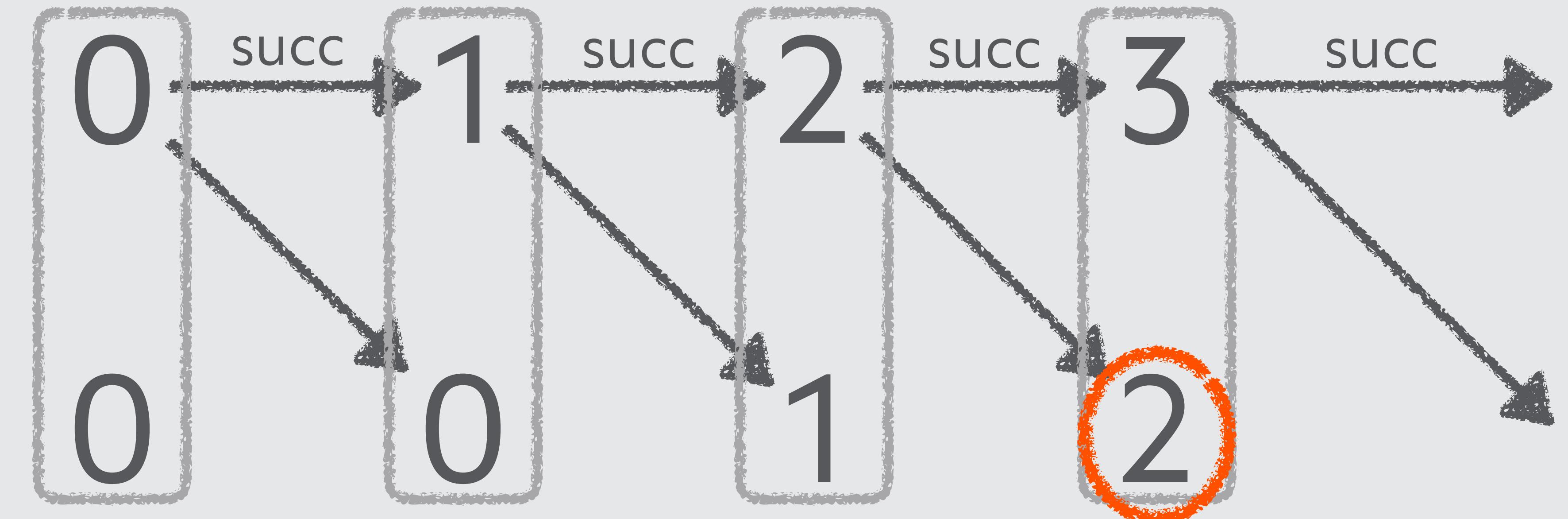
$\text{pair} = \lambda a b f. fab$

$\text{first} = \lambda p . p (\lambda x y . x)$

$\text{second} = \lambda p . p (\lambda x y . y)$

Predhodnika lahko izračunamo s pari

$\lambda n . \text{second}$



$$\left(n \left(\lambda p . \text{pair} (\text{succ} (\text{first } p)) (\text{first } p) \right) (\text{pair } \underline{0} \underline{0}) \right)$$

四
三
二

Rekurzijo dosežemo s **kombinatorjem Y**

$$\begin{aligned}Yf &= (fx\ x)\ (fx\ x) \\&= f((fx\ x)\ (fx\ x)) \\&= f(Yf) \\&= f(f(Yf)) \\&= f(f(f(Yf)))\end{aligned}$$

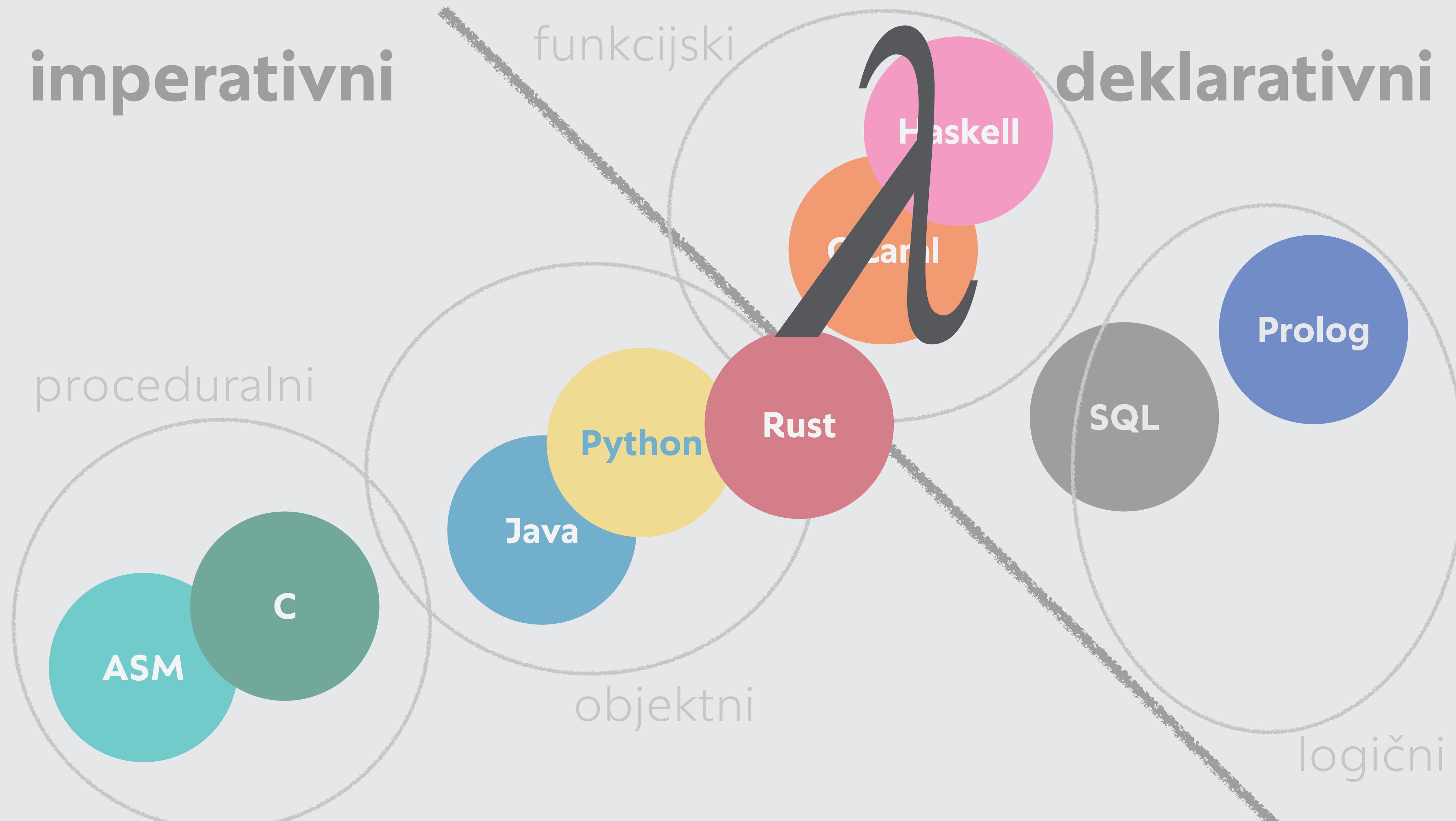
fact 5

prihodnjič...

Z imperativnega bomo prešli na deklarativno programiranje

imperativni

deklarativni



Spoznali bomo **konstrukcije na množicah**

$$A \times B$$

$$A + B$$

$$B^A$$

Začeli bomo delati s **programskim jezikom OCaml**



OCaml