leanCoP: Classical First-Order Logic

```
prove([],_,_,_,_).
prove([L|C],P,I,Q,S) :- \+ (member(A,[L|C]), member(B,P),
    A==B), (-N=L;-L=N) -> ( member(D,Q), L==D ;
    member(E,P), unify_with_occurs_check(E,N) ; lit(N,F,H),
    (H=g -> true ; length(P,K), K<I -> true ;
    \+p -> assert(p), fail), prove(F,[L|P],I,Q,S) ),
    (member(cut,S) -> ! ; true), prove(C,P,I,[L|Q],S).
```

leanCoP-Ω: First-Order Logic with Arithmetic

```
prove([],_,_,_,_,[],Eq,Eq).
prove([L|C],P,I,Q,S,Pr,Eq,Eq1) :-
    Pr=[[[R|F]|Pr1]|Pr2], \+ (member(A,[L|C]), member(B,P), A==B),
    (-N=L;-L=N) -> ( member(D,Q), L==D, F=[], Pr1=[], Eq2=Eq ;
    member(E,P), unify_with_arith(E,N,EqU,S), append(EqU,Eq,Eq2),
    omega(Eq2), F=[], Pr1=[] ; lit(N,E,F,H), unify_with_arith(E,N,EqU,S),
    append(EqU,Eq,Eq3), omega(Eq3), (H=g -> true ; length(P,K), K<I ->
    true ; \+ pathlim -> assert(pathlim), fail),
    prove(F,[L|P],I,Q,S,Pr1,Eq3,Eq2) ;
    (L=(_=_);-(_=_)=L;L=(_<_);-(_<_)=L) -> (leanari(L) ->  Eq2=Eq, F=[],
    Pr1=[] ; member(eq(_),S), path_eq(P,L,EqP), (omega([EqP|Eq]) ->
    Eq2=[EqP|Eq], F=[], Pr1=[] ; member(eq(2),S), lit(_,R,F,H),
    (R=(_=_);-(_=_)=R;R=(_<_);-(_<_)=R), (H=g -> true ; length(P,K),
    K<I -> true ; \+ pathlim -> assert(pathlim), fail),
    prove([R|F],[L|P],I,Q,S,Pr1,Eq,Eq2) ) ) ), (var(R) -> R=N ; true),
    ( member(cut,S) -> ! ; true ), prove(C,P,I,[L|Q],S,Pr2,Eq2,Eq1).
```

+ Omega test system [Pugh '92] for linear integer arithmetic

leanCoP-Ω

J. Otten, H. Trölenberg, T. Raths

University of Potsdam

www.leancop.de