Introduction to symbolic CL (684.01) — OSU Linguistics
Detmar Meurers — Winter 2001

Exercise sheet 2

(Due: Wednesday, 18. January at noon, best via email to dm@ling.ohio-state.edu)

Provide PROLOG definitions for the following relations. Please test before handing them in.

1. `last/2`: a two place relation which takes a list as first argument and returns the last element of that list (if there is one) as second argument; i.e., `last(+`*`List`*`,−`*`Last-List-element`*`)`

   Example queries:

   - `?- last([a,b,c,d]),X).` ⇒ `X=d`
   - `?- last([a,b,c],X).` ⇒ `X=c`
   - `?- last([],X).` ⇒ `no`

2. `firstLastSwap/2`: a two place relation which takes a list and returns the same list with one difference: the first list element and the last list element are exchanged; i.e., `firstLastSwap(+`*`List`*`,−`*`List-With-First-Last-Swapped`*`)`

   Example queries:

   - `?- firstLastSwap([a,b,c,d],X).` ⇒ `[d,b,c,a]`
   - `?- firstLastSwap([a,b,c],X).` ⇒ `[c,b,a]`
   - `?- firstLastSwap([],X).` ⇒ `no`

3. `delete_a/2`: a two place relation which takes a list and deletes one first occurrence of `a` (if there is one); i.e., `delete_a(+`*`List`*`,−`*`List-with-one-a-less`*`)`

   Example queries:

   - `?- delete_a([a,b,a,d],X).` ⇒ `X=[b,a,d]; X=[a,b,d]`
   - `?- delete_a([b,a,c,a,g,h],X).` ⇒ `X=[b,c,a,g,h]; X=[b,a,c,g,h]`
   - `?- delete_a([b,g,a],X).` ⇒ `X=[b,g]`
   - `?- delete_a([b,g,a,a],X).` ⇒ `X=[b,g,a]; X=[b,g,a]`
   - `?- delete_a([b,c],X).` ⇒ `X=[b,c]`

   Could you also write the relation so that it only removes the first occurrence of an `a` as exemplified below? What is needed to do this?

   - `?- delete_a([a,b,a,d],X).` ⇒ `X=[b,a,d]`
   - `?- delete_a([b,a,c,a,g,h],X).` ⇒ `X=[b,c,a,g,h]`
   - `?- delete_a([b,g,a],X).` ⇒ `X=[b,g]`
   - `?- delete_a([b,g,a,a],X).` ⇒ `X=[b,g,a]`
   - `?- delete_a([b,c],X).` ⇒ `X=[b,c]`

4. `containsList/2`: a two place relation which succeeds if the second list is part of the first; i.e., `containsList(+`*`List`*`,+`*`Sublist`*`)`

   Example queries:

   - `?- containsList([a,b,c,d],[b,c]).` ⇒ `yes`
   - `?- containsList([a,b,b,c,d],[b,c]).` ⇒ `yes`
   - `?- containsList([a,b,c,d],[a,b]).` ⇒`yes`
   - `?- containsList([a,b,c,d],[a,b,c]).`⇒ `yes`
   - `?- containsList([a,b,c,d],[a]).` ⇒ `yes`
   - `?- containsList([a,b,c,d],[]).` ⇒ `no`
   - `?- containsList([a,b,c,d],[a,c]).` ⇒ `no`
   - `?- containsList([a,b,c,d],[b,d]).` ⇒ `no`

5. `permute/2`: a two place relation which takes a list as its first argument and returns as second argument each list that consists of all and only the elements of the input list in any order of occurrence; i.e., `permute(+`*`List`*`,−`*`Permuted-list`*`)`

   Example queries:

   - `?- permute([a,b],X).` ⇒ `X=[a,b] ; X=[b,a]`

   Hint: in defining `permute` it is useful to define an auxiliary relation `insert` which inserts a single element into an input list at any arbitrary position of the list and returns this newly constructed list.