Lithp A slow and verbose programming language: The standard

Hussam Samir Yousif

2016

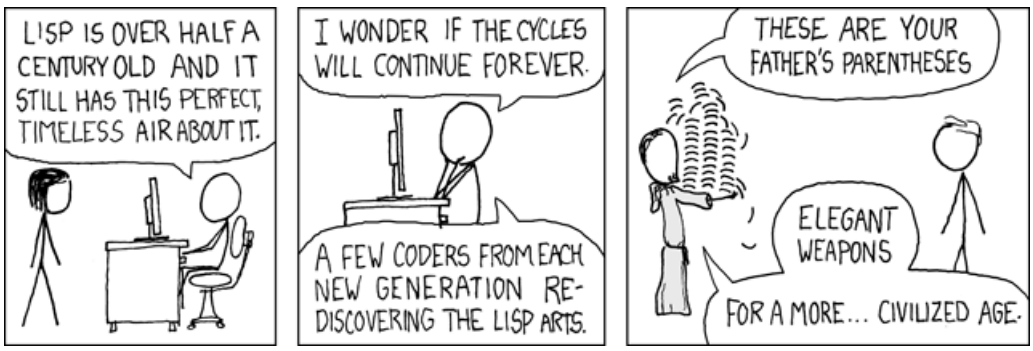
1 Introduction

Definition 1. *For simplicity's sake, let's refer to Common Lisp as Lisp.*

Lithp is a subset of the programming language Lisp implemented in Haskell. The goal of Lithp is to further my (very limited) understanding of Lisp and Haskell, the Language Lithp is implemented in. I'd be extremely happy if I can write recursive functions in Lithp, then it would be an actual programming language and not a glorified calculator.

I chose to implement Lisp because Lisp seemed to me like the closest thing one could come to a "classic" within programming. An ancient Language that has survived throughout the ages. It survived while languages like Fortran and Cobol died out (mostly). There must be a reason why people chose it and keep sticking up to it even though it is dynamically typed and as such a blight upon all decency. What are Sexpressions? Why would you have a fundemental part of your language have such an inappropriate name? How can a Lisp Interpreter be implemented with so few lines? Why is Lisp the only language to be so insanely multiparadigm to the point where some consider it functional while other use CLOS an OOP system of programming?

If I can answer some of these questions, I'd be satisfied.



2 Language description

2.1 Data types

Lithp contains the following data types

- Integer
- boolean
- String
- Atom
- List
- Nil
- Function

2.2 Type System

Lithp uses a dynamic typing system, meaning that the interpreter determines the type at run time, more specifically at the parsing phase. Also there are no declaration of the types of variables or functions for that matter making Lithp very vulnerable to type errors.

2.3 features

Here is a comprehensive list of all of Lithps functionality:

- head: Given a list l returns the head of l.
- tail: Given a list l returns the tail of l.
- if statements.
- functional declaration and application (with recursion!).

3 Implementation

3.1 Implementation Language

I'm by no means fluent or even good at Haskell. I just enjoy functional programming which is one reason I chose to do this in Haskell. When I got introduced to Haskell programming I really enjoyed it as it taught me a new way of programming and a new way to think. I've been fond over Haskell ever since then and chose to write Lithp in Haskell due to that fascination and fondness. I also wanted to pick up concepts in Haskell which I don't understand such as Monads. I had two main issues with this idea though. First off I didn't really need Monads besides Parsing and IO. The other issue is that although I did get to watch some lectures and read some notes on Monads, I still didn't really have that much time which could be dedicated to monads.

3.2 Parsing

The parsing part of Lithp is ”inspired” by ”Write yourself a scheme in 48 hours tutorial”. By inspired I obviously mean shamelessly ripped off. Although I’ve gone through the entire thing slowly to understand the monads at work.

`https://en.wikibooks.org/wiki/Write_Yourself_a_Scheme_in_48_Hours/Parsing`

Note: Lithp and the end product of that tutorial are implemented vastly different. The tutorial actually provides a decent programming language. The only parts that are similar are the parsing.