Kava

* 3. loeng
  T: substitutsioon

I: Range ja listikomprehensioon
I: listidega programmeerimine

Substitutsioon

- λ-arvutuse alusoperatsiooniks on formaalsete parameetrite asendamine tegelike argumentidega.
- Avaldise $(\lambda x.\, e_1)\, e_2$ väärtustamiseks tuleb termis e_1 asendada muutuja x kõik vabad esinemised termiga e_2.
- Substitutsiooni tähistame $e_1[x \mapsto e_2]$.
- Peab vältima vabade muutujate "vangistamist".
- Näited:

$$
\begin{array}{lcl}
(\lambda x.\, y\, x)[y \mapsto \lambda z.\, z] & = & \lambda x.\, (\lambda z.\, z)\, x \\
(\lambda x.\, y\, x)[x \mapsto \lambda z.\, z] & = & \lambda x.\, y\, x \\
(\lambda x.\, y\, x)[y \mapsto \lambda z.\, x] & \neq & \lambda x.\, (\lambda z.\, x)\, x
\end{array}
$$

Substitutsioon

- λ-arvutuse alusoperatsiooniks on formaalsete parameetrite asendamine tegelike argumentidega.
- Avaldise $(\lambda x.\, e_1)\, e_2$ väärtustamiseks tuleb termis e_1 asendada muutuja x kõik vabad esinemised termiga e_2.
- Substitutsiooni tähistame $e_1[x \mapsto e_2]$.
- Peab vältima vabade muutujate "vangistamist".
- Näited:

$$
\begin{array}{lcl}
(\lambda x.\, y\, x)[y \mapsto \lambda z.\, z] & = & \lambda x.\, (\lambda z.\, z)\, x \\
(\lambda x.\, y\, x)[x \mapsto \lambda z.\, z] & = & \lambda x.\, y\, x \\
(\lambda x.\, y\, x)[y \mapsto \lambda z.\, x] & \neq & \lambda x.\, (\lambda z.\, x)\, x
\end{array}
$$

Substitutsioon

Substitutsiooni definitsioon:

$$
\begin{aligned}
y[x \mapsto e] &= \begin{cases} e & \text{kui } x = y \\ y & \text{muidu} \end{cases} \\
c[x \mapsto e] &= c \\
(e_1\ e_2)[x \mapsto e] &= (e_1[x \mapsto e])\ (e_2[x \mapsto e]) \\
(\lambda y.\ e_1)[x \mapsto e] &= \begin{cases} \lambda y.\ e_1 & \text{kui } x = y \\ \lambda y.\ e_1[x \mapsto e] & \text{kui } y \notin \mathrm{FV}(e) \\ \lambda z.\ e_1[y \mapsto z][x \mapsto e] & \text{muidu, kus } z \text{ on värske} \end{cases}
\end{aligned}
$$

ül 1.

Tahvlil lahendamise ülesanne!

$$(\lambda y.\ (\lambda x.\ y\ x)\ x)[x \to \lambda x.\ x] = \ldots$$

ül 2.

Tahvlil lahendamise ülesanne!

$$(\lambda y.\ (\lambda x.\ y\ x)\ x)[x \to \lambda x.\ y] = \ldots$$

Enumeratioonid

- `[m..n]` on list $[m, m+1, \ldots, n]$

  Näiteks: `[2..5] == [2,3,4,5]` ja `[3..1] == [3, 2, 1]`
  Näiteks: `['a'..'g'] = ['a', 'b', 'c', 'd', 'e', 'f', 'g']`

- `[m,n..p]` on list $[m, m+(n-m), m+2(n-m), \ldots, q]$ kus $q \leq p$

  Näiteks: `[3,5..15] == [3, 5, 7, 9, 11, 13, 15]`
  Näiteks: `['z', 'x'..'q'] == ['z', 'x', 'v', 't', 'r']`

ül 3.

Tahvlil lahendamise ülesanne!

Olgu defineeritud funktsioon `f`.

```
f : Int → List Int
f x = [x..x+5]
```

Mis on `f 10` väärtus?

ül 4.

Tahvlil lahendamise ülesanne!

Olgu defineeritud funktsioon `f`.

```
f : Int → List Int
f x = [x, x`div`2..0]
```

Mis on `f 20` väärtus?

Kuidas töötab see Pythonis?

