Logical reasoning and programming, task III
(December 11, 2017)

Problem

Your task is to modify the following leantap Prolog program in such a way that it is able to produce a tableau proof tree (at the last additional 6th argument of original predicate prove/5) that exactly correspond to the original leantap computation.

% prove(+Fml,+UnExp,+Lits,+FreeV,+VarLim)

prove((A,B),UnExp,Lits,FreeV,VarLim) :- !,
  prove(A,[B|UnExp],Lits,FreeV,VarLim).

prove((A;B),UnExp,Lits,FreeV,VarLim) :- !,
  prove(A,UnExp,Lits,FreeV,VarLim),
  prove(B,UnExp,Lits,FreeV,VarLim).

prove(all(X,Fml),UnExp,Lits,FreeV,VarLim) :- !,
  \+ length(FreeV,VarLim),
  copy_term((X,Fml,FreeV),(X1,Fml1,FreeV)),
  append(UnExp,[all(X,Fml)],UnExp1),
  prove(Fml1,UnExp1,Lits,[X1|FreeV],VarLim).

prove(Lit,_,[L|Lits],_,_) :-
  ( Lit = -Neg; -Lit = Neg ) -> ( unify_with_occurs_check(Neg,L)
    ; prove(Lit,[],Lits,_,_) )
  ;
prove(Lit,[Next|UnExp],Lits,FreeV,VarLim) :-
  prove(Next,UnExp,[Lit|Lits],FreeV,VarLim).

Tableau Proof Format

The tableau proof tree format is a Prolog representation of Semantic Tableau tree. Every node in the tree has its unique identifier that can be Prolog integer or Prolog atom. Every tableau rule can add one (in case of add_rule, all_rule, and closed_by) or two (in case of or_rule) leaves to the current working node. Every node has the following format:

node(identifier,corresponding NNF formula,used rule,list of its sub-nodes/sons)

where used rule can be a term add_rule, or_rule, all_rule, or closed_by with arguments that contain identifiers as references to previous nodes that are needed by this rule to infer the current node. The root of the Semantic Tableau tree has a node with top identifier. Our tableau proof tree does not have this node but you can refer top as identifier. The corresponding NNF formula, that is inferred by closed_by rule, is false.
The tableau proof tree starts with a list of sub-nodes of top. All substitutions, that are needed, are already applied on the resulting tableau proof tree.

**Program**

You are supposed to upload a program `leantap2.pl`, in an archive, containing a predicate `prove/6`, where the last argument is the output tableau proof and the other arguments correspond one to one to the original `leantap` implementation. Of course, you can use additional predicates in your solution.

**Example 1**

```prolog
?- prove((fact, -fact), [], [], [], 0, Proof).
Proof = [node(1, fact, and_rule(top),
          [node(2, -fact, and_rule(top),
            [node(3, false, closed_by(2, 1), [])])])
```

The example shows the proof tree structure with nodes labeled by `fact`, `-fact`, and `false` and rules applied as `and_rule(top)`. The tree is rooted at `top` with branches leading to sub-nodes that represent the logical structure of the proof.

```prolog
```

```prolog
```
Example 2

?- prove((all(X,p(X)) , (-p(c);-p(d))), [], [], [], 3, Proof).
Proof = [node(0, all(B, p(B)), and_rule(top), [node(2, p(c), all_rule(0), [node(3, (-p(c);-p(d)), and_rule(top), [node(4, -p(c), or_rule(3), [node(5, false, closed_by(4, 2), [])]), node(6, -p(d), or_rule(3), [node(7, p(d), all_rule(0), [node(8, false, closed_by(7, 6), [])])])])])])

```
0: all(B, p(B))

2: p(c)

3: -p(c);-p(d)

4: -p(c)

5: false

6: -p(d)

7: p(d)

8: false
```

3