CS570 – Artificial Intelligence Project 4

Chihsiang Wang

Genealogy with Prolog

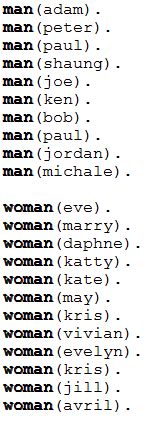
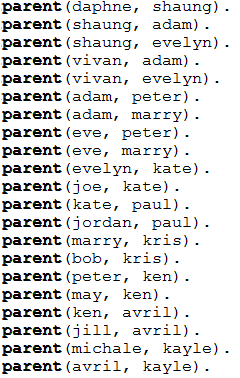
By Chihsiang Wang

**Abstract**

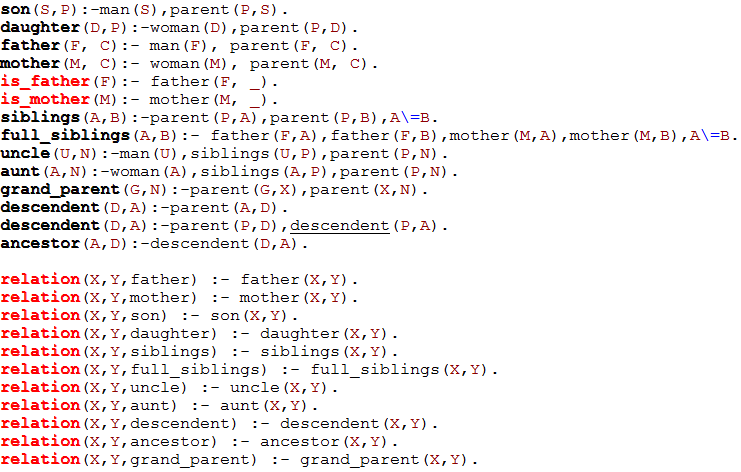
In this project I am going to use Prolog language to implement a genealogy. To create a genealogy, I utilized Prolog, a general-purpose logic programming language. The Prolog program describes the relationships between two or more objects with the help of two clauses: facts and rules. Two kinds of facts are given to define the genders of each person in the genealogy. For the male gender, ‘man (name)’ is used, while ‘woman (name)’ is used for the female gender. To determine parentage, ‘parent (name1, name2)’ is used to show that name1 is the parent of name2. Combining the different facts, I can create various rules to define the relationship between each person. These are the relationships I determined for this project: son, daughter, father, mother siblings (same mother but different father, or vice versa), full\_silbings (same mother and father), uncle, aunt, grandparent, descendant, and ancestor. After building the knowledge base, I can ask a question and receive an answer (true or false). I can also ask to list all the names in the genealogy—which is desired goal. The results of Prolog always present satisfactory answers – nothing can be answered incorrectly.

**Knowledge Base**

There are three kinds of facts utilized in the program: man, woman, and who is whose parent. These three facts are enough to build a complete genealogy. The genealogy graph can be seen on the appendix page.

 **Figure 1. Facts.**

Applies by these facts, I can build other relation rules; it’s pretty logic things. In this project I build 25 rules, it shows as figure. From these three facts, I can build a multitude of relationship rules. The figure below shows the 25 rules I have built.

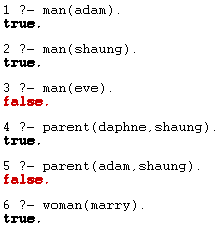


**Figure 2. Rules.**

After building a knowledge base, the Prolog language will be able to use logical reasoning to connect each person. The program is now ready for me to ask questions.

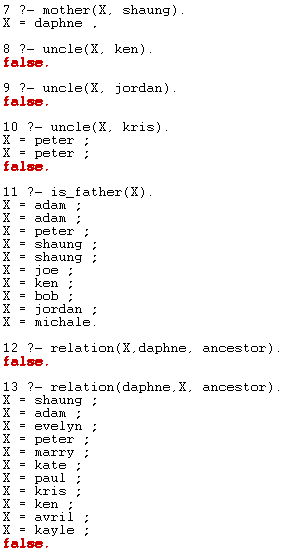
**Result**

To begin, simple questions can be asked to receive true or false answers. The sample output list is numbered 1 through 6.



1. Is Adam a man? Answer: true.
2. Is Shaung a man? Answer: true.
3. Is Eve a man? Answer: false.
4. Is Daphne a parent of Shaung? Answer: true.
5. Is Adam a parent of Shaung? Answer: false.
6. Is Marry a woman? Answer: true.

Or I can use a variable to show what can make rules to be true, lists number 7 – 13.



7. Who is mother of Shaung? Answer: Daphne.

8. Who is uncle of Ken? Answer: false, Ken has no uncle.

9. Who is uncle of Jordan? Answer: false.

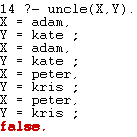
10. Who is uncle of Kris? Answer: Peter.

11. Who is a father? Answer: Adam, Peter, Shaung, Joe, Ken, Bob,Jordan.

12. Who is ancestor of Daphne? Answer: No one.

13. Who is Daphne’s ancestor? Answer: Shaung, Adam, Evelyn, Peter, Marry, Kate, Paul, Kris, Ken, Avril, Kayle.

When the answer is more than one, I will need to use a “;”to show each answer. If there are no answer, the compile will return false. I also can use two variables to receive complex answers. For example, I use uncle(X,Y) to show that who is whose uncle in a whole genealogy.

****

14. Adam is Kate’s uncle, Peter is Kris’s uncle

**Discussion**

Prolog language is really different from other programming languages like C, Java, and Python. It uses no if/else and for/while statements. Prolog’s programming style is more human-like thinking and logic reasoning. A knowledge base determines how good a system works. As a result, the correct facts and rules are most important settings. I have to be meticulous while operating a prolog program, because just one mistake or typo can destroy the whole logic. Facts build rules, and facts are also rules. How to use the facts to create rules and use these rules to create more rules, is the trickiest part while coding. When I first used the program, I wrote too many facts; for example, defining the relationship between “father” and “mother” consists of an enormous amount of hard typing. After finding some references about Prolog, I understood how to use facts to build rules. As a result, I modified my knowledge base to only contain three kinds of facts—which is enough for a successful project. Prolog is a very logical programming language, but a code designer will need good and clear logical thinking as well. After building a good knowledge base, the answers received can always be satisfactory—outputs of 100% accurate answers.

**Appendix**

The figure 1 and 2 shows all of my knowledge base.

The genealogy graph list below:

