University of Copenhagen

Assignment 1: Curvy Syntax

Jens Fredskov (chw752)

September 15, 2013

1 Introduction

The following report describes the implementation and testing of a Haskell library to parse curve programs to an abstract syntax tree.

2 Implementation

The library is implemented in the file `CurvySyntax.hs` and uses the abstract syntax from `CurveAST`, which has not been modified. The library is implemented using SimpleParse.

The implementation follows the given grammar. This also means that `width` and `height` binds stronger than any curve operations, since they are further down the list.

In general the implementation uses a parser combinator for each rule in the grammar. The exception is *Defs* which is not need since it can be integrated in *Program*. Furthermore there also exists combinators for *Number* and *Ident*.

To establish precedence between operators `chainl1` has been used in `expr` and `curve`. Furthermore `curve` uses a variant of `chainl1` called `eval` which allows to raise an operator and apply two different parsers to the sides instead of one.

When parsing for strings `symbol` is in general used. The exception being in `curve` and `number`. The first because some operators require at least one space and other can have zero or more (thus either `many space` or `many1 space` is used).

The type *Error* has been defined as a string, and when returned in `parseString` gives an appropriate error message, to let the user know that the parser failed.

3 Testing

The implementation have been tested using the curve programs defined in the directory `tests`. Test 1 through 5 all parse, while 6 through 9 fail. Here a list of the tests is given, summing why what they test.

test1	Tests creation of a curve. Either a point or another curve.
test2	Tests the precedence of expression operators over curve operators.
test3	Tests precedence between curve operators and the use of parentheses to change the precedence.
test4	Tests the use of *where*.
test5	Tests the use of several curve operators, their precedence and spacing between operators.
test6	Fails because a keyword is used as identifier.
test7	Fails because of incorrect spacing between operators.
test8	Fails because the *where*-clause is defined but empty, which is wrong according to the grammar.
test9	Fails because only a curve with no definition is given.

All the tests yielded the expected result, test5 however is somewhat slow. We can thus conclude that the library works correct in our tests, and seemingly in general.

4 Conclusion

We have now described the implementation of the library, and accounted for the overall structure of the library and the parser combinators used to construct the complete parser. The testing of the library have been described, and we have concluded that the library has worked as expected in all of our tests.