Rekurzija

$rek : \underbrace{((\alpha \to \beta) \to (\alpha \to \beta))}_{telo} \to \underbrace{(\alpha \to \beta)}_{f}$

self: $(\alpha \to \beta)$ — $n \mapsto$ if $n = \ldots$: $(\alpha \to \beta)$

$f = rek\ telo$

```
let rec f n =
      if n = 0 then 1 else n * f (n - 1)
```

$f := \lambda n.\ \text{if } n=0 \text{ then } 1 \text{ else } n \cdot f(n-1)$

$f = t\ f$ kjer je $t = \lambda g.\lambda n.\ \text{if } n=0 \text{ then } 1 \text{ else } n \cdot g(n-1)$

Razprli smo rekurzijo

$f = rek\ t$

$f = t\ f$

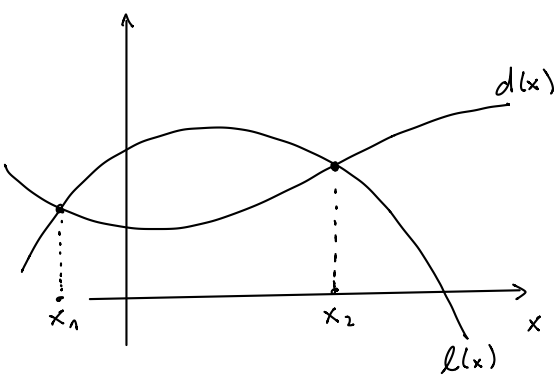
f je negibna točka za t

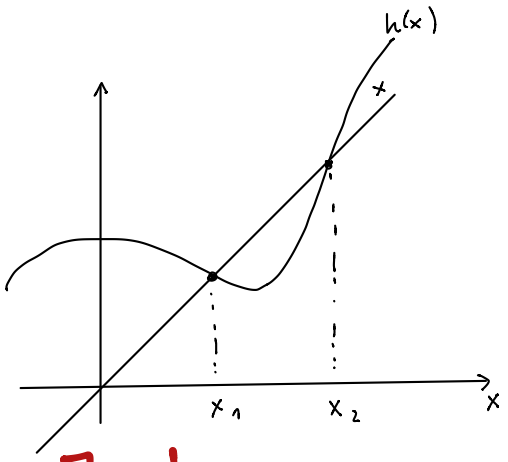
Negibna točka:

$l(x) = d(x)$ enačba npr. $\underbrace{x^2 - x + 1}_{l(x)} = \underbrace{x^3 \cdot \sin(x)}_{d(x)}$

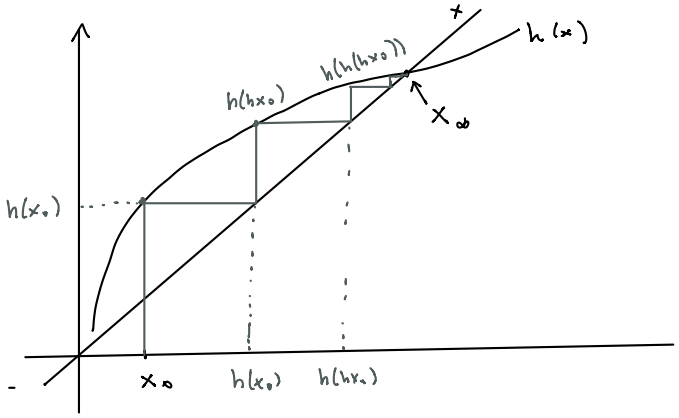
$x = h(x)$ npr. $x = x^2 + x^3$

x je negibna točka za h





Rekurzija je negibna točka!



$x_\infty = h(h(h(h(h(h \dots ( ) \dots)$

x_0

$x_1 = h(x_0)$

$x_2 = h(x_1) = h(h(x_0))$

$x_{n+1} = h(x_n)$

$x_\infty = \lim_{n \to \infty} x_n$

$S\ \widetilde{(n,k)} = \dots$

$S''\ n\ k = \dots$

$S : int \times int \longrightarrow int$

$S' : int \to int \to int$ } currying

λ-račun Alonzo Church → Dana Scott → Andrej Bauer → vi

Iteracija

Zanka kot negibna točka

(while b do c done) ≡ if b then (c; while b do c done) else skip end

(while b do c done) = W

(c; while b do c done) = W

$W \equiv$ if b then (c ; W) else skip end

$t := \lambda w'.$ if b then (c ; w') else skip end

$W \equiv t\ W \equiv t(t\ W) = t(t(t\ W))) \dots$

Faza 0: while

Rekurzivne podatkovne strukture

Primer: $\ell = [1; 2; 1; 2; \ldots]$ neskončen seznam

$\ell = 1 :: (2 :: \ell)$ rekurzija / negibna točka

Seznam: } rekurzivni podatkovni tip

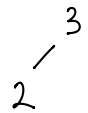
- prazen:
- sestavljen: $x :: \ell$ (x: podatek, ℓ: seznam; x: glava / head, ℓ: rep / tail)

LISP nil (prazen seznam)
cons (sestavljen seznam)
(cons x ℓ)

1. Induktivne strukture ← dopuščamo samo končne strukture
2. Koinduktivne strukture ← dopuščamo končne in neskončne strukture

