Answers to the final exam on Prolog

Christian Rinderknecht

14 December 2006

1. Let `delete(X,S,T)` be a relation true when the list `T` contains the same items as the list `S`, in the same order, except the first `X` in `S` (starting from the top). One possible definition is

```
delete(X,[X|A],A).                          % Rule 2
delete(X,[Y|A],[Y,B]) :- delete(X,A,B).     % Rule 4
```

   We want to modify this definition so that the relation is true when `S` does not contain any `X`.

   Question. What happen if we add the fact

```
delete(_,X,X).                              % Rule 1, 3 or 5?
```

   as rule 1, 3 or 5? Give counter-examples if needed.

   Answer. Consider the all the possible cases:

   - If the new rule is first (i.e. number 1), then the relation is not what we want since, for instance

```
?- delete(a,[a],[a]).
Yes
```

     instead of `No`.

   - If the new rule is second (i.e. number 3), then the relation is broken too since, for example

```
?- delete(a,[b,a],[b,a]).
Yes
```

     instead of `No`.

   - If the new rule is last (i.e. number 5), then the relation is correct. Rule 2 handles the case when `X` is found on the top of `S`; rule 4 handles the case when `X` is not found on the top of `S` *but* is below (i.e. in `A`). Therefore, the new rule in last position will handle the case where `S`, perhaps empty, does not contain `X`. Since the heads of rules 2 and 4 match a non empty `S`, `X` must only match `[]` in the new rule 5, which can then be further simplified as

```
delete(_,[],[]).                                % Rule 5
```

Then this rule can be moved at any position, since rules 2 and 4 do not match an empty `S`. For example:

```
delete(_,[],[]).
delete(X,[X|A],A).
delete(X,[Y|A],[Y,B]) :- delete(X,A,B).
```

2. **Question.** Define a relation `catenate(U,V,W)` to be true when list `W` is made of list `U` followed by list `V`. For example

```
?- catenate(U,[3,4],[0,1,2,3,4]).
U = [0,1,2] ;
No
```

Answer.

```
catenate([],B,B).
catenate([X|A],B,[X|C]) :- catenate(A,B,C).
```

3. **Question.** Define a relation `flatten(A,B)` to be true when `B` is the list of the items found in the list of lists `A`. (**Hint.** Use `catenate`.)

```
?- flatten([[3,-1,-1],[0],[]],B).
B = [3,-1,-1,0] ;
No
```

Answer.

```
flatten([],[]).
flatten([L|Lists],B) :- catenate(L,Flat,B),flatten(Lists,Flat).
```

4. **Question.** Define a relation `split(L,P,N)` to be true if `P` contains the positive or nul items of `L` (same order as in `L`) and `N` contains the negative items (same order). For example:

```
?- split(L,[1,0],[-3,-2]).
L = [1, 0, -3, -2] ;
L = [1, -3, 0, -2] ;
L = [1, -3, -2, 0] ;
L = [-3, 1, 0, -2] ;
L = [-3, 1, -2, 0] ;
L = [-3, -2, 1, 0] ;
No
```

Answer.

```
split([],[],[]).
split([X|T],[X|P],N) :- X >= 0, split(T,P,N).
split([X|T],P,[X|N]) :- split(T,P,N).
```