**CS570 Artificial Intelligence**

**Spring 2012**

**Project #4 Prolog**

**Sanqing Yuan**

**May 6**

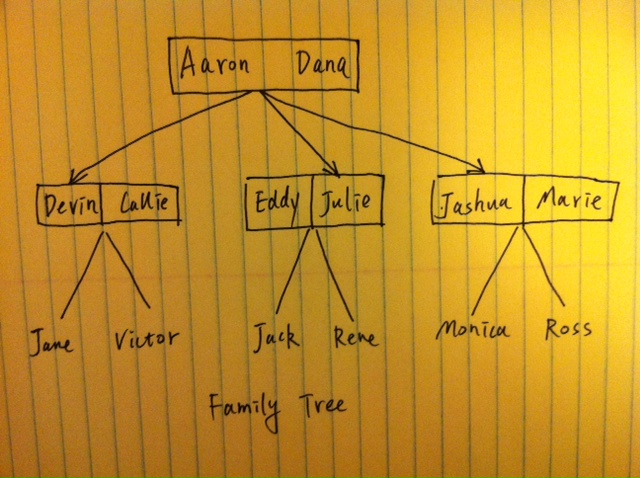
**Implementation of a family tree using prolog**

**Abstract**

In this project, I used a logic programming language “prolog” to create and define a family tree. A simple knowledge base, including information regarding members’ parent and gender, was included in the prolog program. Also some inferring rules were used to define the relationships between the family members. Based on the testing results, this prolog program is able to answer all the questions I can ask. For example, it can give you the relationship between any two members in the family tree. It can list all the correct answers to complex queries. All possible relationships are provided and available for query in the program, including sibling in law, son in law, uncle, aunt and cousin.

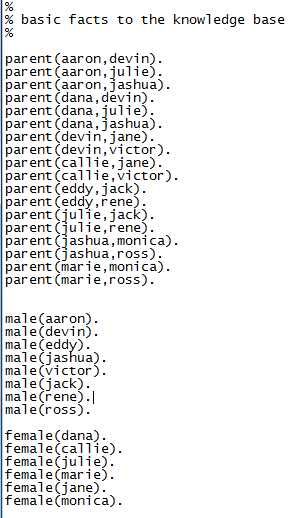
**Knowledge base**

In the knowledge base, only the basic facts are given. In this case, the only information is the parent relationship between two members (who is whose parent) and members’ gender. Actually this information is already enough to define all different kinds of other relationships. Below (Figure 1) is the family tree, which contains 14 members and 4 couples.



**Figure 1** The family tree used in this project

The knowledge base in the program is shown below in Figure 2. Note that the first person in the parenthesis is the parent of the second person.



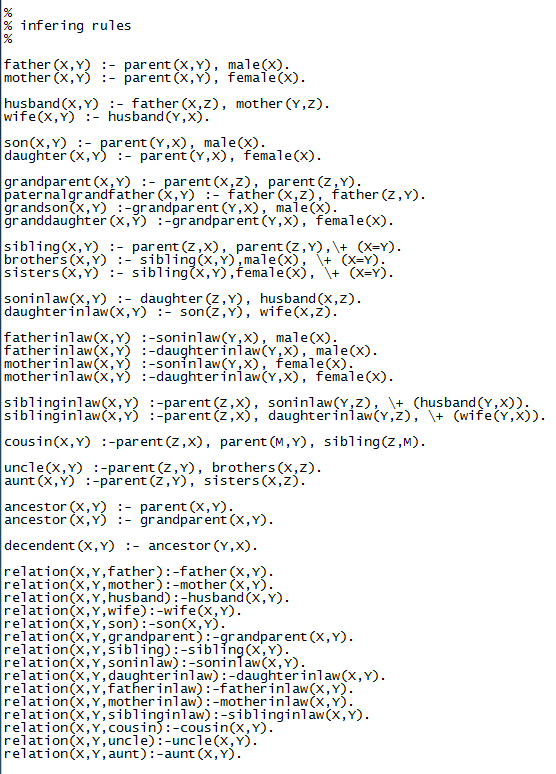
**Figure 2** The knowledge base in prolog program

**Inferring rules**

In this project, 23 different kinds of relationships were defined. Some relationships are explained as follows:

* sibling in law: includes brother in law and sister in law. Those are persons that marry to their brother or sister.
* ancestor: a member of your family who lived in past times.
* decendent: the traditions of their ancestors.

