Haskell Unit 8: Interactive Programs

Antoni Diller

26 July 2011

Introduction

There are several ways of handling I/O in Haskell. The simplest is to use what is know as stream-based I/O. More information can be found in Richard Bird, *Introduction to Functional Programming using Haskell* (1998), section 9.6, pp. 319-322. Much more information can be found in Bird and Wadler, *Introduction to Functional Programming* (1988), section 7.8, pp. 196-203. In stream-based I/O the hard work of reading and writing is done for you; all that you have to do is to define a function from `String` to `String`, where `String` is just `[Char]`. A very simple example follows:

```
capitalises :: String -> String
capitalises = map toUpper
```

To run this as an interactive program you just issue the command `interact capitalises`. What you type is passed to the function `capitalises` as its argument and the value of the function `capitalises` appears on the terminal. The following is a more complicated example:

```
before x = takeWhile (/= x)
after x  = tail . dropWhile (/= x)

type StCo = String -> String

read1 :: String -> (String -> StCo) -> StCo
read1 msg g input
  = msg ++ line ++ "\n" ++ g line input'
    where line   = before '\n' input
          input' = after  '\n' input
```

```
read2 :: (String,String)
      -> ((String,String) -> StCo)
      -> StCo
read2 (msg1, msg2) g
  = read1 msg1 g1
    where g1 line1 = read1 msg2 g2
                     where g2 line2 = g (line1, line2)

write :: String -> StCo -> StCo
write msg g input = msg ++ g input

end _ = ""

enter :: a -> b -> [(a,b)] -> [(a,b)]
enter k v t = (k, v) : t

lookup1 :: Eq a => [(a,String)] -> a -> String
lookup1 t k
  | vs /= []  = head vs
  | otherwise = "No entry"
                where vs = [w | (c, w) <- t, c == k]

newtable = []

table :: [(String,String)] -> StCo
table t
  = read1 "Command: " tcommand
    where
    tcommand "enter"  = read2 ("Key: ", "Value: ") tenter
    tcommand "lookup" = read1 "Key: " tlookup
    tcommand "end"    = write "Exit program\n" end
    tcommand _        = write "Unknown command\n\n" (table t)
    tenter (k, v)     = write "\n\n" (table (enter k v t))
    tlookup k         = write (lookup1 t k ++ "\n\n") (table t)
```