

An Example of Formula Simplification

The formula:

$$\neg((a \wedge \neg((\text{true} \wedge \text{false} \wedge \text{true})) \wedge (\neg a \vee c) \wedge \text{true}))$$

will be represented as the Prolog term:

```
- and([ a,  
        - and([true, false, true]),  
        or([- a, c]),  
        true  
      ])
```

We allow the negation of ANY formula.

We can simplify such formulas
to produce equivalent formulas
in which only atoms are negated.

Then we can use the "simplify" rules
to form a powerful formula simplification system:

```
nsimplify(Wff, SWff) :- nsimp(Wff, NNF),  
                        simplify(NNF, SWff).
```

nsimp(- true, false) :- !.
nsimp(- false, true) :- !.
nsimp(Lit, Lit) :- literal(Lit), !.
nsimp(- (- W), NW) :- !, nsimp(W, NW).

nsimp(and(Args), and(NsimpArgs)) :-
 !, nsimplist(+, Args, NsimpArgs).
nsimp(or(Args), or(NsimpArgs)) :-
 !, nsimplist(+, Args, NsimpArgs).

nsimp(- and(Args), or(NsimpArgs)) :-
 !, nsimplist(-, Args, NsimpArgs).
nsimp(- or(Args), and(NsimpArgs)) :-
 !, nsimplist(-, Args, NsimpArgs).

nsimplist(Sign, [], []) :- !.

nsimplist(+, [Arg1 | Rest], [NArg1 | NRest]) :-
 nsimp(Arg1, NArg1), nsimplist(+, Rest, NRest).

nsimplist(-, [Arg1 | Rest], [NArg1 | NRest]) :-
 nsimp(- Arg1, NArg1), nsimplist(-, Rest, NRest).

literal(-X) :- atom(X).

literal(X) :- atom(X).