I/O and Compiler Programs

Haskell for Real Life

Prof Hans Georg Schaathun

Høgskolen i Ålesund

February 2, 2015

hials

Motivation

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

hials

Outline

hials

The I/O problem

I/O breaks regular assumptions in functional programming

1. Evaluations like `f x` are simple
   1. always the same return value for the same `x`
   2. can be evaluated in any order
   3. no side effects
2. What about I/O operations?
   1. `putStr` (output)
   2. `getLine` (input)

hials

A simple example

Sequencing IO operations

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

hials

The IO type

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

1. `IO` is a type constructor
   1. `IO String` and `IO ()` are different types
   2. ... with something in common
2. Objects of `IO` types are actions
   1. to be sequenced with `do` notation
   2. may wrap a value, to be retrieved with the `<-` operation
3. Syntactic sugar — details next week

hials

Outline

hials

A standalone Haskell program

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

hials

Outline

hials

The CSV files

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

hials

Reading and parsing

- Two operations
  1. Reading the file into memory
     1. IO operations
  2. Parsing strings read from file
     1. CSV library

- The tutorial gives a simplerecipe

hials

Outline

hials

Summary

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

hials