More Prolog Mini Exercises
Derivation Trees; Difference Lists; Controlling Search — Answer Key

These questions use the Prolog rules in the lecture notes (both the basics and the ones on controlling search).

1. Draw the derivation tree for the following goals:
   
   `?- reverse([1],R).`

   Please see the separate scan of the hand-drawn tree. Also try running the goal with the Prolog trace facility.

2. Consider `mymember` and also the `member_cut` rule defined in the notes on controlling search. What are all the answers that Prolog returns for the following goals?

   `?- mymember(1,[A,B,C]).`
   
   A = 1 ;
   B = 1 ;
   C = 1 ;
   false.

   `?- member_cut(1,[A,B,C]).`
   A = 1.

3. What are all the answers that Prolog returns for the following goals?

   `?- mymember(X,[1,2]), mymember(X,[0,2,2]).`
   
   X = 2 ;
   X = 2 ;
   false.

   (Note that you get the same answer twice!)

   `?- member_cut(X,[1,2]), mymember(X,[0,2,2]).`
   false.

   `?- mymember(X,[1,2]), member_cut(X,[0,2,2]).`
   X = 2 ;
   false.

   `?- member_cut(X,[1,2]), member_cut(X,[0,2,2]).`
   false.

4. What are all the answers that Prolog returns for the following goals?

   `?- not(mymember(1,[1,2,3])).`
   false.
?- not(mynmember(5,[1,2,3])).
   true.

?- not(mynmember(X,[1,2,3])).
   false.

?- mymember(X,[1,2,3]), not(mynmember(X,[1,2,4])).
   X = 3 ;
   false.

?- not(mynmember(X,[1,2,4])), mymember(X,[1,2,3]).
   false.

5. Consider the standard version of append:

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

If you know that the first argument is ground (that is, fully instantiated, containing no variables), there is a more efficient version that you can write by including a cut.

(a) Define such a version.

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

(b) Give an example of a query that has exactly the same behavior for both the standard version and the version with a cut.

append([1,2],[3,4,5],X).

(c) Give an example of a query that behaves differently for the standard version and the version with a cut.

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

(d) What restrictions do we need on the inputs for the two versions to behave exactly the same? (Is it that the first argument is ground?)

No, it’s a little more general: just that the first argument not be a variable or a list with a variable as the tail of the list.

6. Which of the following lists represent valid difference lists? For valid difference lists, what list do they represent?

[1,2|T]\T -- valid, represented [1,2]
[1,2,3]\[] -- valid, represents [1,2,3]
[1,2,3]\[1,2] -- not valid
[1,2,3|T]\[3|T] -- valid, represents [1,2]
[1,2,3]\[1,2,3] -- valid, represents []
7. Write the list [squid, clam] as a difference list (in the most general possible way). Also draw a box-and-arrow diagram of the first part of the difference list.

\[\text{[squid, clam|T]} \backslash T\]

\[
\begin{array}{c}
\text{o} \\
\text{---} \\
| \text{squid} \\
\text{clam} \\
\end{array}
\begin{array}{c}
\text{o} \\
\text{T} \\
\text{---} \\
| \text{clam} \\
\end{array}
\]

Notice that this remains a valid difference list representation of [squid, clam] no matter what we unify with T. For example, if we unify T with [octopus], we get this difference list:

\[\text{[squid, clam, octopus]} \backslash \text{[octopus]}\]

which still represents [squid, clam]. Here’s the box-and-arrow representation of what happens to [squid, clam|T]:

\[
\begin{array}{c}
\text{o} \\
\text{---} \\
| \text{squid} \\
\text{clam} \\
\end{array}
\begin{array}{c}
\text{o} \\
\text{T} \\
\text{---} \\
| \text{clam} \\
\end{array}
\begin{array}{c}
\text{o} \\
\text{T} \\
\text{---} \\
| \text{octopus} \\
\end{array}
\]

8. Using the clpr library, write a rule mymin such that if you call mymin(A, B, C), C will be the minimum of A and B.

\[
\begin{align*}
\text{mymin}(X, Y, X) & :\{X \leq Y\}. \\
\text{mymin}(X, Y, Y) & :\{X > Y\}.
\end{align*}
\]

9. Write a rule solve using the clpr library that solves the simultaneous equations \(2x + 3y = 8\) and \(x + y = 3\).

\[
\text{solve}(X, Y) :\{2X + 3Y=8, \ X+Y=3\}.
\]

10. Again using the clpr library, write a rule sum such that for sum(Xs, S), S is the sum of the numbers in the list Xs. You can assume the list consists only of numbers. For example sum([], S) should succeed with S=0.0, sum([3, 4], S) should succeed with S=7.0, and sum([A, A], 10) should succeed with A=5.0.

\[
\begin{align*}
\text{sum}([], 0). \\
\text{sum}([X|Xs], S) & :\text{sum}(Xs, S1), \{X+S1=S\}.
\end{align*}
\]