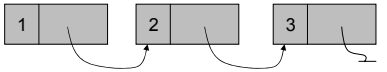
Seznamy v jazyce Haskell

Ing. Lumír Návrat
katedra informatiky, D-403
☎ 59 732 3252

Definice seznamu

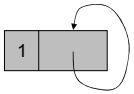
data List a = Cons a (List a) (x:xs)
| Nil []



1 : (2 : (3 : []))

Nekonečné seznamy

ones = 1 : ones



natFrom n = n : natFrom (n+1)
natFrom 1 = 1 : 2 : 3 : … = [1..]

Funkce pro seznamy

- Přístup k prvkům seznamu
  - `head [1,2,3]  = 1`
  - `tail [1,2,3]  = [2,3]`
  - `last [1,2,3]  = 3`
  - `init [1,2,3]  = [1,2]`
  - `[1,2,3] !! 2  = 3`
  - `null []       = True`
  - `length [1,2,3] = 3`

Funkce pro seznamy

- Spojení seznamů
  - `[1,2,3] ++ [4,5]  = [1,2,3,4,5]`
  - `[[1,2],[3],[4,5]] = [1,2,3,4,5]`
  - `zip [1,2] [3,4,5] = [(1,3),(2,4)]`
  - `zipWith (+) [1,2] [3,4] = [4,6]`
- Agregační funkce
  - `sum [1,2,3,4]     = 10`
  - `product [1,2,3,4] = 24`
  - `minimum [1,2,3,4] = 1`
  - `maximum [1,2,3,4] = 4`

Funkce pro seznamy

- Výběr části seznamu
  - `take 3 [1,2,3,4,5] = [1,2,3]`
  - `drop 3 [1,2,3,4,5] = [4,5]`
  - `takeWhile (> 0) [1,3,0,4] = [1,3]`
  - `dropWhile (> 0) [1,3,0,4] = [0,4]`
  - `filter (>0) [1,3,0,2,-1] = [1,3,2]`
- Transformace seznamu
  - `reverse [1,2,3,4] = [4,3,2,1]`
  - `map (*2) [1,2,3] = [2,4,6]`

Aritmetické řady

- [m..n]
  - [1..5] = [1,2,3,4,5]
- [m1,m2..n]
  - [1,3..10] = [1,3,5,7,9]
- [m..]
  - [1..] = [1,2,3,4,5,…]
- [m1,m2..]
  - [5,10..] = [5,10,15,20,25,…]

Funkce filter

Výběr všech prvků seznamu splňujících zadanou podmínku (predikát)

```
filter :: [a] -> (a->Bool) -> [a]
filter _ [ ] = [ ]
filter p (x:xs) | p x = x : filter p xs
                | otherwise = filter p xs
```

```
filter even  [1..10]      = [2,4,6,8]
filter (> 0) [1,3,0,2,-1] = [1,3,2]

delitele n = filter deli [1..n]
             where deli m = n `mod` m == 0
```

Poznámka – lambda abstrakce

- **Zadání funkce jako parametru**

```
nenulove xs = filter p xs
              where p x = x /= 0

nenulove xs = filter (/= 0) xs

nenulove xs = filter (\x -> x/=0) xs
```

- \ x -> e … λx . e
  - inc = \x -> x+1
  - plus = \(x,y) -> x + y
  - delitele n = filter (\m -> n `mod` m == 0) [1..n]

Funkce map

- Transformace prvků v seznamu

```
map :: [a] -> (a->b) -> [b]
map f [ ]    = [ ]
map f (x:xs) = f x : map f xs
```

```
map (+1)    [1,2,3] = [2,3,4]
map toUpper "abcd"  = "ABCD"

mocniny = map (\x -> x * x) [1..]
```


Generátory seznamů

Příklad: Množina všech sudých čísel z intervalu 1..10
- { x | x ∈1..10, x je sudé }
- [x | x <- [1..10], even x]

- `[ x | x <- xs ]  = xs`
- `[ f x | x <- xs ]  = map f xs`
- `[ x | x <- xs, p x ] = filter p xs`

-
```
[ (x,y) | x<-xs, y<-ys ] =
   [(x1,y1),(x1,y2),(x1,y3),…,
    (x2,y1),(x2,y2),(x2,y3),…,
     …  ]
```


Příslušnost prvku do seznamu

elem 1 [2,3,4] = False
elem 2 [2,3,4] = True
elem 3 [] = False

```
elem _ [ ]                = False
elem x (y:ys) | x == y    = True
              | otherwise = elem x ys
```

Jakého typu je funkce elem?
`elem :: a -> [a] -> Bool` --- NE!!! (elem sin [sin,cos,tan] = ?)

