University of Copenhagen

Assignment 2: Counting parentheses

Jens Fredskov (chw752)

September 22, 2013

1 Introduction

The following report describes the implementation and testing of a Haskell library to intepret the APLisp dialect.

2 Implementation

The library has been implemented in the file `APLisp.hs`. At the moment the *let*-keyword and the use of lambdas has not been implemented.

2.1 Types and data structures

The environment type has been implemented as:

```
newtype APLispExec a = RC { runLisp :: Environment -> Either String a }
```

We use *Either* because it allows us to propagate an error message up through the interpretation, whereas we with e.g. *Maybe* would be able to have only one error message. This also means that the *fail* keyword is not used anywhere.

In the same way *Result* has been defined as

```
type Result = Either String SExp
```

Allowing us to pass an error message from either the parse or the interpreter if necessary.

2.2 Functions

The main function `interpret` takes in a string parses it and then tries to interpret it using `eval` with an empty environment (since we have yet to bind anything).

`eval` evaluates the expressions and in case of a function application (such as an arithmetic operator) sends the evaluated arguments and the operator to `apply`.

`apply` applies the operator to the given arguments (if valid) and returns the result. In case of arithmetic operators it expects the arguments to be *IntVal* as `eval` is expected to have evaluated all arguments beforehand.

Both `eval` and `apply` uses the function `invalid` as a shorthand for defining a *Left* when an error occurs.

3 Testing

As most of the time was spent trying to implement the missing parts of the assignment, the testing of the implemented part is not complete. A test suite can be found in `APLispTest.hs`.

Currently the suite uses HUnit to test different cases of the implemented parts. The suite tests wheter a program gives the expected result, where the result is either a *SExp* or an error, e.g. in the case of division by zero.

In all the tried test cases the implementation returns the expected result, thus it seems that what has been implemented works. The tests however are still not exhaustive.

4 Conclusion

We have now described the implementation of the library, and accounted for the overall structure of the library.. The testing of the library have been described, and we have concluded that the implemented parts of the library has worked as expected in all of our tests.