PROLOG: non linear data structures

Lecture 3

Summary

- Lists of lists
- Binary trees

Lists of lists

`memberlist(X,L)` X is an element of the list of lists L.

`memberlist(X,Y)` is true if one of the following conditions holds:
- X is the first element of L
- X is not the first element of L, the first element of L is a list and X is a member of the first element of L.
- X is a member of the rest of L.

```
memberlist(X,[X|_Xs]).
memberlist(X,[Y|_Ys]) :- memberlist(X,Y).
memberlist(X,[_Y|Ys]) :- memberlist(X,Ys).
```

Prolog: binary trees

`binary_tree(X)` is true if X is a binary tree, that is:
- X is the empty tree.
- X is a structure with a label and 2 binary trees.

```
binary_tree(void).
binary_tree(tree(_Element,Left,Right)) :-
        binary_tree(Left), binary_tree(Right).
```

Binary trees: isomorphism

`isotree(X,Y)` is true if X and Y are isomorphic, that is:
- X and Y are both the empty tree
- X and Y have the same root element and the left and right subtrees are isomorphic

```
isotree(void,void).
isotree(tree(X,Left1,Right1),tree(X,Left2,Right2)) :-
        isotree(Left1,Left2), isotree(Right1,Right2).
isotree(tree(X,Left1,Right1),tree(X,Left2,Right2)) :-
        isotree(Left1,Right2), isotree(Right1,Left2).
```

Binary trees: pre-order

`preorder (X,Y)`: Y is the list containing the elements of the tree X, as encountered in the pre-order visit.

```
preorder (void, [void]).
preorder (tree(X, Left, Right), CompleteList):-
        preorder(Left, LeftList),
        preorder(Right, RightList),
        append([X|LeftList], RightList, CompleteList).
```

Binary trees: breadth-first visit

`breadth_first(X,Y)`: Y is the list containing the elements of the tree X, as encountered in the breadth-first visit.

```
bf([], []).
bf([void | Rest], Ls):- bf(Rest, Ls).
bf([tree(I, Dx, Sx)| Rest], [I|Ls]):-
    append(Rest, [Sx, Dx], Nodes), bf(Nodes, Ls).
```

Evaluating Expressions

Write a PROLOG program, whose input is an arithmetic expression represented by a term and computes its value.

```
evalExpression(plus(A,B), N):- evalExpression(A, N1),
        evalExpression(B, N2), N is N1+N2.
evalExpression(minus(A,B), N):- evalExpression(A, N1),
        evalExpression(B, N2), N is N1-N2.
evalExpression(mult(A,B), N):- evalExpression(A, N1),
        evalExpression(B, N2), N is N1*N2.
evalExpression(frac(A,B), N):-  evalExpression(A, N1),
        evalExpression(B, N2), N is N1 // N2.
evalExpression(X, X).
```

Exercises

1. Write a PROLOG program counting the elements of a list of lists.
2. Write a PROLOG program which implements member for a binary tree.
3. Write a PROLOG program which returns a list containing all the nodes at a given depth D of a binary tree.

Exercises on binary trees

a) Consider the PROLOG terms representing the binary trees whose nodes are labelled by a constant symbol and, in addition, store the depth of the node. Write a PROLOG program that returns true if its argument is a binary tree as above specified.

b) Write a PROLOG program that, given in input a binary tree and a constant, returns the depth of a node containing the given constant.

c) Write a PROLOG program that, given in input a binary tree without the depth information on the nodes and a constant, returns the depth of a node containing the given constant.

d) Write a PROLOG program that, given in input a binary tree without the depth information on the nodes, returns an iso-

morphic binary tree with the depth information stored in the nodes.