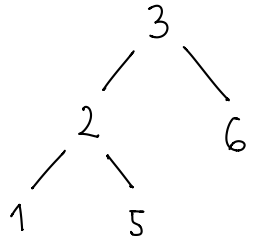
$\mathbb{N}$: 0 nič
Succ n naslednik

tree:



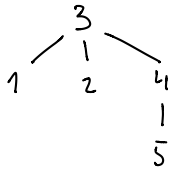
Tree(3, Tree(2, Empty, Empty), Empty)

Tip option: α option → None "ni vrednosti"
→ Some v "je vrednost v"



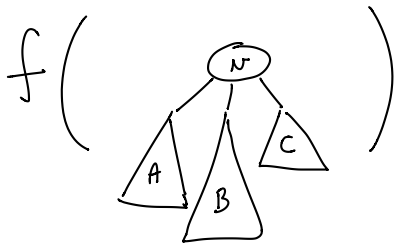

```
Tree'(3,
    Tree'(2,
        Tree'(1, None, None),
        Tree'(5, None, None)),
    Tree'(6, None, None))
```

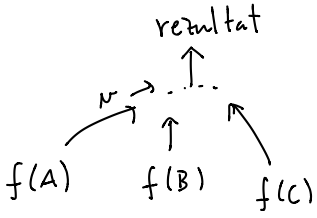
tree'''




```
Tree'''(3, [Tree(1,[]);
            Tree(2,[]);
            Tree(4,[Tree(5,[])])
           ]
)
```

Strukturna rekurzija

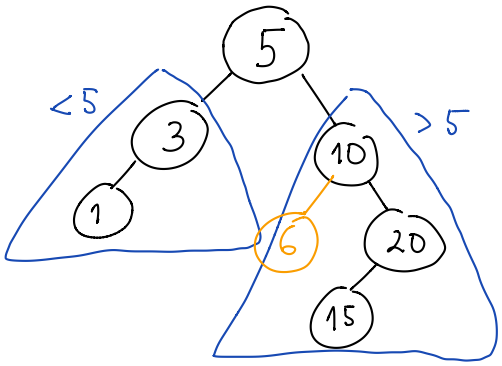


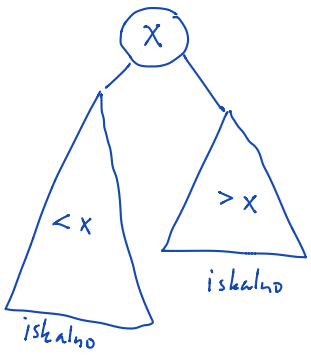


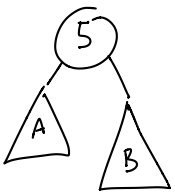
(fold)

"sprehod po drevesu"

Iskalno drevo





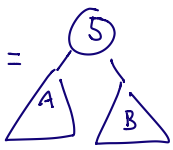


Node(y, l, r) as t →

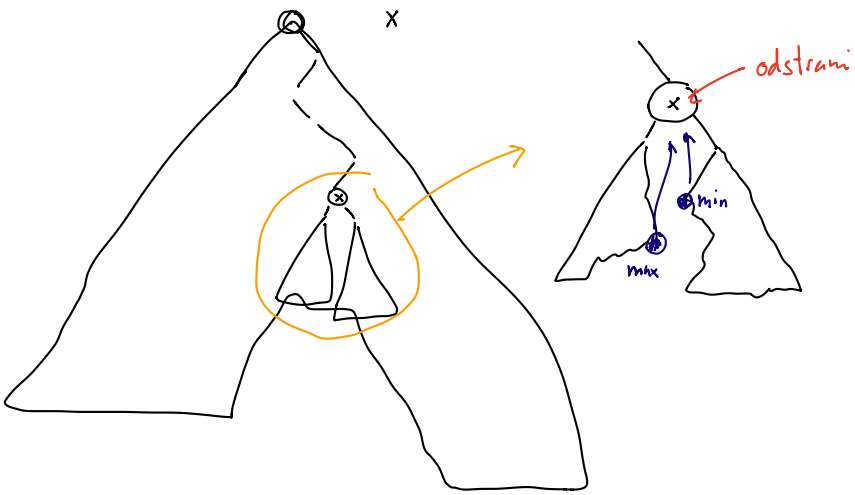
y = 5

l = 

r = 

t = 

Odstrani x iz drevesa:



Koinduktivni tipi (pod. strukture)

Podatkovni tok:

→ konec podatkov

→ sporočilo, kaznje še preostanek toka

Haskell:		množice	OCaml	Haskell	Java
	produkt	$A \times B$	`a * b`	`(A, B)`	simuliraj z objekti
	vsota	$A + B$	`type t = | Foo of a | Bar of b`	`data T = | Foo A | Bar B`	simuliraj z objekti
	induktivni		✓		simuliraj z objekti
	koinduktivni			✓	simuliraj z objekti

$$fib = 0 : 1 : (\underbrace{1 : 2 : 3 : 5 :}\ \text{zipWith}\ (+)\ fib\ (\text{tail}\ fib))$$

$0 : 1 : 1 : 2 : 3$ $\qquad$ $1 : 1 : 2 : 3$

Thunk:

$c' \equiv \underbrace{\text{fun}\ () \to c}_{\text{thunk}}$

$c'()$

```
def c'():
    c
```

$$\sum_{k=0}^{\infty} a_k \cdot x^k \qquad a_0, a_1, a_2, a_3, \ldots$$