ESC101 Advanced Track

Project: Scheme REPL written in Haskell

Project Goals

- Learn functional programming through Haskell
- Learn parsing in Haskell
- Build a Scheme REPL that implements a decent subset of the R5RS standard

What I did

I first worked through Learn You a Haskell for Great Good. Due to the lack of exercises in the book I implemented myself several library functions, especially those in Data.List, with the official documentation as a fallback. Working through the book took me quite some time.

The original plan was to build a web scraper, but I lost interest and stopped working for a while. When I finally realized what I wanted to do was parse a programming language, not much time was left and I ended up following a tutorial (Write Yourself a Scheme in 48 Hours).

I then wrote solutions to 25 of the 99 Problems in Lisp to test the REPL.

REPLica

REPLica is a Scheme REPL that implements a decent subset of the R5RS standard, with a few extensions and a few exceptions, and a small standard library. Omissions and deviations from the standard are largely due to Haskell's high level nature.

How REPLica is Scheme

- Has mutable variables and supports define syntax

- Can load scripts from the computer
- Can evaluate almost all Scheme expressions, barring a few primitive functions and syntactic keywords that it does not include (yet)

How REPLica is not Scheme

- Load and apply are syntactic keywords
- No local scope (i.e. let syntax)
- No support for exact and inexact numbers
- Eqv? and eq? differ from standard (no pointer comparisons etc in Haskell)
- No hygeinic macros

Supported primitives

- **Numerical operations:** +, -, *, /, quotient, remainder
- **Type predicates:** symbol?, string?, bool?, list?, pair?, vector?, number?, complex?, real?, rational?, integer?
- **Type conversions:** symbol->string, string->symbol, string->list, list->string
- **Comparisons:** =, <, >, /=, >=, <=, string=?, string<?, string>?, string<=?, string>=?, string-ci=?, string-ci<?, string-ci>?, string-ci<=?, string-ci>=?, char=?, char<?, char>?, char<=?, char>=?
- **String functions:** make-string, string, string-length, string-ref, substring, string-append
- **Cons cells:** car, cdr, cons
- **Equality predicates:** eq?, eqv?, equal?
- **IO:** open-input-file, open-output-file, close-input-port, close-output-port, read, write, read-contents, read-all

Standard Library

- not
- null?
- id
- **Numerical predicates:** zero?, positive?, negative?, odd?, even?

- **Higher order functions:** flip, curry, compose
- **List manipulations:** list, map, filter, foldr/reduce, foldl/fold, unfold, sum, product, max, min, length, reverse, memq, memv, member, assq, assv, assoc