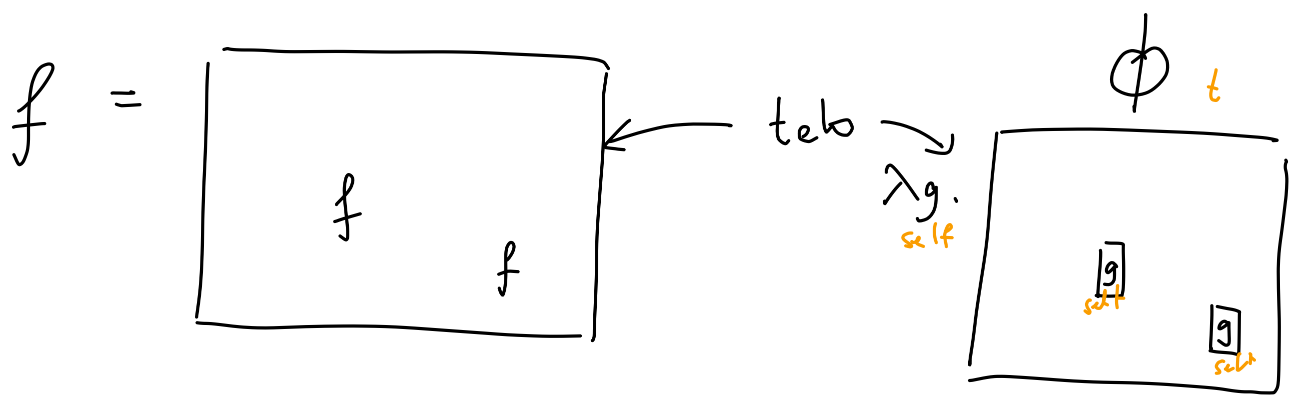
Rekurzija



$$f = \Phi\, f$$

$$\text{rek}\ t = \lambda x \to t\,(\text{rek}\ t)\ x$$

$$\text{rek}\ t = t\ (\text{rek}\ t)$$

$$\underbrace{(\overbrace{(\alpha \to \beta)}^{\text{self}} \to (\alpha \to \beta))}_{\text{red } 2} \xrightarrow{\text{red } 3} \underbrace{(\alpha \to \beta)}_{f}$$

t – telo definicije rekurzivne funkcije

f – rekurzivna funkcija, ki jo ustvari rek

int	red 0
int → int	red 1
(int → int) → int	red 2
((int → int) → int) → int	red 3

int → (int → int)

$\mathbb{R}$

$\mathbb{R} \to \mathbb{R}$

$(\mathbb{R} \to \mathbb{R}) \to \mathbb{R}$ $\quad \int_0^1$

Negibna točka za funkcijo h je x da velja $x = h(x)$

$$y'' + a\cdot y = 0 \Longrightarrow Dy = 0 \qquad D(z) = z'' + az$$

$$y'' + a\cdot y + y = y$$

$$y'' + (a+1)\,y = y \qquad \phi(z) := z'' + (a+1)z$$

$$\phi(y) = y$$

REKURZIJA = NEGIBNA TOČKA

funkcije
število
seznam

Rekurzivne definicije vrednosti, ki niso funkcije:

neskončen seznam

$$\ell = 1 :: 2 :: \ell \rightsquigarrow \ell = 1 :: 2 :: 1 :: 2 :: 1 :: 2 :: \cdots$$

$$\ell = t\ \ell \qquad t = \lambda s.\ 1 :: 2 :: s$$

$$\underset{\text{glava}}{x} :: \underset{\text{rep}}{\ell}$$

Seznam celih števil je:

1. Prazen []
2. Sestavljen $x :: \ell$ iz glave $x \in \mathbb{Z}$ in seznama ℓ

Rekurzivni tip: definicija seznamov se sklicuje na sezname

Ostali primeri:

- dvojiška drevesa:
  - prazno
  - sestavljeno iz korena in dveh dreves
- naravna števila:
  - nič 0
  - naslednik naravnega števila

Seznami števil kot negibna točka:

S množica seznamov $\qquad \mathbb{1} = \{*\}$

$$S = \underbrace{\mathbb{1}}_{\text{prazen}} + \mathbb{Z} \times S$$

$T : \mathrm{Set} \to \mathrm{Set}$

$T(X) := \mathbb{1} + \mathbb{Z} \times X$

$S = T(S)$ negibna točka za T

$$S = \mathbb{1} + \mathbb{Z} \times S$$

$\iota_1(*)$ prazen seznam

$\iota_2(42, \iota_1(*))$ $\qquad$ [42]

$\iota_2(13, \iota_2(42, \iota_1(*)))$ $\qquad$ [13, 42]

```
type seznam =
  | Nil
  | Cons of int * seznam
```

Ali dovolimo neskončne sezname:

Cons(1, Cons(2, Cons(3, ...)))

DA, dovolimo neskončne strukture

KOINDUKTIVNI TIPI

tok podatkov

NE, samo končno mnogo korakov

INDUKTIVNI TIPI

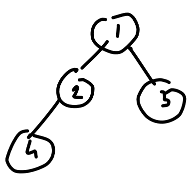
končni seznami
končna drevesa
naravna števila

Induktivni seznami = vsi končni seznami (neskončno jih je)
Koindukivni seznami = vsi končni in neskončni seznami (neskončno jih je)

Induktivna naravna števila: $0, 1, 2, \underset{\text{suc(suc(suc 0)))}}{\underset{\shortparallel}{3}}, \ldots$

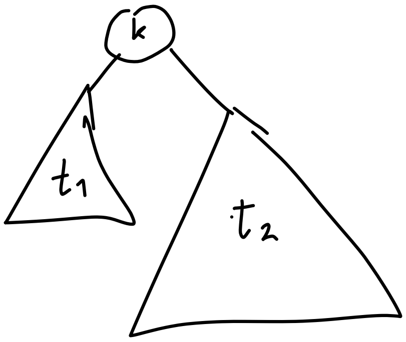
Koinduktivna naravna števila: $0, 1, 2, 3, \ldots, \underset{\text{suc(suc(suc(...)))}}{\underset{\shortparallel}{\infty}}$

Strukturna rekurzija



```
type tree =
  | Empty
  | Tree of int * tree * tree
```

```
Tree(1, Tree(2, Tree(4, Empty, Empty),
             Empty),
        Tree (5, Empty, Empty)
)
```



Koinduktivni tipi

Tok podatkov tipa α:

→ sporočilo/paket tipa α in preostanek toka

data Stream a = Cons (a, Stream a) Haskell

type 'a stream = Cons of 'a * Stream 'a

INDUKTIVNO: prazen tip

KOINDUKTIVNO: Cons (p_0, Cons (p_1, Cons (p_2, ...)))

v nedogled

	Matematika	OCaml	Haskell	C
produkt	$A \times B$	a * b	(A, B)	/
par	(x, y)	(x, y) x, y	(x, y)	/
enotski tip	1 $\{*\}$	unit ()	() ()	void
seznam	A^*	'a list	[a]	pointer → pointer → pointer → NULL

1, 2, 3, 4, ...

n, n+1, n+2, n+3,

Thunk

$E : \text{int}$

$\text{fun}\,() \to E \;: \text{unit} \longrightarrow \text{int}$

thunk

zakasnitev

```
int x = ..E.. ;

int f() {
  return ..E.. ;
}
```

izračuna šele ko pokličemo f:

.... f() .....