Monaadiline prelüüd

Sissejuhatus

- Monaadid on abstraktsioon kõrvalefektidega arvutuste modelleerimiseks.
- Annab ühtse liidese triviaalsete arvutuste konstrueerimiseks ning arvutuste järjestikustamiseks koos väärtuste edasiandmisega.
- See võimaldab realiseerida erinevaid funktsioone, mis on defineeritud kõigil monaadidel ühtmoodi.
- Suur hulk selliseid funktsioone on eeldefineeritud standardteekides.
  - Enamik neist vastavad imperatiivsetest keeltest tuntud juhtimisstruktuuridele.

Monaadilised funktsioonid standardprelüüdis

Klassidefinitsioon

$$
\begin{aligned}
&\textbf{class } Monad\ m \textbf{ where}\\
&\quad return :: a \to m\ a\\
&\quad (\gg\!=) \quad :: m\ a \to (a \to m\ b) \to m\ b\\
&\quad (\gg) \quad :: m\ a \to m\ b \to m\ b\\
&\quad fail \quad :: String \to m\ a\\
&\quad m \gg k = m \gg\!= \lambda\_ \to k\\
&\quad fail\ s = error\ s
\end{aligned}
$$

Monaadi ekstensioon

$$
\begin{aligned}
&(=\!\ll) \quad :: Monad\ m \Rightarrow (a \to m\ b) \to m\ a \to m\ b\\
&f =\!\ll x = x \gg\!= f
\end{aligned}
$$

Monaadilised funktsioonid standardprelüüdis

Üldistatud järjestikustamine

$$
\begin{aligned}
&sequence \;\; :: Monad\ m \Rightarrow [m\ a] \to m\ [a] \\
&sequence \;\; = foldr\ mcons\ (return\ [\,]) \\
&\quad \textbf{where}\ mcons\ p\ q = p \gg\!= \lambda x \to q \gg\!= \lambda y \to return\ (x : y) \\
&sequence\_ :: Monad\ m \Rightarrow [m\ a] \to m\ () \\
&sequence\_ = foldr\ (\gg)\ (return\ ())
\end{aligned}
$$

Monaadilised funktsioonid standardprelüüdis

Näide

```
Main> sequence [print 1, print 'a']
1
'a'
[(), ()]
Main> sequence_ [print 1, print 'a']
1
'a'
Main> it
()
Main> sequence [Just 1, Just 2, Just 3]
Just [1,2,3]
Main> sequence [Just 1, Nothing, Just 3]
Nothing
```

Monaadilised funktsioonid standardprelüüdis

Monaadiline *map*

```
mapM   :: Monad m => (a -> m b) -> [a] -> m [b]
mapM f as = sequence (map f as)
mapM_  :: Monad m => (a -> m b) -> [a] -> m ()
mapM_ f as = sequence_ (map f as)
```

map vs. *mapM*

```
map  ::                  (a ->   b) -> [a] ->   [b]
mapM :: Monad m => (a -> m b) -> [a] -> m [b]
```

Monaadilised funktsioonid standardprelüüdis

Näide (1)

```
Main> mapM print [1..5]
1
2
3
4
5
[(),(),(),(),()]
```

Näide (2)

```
putString   :: [Char] -> IO ()
putString s = mapM_ putChar s
```

Monaadiline prelüüd

"*for*-tsükkel"

```
forM  :: Monad m ⇒ [a] → (a → m b) → m [b]
forM  = flip mapM
forM_ :: Monad m ⇒ [a] → (a → m b) → m ()
forM_ = flip mapM_
```

NB!

Need ja järgnevad definitsioonid asuvad standardteegis *Control.Monad*

Näide

```
main = do
    forM_ [1..10] $ λi → do
      let n = fact i
      putStrLn $ show i ++ "! = " ++ show n
```

Monaadiline prelüüd

"Tingimuslik täitmine"

```
when       :: Monad m ⇒ Bool → m () → m ()
when p s   = if p then s else return ()
unless     :: Monad m ⇒ Bool → m () → m ()
unless p s = when (not p) s
```

Näide

```
import System.Environment (getArgs)
import System.Directory   (doesDirectoryExist)
main = do names ← getArgs
          forM_ names $ λdir → do
                  b ← doesDirectoryExist dir
                  when b $ putStrLn dir
```

Monaadiline prelüüd

Monaadiline *filter*

```
filterM :: Monad m ⇒ (a → m Bool) → [a] → m [a]
filterM _ []       = return []
filterM p (x : xs) = do
  b  ← p x
  ys ← filterM p xs
  return (if b then x : ys else ys)
```

