WATER JUG PROBLEM

:- include(bb\_planner).

initial\_state( [initial, [a,3,0], [b,5,0], [c,8,8]]).

goal\_state( [\_, [a,\_,4], [b,\_,\_], [c,\_,\_]]).

goal\_state( [\_, [a,\_,\_], [b,\_,4], [c,\_,\_]]).

goal\_state( [\_, [a,\_,\_], [b,\_,\_], [c,\_,4]]).

transition( [\_, A1,B1,C], [pour\_a\_to\_b, A2,B2,C] ) :- pour(A1,B1,A2,B2).

transition( [\_, A1,B,C1], [pour\_a\_to\_c, A2,B,C2] ) :- pour(A1,C1,A2,C2).

transition( [\_, A1,B1,C], [pour\_b\_to\_a, A2,B2,C] ) :- pour(B1,A1,B2,A2).

transition( [\_, A,B1,C1], [pour\_b\_to\_c, A,B2,C2] ) :- pour(B1,C1,B2,C2).

transition( [\_, A1,B,C1], [pour\_c\_to\_a, A2,B,C2] ) :- pour(C1,A1,C2,A2).

transition( [\_, A,B1,C1], [pour\_c\_to\_b, A,B2,C2] ) :- pour(C1,B1,C2,B2).

pour( [Jug1, Capacity1, Initial1], [Jug2, Capacity2, Initial2], % initial jug states

[Jug1 ,Capacity1, 0], [Jug2, Capacity2, Final2] % final jug states

):-

Initial1 =< (Capacity2 - Initial2),

Final2 is Initial1 + Initial2.

pour( [Jug1, Capacity1, Initial1], [Jug2, Capacity2, Initial2], % initial jug states

[Jug1 ,Capacity1, Final1], [Jug2, Capacity2, Capacity2] % final jug states

):-

Initial1 > (Capacity2 - Initial2),

Final1 is Initial1 - (Capacity2 - Initial2).

legal\_state( \_ ).

equivalent\_states( X, X ).

loopcheck(on).

?- find\_solution.

OUTPUT:

?- find\_solution.

== Starting Search ==  
Found 1 states reachable in path length 0  
Computing extensions of length : 1  
Found 6 states reachable in path length 1  
Computing extensions of length : 2  
Found 8 states reachable in path length 2  
Computing extensions of length : 3  
Found 13 states reachable in path length 3  
Computing extensions of length : 4  
Found 14 states reachable in path length 4  
Computing extensions of length : 5  
Found 8 states reachable in path length 5  
Computing extensions of length : 6  
Found 20 states reachable in path length 6  
  
\*\* FOUND SOLUTION of length 6 \*\*  
[initial, [a, 3, 0], [b, 5, 0], [c, 8, 8]]  
[pour\_c\_to\_b, [a, 3, 0], [b, 5, 5], [c, 8, 3]]  
[pour\_b\_to\_a, [a, 3, 3], [b, 5, 2], [c, 8, 3]]  
[pour\_a\_to\_c, [a, 3, 0], [b, 5, 2], [c, 8, 6]]  
[pour\_b\_to\_a, [a, 3, 2], [b, 5, 0], [c, 8, 6]]  
[pour\_c\_to\_b, [a, 3, 2], [b, 5, 5], [c, 8, 1]]  
[pour\_b\_to\_a, [a, 3, 3], [b, 5, 4], [c, 8, 1]]

*1***true**