Lecture 9 (Oct 7)

Lecture outline:

- Prolog lists: membership, insertion, concatenation

First, an exercise on list unification:

Example 1. What will be Prolog's response to the following query:

$$a([H|T],[H2|T2]) = a([1,2],T).$$

To answer "confusing" questions about list unification it helps in terms of "." notation for lists. So, the above query is equivalent to

$$a(.(H,T),.(H2,T2)) = a(.(1,.(2,[])),T).$$

Applying unification algorithm, we obtain the following m.g.u:

$$[H/1,T/[2],H2/2,T2/[]]$$

List membership

Our goal is to define predicate $is\_member(X,L)$ which is true if X is a member of the list L. We will use the following recursive definition: $is\_member(X,L)$ is true if

1. X is a head of L, or
2. X is a member of the tail of L.

This definition gives us the following Prolog program (see `is_member.pl` in the Lecture Resources section):

$$is\_member(X,[X|T]).$$
$$is\_member(X,[H|T]) :- member(X,T).$$

Now, try the following queries, construct a refutation tree to see how they get answered, and test it with Prolog.

$$?- is\_member(a,[b,a,c]).$$
$$?- is\_member(X,[b,a,c]).$$
$$?- is\_member(a,X).$$

Element insertion

Lets define predicate $insert\_elem(X, L1, L2)$ which is true if $L2$ is a list obtained by inserting an element X into some position of list $L1$. We will use the following recursive definition: $insert\_elem(X, L1, L2)$ is true if

1. $L2$ is X appended to $L1$, or
2. $L2$ is obtained by taking the head off $L1$, appending X to the resulting list, and putting the head of $L1$ back.

This definition gives the following Prolog program (see `insert_elem.pl`):

$$insert\_elem(X, L1, [X|L1]).$$
$$insert\_elem(X, [H|T], [H|T2]) :- insert\_elem(X, T, T2).$$

Try the following queries, construct a refutation tree to see how they get answered, and test it with Prolog.

$$?- insert\_elem(a, [b, c], X).$$
$$?- insert\_elem(X, [b, c], [b, c, a]).$$
$$?- insert\_elem(a, X, [b, c, a, d]).$$

Note that the last query demonstrates that the *insert_elem* predicate that we defined can be use to *remove* elements from the list.

Concatenation of lists

Now we want to define predicate $append(X, Y, Z)$ which is true if Z is a list obtained by concatenation of lists X and Y. Again, the following recursive definition will do the job: $append(X, Y, Z)$ is true if

1. X is [], and Z is Y, or
2. if H is the head of X, and T is the tail, then Z has H as the head, and the result of concatenation of T and Y and the tail.

This definition gives the following Prolog program (see `append.pl`):

$$append([], Y, Y).$$
$$append([H|T], Y, [H|T2]) :- append(T, Y, T2).$$

If the second clause is confusing, the alternative, longer version that follows the item 2. of the above recursive definition, can be written as:

$$append(X, Y, Z) :- X = [H|T], Z = [H|T2], append(T, Y, T2).$$

Note that this as far as resolution is concerned there is no difference between this and the previous versions. Try some queries with Prolog.
Read Section 7.2 of Nilsson, Maluszynski, *Logic, Programming and Prolog (2ed)* for more on lists.