I/O and Compiler Programs

Haskell for Real Life

Prof Hans Georg Schaathun

Høgskolen i Ålesund

18th January 2016

Motivation

1. GHCi is an interactive interpreter
    - evaluate any function
    - very convenient for testing
2. Users want standalone programs
    2.1 do not want to learn GHCi
3. Programs need I/O to communicate

A simple example

```
greeting :: String -> String
greeting i = "Pleased to meet you, " ++ i ++ "!"
```

I/O breaks regular assumptions in functional programming

```
foobar = do
     putStr "What is your name?"
     i <- getLine
     putStr "Pleased to meet you, " ++ i ++ "!"
```

The IO type

```
Prelude> :type putStr "What is your name?"
putStr "What is your name?" :: IO ()

Prelude> :type getLine
getLine :: IO String
```

A standalone Haskell program

1. A `Main` module
2. A `main ::  IO ()` object
3. Compile: `ghc Main.hs`
4. Run: `./Main`

The CSV files

1. Data sets as comma separated values
2. Each row is an object
   2.1 Feature values
   2.2 Class labels
3. How do we read the data set for use?

— Two operations
   1. Reading the file
      1.1 IO operations
   2. Parsing strings
      2.1 CSV library

- The tutorial gives a simple recipe

Summary

1. IO is different from regular evaluations
2. We have IO actions
   2.1 sequenced with `do` notation
3. Next week we will dig under the syntactic sugar
4. Haskell programs can be compiled with GHC