Joseph Camacho-Terrazas
11/25/2020
List Manipulation in Prolog

The problems for this assignment were given a binary tree written as a nested list, we were to write Prolog procedures to return a unique list of leaves from the tree, and another to return the longest path from root to leaf. To accomplish the unique list, I combined the flatten and unique functions that were provided in lecture into one file. Then, I wrote the predicate mytreeunique that when given a nested list, it would first flatten the list, then find the unique leaves of that list. Next, for the depth problem, I started out by writing predicates for the depth of nil being 0 and the depth of an atom being 0. Then, I wrote a predicate that takes a nested list and will split the head and the tail. It will then take the max depth of the head and tail and add 1. This performs recursively until you reach the deepest leaf of the tree.

Joseph Camacho-Terrazas
11/25/2020
List Manipulation in Prolog

```
% Joseph Camacho-Terrazas
% 11/25/2020
% Input: A binary tree written as a nested list
% Output: A list of unique leaves
% Precondition: The user gives a valid list as input
% Postcondition: The program will give a list of unique leaves in the tree

% mytreeunique combines flatten and myunique
% Flattens the list then returns the unique elements of that list

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

% Flattens a nested list
% Credit Shaun Cooper

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

% We assume we always get two lists to append

append([], L, L).
append([H | T], L, [H | Z]) :- append(T, L, Z).

% Finds the unique elements of a list
% Credit Shaun Cooper

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

Joseph Camacho-Terrazas
11/25/2020
List Manipulation in Prolog

```
% Joseph Camacho-Terrazas
% 11/25/2020
% Input: A binary tree typed as a nested list
% Output: The longest path from root to leaf
% Precondition: The user provides a valid binary tree as a nested list
% Postcondition: The program will output the longest path from root to leaf

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

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

% mydepth of a nested list is max(depth(L),depth(R)) + 1
% L represents the left node, R represents the right node
% D returns the max depth of the left and right nodes + 1
mydepth([L | R], D) :- mydepth(L, D1), mydepth(R, D2), D is max(D1, D2) + 1.
```

Joseph Camacho-Terrazas
11/25/2020
List Manipulation in Prolog