Worksheet 3: Non-deterministic Prolog Programs

Due: Midnight, Friday, 4 March 2010.
Total marks: 40

Instructions

If you detect or suspect any errors, post a question on the Worksheet 2 discussion forum for clarification, or email the instructor!

The Worksheet should be solvable in 10 to 12 hours of work. If after two hours (not including reading the text and notes), you have not made significant progress, waste no further time, and consult with the TA or the instructor.

Question 1 and 2 consist of a couple of search trees. These can be drawn electronically, but be warned that drawing by hand, using Good Old Fashioned Pencil and Paper (GOFPP) may be up to 10 times faster. You may hand in your search trees on Tuesday 1 March in lecture at 10am. After 10am, the submission will be considered late.

Submit the rest of your solutions electronically using the CMPT 340 Moodle page for Worksheet 3.

Question 1: If a tree falls (10 marks)

1. Consider the following program.

```
h(X,1) :-
   X < 1,
   X > 0.
h(X,K) :-
   X >= 1,
   Y is X / 2.0,
   h(Y,L),
   K is L + 1.
```

   Draw a search tree for the query

```
?- h(3,3).
```

2. Consider the program `append/3`.

```
append([],L,L).
append([X|Xs],Ys,[X|Zs]) :-
   append(Xs, Ys, Zs).
```

   Draw a search tree for the query

```
?- append(A,B,[1,2,3]).
```

   There are 3 answers, and your search tree should show them all.

Evaluation

Marking will be based on the following attributes:

- Showing all branches, even DNU branches. The search demonstrates that you know how Prolog handles rules and facts.
- All solutions are shown, not just the first. You show that you understand backtracking.
- Every non-DNU branch must be labelled with a MGU. This shows you understand unification.
- The resolvent is correctly updated at each node.
- All steps are shown. Don't skip any!
- Legibility. If the marker canot read it, it cannot be graded.

The following will not be penalized

- Minor typographical errors.
- Minor omissions (e.g., commas, abbreviations, etc).

What to hand in

Two search trees. It is advisable to draw these by hand, on paper, and submit it to Mike in the lecture on 1 March at 10am. but electronic submissions are perfectly fine (they may take 10 times longer to do). If you hand it in electronically, use the file names `w3q1a` and `w3q1b`, with an appropriate file-type suffix.

Question 2: Half it your way (12 marks)

Write a program called `splitInHalf/3`, which takes a list as input, and splits it in half (roughly: if the list is odd length, one of the returned lists can be longer than the other by one element). You'll do two versions, as follows:

1. In this part of the question, you will write a deterministic program. I know you want to write this, because that's the way you have been taught to program in all your other programming experience. So get it out of the way here.

   There are several ways to split a list. One way is to use a counter. Another way is to remove an element from the original list, one at a time, and alternate dropping it into 2 other lists. The only requirement is that you do not use non-determinism in any way.

   Show that your program works by copy/paste a demo into a textfile.

2. The work you did on Question 1 should help you imagine ways of using `append/3`.

   Write a version of the `splitInHalf/3` program that uses `append/3` non-deterministically.

   Hint: the program `length/2` is commonly used to count the number of elements in a list, but it can also be used to create a list of a given length:

```
| ?- length(L,3).
L = [_A,_B,_C] ?
yes
```

Note that the definition of `length/2` that I gave in class cannot do this, but the built-in versions of `length/2` can.

Show that your program works by copy/paste a demo into a textfile, which you should submit with your homework.

Evaluation

Marking will be based on the following attributes:

- Format: 1 mark each. A mark will be deducted if the format of your program is unacceptable, even if it is correct.
- Testing: 1 mark each. This mark will be deducted if you do not include a text file that shows your program working.
- Correct implementation (1 mark each). A mark will be deducted if there are rules or facts that are redundant or unnecessary. The mark may be deducted if other errors are found, as well.
- Non-determinism: 2 marks each Two marks will be deducted if your program for part 1 is non-deterministic, and 2 marks will be deducted from part 2 if your program is deterministic.
- Documentation: 1 mark each. The documentation should include the informal contract, and a description of the variables and the relationship being defined.

