TLLP interpreter in Prolog

```
/*
        TLLP interpreter in Prolog
*/
:- op(1060, xfy, (&)).
:- op( 950, xfy, [-<>, =>]).
:- op( 900,  fy, [!, @, #]).

prove(G) :-
        prove(G, 0, [], []).

prove(true, T, I, I) :- !.
prove(top, T, I, O) :- !,
        subcontext(T, O, I).
prove((G1, G2), T, I, O) :- !,
        prove(G1, T, I, M),
        prove(G2, T, M, O).
prove((G1 & G2), T, I, O) :- !,
        prove(G1, T, I, O),
        prove(G2, T, I, O).
prove((G1 ; G2), T, I, O) :- !,
        (prove(G1, T, I, O) ;
         prove(G2, T, I, O)).
prove((R -<> G), T, I, O) :- !,
        count_next(R, N, R1),
        T1 is T + N,
        prove(G, T, [(R1,T1)|I], [1|O]).
prove((S => G), T, I, O) :- !,
        prove(G, T, [(!S,0)|I], [(!S,0)|O]).
prove(!G, T, I, I) :- !,
        prove(G, T, I, I).
prove(@G, T, I, O) :- !,
        T1 is T + 1,
        prove(G, T1, I, O).
prove(A, T, I, O) :-
        pick(T, I, O, A).
prove(A, T, I, O) :-
        pick(T, I, M, (G -<> A)),
        prove(G, T, M, O).
```

```
/*
        TLLP interpreter in Prolog
*/
:- op(1060, xfy, (&)).
:- op( 950, xfy, [-<>, =>]).
:- op( 900,  fy, [!, @, #]).

prove(G) :-
        prove(G, 0, [], []).

prove(true, T, I, I) :- !.
prove(top, T, I, O) :- !,
        subcontext(T, O, I).
prove((G1, G2), T, I, O) :- !,
        prove(G1, T, I, M),
        prove(G2, T, M, O).
prove((G1 & G2), T, I, O) :- !,
        prove(G1, T, I, O),
        prove(G2, T, I, O).
prove((G1 ; G2), T, I, O) :- !,
        (prove(G1, T, I, O);
         prove(G2, T, I, O)).
prove((R -<> G), T, I, O) :- !,
        count_next(R, N, R1),
        T1 is T + N,
        prove(G, T, [(R1,T1)|I], [1|O]).
prove((S => G), T, I, O) :- !,
        prove(G, T, [(!S,0)|I], [(!S,0)|O]).
prove(!G, T, I, I) :- !,
        prove(G, T, I, I).
prove(@G, T, I, I) :- !,
        T1 is T + 1,
        prove(G, T1, I, O).
prove(A, T, I, O) :-
        pick(T, I, O, A).
prove(A, T, I, O) :-
        pick(T, I, M, (G -<> A)),
        prove(G, T, M, O).
```

```
count_next(@R, N, R1) :- !,
        count_next(R, N1, R1),
        N is N1 + 1.
count_next(R, 0, R).

pick(T, I, O, S) :-
        pick1(T, I, O, S).
pick(T, I, I, S) :-
        rule(S).
pick(T, I, I, (G -<> A)) :-
        rule((A :- G)).

pick1(T, [(!S,0)|I], [(!S,0)|I], S).
pick1(T, [(#R,T0)|I], [1|I], S) :-
        T >= T0,
        select(R, S).
pick1(T, [(R,T)|I], [1|I], S) :-
        \+(R = (!_)), \+(R = (#_)),
        select(R, S).
pick1(T, [R|I], [R|O], S) :-
        pick1(T, I, O, S).

select((R1 & R2), R) :- !,
        (select(R1, R) ; select(R2, R)).
select(R, R).

subcontext(T, [], []).
subcontext(T, [(!S,0)|O], [(!S,0)|I]) :-
        subcontext(T, O, I).
subcontext(T, [R1|O], [(#R,T0)|I]) :-
        (R1 = (#R,T0) ; R1 = 1),
        subcontext(T, O, I).
subcontext(T, [R1|O], [(R,T0)|I]) :-
        \+(R = (!_)), \+(R = (#_)),
        T0 >= T,
        (R1 = (R,T0) ; R1 = 1),
        subcontext(T, O, I).

rule(( p(V,V,[V]) :- v(V) )).
rule(( p(U,V,[U|P]) :-
         v(U), e(U,W), @p(W,V,P) )).
rule(( e(U,V) )).
rule(( goal(P) :- #v(a) -<> @ @v(b) -<>
         @ #v(c) -<> #v(d) -<> p(a,d,P) )).
```

```
count_next(R, N, R1) :- !,
        count_next(R, N1, R1),
        N is N1 + 1.
count_next(R, 0, R).

pick(T, I, O, S) :-
        pick1(T, I, O, S).
pick(T, I, I, S) :-
        rule(S).
pick(T, I, I, (G -<> A)) :-
        rule((A :- G)).

pick1(T, [(!S,0)|I], [(!S,0)|I], S).
pick1(T, [(#R,T0)|I], [1|I], S) :-
        T >= T0,
        select(R, S).
pick1(T, [(R, T)|I], [1|I], S) :-
        \+(R = (!_)), \+(R = (#_)),
        select(R, S).
pick1(T, [R|I], [R|O], S) :-
        pick1(T, I, O, S).

select((R1 & R2), R) :- !,
        (select(R1, R) ; select(R2, R)).
select(R, R).

subcontext(T, [], []).
subcontext(T, [(!S,0)|O], [(!S,0)|I]) :-
        subcontext(T, O, I).
subcontext(T, [R1|O], [(#R,T0)|I]) :-
        (R1 = (#R,T0) ; R1 = 1),
        subcontext(T, O, I).
subcontext(T, [R1|O], [(R,T0)|I]) :-
        \+(R = (!_)), \+(R = (#_)),
        T0 >= T,
        (R1 = (R, T0) ; R1 = 1),
        subcontext(T, O, I).

rule(( p(V,V,[V]) :- v(V) )).
rule(( p(U,V,[U|P]) :-
         v(U), e(U,W), @p(W,V,P) )).
rule(( e(U, V) )).
rule(( goal(P) :- #v(a) -<> @ @v(b) -<>
         @ #v(c) -<> #v(d) -<> p(a,d,P) )).
```