Assignment #1

Date Assigned: Friday, Jan. 23, 1998
Date Due: Friday, Feb. 6, 1998

---

1. (Small warm up exercises in PROLOG)

   - (i) This problem will help you to switch between the RDBMS and PROLOG modes of thinking about data models and database systems. Consider the relational database schema:

     lives (person-name, street, city)
     works (person-name, company-name, salary)
     located-in (company-name, city)
     manages (person-name, manager-name)

     Write queries (in predicate logic) to:
     - Find the name, street, and city of all employees who work for First Bank Corporation and earn more than $10,000.
     - Find all employees who live in the same city and on the same street as their manager
     - Find all employees who earn more than every employee of Small Bank Corporation
     - Find all employees working (directly or indirectly) under the supervision of "Kathy"

   - (ii) Can all of the above queries be expressed by SQL? (i.e., can we just use the relational model of databases to express all this functionality) Why/Why not?

   - (iii) Consider the following PROLOG clauses:

     ```
     ancestor(X,Y) :- ancestor(Z,Y), parent(X,Z).
     ancestor(X,X).
     parent(amy,bob).
     ```

     Why does PROLOG fail to answer (amy) for the query `ancestor(X,bob)`, even though amy is an ancestor of bob?

---

2. One major use of PROLOG is to create diagnostic expert systems. For example, if there is no fuel in your car, then it will not start. If the car doesn't start, we would like to diagnose that fault and perhaps, hypothesize that there is no gas, assuming we have no reason to believe otherwise (maybe you just filled the tank). So, assume that we say:

   ```
   nostart(X) :- nogas(X).
   nostart(pontiac).
   ```

   Is it possible to make PROLOG infer that `pontiac` has no gas? If so, how would you do it? If not, how would you go about doing this kind of "abductive" inference?

---

3. (Reverse Engineering) Here's a logical description of Euclid's algorithm for finding the greatest common divisor (gcd) of two numbers:

   - The gcd of u and 0 is u.
   - The gcd of u and v, if v is not 0, is the same as the gcd of v and the remainder of dividing v into u.

   and here's one PROLOG version of it:

   ```
   gcd(U,0,U).
   gcd(U,V,W) :- not(V=0), R is U mod V, gcd(V,R,W).
   ```

   Of course, there is a lot of syntactic sugar in this, but the basic idea is the same. Alternatively, the last rule can be thought of as:

   ```
   gcd(U,V,W) :- notzero(V), modval(U,V,R), gcd(V,R,W).
   ```

   if you find this more intuitively appealing (But take care to note that **`notzero`** and **`modval`** need to be appropriately defined by you).

   The task of this problem is two fold :

   - (i) Use an ILP system like PROGOL to mine the above two rules!! This means that you will have to think of how the data is to be specified to PROGOL, positive examples of gcd, negative examples of gcd, background knowledge (this is the tricky part) and the header information. This should not be too difficult because you know what exactly you are mining for! (This exercise will demonstrate that you can use induction to "synthesize" logic programs) This is similar to the Eleusis card game of induction. (Someone else plays the game and you determine the rules! :-))

   - (ii) Now note down the support and the confidence (%) for each one of the two rules (Recall that support and confidence are based solely on the data that you supply to PROGOL). Now start decreasing the number of positive and negative examples steadily (say, in units of one). What parameters of PROGOL (look in Chapter 5 of the manual) do you now have to tune to obtain the same rules with the same confidence levels? (This exercise will give you an understanding of how the various parameters affect the process of rule induction). A simple graph will help you to see the trend.

---

4. Conduct data mining on the Car Evaluation Database (it's really a file) from the UCI Machine Learning Database Repository (Scroll down the page to get to this database). The purpose of this database is to record car prices, technical and safety features and use this information to relate to the car's acceptability in the market. You can also obtain it from the ftp address **`ftp://ftp.ics.uci.edu/pub/machine-learning-databases/car/`** (use the files **`car.names`** and **`car.data`**). Notice that you will have to change the format of this file from its present form to the format required by PROGOL (a knowledge of UNIX shell scripting would come in useful for this).

- (i) Try to relate the car acceptability to the different attributes. As mentioned in the **`car.names`** file, structural information about features has been removed from the file. i.e., there is nothing to specify that the buying price and the maintenance price are both "prices" and so on.. How would you incorporate this structural information into your data mining process? Do you need this at all to do data mining? Notice that some of the attribute information is numeric: How would you model this aspect using PROGOL?

- (ii) What would be the methodology that you would use? How would you form the training set for PROGOL, and how would you validate your results, in terms of accuracy, generalization etc.? Is there a way to "intelligently" construct the training and test sets from what you know about this database?

- (iii) The paradigm of ILP, as used by PROGOL is probably an overkill because we are trying to do data mining with a single table, as opposed to multiple tables in a real-world database. Nevertheless, can you state some advantages of trying this approach on this simplistic domain?

Return Home