More details can be found in Figure 3. Note that in all inferring rules, the first variable denotes the role of the rule name. For example*, father(X,Y)* means X is Y’s father. *soninlaw(X,Y)* means that X is Y’s son in law. *relation(X,Y,wife)* means X is Y’s wife.

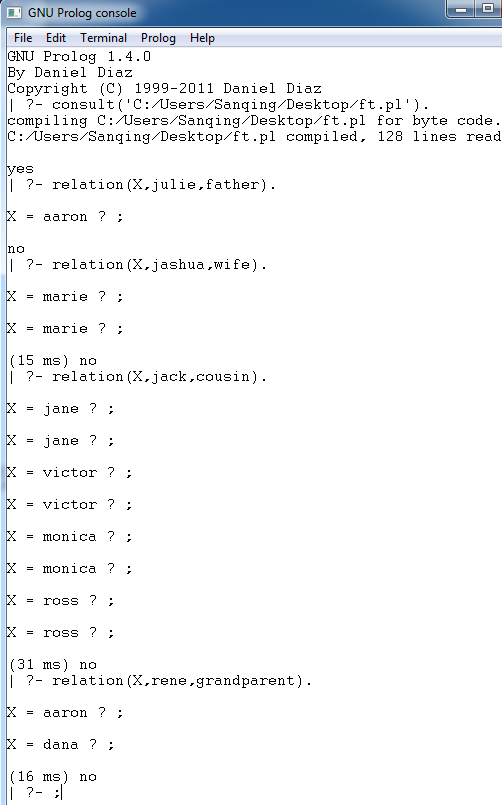


**Figure 3** Inferring rules to define relationships

**Results**

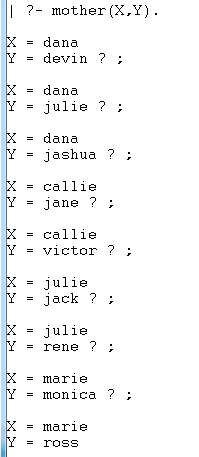
Simple queries

* Who is Julie’s father?
* Who is Jashua’s wife?
* Who is Jack’s cousin?
* Who is Rene’s grandparent?

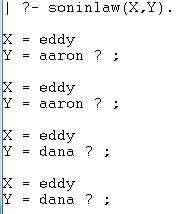


Complex queries

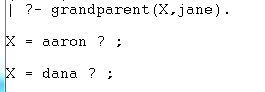
* List all of the mothers



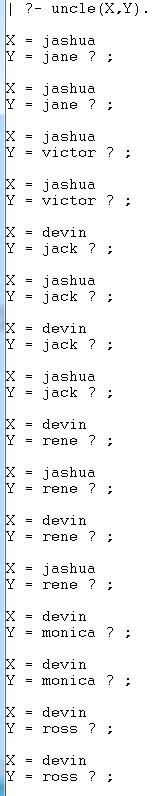
* List all sons in law



* List all grandparent of Jane



* List all uncles



As I mentioned in abstract, I didn’t find any queries that can’t be answered correctly in this program.

**Discussion**

In this project, a relatively simple knowledge base was used. Only the very basic facts were listed in the knowledge base. Although it is quite simple, the information is enough to define all the other relationships in the family tree. For a simple family tree like the one in this project, we may not see the advantage of a simple knowledge base. For example, you can easily see the relationships between any two members in the family tree since the tree is very simple, and put them in the knowledge base. In this way, much less inferring rules might be used because the information is already in the knowledge base. However, for a huge family tree, we don’t want to put a lot of information in the knowledge base because it is complicated and time-consuming. Instead we can put the basic facts like the one in this project (parent and sex) and let the inferring rules do their jobs to define all the other relationships.

One thing I’m not totally satisfied is that for some queries, I get the same correct answers multiple times (see the outputs above for examples). I think my codes can be improved to solve this problem.

**Appendix for knowledge base**

Please see Figure 2 for the full knowledge base.