Tony Maldonado
CS 471
November 30, 2020

Program 10 – List manipulation in Prolog

Problem Description:

In this program, we dealt with traversing a list in prolog. The lists were representations of binary trees. In one of the procedures, we were to output a unique list of leaves on a tree, given a binary tree that was represented as a list. In the second, we had to report the longest path from the root to a lead in the tree.

Both procedures' implementation weren't very long codes. The first only required a couple of lines be added to the code that Dr. Cooper had already provided for us. Essentially, it was the combination of two other functions that we needed to add. In the second procedure, we had to write three lines of code to the program for which we were provided pseudocode for.

Code:

'mytreeuniq'

```
% Name: Antonio Maldonado
% Date: November 29, 2020
% Input: A binary tree represented as a list
% Output: A list of the unique leaves on the tree
% Preconditions: The user has to give a valid list/tree as input
% Postconditions: n/a

% mytreeuniq will combine flatten and myunique
% Flattens the list then returns the unique of the flattened list

mytreeuniq([],[]).
mytreeuniq([H | T], Z) :- flatten([H | T], Y), myunique(Y,Z).

% Flattens a list

flatten([],[]).
flatten(X,[X]) :- atom(X),!.
flatten([H | T], Z) :- flatten(H, T1), flatten(T, T2), myappend(T1, T2, Z).

% We assume we always get two lists for append

myappend([], L, L).
myappend([H | T], L1, [H | L2]) :- myappend(T, L1, L2).

% Make sure that a list has unique elements
```

Tony Maldonado
CS 471
November 30, 2020

```
myunique([], []).
myunique([H | T], L) :- member(H, T), ! , myunique(T, L).
myunique([H | T], [H | L]) :- myunique(T, L).
```

‘treedepth’

```
% Name: Antonio Maldonado
% Date: November 29, 2020
% Input: A binary tree represented as a list
% Output: The longest path from root to leaf
% Preconditions: The user has to give a valid list/tree as input
% Postconditions: n/a

% The depth of nil is 0
mydepth([], 0).

% The depth of an atom is 0
mydepth(X, 0) :- atomic(X).

% mydepth of a nested list is max(depth(H), depth(T)) + 1
mydepth([H | T], D) :- mydepth(H,D1), mydepth(T, D2), D is max(D1, D2) + 1.
```

Tony Maldonado
CS 471
November 30, 2020

Output/Screenshot:

```
newton cs471/program10> swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['mytreeuniq'].
true.

?- mytreeuniq([a,[b,[a,[c,d]]]],X).
X = [b, a, c, d] .

?- mytreeuniq([],X).
X = [].

?- ['treedepth'].
true.

?- mydepth([a,[b,[a,[c,d]]]], X).
X = 8.

?-
```