

CSCI 360 — Survey of Programming Languages
Spring 2008
Assignment #7, Due 4/15/2008

Consider the following tree diagram.

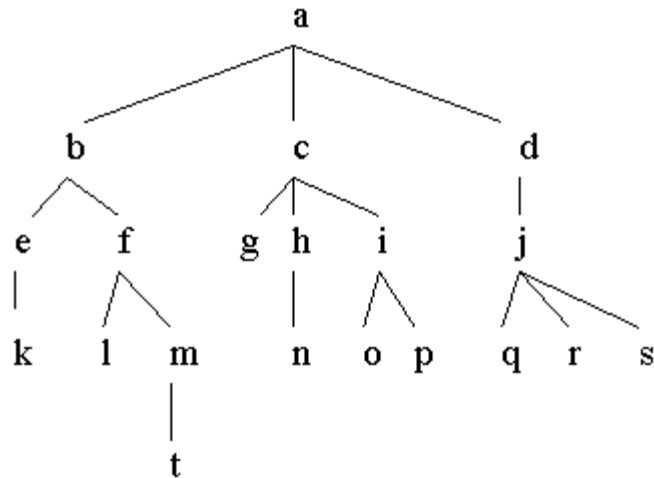


Illustration 1: Tree diagram.

The file on the classnotes website has a representation for this tree and predicate definitions to do some processing of the tree. The contents of that file are shown below. Note the use of Prolog operators in some of the definitions.

```
/* The tree database */

:- op(500,xfx,'is_parent').

a is_parent b.      c is_parent g.      f is_parent l.      j is_parent q.
a is_parent c.      c is_parent h.      f is_parent m.      j is_parent r.
a is_parent d.      c is_parent i.      h is_parent n.      j is_parent s.
b is_parent e.      d is_parent j.      i is_parent o.      m is_parent t.
b is_parent f.      e is_parent k.      i is_parent p.

/* X and Y are siblings */
:- op(500,xfx,'is_sibling_of').
X is_sibling_of Y :- Z is_parent X,
                    Z is_parent Y,
                    X \== Y.

/* X and Y are on the same level in the tree. */
:-op(500,xfx,'is_same_level_as').

X is_same_level_as X .
X is_same_level_as Y :- W is_parent X,
                       Z is_parent Y,
```

```

                                W is_same_level_as Z.

/* Depth of node in the tree. */
:- op(500,xfx,'has_depth').

a has_depth 0 :- !.
Node has_depth D :- Mother is_parent Node,
                   Mother has_depth D1,
                   D is D1 + 1.

/* Locate node by finding a path from root down to the node. */
locate(Node) :- path(Node),
                write(Node),
                nl.

path(a).
path(Node) :- Mother is_parent Node, /* Can start at a. */
              path(Mother),          /* Choose parent, */
              write(Mother),         /* find path and then */
              write(' --> ').

/* Calculate the height of a node, length of longest path to
   a leaf under the node. */

height(N,H) :- setof(Z,ht(N,Z),Set), /* See section 2.8 for 'setof'.
*/
              max(Set,0,H).

ht(Node,0) :- leaf(Node), !.
ht(Node,H) :- Node is_parent Child,
              ht(Child,H1),
              H is H1 + 1.

leaf(Node) :- not(is_parent(Node,Child)). /* Node grounded */

max([],M,M).
max([X|R],M,A) :- (X > M -> max(R,X,A) ; max(R,M,A)).

```

The 'is_sibling_of' relationship tests whether two nodes have a common parent in the tree. For example,

```

?- h is_sibling_of S.
S=g ;
S=i ;
no

```

Note the use of the literal $X \neq Y$, which succeeds just in case X and Y are not co-bound (bound to the same value).

The 'is_same_level_as' relationship tests whether two nodes are on the same level in the tree. The 'depth' predicate computes the depth of a node in the tree (how many edges from the root). For example,

```
?- t has_depth D.  
D=4
```

Here is an alternate definition of 'depth' using Prolog implication:

```
N has_depth D :- N == 'a' -> D=0 ;  
                Mother is_parent N,  
                Mother has_depth D1,  
                D is D1 + 1.
```

The 'locate' predicate computes and prints a path from the root to a node. For example,

```
?- locate(n).  
a --> c --> h --> n
```

The 'leaf' predicate defines a leaf to be a node which is not a parent. Note the free variable inside the negation. This is correct, since if the node has any child then the node is not a leaf.

The 'height' predicate computes the height of a node -- defined as the length of the longest path to a leaf under the node. This definition uses lists and the second-order Prolog predicate 'setof'.

Load the program into the Prolog environment and test the program by issuing various goals.

Exercise 1 Write a Prolog definition for 'ancestor(X,Y)' with the intended meaning that "X is an ancestor of Y in the tree". Pay attention: recursion from the top of the tree or from the bottom of the tree?

Exercise 2 As written `leaf/1` is intended as a test when `Node` is grounded. Reformulate `leaf/1` so that the goal `?- leaf(X).` will return values of `X` which are leaves in the tree.

Exercise 3 Formulate definitions for a human family tree using relations 'male', 'female', 'parent', 'father', 'mother', 'sibling', 'grandparent', 'grandmother', 'grandfather', 'cousin', 'aunt', and 'uncle'. Let 'male', 'female', 'parent' be the fundamental relations and define the others in terms of these.

You will need to email me a file(s) containing the completed programs so I can test them.

Taken from the Prolog Tutorial by J. R. Fisher,

http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/