Forward Chaining in Prolog

FC Core

% add(P) adds assertion P to database and triggers forward chaining rules.
add(P) :- clause(P,true), !.
add(P) :-
    dbug("Adding ~p.~n",[P]),
    assert(P),
    foreach(ifAdded(P,Actions), call(Actions)).

% remove(P) removes P from database and triggers ifRemoved rules.
remove(P) :-
    dbug("Removing ~p.~n",[P]),
    retract(P),
    foreach(ifRemoved(P,Actions), call(Actions)).
Mapping rules into triggers

% A=>B adds a forward chaining rule that will satisfy B whenever the assertions in A have all been added to the database.

((P1,P2)=>Q) :- !, (P1=(P2=>Q)).

((P1;P2)=>Q) :- !, (P1=>Q), (P2=>Q).

(P=>Q) :- ifAdded(P,Q), !.

(P=>Q) :-
  assert(ifAdded(P,Q)),
  foreach(clause(P,true),fcDo(Q)).

If-removed rules

% A=>B adds a ifRemoved rule that will satisfy B whenever the assertions in A have all been removed from the database.

((P1,P2)=/>Q) :- !, (P1=(P2=>Q)).

((P1;P2)=/>Q) :- !, (P1=>Q), (P2=>Q).

(P=>Q) :- assert(ifRemoved(P,Q)).
A test file

% FCTEST

:- spouse(P1,P2) => add(spouse(P2,P1)).
:- spouse(P1,P2) =/> remove(spouse(P2,P1)).
:- add(spouse(adam,eve)).

:- a(X), b(X), c(X) => add(d(X)).