Weekly Assignment 2

Advanced Programming 2014 @ DIKU

Martin Jørgensen
University of Copenhagen
Department of Computer Science
tzk173@alumni.ku.dk

Casper B. Hansen
University of Copenhagen
Department of Computer Science
fvx507@alumni.ku.dk

September 27, 2014

Abstract

A parser should be implemented for a domain specific language, describing curves and operations on them.

Tasks

Introduction

For the resubmission we have changed to a new parser library, instead of going with Parsec we switched to ReadP which proved to be easier to work with for our simple purposes.

Parsing

Partwise parsers

To allow for nore readable code the parser is split into different smaller parsers/methods. Topmost in the file we have a number of smaller convenience parsers such as `charToken`, `stringToken`, `number` and so forth. These are meant to catch and parse small components that are likely to be used by severeal other parsers.

The main parser is the one that parses "programs", it is called form `parseString` and will in turn call the rest of the parsers (indirectly of course, since it only really calls the `defs` parser itself. The code for the method can be seen in Figure 1.

```
89  -- Parse a program
90  prog :: ReadP [Def]
91  prog = do
92    d <- defs
93    eof
94    return d
```

Figure 1: The implementation of the `prog` method which parses programs/lists of definitions. (../CurvySyntax2.hs)

`parseString`

Uses readP to parse a string with the parsers defined earlier in the program.

```
226 -- Parses a string into a program.
227 parseString :: String -> Either Error Program
228 parseString s = case opt of
229   [] -> Left "Parser error."
230   (x:_) -> Right (fst x)
231   where opt = readP_to_S prog s
```

Figure 2: The implementation of the `parseString` method. (../CurvySyntax2.hs)

`parseFile`

`parseFile` was implemented with the suggestion from the assignment text and can be seen in Figure 3.

```
233 -- Reads and parses a file to a program.
234 parseFile :: FilePath -> IO (Either Error Program)
235 parseFile filename = fmap parseString $ readFile filename
```

Figure 3: The implementation of the `parseFile` method. (../CurvySyntax2.hs)

Testing