Three Scala solutions
How to make your Scala controll effects a-la Haskell

George Leontiev

folone.info

February 28, 2013

Introduction

Note: I intentionally made it more "interesting" to show more neat scalaz stuff
I won't cover everything though. If something seems strange, please ask.

Core

Main functions

```
wordCount :: String → Map (String, Int)
acceptedChars :: Char → Boolean
splitWords :: String → [String]
```

Helper functions

```
time :: (a → IO b) → IO b
close :: Closeable a  a → IO ()
```

Executing

```
def main(args: Array[String]) = {
  val path   = args(0)
  val action = for {
    result ← time(function(path))
    _ ←          putStrLn(result)
  } yield ()
  // Yuck!
  action.unsafePerformIO()
}
```

First attempt

```
def wholeFile(path: String): IO[String] =
  IO { Source.fromFile(path) }.bracket(close) {
    source =>
      IO {
        val text   = source.mkString
        val result = wordCount(text)
        result.toList.shows
      }
  }
```

First attempt

Works fine, but eats all the heap on a large enouth file.

Second attempt

```
def byLine(path: String): IO[String] =
  IO { Source.fromFile(path) }.bracket(close) {
    source =>
      IO {
        val stream = source.getLines.toStream
        val result = stream.map(wordCount)
          .foldLeft(Map.empty[String, Int]) {
            case(acc, v) => acc |+| v
        }
        result.toList.shows
      }
  }
```

Second attempt

Just what is this $|+|$?

Typeclasses

```
implicit val mapInstances =
  new Show[List[(String, Int)]]
  with Monoid[List[(String, Int)]]

  is the same as

instance Show [(String, Int)] where ...
instance Monoid [(String, Int)] where ...
```

Second attempt

Pretty good, but can we do better?

Iteratees

```
def getFileLines[A](f: File,
  m: Process[String, A]): Procedure[IO, A] =
    new Procedure[IO, A] {
      type K = String => Any
      val machine = m
      def withDriver[R](k: Driver[IO, K] =>
        IO[R]): IO[R] = {
          bufferFile(f).bracket(close)(r => {
            val d = new Driver[IO, String => Any] {
              val M = Monad[IO]
              def apply(k: String => Any) =
              rReadLn(r) map (_ map k)
            }
            k(d)
          })
        }
    }
```

Iteratees

```
def bufferFile(f: File): IO[BufferedReader] =
  IO { new BufferedReader(new FileReader(f)) }

  /** Read a line from a buffered reader */
def rReadLn(r: BufferedReader):IO[Option[String]] =
  IO { Option(r.readLine) }
```

Iteratees

```
def wordFreq(path: String) =
  getFileLines(new File(path),
    id outmap wordCount) execute
```

Wordcounting at large

Scoobi	http://nicta.github.com/scoobi/
Spark	http://spark-project.org/
Scalding	https://github.com/twitter/scalding/wiki/Type-safe-api-reference

Wordcounting

Turns out, this code will work for these "as is".