Poznámka k typovým třídám

- Třída – množina typů s určitými operacemi
  - Num: +, -, *, abs, negate, signum, …
  - Eq: ==, /=
  - Ord: >, >=, <, <=, min, max
- Omezení specifikace typu
  - elem :: Eq a => a -> [a] -> Bool
  - minimum :: Ord a => [a] -> a
  - sum :: Num a => [a] -> a

Příklady

- Seznam jako množina
  - průnik

```
intersect xs ys = [y | y <-ys, elem y xs]
```

  - sjednocení

```
union xs ys = xs ++ [y | y<-ys, notElem y xs]
```

  - rozdíl

```
diff xs ys = [x | x<-xs, notElem x ys]
```

  - podmnožina

```
subset xs ys = [x | x<-xs, notElem x ys] == []
subset xs ys = all (\x -> elem x ys) xs
```


Dokazování pomocí indukce

- Matematická indukce
  a) dokážeme pro n = 0
  b) za předpokladu, že platí pro n, dokážeme pro n+1
- Strukturální indukce pro seznamy
  a) dokážeme pro []
  b) za předpokladu, že platí pro xs, dokážeme pro (x:xs)

Příklad – asociativita ++

(xs ++ ys) ++ zs = xs ++ (ys ++ zs)

[] ++ ys = ys (++.1)
(x:xs) ++ ys = x: (xs ++ ys) (++.2)

a) [ ] => xs

([] ++ ys) ++ zs
= ys ++ zs (++.1)
= [] ++ (ys ++ zs) (++.1)

Příklad – asociativita ++

(xs ++ ys) ++ zs = xs ++ (ys ++ zs)

```
[] ++ ys       = ys                  (++.1)
(x:xs) ++ ys = x: (xs ++ ys)         (++.2)
```

b) (x:xs) => xs

```
((x:xs)++ys)++zs
   = x:(xs++ys)++zs                  (++.2)
   = x:((xs++ys)++zs)                (++.2)
   = x:(xs++(ys++zs))                (předpoklad)
   = (x:xs)++(ys++zs)                (++.2)
```


Příklad – length (xs++ys)

length (xs++ys) = length xs + length ys

```
length []       = 0                  (len.1)
length (_:xs) = 1 + length xs        (len.2)
```

a) [] => xs

```
length ([] ++ ys)
   = length ys                       (++.1)
   = 0 + length ys                   (nulový prvek +)
   = length [] + length ys           (len.1)
```


Příklad – length (xs++ys)

length (xs++ys) = length xs + length ys

```
length []       = 0                  (len.1)
length (_:xs) = 1 + length xs        (len.2)
```

b) (x:xs) => xs

```
length ((x:xs) ++ ys)
   = length (x:(xs++ys)      (++.2)
   = 1 + length (xs++ys)     (len.2)
   = 1 + (length xs + length ys)     (předpoklad)
   = (1 + length xs) + length ys           (asociativita +)
   = length (x:xs) + length ys             (len.2)
```


Příklady na cvičení

- Funkce reverse
  - varianta s časovou složitostí n^2
  - varianta se složitostí n
    - [1,2,3] [] -> [2,3] [1] -> [3] [2,1] -> [] [3,2,1]
- Spojení seznamů pomocí funkce
  - zip [1,2,3] [4,5] = [(1,4), (2,5)]
  - zipWith f [1,2,3] [4,5] = [f 1 4, f 2 5]
- Skalární součin dvou vektorů
  - [1,2,3] * [4,5,6] = 1*4+2*5+3*6
- Kartézský součin dvou množin (seznamů)
  - [1,2,3] x [‘a’,’b’] =
    [(1,’a’),(1,’b’),(2,’a’),(2,’b’),(3,’a’),(3,’b’)