Matt Burton

CSC 4510

10/4/23

**Functional and Logical Languages**

Haskell and Prolog are based on two different programming paradigms, each with its own advantages and limitations. Haskell is a functional language which embraces immutability and pure functions, making it a powerful choice when correctness is the top priority. Its syntax is not entirely obvious to a new programmer, but its expressive structure makes sense after diggin into its function declaration, parameter passing, and keywords. Prolog is a logical language used for rule-based problem-solving and knowledge representation. The declarative nature of Prolog makes expressing complex relations and constraints simple and easy to pick up. However debugging can be increasingly difficult due to the repetitive declarations and predicates.

These two paradigms offer different advantages over traditional sequential programs, and the option to use these paradigms when suitable can dramatically simplify and reduce the code required to solve certain problems. Having a deep understanding of how different paradigms are translated, enforced, optimized, and executed is essential for effectively applying different tools and strategies when programming.

In terms of documentation, getting started with Haskell on my personal machine was difficult for me. I downloaded Haskell-stack using Homebrew and initialized a new project using stack new <project name> simple. However, as I tried to change the project to support two Hello World programs, I quickly ran into issues. After much trial and error, I checked Spock and with apropos Haskell I found the runghc command. I created two distinct Haskell programs, compiled them, and ran them using runghc program.hs. As far as documentation goes, the information I found online mislead me more than it helped me. After pivoting away from using Haskell stack, the issues I had resolved. Moving beyond compilation, the Haskell website linked to a useful ebook which explained the syntax and some basic keywords. When working on the advanced program, I used chatGPT to learn about more complicated features of the language like the concatMap which allowed me to build strings based on matched conditions.

The documentation for prolog was simple and straightforward for writing the knowledgebase, however I trouble when compiling my program. I knew I had to load the program into the current gprolog before querying, however I was getting this error: uncaught exception: error(type\_error(atom,family-tree),consult/1) because the file name had a hyphen in it. The documentation I found in Daniel Diaz’s, *GNU PROLOG,* helped me understand the type error, and after watching a video on how to compile a Prolog file, I renamed the file to just one word and it resolved the issue. In the section where I declared the predicates, I got the following error because I declared the names without single quotes: warning: singleton variables [Allen] for male/1. The documentation I found on this error revealed that without single quotes the names were interpreted as variables not constants. Additionally, I input the predicates in the order of the family tree which produced this error: warning: discontiguous predicate male/1 - clause ignored. Simply reordering the predicates into groups of similar types fixed this problem. Overall, the documentation available online was sufficient when it comes to writing programs, but confusing for beginners who are not familiar with how to compile or what errors may occur when doing so.

**Works Cited**

Abrahamson, J. “What Exactly Makes the Haskell Type System so Revered?” *Software Engineering Stack Exchange*, 17 Apr. 2015, [www.softwareengineering.stackexchange.com/questions/279316/what-exactly-makes-the-haskell-type-system-so-revered-vs-say-java](http://www.softwareengineering.stackexchange.com/questions/279316/what-exactly-makes-the-haskell-type-system-so-revered-vs-say-java).

Diaz, Daniel. *GNU PROLOG a Native Prolog Compiler with Constraint Solving over Finite Domains Edition 1.50, for GNU Prolog Version 1*. 2021.

“Expert Systems in Prolog.” *Www.metalevel.at*, [www.metalevel.at/prolog/expertsystems#:~:text=An%20expert%20system%20emulates%20the](http://www.metalevel.at/prolog/expertsystems#:~:text=An%20expert%20system%20emulates%20the).