Assignment1: Prolog, PEAS
– Given Oct 29, Due Nov 6 –

This exercise sheet gets you started with Prolog. To complete the assignments you can use

- a Prolog interpreter, e.g., `http://www.swi-prolog.org/`
- an online interpreter such as `https://swish.swi-prolog.org`.

However, for submission you will have to upload a single Prolog file for each problem. The files must run using the SWI Prolog interpreter. The manual for SWI Prolog is available at `http://www.swi-prolog.org/pldoc/`.

Now and for all future programming problems, your solutions must

- strictly follow the specification in the problem statement, in particular regarding the names and arities of the predicates — we may do automated testing!
- include comments that explain how your code works including example invocations of your programs — a key grading criterion is how easy it is for the grading tutor to verify that you solved the problem correctly.

Problem 1.1 (Basic Prolog Functions) 35 pt

Implement the functions listed below in Prolog. Note that many of them are built-in, but we ask you create your own functions.

1. a function reversing a list
   Test case:
   ```
   ?- myReverse([1,2,3,4,2,5],R).
   R = [5, 2, 4, 3, 2, 1].
   ```

2. a function removing multiple occurrences of elements in a list
   Test case:
   ```
   ?- removeDuplicates([1,1,1,1,2,2,3,4,1,2,7],A).
   A = [1, 2, 3, 4, 7].
   ```

Hint: You may want to implement a helper method `delete(X,LS,RS)`, that removes all instances of `X` in `LS` and returns the result in `RS`.

---

3. a function for zipping two lists
   `zip` takes two lists and outputs a list of pairs (represented as 2-element lists) of elements at the same index in the two lists. If the lists do not have the same length, the zipped list contains only as many pairs as the shorter list.

   Create a Prolog predicate with 3 arguments: the first two are the two lists to zip and the third one the result. For instance:

```
?- zip([1,2,3],[4,5,6],L).          ?- zip([1,2],[3,4,5],L).
L = [[1, 4], [2, 5], [3, 6]].       L = [[1, 3], [2, 4]].
```

4. a function for computing permutations of a list
   Try it out on paper first and understand why this is difficult.

   Test case:

   ```
   ?- myPermutations([1,2,3],P).
   P = [1, 2, 3] ;
   P = [2, 1, 3] ;
   P = [2, 3, 1] ;
   P = [1, 3, 2] ;
   P = [3, 1, 2] ;
   P = [3, 2, 1].
   ```

   Note that there are two ways for specifying such a function:

   (a) return a list of all permutations

   (b) return a single permutation each time such that Prolog finds them one by one.

   Here we are using the second way, i.e., `myPermutations(L,P)` must in particular be true if `P` is some permutation of `L`.

Hint: One possible solution is to start with a helper predicate `takeout(X,L,M)` that is true iff `M` is the result of removing the first occurrence of `X` from `L`. Or equivalently: `M` arises by adding `X` somewhere in `L`. How does this allow you to define the notion of permutation recursively?

Problem 1.2 (Binary Tree) 25 pt

A binary tree of (in this case) natural numbers is inductively defined as either

- an expression of the form `tree(n,t1,t2)` where `n` is a natural number (the label of the node) and `t1` and `t2` are themselves binary trees (the children of that node)
- or `nil` for the empty tree. (Normally a tree cannot be empty, but it is more convenient here to allow an empty tree as well.)

In particular, the nodes of the form `tree(n,nil,nil)` are the *leaf* nodes of the tree, the others are the *inner* nodes.

An example tree in Prolog would be:

```
tree(1,tree(2,nil,nil),tree(2,nil,nil))
```

1. Write a Prolog function `construct` that constructs a binary tree out of a list of (distinct) numbers such that for every subtree `tree(n,t1,t2)` all values in `t1` are smaller than `n` and all values in `t2` are larger than `n`.

   Note that there are usually multiple such trees for every list. One example is:

   ```
   ?- construct([3,2,4,1,5],T).
   T = tree(3, tree(2, tree(1, nil, nil), nil), tree(4, nil, tree(5, nil, nil))).
   ```

2. Write Prolog functions `count_nodes` and `count_leafs` that take a binary tree and return the number of nodes and leaves, respectively.

3. Write a Prolog function `symmetric` that checks whether a binary tree is symmetric.

Problem 1.3 40 pt

For each of the following agents, develop a PEAS description of the task environment.

1. Robot soccer player
2. Internet book-shop agent (that is: an agent for book shops that stocks up on books depending on demand)
3. Autonomous Mars rover
4. Mathematical theorem prover
5. First-person shooter (Counterstrike, Unreal Tournament etc.)

Additionally, characterize the environments of these agents according to the properties discussed in the lecture. Where not "obvious", justify your choice with a short sentence.

Finally, choose suitable designs for the agents.