Listikomprehensioon

Süntaks:

$$[e \mid q_1, q_2, \ldots, q_n]$$

e on avaldis ja q_i võib olla

- generaator, kujul $x \leftarrow l$, kus x on muutuja ja l list,
- let, kujul `let x = e`, kus x on muutuja ja e avaldis, või
- valvur, kujul e, kus e on `Bool` tüüpi avaldis.

Avaldis e võib kasutada "paremal" defineeritud muutujaid, q_i-d võivad kasutada "vasakul" defineeritud muutujaid.

Näiteks:

- `[x*x | x←[1..10]] == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]`
- `[m | x←[1..10], let m = x*x, m < 50] == [1, 4, 9, 16, 25, 36, 49]`
- `[(a,b) | a←[1..2],b←[1..3]] == [(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)]`

ül 5.

Tahvlil lahendamise ülesanne!

Olgu defineeritud funktsioon `f`.

```
f : List Int → List Int
f xs = [ x | x←xs, x`mod`2 == 0]
```

Mis on `f [2..8]` väärtus?

ül 6.

Tahvlil lahendamise ülesanne!

Olgu defineeritud funktsioon `f`.

```
f : List Int → List Int
f xs = [ x+10 | x←xs]
```

Mis on `f [2..8]` väärtus?

ül 7.

Tahvlil lahendamise ülesanne!

Olgu defineeritud funktsioon `f`.

```
f : List Int → List Int
f xs = [ 10*x+y | x←xs, y←xs, x<y]
```

Mis on `f [2,6,8]` väärtus?

Kuidas töötab see Pythonis?

Listifunktsioonid

(++): $[x_1, x_2, \ldots, x_n] \mathbin{++} [x_{n+1}, x_{n+2}, \ldots, x_{n+m}] = [x_1, x_2, \ldots, x_{n+m}]$

```
(++) : List a → List a → List a
[]     ++ ys = ys
(x::xs) ++ ys = x :: (xs ++ ys)
```

take: $\mathtt{take}\ n\ [x_1, x_2, \ldots, x_m] = [x_1, x_2, \ldots, x_{min(n,m)}]$

```
take : Int → List a → List a
take 0 _      = []
take n []     = []
take n (x::xs) = x :: take (n-1) xs
```

drop: $\mathtt{drop}\ n\ [x_1, x_2, \ldots, x_m] = [x_{n+1}, x_{n+2}, \ldots, x_{n+m}]$

```
drop : Int → List a → List a
drop 0 xs     = xs
drop n []     = []
drop n (x::xs) = drop (n-1) xs
```

Näited

- väikesed algarvud
- quicksort
- mergesort
- binaararvud

väikesed algarvud

Wikipedia:
The Sieve of Eratosthenes is an efficient algorithm for computing the successive primes. Rendered informally, it is as follows:

1. Write down the successive "plurals": 2, 3, 4, . . .
2. Repeat:
   - Take the first number that is not circled or crossed out.
   - Circle it.
   - Cross out its proper multiples.
3. What is left (i.e. the circled numbers) are the successive prime numbers.

quicksort

Implementeeri (kopeerimisega) kiirsorteerimine.

```
quicksort : List Int → List Int
quicksort = ?qs
```

mergesort

- tühi ja üheelemendiline list on sorteeritud
- pikem list jagada kaheks võimalikult ühepikkuseks listiks
  1. sorteeri jagatud listid
  2. ühilda kaks sorteeritud listi

Idris kood:

```
split : List Int → (List Int, List Int)
split xs = ?a1

merge : List Int → List Int → List Int
merge xs ys = ?a2

mergeSort : List Int → List Int
mergeSort xs = ?a3
```

binaararvud

Implementeeri binaararvud tõeväärtuste listina.

```
viis : List Bool
viis = [True, False, True]
-- |      |       |       |
-- v      v       v       v
-- 5 =    1 + 2*(0 + 2*(1 + 2*0))
```

NB! Bitid on listis vastupidises järjestuses kui tavaliselt.

```
toInteger : List Bool → Integer
toInteger xs = ?toInteger_rhs

fromInteger : Integer → List Bool
fromInteger n = ?fromInteger_rhs

incr : List Bool → List Bool
incr xs = ?incr_rhs

add : List Bool → List Bool → List Bool
add xs ys = ?add_rhs
```