Prolog Evaluation

Riley Fiske – November 15th, 2023

**Initial Impressions**

Prolog is unlike any other language I have really worked with, and it took me a second to wrap my head around what was really going on. The idea of writing items, or Prolog facts, that have a type and contain different pieces of information took a second to fully understand, especially when having to then write queries, or predicates, as methods to extrapolate meaning from the attributes of the prolog fact is very logical and I enjoy it, but it is very different from the approach I am used to attacking coding problems with.

Prolog files can also serve as a query-able environment within a Prolog Compiler, such as SWI Prolog. In the terminal, a user can consult a Prolog file in the directory being pointed at and write queries on the facts within the consulted Prolog file. For the purposes of this assignment, I generated these queries with output from a batch file to show that the facts and predicates fulfill the assignment goals of identifying travelers, healthy individuals, and wealthy individuals independently.

I would like to explore Prolog more one day to see what it can do on a larger scale since we only got a glimpse of what it can do with smaller facts.

**Comparison to Imperative Languages**

***Java***

Having come from a heavy Java background, I am glad that Prolog came later in the semester as it feels, looks, and operates quite differently from Java. Firstly, there is no typing, which any Java programmer will testify is a huge part of that language and it feels odd to not type literally anything (perhaps this is supported and I simply did not encounter it). As a logical language, a Prolog file contains facts and predicates to glean information from data, which is different from a Java file that creates objects and calls methods on objects to manipulate them. Syntactically, you end statements with periods (.) in Prolog and semicolons (;) in Java; not all languages include an end-line character, so there is a similarity there. When creating a predicate, you declare its name, then give a list of statements to be true after a “:-“, ending with a period. When creating a method in Java, you encapsulate all statements with a set of curly braces {}.

***Python***

Python is a quite flexible language, and thus there are some conventions in Prolog that might feel more familiar to a Python developer. For example, in Python when creating a method or class, you place a colon (:) after the method name to state that everything following belongs to the new method or class. This syntax is similar to the “:-“ utilized by Prolog when creating predicates. The lack of braces or begin/end statements will also be pleasing to a Python developer who is used to using indentation to accomplish this. Although indentation is not required for Prolog, you can indent your program like you would a Python program to visually see what belongs to what like you do in Python.

**Screenshot of Program**

As previously mentioned, a batch file was created to call the Prolog file with the required queries pre-called and text to designate what belongs to what.

