Functional Design & Parser combinators

Lucas Nouguier

Polytech Montpellier

13 November 2023

Outline

Outline

Binary expression ADT

Definition

```
sealed trait Expr:
  def +(that: Expr): Expr

case class Num(value: Int) extends Expr
case class Var(name: String) extends Expr
case class Add(left: Expr, right: Expr) extends Expr
```

Addition parser

Given the following parsers:

number [0-9]+

variable [a-zA-Z]+

plus [ ]*+[ ]*

expr variable | number

```
val parser0: Parser[Add] =
  (expr, plus, expr).mapN((l, _, r) => l + r)
parser0.parse("x + 1 + a").get // Add(Var("x"), Num(1))

val parser1: Parser[Add] =
  (expr, plus, parser1 orElse expr).mapN((l, _, r) => l + r)
parser1.parse("x + 1 + a").get // NullPointerException

def parser2: Parser[Add] =
  (expr, plus, parser2 orElse expr).mapN((l, _, r) => l + r)
parser2.parse("x + 1 + a").get // StackOverflowError
```

Addition parser

Solution Delay the parser's evaluation

```scala
// In Parser.scala
object Parser:
  def lzy[A](parser: => Parser[A]) = Delayed(parser)

class Delayed[A](p: => Parser[A]) extends Parser[A]:
  lazy val cached = p
  def parse(input: String, index: Int): Result[A] =
    cached.parse(input, index)

// Addition parser
val parser3: Parser[Add] =
  (expr, plus, lzy(parser3) orElse expr).mapN((l,_,r) => l+r)

parser3.parse("x + 1 + a").get
// : Add = Add(Var("x"), Add(Num(1), Var("a")))
```