Answers to the Final Exam on Prolog Programming

Christian Rinderknecht

3 December 2008

1 Sorting leaves in a binary tree

Question. Design a simple data structure to represent binary trees which hold integers at their leaves[1] only. For example:

Write an efficient predicate `tmerge/2` such that `tmerge(Tree,Nodes)` is provable if and only if `Nodes` is the list of all the leaves of the tree `Tree` in increasing order of their integer values. In the example above, we have `Nodes=[1,5,7,9]`.

Answer.

```
tmerge({Left,Right},Sorted) :-
  tmerge(Left,SLeft),
  tmerge(Right,SRight),
  merge(SLeft,SRight,Sorted).
tmerge(empty,[]).
tmerge(Leaf,[Leaf]).

merge(   [],    Q,    Q).
merge(    P,   [],    P).
merge([I|P],[J|Q],[I|R]) :- I < J, merge(P,[J|Q],R).
merge(    P,[J|Q],[J|R]) :- merge(P,Q,R).
```

[1] A leaf is an internal node whose both children are external nodes.

2 Depth-first traversal revisited

Question. Given a predicate `edge/2` encoding a general tree containing information at all of its nodes, write a predicate `depth_first/2` such that `depth_first(Tree,Nodes)` is provable if and only if `Nodes` is the list of all the nodes of the tree `Tree` in depth-first order (in general, several are possible). For example, given

```
edge(1,3).
edge(2,4).
edge(1,2).
edge(2,5).
edge(1,6).
```

then `Nodes = [1,3,2,4,5,6]`.

Answer.

```
% General tree (the subtrees are not ordered)
%
%          1
%         /|\
%        2 6 3
%       / \
%      4   5
%
edge(1,3).
edge(2,4).
edge(1,2).
edge(2,5).
edge(1,6).

% Reflexive and transitive closure of ‘edge’
%
rt(A,A).
rt(A,B) :- edge(A,X), rt(X,B).

% Depth-first traversal (not unique: depends on the
% order of the facts defining ‘edge’.
%
depth_first(Root,Nodes) :- findall(Node,rt(Root,Node),Nodes).
```