dOvs - warmup project

Group 9
Miran Hasanagić - 20084902
Jakob Graugaard Laursen - 20093220
Steven Astrup Sørensen - 201206081

September 6, 2015

1 Introduction

This report describes the approach in order to develop a straight line interpreter. Additionally, it provides an overview of problems encountered and intersting experiments during development process. Finally, five test expressions are interpreted and shown.

2 Development process and collaboration

You should not write how you wrote the interpreter, but why you wrote it that way The group members met physically in order to both unterstand the problem and furthermore outline and discuss the solutions.

During the development different parts raised different challenges. Describe the challenges here. After the main functionality of the interpreter was woking correctly, different experiments led to the improvement of different special cases. For example, the two exceptions `DivisionByZero` and `IdExpNotFound`. Both can be interpreted with valid syntax, but can not be interpreted to valid values. Hence they had to be handled. The syntax is ensured by the **datatype**s `stm` and `exp`. Hence if a `stm` uses non valid constructs the SML interpreter will give an error.

Generally the suggestions in the book were followed. Since a `stm` can be viewed as having a TREE structure as described in the book, it was naturally to use mutual recursive functions in order to interpret an expression, such as `prog` from the book. In the end different parts of the code were refactored in order to make them more readable and use more fitting SML syntax. As an example `if` expressions were change to use pattern matching in relevant places.

3 Problems experienced

If you encountered any problems describe how you solved them

This is already mentioned challenges in the previous section, maybe we should move it to here?? What do you guys think??

Other notes:

The first major problem we encountered were to ambitious when writing the mutual recursive functions interpStm, interpExp and all of their auxiliary functions.

We simply wrote all of code at once, never checking if what we already had written was sound. Naturally, the SML interpreter complained and threw error messages that filled several terminal screens. Defeated, after about an hour or two of trying to make sense of the errors along with rewriting the code, we decided to throw away the code and start from scratch.

We then build the functions one line at a time, each time checking if the functions as currently written had the correct type. We learned a very useful technique, namely having a catch all base case, something like funct (_) = NONE, until we had covered all possible cases we were interested in.

This approach, which essentially is Test Driven Development, allowed us to write all the desired functionality in about 2 hours.

4 Additional statements tested

5 statements as source code, and as values of type 'stm' plus a description of test including the outcome.

5 Conclusion

This report described the development process of the straight line interpreter. This project was used as a "warum up" project for both learning programming in SML and general principles of interpreting a language. Both aspects provide valuable lessons in order to start work on the compiler project.