What to hand in

Your implementation of the predicates above, in a file called `w3q2.pl`. Make sure that you put student information in the file, and document anything that you might think will help the marker read your program. Also submit your testing (a few queries for each program, showing that it works) in a file called `w3q2_testing.txt`.

Question 3: Eight times (8 marks)

Write a Prolog program to determine the identity of two 5-digit numbers X and Y with the following properties:

- The digits in X are unique (no repetitions)
- The digits in Y are unique (no repetitions)
- The numbers X and Y share no digits in common.
- The leading digits of X and Y are not zero.
- $Y = 8 \times X$

- The sum of the digits in X is greater than the sum of the digits in Y.

Copy/paste from the Prolog interaction to show your program working, and the 2 numbers your program finds.

You may use the program `select/3`. The definition is

```
% select(?X,?List,?Remaining)
% select X from the List, leaving Remaining
select(X,[X|L],L).
select(X,[Y|L],[Y|M]) :-
   select(X,L,M).
```

In SWI Prolog, `select/3` is pre-defined, exactly as above. For SICStus Prolog, you have to load the lists library, as follows:

```
?- use_module(library(lists)).
% loading .../library/lists.po
% module lists imported into user
% loaded .../library/lists.po in module lists, 10 msec 13696 bytes
yes
```

Evaluation

Marking will be based on the following attributes:

- Format: 1 mark. A mark will be deducted if the format of your program is unacceptable, even if it is correct.
- Correct implementation: 8 marks. A mark will be deducted if there are rules or facts that are redundant or unnecessary. The mark may be deducted if other errors are found, as well. If you don't submit the text file showing your program working, a mark may be deducted. Four marks will be deducted if your program does not use Prolog's non-determinism.
- Documentation: 1 mark. The documentation should include the informal contract, and a description of the variables and the relationship being defined.

What to hand in

Your implementation of the program above, in a file called `w3q3.pl`. Make sure that you put student information in the file, and document anything that you might think will help the marker read your program. Also submit a short text file showing your program working (copy/paste from the PRolog interaction).

Question 4: A la carte (10 marks)

A certain restaurant has an appetizer menu with 6 items on it, with the following costs:

Mixed Fruit	\$2.15
French Fries	\$2.75
Side Salad	\$3.35
Hot Wings	\$3.55
Mozzarella Sticks	\$4.20
Sampler Plate	\$5.80

The problem to solve is to find combinations of these items so that the total cost is exactly a given dollar amount. For example, for \$15.05, you can get 7 orders of Mixed Fruit; or 1 order of Mixed Fruit along with 2 orders of Hot Wings and 1 order of the Sampler Plate.

Your program should return one combination at a time, and should try to find more combinations for the same cost when you press ";". You should probably use integers to represent the dollar amounts in cents, rather than using real values. It's okay if your program finds the same answer more than once.

Copy/paste a short demo of your program into a text file and submit it as well.

Evaluation

Marking will be based on the following attributes:

- Format: 1 mark. A mark will be deducted if the format of your program is unacceptable, even if it is correct.
- Correct implementation: 6 marks. A mark will be deducted if there are rules or facts that are redundant or unnecessary. The mark may be deducted if other errors are found, as well. If you don't submit the text file showing your program working, a mark may be deducted.

  Three marks will be deducted if your program does not use Prolog's non-determinism.
- Documentation: 1 mark. The documentation should include the informal contract, and a description of the variables and the relationship being defined.

What to hand in

Your implementation of the program, in a file called `w3q4.pl`. Make sure that you put student information in the file, and document anything that you might think will help the marker read your program. Also, submit your demo of your program (a textfile).