Two Beer Glass Problem

State Representation
2-tuple
(Integer, Integer)
First Integer is liquid in 7 pint glass
Second Integer is liquid in 5 pint glass

Initial State
(0,0)

Goal State
(4,_)

Actions
Fill 7 Fill 5
Empty 7, Empty 5,
Pour 7 to 5
Pour 5 to 7

```
state_change( fill7, (S,F), (7,F) ) :-
        S < 7.

state_change( fill5, (S,F), (S,5) ) :-
        F < 5.

state_change( empty7, (S,F), (0,F) ) :-
        S > 0.

state_change( empty5, (S,F), (S,0) ) :-
        F > 0.

state_change( pour7to5, (S,F), (R,5) ) :-
        All is S + F,
        All > 5,
        R is All – 5.
state_change( pour7to5, (S,F), (0,All) ) :-
        All is S + F,
        All =< 5.

state_change( pour5to7, (S,F), (7,R) ) :-
        All is S + F,
        All > 7,
        R is All - 7.
state_change( pour5to7, (S,F), (All,0) ) :-
        All is S + F,
        All =< 7.
```

ISN System

State Representation
List of [i|s|n]

Initial State
[i,s]

Goal State
[i,n]

Actions
The four rules

```
state_change( 1, Xs, Xsn ) :-
        append( X, [s], Xs ),
        append( X, [s,n], Xsn ).

state_change( 2, [I|X], IXX ) :-
        append( [I|X], X, IXX ).

state_change( 3, XsssY, XnY ) :-
        append( X, [s,s,s|Y], XsssY ),
        append( X, [n|Y], XnY ).

state_change( 4, XnnY, XY ) :-
        append( X, [n,n|Y], XnnY ),
        append( X, Y, XY ).
```

7 Ball Solver

State representation
(List1, List2, List3, N)
List1 is Balls not in a Pan
List2 is LeftPan
List3 is RightPan
N is number of weighs

Start state
([1,2,3,4,5,6,7], [], [], 0)

Goal State
(L, [], [], N), length(L, 1), N < 3

Action
Distribute
Weight

```
state_change( distribute, (OldP, [], [], N ), (NewP, LP, RP, N ) ) :-
        append( LP, Temp, OldP ),
        append( RP, NewP, Temp ),
        length( LP, X ),
        length( RP, X ).

state_change( weigh, (OldP, LP, RP, N), (NewP, [], [], N1 ) :-
        weigh( LP, RP, Result ),
        %append( Result, OldP, NewP ),
        in_pan_or_out( Result, OldP, NewP ),
        N1 is N + 1.

in_pan_or_out ( [], OldP, OldP ) :- !.
in_pan_or_out ( L, _, L ).

weigh( LP, _, LP ) :-
        htto( H ),
        member( H, LP ), !.
weigh( _, RP, RP ) :-
        htto( H ),
        member( H, RP ), !.
weigh( _, _, [] ).
```

Euler's Bridge

State representation
(Term,List)
Term is land mass a, b, c, d
List of crossed bridges

Start state
(a, [])

Goal State
(_,L), length(L, 7)

Actions
Cross a bridge

```
state_change( cross, (Location, Bridges), (NewLocation, [B|Bridges]) ) :-
        is_connected( Location, Bridge, NewLocation ),
        not( member( B, Bridges ) ).

is_connected( X, B, Y ) :-
        connected( X, B, Y ).
is_connected( X, B, Y ) :-
        connected( Y, B, X ).

connected( a, 1, b ).
connected( a, 2, b ).
connected( a, 3, c ).
connected( a, 4, c ).
connected( a, 5, d ).
connected( b, 6, d ).
connected( c, 7, d ).
```