Collin Gros
11-30-2020
cs471

PROLOG REPORT

PROBLEM DESCRIPTION

Create two Prolog procedures, along with any supporting procedures, to:

1. Given a Binary tree represented as a list, provide a unique list of leaves of the tree.
2. Given a Binary tree, report the longest path from the root to a leaf.

PROLOG CODE

PART 1

flatten

```
% collin gros
% 11-30-2020
% cs471
% prolog.flatten
%
% gets a list and flattens it
%
% (largely based on cooper's code 11-18-2020)
%

flatten([], []).
% if what we have is not a list at all, make it a list
% cut (stop possible choices) at the end
flatten(X, [X]) :- atom(X), !.

% flatten head and tail into temporary values, then append them
% for final result
flatten([H|T], Z) :- flatten(H, T1), flatten(T, T2), myappend(T1, T2, Z).
% equivalent to append(flatten(H), flatten(T))

% append always gets 2 lists
myappend([], L, L).
myappend([H|T], L, [H | Z]) :- myappend(T,L,Z).
```

myappend

```
% collin gros
% 11-30-2020
% cs471
% prolog.myappend
%
% program that appends two lists
%
% (largely based off of cooper's code 11-18-2020)
%

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

mytreeuniq

```
% collin gros
% 11-30-2020
% cs471
% prolog.mytreeuniq
%
% takes a binary tree and provides a unique list of leaves of the tree.
```

```
% input: X (unflattened, non-unique binary tree list)
% output: Y (flattened, unique binary tree list representing leaves)
%

% flatten the tree and store into temporary variable T1
mytreeuniq(X, Y) :- flatten(X, T1), myuniq(T1, Y).


% get a unique tree list from the flattened non-unique tree list
```

myuniq

```
% collin gros
% 11-30-2020
% cs471
% prolog.myuniq
%
% determines that a list is uniq

myuniq([], []).

% if there's more than 1 H in T, we want to delete it
% !: we never want to get to the next part if H is
% a member.
myuniq([H | T], L) :- member(H, T), !, myuniq(T, L).

% we know it's not a member. append H to L (result)
myuniq([H | T], [H | L]) :- myuniq(T, L).
```

PART 2

mydepth

```
% collin gros
% 11-30-2020
% cs471
% prolog.mydepth
%
% this program gets the value of the deepest leaf in a tree
%
% (based heavily on cooper's code 11-30-2020)


% depth of empty list is 0
mydepth([], 0).
% only if X is atomic, depth is 0
mydepth(X, 0) :- atom(X).

% depth is equal to the max of the depth of the left side, and the
% max of the depth of the right side, +1.
mydepth([H|T], L) :- mydepth(H, T1), mydepth(T, T2), L is max(T1, T2) + 1.
```

OUTPUT (SCREENSHOT)

PART 1

```
cgros@borg:~/Documents/school/cs-471/prolog/1> !sw
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).

?- ['myappend'].
true.

?- ['myuniq'].
true.

?- ['flatten'].
Warning: /home/ugrad20/cgros/Documents/school/cs-471/prolog/1/flatten:22:
        Redefined static procedure myappend/3
        Previously defined at /home/ugrad20/cgros/Documents/school/cs-471/prolog/1/myappend:11
true.

?- ['mytreeuniq'].
true.

?- mytreeuniq([a, b, c, [c, d, e, [f, g, f, g]]], L).
L = [a, b, c, d, e, f, g] .

?-
```

PART 2

```
cgros@borg:~/Documents/school/cs-471/prolog/2> !sw
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).

?- ['mydepth'].
true.

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

?-
```