Näide

```
main = do names ← getArgs
          dirs  ← filterM doesDirectoryExist names
          mapM_ putStrLn dirs
```

Monaadiline prelüüd

Monaadiline *fold*

$$
\begin{aligned}
&foldM :: Monad\ m \Rightarrow (a \to b \to m\ a) \to a \to [b] \to m\ a \\
&foldM\ f\ a\ [\,] \qquad\quad = return\ a \\
&foldM\ f\ a\ (x : xs) = f\ a\ x \gg\!= \lambda fax \to foldM\ f\ fax\ xs
\end{aligned}
$$

foldM intuitsioon

$$
\begin{aligned}
foldM\ f\ a_1\ [x_1, x_2, \ldots, x_n] = \mathbf{do}\ & a_2 \leftarrow f\ a_1\ x_1 \\
& a_3 \leftarrow f\ a_2\ x_2 \\
& \ldots \\
& f\ a_n\ x_n
\end{aligned}
$$

foldl vs. *foldM*

$$
\begin{aligned}
&foldl\ \ :: \qquad\qquad\quad (a \to b \to \quad\ a) \to a \to [b] \to \quad\ a \\
&foldM :: Monad\ m \Rightarrow (a \to b \to m\ a) \to a \to [b] \to m\ a
\end{aligned}
$$

Monaadiline prelüüd

"Liftimine" (1)

$$liftM :: Monad\ m \Rightarrow (a \to b) \to m\ a \to m\ b$$
$$liftM\ f\ m = \mathbf{do}\ \{x \leftarrow m; return\ (f\ x)\}$$

fmap vs. *liftM*

$$fmap :: Functor\ f \Rightarrow (a \to b) \to f\ a \to f\ b$$
$$liftM :: Monad\ m \Rightarrow (a \to b) \to m\ a \to m\ b$$

Näide

$$countLines :: FilePath \to IO\ Int$$
$$countLines = liftM\ (length \circ lines) \circ readFile$$

Monaadiline prelüüd

"Liftimine" (2)

$$liftM2 :: Monad\ m \Rightarrow (a \to b \to c) \to m\ a \to m\ b \to m\ c$$
$$liftM2\ f\ m1\ m2 = \mathbf{do}\ \{x1 \leftarrow m1; x2 \leftarrow m2; return\ (f\ x1\ x2)\}$$

NB!

Analoogiliselt on defineeritud *liftM3*, *liftM4* ja *liftM5*.

Näide

```
Main> liftM2 (+) (Just 1) (Just 2)
Just 3
Main> liftM2 (+) (Just 1) Nothing
Nothing
Main> liftM2 (+) [0,3] [5,6,7]
[5,6,7,8,9,10]
```

Monaadiline prelüüd

Monaadiline aplikatsioon

$$ap :: Monad\ m \Rightarrow m\ (a \to b) \to m\ a \to m\ b$$
$$ap = liftM2\ id$$

ap vs. *liftMn*

$$liftMn\ f\ x1\ x2\ \ldots\ xn = return\ f\ `ap`\ x1\ `ap`\ x2\ `ap`\ \ldots\ `ap`\ xn$$

Näide

```
Main> [(+2)] `ap` [1,2,3]
[3,4,5]
Main> [(+2),(*2)] `ap` [1,2,3]
[3,4,5,2,4,6]
```

Monaadiline prelüüd

Klass *MonadPlus*

```
class Monad m ⇒ MonadPlus m where
    mzero :: m a
    mplus :: m a → m a → m a
```

Üldistatud konkatenatsioon (paralleelkompositsioon)

```
msum :: MonadPlus m ⇒ [m a] → m a
msum = foldr mplus mzero
```

Monaadiline "valvur"

```
guard          :: MonadPlus m ⇒ Bool → m ()
guard True     = return ()
guard False    = mzero
```

Monaadiline prelüüd

Listide komprehensioon vs. do-süntaks

$$list = [r \mid x_1 \leftarrow xs_1, x_2 \leftarrow xs_2, \ldots, p_1, p_2, \ldots]$$

$$\begin{aligned} list = \mathbf{do}\ & x_1 \leftarrow xs_1 \\ & x_2 \leftarrow xs_2 \\ & \ldots \\ & guard\ p_1 \\ & guard\ p_2 \\ & \ldots \\ & return\ r \end{aligned}$$