Lambda-termide redutseerimine

Lambda-termid

```
type Var  = String
data Term = Var Var
          | App Term Term
          | Lam Var Term
```

Vabade muutujate leidmine

```
freeVars :: Term → [Var]
freeVars (Var x)       = [x]
freeVars (App e1 e2) = freeVars e1 `union` freeVars e2
freeVars (Lam x e)    = delete x (freeVars e)
```

Lambda-termide redutseerimine

Olekuteisendusmonaad

$$
\begin{aligned}
&\textbf{newtype}\ S\ s\ a = S\ (s \to (a, s))\\
&\textbf{instance}\ Monad\ (S\ s)\ \textbf{where}\\
&\quad (S\ f) \gg\!= k = S\ (\lambda s \to \textbf{case}\ f\ s\ \textbf{of}\\
&\qquad\qquad\qquad\qquad\qquad\qquad (x, s') \to \textbf{case}\ k\ x\ \textbf{of}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad S\ g \to g\ s')\\
&\quad return\ x \quad = S\ (\lambda s \to (x, s))\\
&getS \quad :: S\ s\ s\\
&getS \quad = S\ (\lambda s \to (s, s))\\
&setS \quad :: s \to S\ s\ ()\\
&setS\ x = S\ (\lambda s \to ((), x))\\
&runS \quad :: S\ s\ a \to s \to (a, s)\\
&runS\ (S\ f)\ s = f\ s
\end{aligned}
$$

Lambda-termide redutseerimine

Uute muutujate genereerimine

$$
\begin{aligned}
&newVar :: S\ Int\ Var \\
&newVar = \mathbf{do}\ i \leftarrow getS \\
&\qquad\qquad\quad setS\ (i + 1) \\
&\qquad\qquad\quad return\ (\texttt{"x"} +\!\!+ show\ i)
\end{aligned}
$$

Lambda-termide redutseerimine

Substitutsioon

```
subst :: Term → (Var, Term) → S Int Term
subst t (x, e) = subs t
  where fvs = freeVars e
        subs (Var y) | x == y      = return e
                     | otherwise   = return (Var y)
        subs (App e1 e2) = do e1' ← subs e1
                              e2' ← subs e2
                              return (App e1' e2')
        subs (Lam y e1)
          | x == y                 = return (Lam y e1)
          | notElem y fvs = do e1' ← subs e1
                               return (Lam y e1')
          | otherwise = do z    ← newVar
                           e1'  ← subst e1 (y, Var z)
                           e1'' ← subs e1'
                           return (Lam z e1'')
```

Lambda-termide redutseerimine

Ühesammuline reduktsioon (aplikatiivjärjekorras)

```
reduA :: Term → S Int (Maybe Term)
reduA (Var x) = return Nothing
reduA (Lam x e)
   = do me' ← reduA e
        case me' of
          Just e'   → return (Just (Lam x e'))
          Nothing → return Nothing
```

Lambda-termide redutseerimine

Ühesammuline reduktsioon (aplikatiivjärjekorras)

```
reduA (App e1 e2)
  = do me1 ← reduA e1
       case me1 of
         Just e1' → return (Just (App e1' e2))
         Nothing →
           do me2 ← reduA e2
              case me2 of
                Just e2' → return (Just (App e1 e2'))
                Nothing →
                  case e1 of
                    Lam x e0 → do e ← subst e0 (x, e2)
                                  return (Just e)
                    _        → return Nothing
```

Lambda-termide redutseerimine

Ühesammuline reduktsioon (normaaljärjekorras)

```
reduN :: Term → S Int (Maybe Term)
reduN (Var x) = return Nothing
reduN (Lam x e) = do me' ← reduN e
                     return (fmap (λe' → Lam x e') me')
reduN (Lam x e1 `App` e2) = do e ← subst e1 (x, e2)
                               return (Just e)
reduN (App e1 e2)
  = do me1 ← reduN e1
       case me1 of
         Just e1' → return (Just (App e1' e2))
         Nothing →
           do me2 ← reduN e2
              return (fmap (λe2' → App e1 e2') me2)
```

