Programming Project 7

Due date: Mon 12/8 11:00 pm

1 Introduction

In this assignment you will handle simple data and list data with Prolog programs. An extra credit part solves a similar problem using higher order functions in ML.

2 Paths on Graphs

In this part we work with labeled directed acyclic graphs using the following Prolog representation. The atom $e(a,b)$ indicates that there is an edge from $a$ to $b$ in the graph. Each node in the graph is also given a single color using the appropriate predicates.

$$e(1,2).$$
$$e(2,3).$$
$$e(3,4).$$
$$e(4,5).$$
$$e(1,3).$$
$$e(2,5).$$

$$\text{blue}(1).$$
$$\text{green}(2).$$
$$\text{black}(3).$$
$$\text{red}(4).$$
$$\text{black}(5).$$

While we illustrate the requirements for the graph above your program should work for any graph given in this format.

Part I: write a function $\text{path1}(X,Y)$ that is true whenever there is a path from $X$ to $Y$ in the graph. For example

?- $\text{path1}(1,5).$
Yes
?- $\text{path1}(1,3).$
Yes
?- $\text{path1}(3,5).$
Yes
?- $\text{path1}(3,1).$
No

Part II: write a function $\text{path2}(X,Y)$ that is true whenever there is a path from $X$ to $Y$ in the graph which does not use an edge whose endpoints are colored red--> black. For example

?- $\text{path2}(1,5).$
Yes
?- $\text{path2}(1,3).$
Yes
?- $\text{path2}(3,5).$
No
Part III: write a function \texttt{path3(X,Y)} that is true whenever there is a path from \(X\) to \(Y\) in the graph which \textit{does} use an edge whose endpoints are colored \texttt{red--> black}. For example

?- path3(1,5).
Yes
?- path3(1,3).
No
?- path3(3,5).
Yes

3 Working with Lists

In this part you will work with a database capturing people and the CDs they own. Each CD is given a name and its main artist is listed. The data is given as a list in the following format:

\[
[\text{[john,[[cd1,clapton],[cd2,moody],[cd3,hip]]],}
\text{[janet,[cd3,hip],[cd4,moody],[cd5,u2]],}
\text{[beatrice,[cd6,doors],[cd4,clapton],[cd3,hip]],}
\text{[robert,[cd7,clapton],[cd3,hip],[cd6,doors]]}]
\]

Each element of the main list represents a person. The person’s name is the first element. Following elements are lists of length 2, each giving a CD name and the artist. As in the previous part we illustrate the requirements using this database but your programs should work for any database in this format.

Part I: write a function \texttt{similarTaste(Data,P1,P2)} that is true when \texttt{Data} is a database as above and \(P_1, P_2\) are two different people who own at least 2 different CDs in common. Notice that the 2 CDs do not need to appear in the same order in the lists. For the data above this holds for janet and beatrice and for beatrice and robert. You do not need to worry about repeated answers.

You may want to break down the task into simpler tasks. For example you can define a function \texttt{ownsCD(Data,Person,CD)} that is true when \texttt{Person} owns the \texttt{CD}. But this is up to you.

Part II: write a function \texttt{sameArtists(Data,P1,P2)} that is true when \texttt{Data} is a database as above and \(P_1, P_2\) are two different people who who have CDs by exactly the same artists. Again the CD lists do not need to be identical - they need to include the same set of elements. For the data above this holds only for beatrice and robert. You do not need to worry about repeated answers.

Here again the structure of the code is up yo you. A useful sub-task is a function \texttt{personArtists(Data,Person,Artists)} that is true when \texttt{Artists} is the list of artists whose CDs are owned by \texttt{Person}.

4 Extra Credit: Higher Order Functions in ML

This part is optional; you can earn up to 20 extra points for doing this part. Here we try to solve the same task as in part II of the previous section. We represent the database in a slightly different format in ML as follows:

```ml
val a = [
  ("john",[["cd1","clapton"],"cd2","moody"],["cd3","hip"])),
  ("janet",[["cd3","hip"],["cd4","moody"],["cd5","u2"])),
  ("beatrice",[["cd6","doors"],["cd4","clapton"],["cd3","hip"])),
  ("robert",[["cd7","clapton"],["cd3","hip"],["cd6","doors"]])
];
```
Write a function \texttt{sameArtists} with the following type:

\begin{verbatim}
val sameArtists = fn : (string * string list list) list -> (string * string) list
\end{verbatim}

(a more general type replacing string with polymorphic values is OK). Given a database as above the function should return all pairs of people that satisfy the same-taste criterion from the previous part. You should avoid duplicates. For example the call

\begin{verbatim}
val R = sameArtists a;
\end{verbatim}

should return

\begin{verbatim}
val R = ["beatrice","robert"] : (string * string) list
\end{verbatim}

NOTE: your code should only use higher order functions to handle lists. Code using a good amount of higher order functions and some regular functions will receive partial credit. Code using only regular functions will not get credit.

5 Files

For your convenience seed files for \texttt{pp7.pl} and \texttt{pp7.sml} including the data given above are available at 
\texttt{/comp/80/files/pp7/}

Submitting your program

Put all the Prolog code in a file \texttt{pp7.pl} and if submitting the ML code put it in \texttt{pp7.sml}. Assuming your files are in the current directory on one of our Sun workstations or servers, you should submit by typing 
\begin{verbatim}
provide comp80 pp7 pp7.pl
\end{verbatim}
or if submitting the extra credit part as well
\begin{verbatim}
provide comp80 pp7 pp7.pl pp7.sml
\end{verbatim}