IO in Haskell

Dr. Heinrich Hördegen
hoerdegen@energiefluss.info

19. Januar 2014

GUTTENBERG
& HÖRDEGEN

Basic IO in Haskell

In Haskell **all** IO action must take place in the IO monad.

```
action :: IO ()
action = do
    putStr "Type something: "
    line <- getLine
    putStrLn line
```

The type `IO a` means: When executing this IO action, we get back something of type `a`.

Is `getLine` a function?

After all, `getLine` returns every time something different...

Is getLine a function?

After all, `getLine` returns every time something different...

But if you think of `IO a` as

```
type IO a  =  RealWorld -> (a, RealWorld)
```

then `getLine` could be just a function like any other function:

```
action :: IO String
action world0 =
    let (a, world1) = getLine world0
        (b, world2) = getLine world1
    in (a ++ "\n" ++ b, world2)
```

That's what the IO monad handles for you!

How to access intermediate results?

Build lazy data structures with intermediate results.

Instead of

```
f :: Int -> Int
```

define something like

```
f :: Int -> [Int]
```

Then, define your original function as

```
func :: Int -> Int
func x = last (f x)
```

For debugging, you can use `f`.

IO Actions are First Class

You can build IO actions in non IO code.

```
readOnlyFromTmp :: FilePath -> Maybe (IO ())
readOnlyFromTmp fileName  =
  if "/tmp/" `isPrefixOf` fileName
     then Just $ do str <- readFile fileName
                    putStrLn str
     else Nothing
```

Useful for building closures.

A Word on Syntax I

```
main = do
    putStrLn "sometext"
```

equals

```
main = putStrLn "sometext"
```

A Word on Syntax I

```
main = do
    putStrLn "sometext"
```

equals

```
main = putStrLn "sometext"
```

```
main = do
    let str = "sometext"
    putStrLn str
```

equals

```
main =
    let str = "sometext"
    in putStrLn str
```

A Word on Syntax II

```
main = do
    putStrLn "sometext1"
    putStrLn "sometext2"
```

equals

```
main = putStrLn "sometext1" >> putStrLn "sometext2"
```

A Word on Syntax II

```
main = do
    putStrLn "sometext1"
    putStrLn "sometext2"
```

equals

```
main = putStrLn "sometext1" >> putStrLn "sometext2"
```

```
main = do
    str <- readFile "file.txt"
    putStrLn str
```

equals

```
main = readFile "file.txt" >>= putStrLn
```

And don’t you ever try this

```
...
case getLine of
     IO line -> line
```

It just won’t work!

Is IO in Haskell pure?

Is IO in Haskell pure?

For further explanation, look here:

http://www.haskell.org/haskellwiki/IO_inside
#Haskell_is_a_pure_language

Have a look at our Munich Haskell Meeting:

http://www.haskell-munich.de/

Thanks for your attention!

GUTTENBERG
& HÖRDEGEN

hoerdegen@energiefluss.info
www.energiefluss.info