Lambda-termide redutseerimine

Reduktsioonijada genereerimine

```
iterateSM :: (a → S Int (Maybe a)) → a → S Int [a]
iterateSM f x = do y ← f x
                   case y of
                     Just y'  → do ys ← iterateSM f y'
                                   return (x : ys)
                     Nothing → return [x]

reduceA :: Term → S Int [Term]
reduceA = iterateSM reduA

reduceN :: Term → S Int [Term]
reduceN = iterateSM reduN
```

Lambda-termide redutseerimine

Parametriseeritud olekuteisendusmonaad

$$
\begin{aligned}
&\textbf{newtype}\ S\ m\ s\ a = S\ (s \to m\ (a, s)) \\
&\textbf{instance}\ Monad\ m \Rightarrow Monad\ (S\ m\ s)\ \textbf{where} \\
&\quad return\ x \quad = S\ (\lambda s \to return\ (x, s)) \\
&\quad (S\ f) \gg\!= k = S\ (\lambda s \to \textbf{do}\ (x, s') \leftarrow f\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad \textbf{case}\ k\ x\ \textbf{of} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad S\ g \to g\ s') \\
&getS \quad :: Monad\ m \Rightarrow S\ m\ s\ s \\
&getS \quad = S\ (\lambda s \to return\ (s, s)) \\
&setS \quad :: Monad\ m \Rightarrow s \to S\ m\ s\ () \\
&setS\ x = S\ (\lambda s \to return\ ((), x)) \\
&runS \quad :: Monad\ m \Rightarrow S\ m\ s\ a \to s \to m\ (a, s) \\
&runS\ (S\ f)\ s = f\ s
\end{aligned}
$$

Lambda-termide redutseerimine

Parametriseeritud olekuteisendusmonaad

```
instance MonadPlus m ⇒ MonadPlus (S m s) where
  mzero                 = S (λs → mzero)
  (S f) `mplus` (S g) = S (λs → f s `mplus` g s)
```

Uute muutujate genereerimine, substitutsioon

```
type StM a = S Maybe Int a
newVar :: StM Var
newVar = ...
subst  :: Term → (Var, Term) → StM Term
subst t (x, e) = ...
```

Lambda-termide redutseerimine

Ühesammuline reduktsioon (aplikatiivjärjekorras)

```
reduA :: Term → StM Term
reduA (Var x)     = mzero
reduA (Lam x e) = reduA e >>= λe' → return (Lam x e')
reduA (App e1 e2)
   = (reduA e1 >>= λe1' → return (App e1' e2)) `mplus`
     (reduA e2 >>= λe2' → return (App e1 e2')) `mplus`
     (case e1 of
        Lam x e0 → subst e0 (x, e2)
        _        → mzero)
```

Lambda-termide redutseerimine

Ühesammuline reduktsioon (normaaljärjekorras)

$$
\begin{aligned}
&reduN :: Term \to StM\ Term \\
&reduN\ (Var\ x) \quad\ = mzero \\
&reduN\ (Lam\ x\ e) = reduN\ e \gg\!= \lambda e' \to return\ (Lam\ x\ e') \\
&reduN\ (Lam\ x\ e1\ `App`\ e2) = subst\ e1\ (x, e2) \\
&reduN\ (App\ e1\ e2) \\
&\quad = (reduN\ e1 \gg\!= \lambda e1' \to return\ (App\ e1'\ e2))\ `mplus` \\
&\qquad (reduN\ e2 \gg\!= \lambda e2' \to return\ (App\ e1\ e2'))
\end{aligned}
$$

Lambda-termide redutseerimine

Reduktsioonijada genereerimine

$$
\begin{aligned}
&iterateStM :: (a \to StM\ a) \to a \to StM\ [a] \\
&iterateStM\ f\ x = (\mathbf{do}\ ys \leftarrow f\ x \gg\!= \lambda y \to iterateStM\ f\ y \\
&\qquad\qquad\qquad\qquad return\ (x : ys))\ `mplus` \\
&\qquad\qquad\qquad return\ [x] \\
&reduceA\ :: Term \to StM\ [Term] \\
&reduceA\ = iterateStM\ reduA \\
&reduceN\ :: Term \to StM\ [Term] \\
&reduceN\ = iterateStM\ reduN
\end{aligned}
$$