Strutture dati non lineari

Esercitazione 3

Sommario

- Liste di liste
- Alberi binari

Valutazione di espressioni

Scrivere un programma che ha come input una espressione aritmetica rappresentata per mezzo si un opportuno termine e ne calcola il valore.

```
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).
```

Prolog: Liste di liste

`memberlist(X,L)` X è un elemento della lista di liste L.

`memberlist(X,Y)` è vero se è verificata una delle seguenti condizioni:
- X è il primo elemento di L
- X non è il primo elemento di L , il primo elemento di L è una lista ed X è membro del primo elemento di L.
- X è membro del resto di L.

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

Prolog: Alberi binari

`binary_tree(X)` è vero se X è un albero, ovvero se una delle seguenti condizioni vale:
- X è l'albero vuoto.
- X è una struttura composta da una informazione e due sotto-strutture, anch'esse alberi binari.

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

Alberi binari: isomorfismo

`isotree(X,Y)` è vero se X ed Y sono isomorfi, ovvero se è vera una delle seguenti:
- X ed Y sono entrambi l'albero vuoto
- X ed Y hanno lo stesso elemento come radice ed i sottoalberi sinistri e destri sono isomorfi

```
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).
```

Alberi binari: visita in preordine

`preorder (X,Y)`: Y è la lista che contiene la visita in preordine dell'albero X.

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

Alberi: visita in ampiezza

`breadth_first(X,Y)`: Y è la lista che contiene il risultato della visita in ampiezza dell'albero Y.

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

Esercizi

1. Scrivere un programma che conta gli elementi di una lista di liste.

2. Scrivere un programma per verificare se una costante appartiene ad un albero binario.

3. Scrivere un programma che compone una lista con tutti i nodi a profondita' D di un albero.

Esercizio d'esame

a) Si considerino dei termini Prolog che rappresentano degli alberi binari i cui nodi contengono un simbolo di costante ed un numero che indica la profondità del nodo. Scrivere un programma PROLOG che restituisce vero se il suo argomento è un albero binario con le caratteristiche specificate.

b) Scrivere un programma PROLOG che dato un albero binario ed una costante restituisca la profondità di un nodo che contiene la costante specificata.

c) Scrivere un programma PROLOG che dato un albero binario in cui le informazioni sulla profondità dei nodi non

sono presenti ed una costante, restituisca la profondità di un nodo che contiene la costante specificata.

d) Scrivere un programma PROLOG che dato un albero binario in cui le informazioni sulla profondità dei nodi non sono presenti restituisca un albero binario contenente le informazioni sulla profondità dei nodi.