**Practical 9**

Write a predicate pptree/1 that takes a complex term representing a tree, such as s(np(det(a),n(man)),vp(v(shoots),np(det(a),n(woman)))), as its argument and prints a nice and readable output for this tree.

pptree(s(NP,VP)) :-

write('s('), nl,

tab(4), pptree(NP,4), nl,

tab(4), pptree(VP,4),

write(')').

pptree(det(DET)) :-

write('det('),

write(DET),

write(')').

pptree(n(N)) :-

write('n('),

write(N),

write(')').

pptree(v(V)) :-

write('v('),

write(V),

write(')').

pptree(np(N),I) :-

Indent is I+4,

write('np('), nl,

tab(Indent), pptree(N),

write(')').

pptree(np(DET,N),I) :-

Indent is I+4,

write('np('), nl,

tab(Indent), pptree(DET), nl,

tab(Indent), pptree(N),

write(')').

pptree(vp(V),I) :-

Indent is I+4,

write('vp('), nl,

tab(Indent), pptree(V),

write(')').

pptree(vp(V,NP),I) :-

Indent is I+4,

write('vp('), nl,

tab(Indent), pptree(V), nl,

tab(Indent), pptree(NP,Indent),

write(')').

In the practical session of Chapter 7, you were asked to write a DCG generating propositional logic formulas. The input you had to use was a bit awkward though. The formula ¬ (p → q) had to be represented as [not, '(', p, implies, q, ')']. Now, that you know about operators, you can do something a lot nicer. Write the operator definitions for the operators not, and, or, implies, so that Prolog accepts (and correctly brackets) propositional logic formulas. For example:

?- display(not(p implies q)).

not(implies(p,q)).

Yes

?- display(not p implies q).

implies(not(p),q)

Yes

:- op(100, fx, not).

:- op(200, xfy, and).

:- op(300, xfy, or).

:- op(400, xfy, implies).