J. Anthony Timberlake, Jonathan Luetze
CS 401 Programming Languages
March 21st, 2018

Assignment 3 : Prolog

Work Statement

The work was divided between the two programs to be written in the assignment. The first problem, in which users state facts and ask questions with the program, was completed by Anthony. The second problem, in which users tell a simulated psychiatrist bot about their problems was completed by Jonathan.

Design

In both parts, the predicate procedures are divided into main functions, keyword definitions, and print functions. Part 1 has an additional section in which rules functions for asserting and querying rules to the database are handled. The main functions section of each part is nearly identical, except for the initial prompt. Both programs are initialized by user input, “start.”. Then, in part 1, the program says, “Say something!” and in part 2, the program asks “What is your problem?” before going into the rest of the functionality of the program itself. Both programs act as an infinite loop that can only be exited by an empty input from the user.

For part 1, the string is parsed and sent to getKeyWord, similarly to part 2. However, there are only 3 cases here, thingIsThing, personIs, and isPerson.These are the three procedures which are defined within the keyword definitions section of the program. Each of them use sub_atom to parse each of the three cases mentioned in the assignment. PersonIs takes a sentence stating that a person is a type of thing as Name and Thing and sends it to the add_rule procedure in the rules function section in order to create a term with those arguments and assert it at the top of the database. Similarly, thingIsThing sends Thing1 and Thing2 as arguments to inheritance_rule, which links types of objects as inheritance, and isPerson sends the arguments Name and Thing to check_rule, where the database is checked for existing rules involving the name of a person and the type of object associated in the question. Any ineritance and additional querying is handled in this method as inheritance is already pointed to within inheritance_rule.

For part 2, the initial structure is the same, but there are twice as many procedures to choose from within getKeyWord. ThisIs handles responses in which the user responds to a question by starting a sentence with “This is...”. The response from the program is to ask what else the user considers to be whatever the user said after “this is”. Family checks the input string from the user for keywords relating to family or kinds of family members and then calls printFamily, which says “Tell me more about your family”. Why handles questions the user asks beginning with “Why”, which are met with sarcastic responses from the program. Feeling responds to the user input of “I don’t like you” with “I guess it’s a mutual dislike. What do you like?”. School will respond to user input about school and ask about what the user’s favorite subject is. If the user answers something involving a given subject (math, science, history, or programming), the program will ask why the user likes the given subject. Finally, continue prompts the user to tell the program more with “I see. Please continue.”.

The remaining portions of the program are the print functions which are handled in a self-explanatory means in each part. In addition the inputs provided for part 1, the program also accounts for differing indefinite articles such as “a” or “an” in whichever portion of the parsed string they may occur. For this reason, there is a great deal of repeated code in each of the keyword definitions procedures of part 1.

Difficulties

The primary difficulties encountered in this language were with learning the language itself and trying to parse the string or do other things that are more easily handled in procedural languages with logic flow.

Pros of this Language

Once the syntax becomes familiar, priority levels within the database are easy to differentiate with asserta and assertz. The levels of priority are also easy to understand and allows saving code when you can just say call 3 different possible functions with the same argument depending merely on whichever will return a true value first. These were really the only pros of the language I could find.

Cons of this Language

Prolog is a very difficult language to work with syntactically. While it has dynamic typing, it is very particular about what inputs it accepts, and working with strings was also very difficult as initially we could only send in integers and values without spaces to our programs. Using sub_atom may or may not have been the best way to parse our string inputs, but it was very obtuse and difficult instead of just using a more natural means of String parsing in another language like Java for example. Use of the OR operator (;) also created a great deal of unnecessary code, since it evaluates all statements which come before it as a whole instead of just allowing an initial branch. While this was attempted in the keyword functions of part 1 initially, the way that cases are evaluated in prolog with sub_atom being what defines when a given procedure is even reached. It is very much a trial and error approach, which seems roundabout and unnecessary. Asserting rules to the database was relatively simple, but finding information that was helpful online for learning the language itself was hard to come by. Most of the documentation is assuming more technical and background information of the language than I felt we really had by starting on this project. Finally, prolog is not widely used in industry. There are simpler declarative database languages like SQL for asserting and querying information in database tables and many procedural languages which interface with SQL that can be used directly instead of ever bothering with something as cumbersome and awkward as prolog.

Test Cases

Part 1 :

```
1 ?- start.
Say something!
|: John is a man.
Okay.
Is John a man?
Yes.
A man is a person.
Okay.
Is John a person?
Yes.
Is Mary a person?
No.
Is Mary a woman?
Unknown.
An a is a b.
Okay.
A b is a c.
Okay.
A c is a d.
Okay.
A d is an e.
Okay.
An e is an f.
Okay.
Jacob is an a.
Okay.
Is Jacob an f?
Yes.

Exiting Program
true .
```

Part 2 :

```
?- start.
What is your problem?
|: This is too much work.
What else do you regard as too much work?
|: Writing letters.
I see. Please continue.
|: Also washing my mother's car.
Tell me more about your family.
|: Why should I?
Why should you what?
|: I don't like you.
I guess it's a mutual dislike. What do you like?
|: Going to school.
Tell me more about being a student. What is your favorite subject?
|: Programming.
Why do you like Programming?
|: Because I enjoy creating new things.
I see. Please continue.
|: why?
Because I said so.
|: why should I answer a machine?
Why should you not answer a machine?
|: because I prefer talking to my brother.
Tell me more about your family.
|: They're nice.
I see. Please continue.
|:
Exiting Program
true .

?-
```