Water Jug problem using BFS

Program:-

```
go :-
	start(Start),
	solve(Start, Solution),
	reverse(Solution, L),
	print(L, _).

solve(Start, Solution) :-
	breadthfirst([[Start]], Solution).

% breadthfirst([Path1, Path2, ...], Solution):
% Solution is an extension to a goal of one of the paths

breadthfirst([[Node | Path] | _], [Node | Path]) :-
	goal(Node).

breadthfirst([Path | Paths], Solution) :-
	extend(Path, NewPaths),
	append(Paths, NewPaths, Paths1),
	breadthfirst(Paths1, Solution).

extend([Node | Path], NewPaths) :-
	findall([NewNode, Node | Path],
		(next_state(Node, NewNode), \+ member(NewNode, [Node | Path])),
		NewPaths),
	!.

extend(_, []).

% States are represented by the compound term (4-gallon jug, 3-gallon jug);
% In the initial state, both jugs are empty:

start((0, 0)).

% The goal state is to measure 2 gallons of water:
goal((2, _)).
goal((_, 2)).

% Fill up the 4-gallon jug if it is not already filled:
next_state((X, Y), (4, Y)) :- X < 4.

% Fill up the 3-gallon jug if it is not already filled:
next_state((X, Y), (X, 3)) :- Y < 3.

% If there is water in the 3-gallon jug (Y > 0) and there is room in the 4-gallon jug (X < 4), THEN use it
to fill up
% the 4-gallon jug until it is full (4-gallon jug = 4 in the new state) and leave the rest in the 3-gallon
jug:
next_state((X, Y), (4, Z)) :-
	Y > 0, X < 4,
	Aux is X + Y,
	Aux >= 4,
	Z is Y - (4 - X).
```

```
% If there is water in the 4-gallon jug (X > 0) and there is room in the 3-gallon jug (Y < 3), THEN use it to fill up
% the 3-gallon jug until it is full (3-gallon jug = 3 in the new state) and leave the rest in the 4-gallon jug:
next_state((X, Y), (Z, 3)) :-
	X > 0, Y < 3,
	Aux is X + Y,
	Aux >= 3,
	Z is X - (3 - Y).

% There is something in the 3-gallon jug (Y > 0) and together with the amount in the 4-gallon jug it fits in the
% 4-gallon jug (Aux is X + Y, Aux =< 4), THEN fill it all (Y is 0 in the new state) into the 4-gallon jug (Z is Y + X):
next_state((X, Y), (Z, 0)) :-
	Y > 0,
	Aux is X + Y,
	Aux =< 4,
	Z is Y + X.

% There is something in the 4-gallon jug (X > 0) and together with the amount in the 3-gallon jug it fits in the
% 3-gallon jug (Aux is X + Y, Aux =< 3), THEN fill it all (X is 0 in the new state) into the 3-gallon jug (Z is Y + X):
next_state((X, Y), (0, Z)) :-
	X > 0,
	Aux is X + Y,
	Aux =< 3,
	Z is Y + X.

% Empty the 4-gallon jug IF it is not already empty (X > 0):
next_state((X, Y), (0, Y)) :-
	X > 0.

% Empty the 3-gallon jug IF it is not already empty (Y > 0):
next_state((X, Y), (X, 0)) :-
	Y > 0.

action((_, Y), (4, Y), fill1).
action((X, _), (X, 3), fill2).
action((_, Y), (4, Z), put(2, 1)) :- Y \= Z.
action((X, _), (Z, 3), put(1, 2)) :- X \= Z.
action((X, _), (Z, 0), put(2, 1)) :- X \= Z.
action((_, Y), (0, Z), put(2, 1)) :- Y \= Z.
action((_, Y), (0, Y), empty1).
action((X, _), (X, 0), empty2).

print([], _).

print([H | T], 0) :-
	write(start), tab(4), write(H), nl,
	print(T, H).

print([H | T], Prev) :-
	action(Prev, H, X),
	write(X), tab(4), write(H), nl,
	print(T, H).
```

Output:-

```
% v:/CSMSS all/7th sem all notes/Ai notes/BFS.pl compiled 0.02 sec, 28 clauses
?- go.
start    0,0
fill2    0,3
put(2,1)    3,0
fill2    3,3
put(2,1)    4,2
true
```