Course Project (CS 653)

12.03.2018

Shashank Shekhar , Roll No 150662
Saket Harsh , Roll No 150617

Overview

In this above project, we aim to design an interpreter of programming language ‘Scheme’.

Goals

1. To learn Haskell mutable state, dynamic typing, error handling and parsing features.
2. To implement parser that will help us understand Haskell in a better way.
3. Build a Scheme REPL , that implements a decent subset of R5RS standards.

Specifications

We want to build an interpreter of Scheme, which will support **Numerical Operations** (such as +, -, *, /, quotient, remainder), **Type Predicates** (symbol?, string?, bool?, list?, pair?, vector?, etc.), **Type Conversions**, **Comparisons** (<, >, <=, >=, string<?, string>?, /=, etc.), **String functions** and **I/O.**

We will also try and add support for standard library such as **Numerical predicates, Higher Order functions** (flip, curry, compose) and **List Manipulations.**

Milestones

I. Learn Monads, I/O features and Error Handling in Haskell .

II. Gain working knowledge of Scheme.

III. Write the interpreter for Scheme in Haskell.

We will take references from :- http://bit.ly/1IU7Xwr.

Achievements

1. I/O features and Error Handling implemented.
2. Conditional clauses added, like if-else and cases.
3. Support for functions, recursion and lambda functions added.
4. Support to include files and libraries.
5. Standard library included, which contains definitions of functions like foldl, foldr, curry, sum, max, length, etc.
6. Higher order functions implemented.

Things Which We Have Not Implemented

1. Building a syntax tree.
2. We had thought of including Data Constructors, but could not do it.
3. Quote - unquote -- only parsing is done. After parse, no more evaluations are defined. Moreover, parsing is done only for quote. The symbol, ',' (comma), throws a parse error.

Project Submission

Readme file contains the packages which need to be installed. LispVal.hs contains the declaration of different data types. SchemeParser.hs is the parser, SchemeEval.hs evaluates the output from the parser, and SchemeInterpreter.hs is the final interpreter. Commands.txt contains various commands which can be run on the interpreter.