CSC 261 **Lab 7: First-Order Logic Programming** **Fall 2011**

Assigned: Tuesday 11 October 2011

Due: Monday 24 October 2011, 11:59 p.m.

Objectives:

- Understand first-order relations
- Practice using logic programming for inference

Collaboration: This laboratory will be completed in pairs of your choice.

In the following questions, you will create a single Prolog program. Your answers may be written in the file in the order the questions are assigned. Please indicate which portion of the file is relevant to which question through the use of comments (using the `%` symbol).

Recall that variables in Prolog are given in capital letters, while constants are lower case letters.

Problem 1

Primitive predicates are those that are used *only* as facts in our knowledge base, rather than as the head of some rule (though they may be part of the bodies of rules). Let `male(X)` and `female(X)` be primitive unary predicates, with `child(C,P)` and `married(H,W)` as primitive binary predicates. Note that `married` and `child` are *not* symmetric; the former takes the husband first with the wife second, while the latter the child first with the parent second.

Using these primitives, we can write the many Prolog rules to express familial relationships.[1] For example,

```
parent(P,C) :- child(C,P).
father(F,C) :- male(F), parent(F,C).
```

Along with `parent(P,C)` and `father(F,C)` given above, express the following familial relationships in a Prolog program. The bodies of your clauses should use as few terms as possible.

`mother(M,C) :-`	`wife(W,H) :-`
`son(S,P) :-`	`husband(H,W) :-`
`daughter(D,P) :-`	`sibling(X,Y) :-`
`grandfather(G,C) :-`	`brother(B,X) :-`
`grandmother(G,C) :-`	`sister(S,X) :-`
`grandson(S,G) :-`	`uncle(U,X) :-`
`granddaughter(D,G) :-`	`aunt(A,X) :-`

Problem 2

The song "I'm My Own Grandpa" (Latham/Jaffe) debuted in 1947. It was summarized by Niklaus Wirth as follows:

> I married a widow (call her w) who has a grown-up daughter (d). My father (f), who visited us quite often, fell in love with my step-daughter and married her. Hence my father became my son-in-law and my stepdaughter became my mother. Some months later, my wife gave birth to a son ($s1$), who became the brother-in-law of my father, as well as my uncle. The wife of my father, that is, my step-daughter, also had a son ($s2$).
>
> *Algorithms + Data Structures = Programs, 1976*

[1] That is, at least those familial relationships that were thought of as "traditional" in one day and age; some states, countries, and provinces may have more modern definitions. For the purposes of making your assignment easier, we'll stick to the traditional ones here.

Part A Step-relations shouldn't have been reflected in your rules above. Add to your Prolog program two new rules for `parent` that take this into account. (Recall that the `married` relation is not symmetric). Note that Prolog requires rules with the same head to be adjacent to each other.

Part B There are many potential in-law relationships, but the one we're interested in for this story is son-in-law. Add a new rule for `soninlaw(S,P)` that defines this relationship.

Part C Using only the four primitives from part A (`male`, `female`, `married`, and `child`), add the facts present in the story above. (Use n for the narrator of the story). Note that Prolog requires that facts using the same primitives be adjacent to each other.

Part D Using Prolog, verify the conclusions drawn by the narrator:

```
grandfather(n,n).
soninlaw(f,n).
mother(d,n).
uncle(s1,n).
```

What to turn in

Your submission should include the following

- Your completed `.pl` program file
- A single PDF containing (merged, see below) your `.pl` file and a transcript of your program's results

How to Submit Your Work

Record your Prolog file and an interaction with the prolog interpreter to a PDF as follows:

1. To begin recording your session, type

   ```
   script output.txt
   ```

2. To echo your program to the file, type

   ```
   cat program.pl
   ```

3. Start the Prolog interpreter, load your program, and use the interpreter to verify the conclusions given in Problem 2(D).
4. Exit the Prolog interpreter by typing

   ```
   halt.
   ```

5. Stop the script session by typing `Ctrl-D`
6. Save the record of your session to a PDF by typing

   ```
   enscript -2 -r -o - output.txt | ps2pdf - submission.pdf
   ```

Acknowledgments

Adapted from P. Norvig, *Paradigms of Artificial Intelligence Programming*, Morgan-Kaufmann, 1992, Exercises 11.9 and 11.10 (p. 385).