Sagarmatha

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

[Document title]

[Document subtitle]

1. Write prolog program to simulate the water jug problem and verify that we can 2 liters of water in first jug.

Also try for goal such that first jug contains 1 liter of water.

* Theory:-

Water Jug Problem :-.

Consider the following problem:-

A Water Jug Problem:- You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

State Representation and Initial State: –

We will represent a state of the problem as a tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug. Note 0 ≤ x ≤ 4, and 0 ≤ y ≤ 3.

Our initial state: (0,0) Goal Predicate – state = (2,y) where 0 ≤ y ≤ 3.

Operators – we must define a set of operators that will take us from one state to another:

1. Fill 4-gal jug (x,y) → (4,y) x < 4

2. Fill 3-gal jug (x,y) → (x,3) y < 3

3. Empty 4-gal jug on ground (x,y) → (0,y) x > 0

4. Empty 3-gal jug on ground (x,y) → (x,0) y > 0

5. Pour water from 3-gal jug (x,y) → (4, y - (4 - x)) to fill 4-gal jug 0 < x+y ≥ 4 and y > 0

6. Pour water from 4-gal jug (x,y) → (x - (3-y), 3) to fill 3-gal-jug 0 < x+y ≥ 3 and x > 0

7. Pour all of water from 3-gal jug (x,y) → (x+y, 0) into 4-gal jug 0 < x+y ≤ 4 and y ≥ 0

8. Pour all of water from 4-gal jug (x,y) → (0, x+y) into 3-gal jug 0 < x+y ≤ 3 and x ≥ 0

Through Graph Search, the following solution is found: -

|  |  |  |
| --- | --- | --- |
| **Gals in 4 Gal Jug** | **Gals in 4 Gal Jug** | **Rule Applied** |
| 0 | 0 |  |
|  |  | 1. Fill 4 |
| 4 | 0 |  |
|  |  | 6. Pour 4 into 3 to fill |
| 1 | 3 |  |
|  |  | 4. Empty 3 |
| 1 | 0 |  |
|  |  | 8. Pour all of 4 into 3 |
| 0 | 1 |  |
|  |  | 1. Fill 4 |
| 4 | 1 |  |
|  |  | 6. Pour into 3 |
| 2 | 3 |  |

* Code:-

DATABASE

rstate(integer,integer)

PREDICATES

state(integer,integer)

CLAUSES

state(2,\_).

state(0,0):-

not(rstate(0,0)),

assert(rstate(0,0)),

state(0,0).

state(X,Y):-

X<4,

not(rstate(4,Y)),

assert(rstate(4,Y)),

write("\n Rule 1 => (4,",Y,")"),

state(4,Y).

state(X,Y):-

X<3,

not(rstate(X,3)),

assert(rstate(X,3)),

write("\n Rule 2 => (",X,",3)"),

state(X,3).

state(X,Y):-

X>0,

not(rstate(0,Y)),

assert(rstate(0,Y)),

write("\n Rule 3 => (0,",Y,")"),

state(0,Y).

state(X,Y):-

Y<3,

not(rstate(X,0)),

assert(rstate(X,0)),

write("\n Rule 4 => (",X,",0)"),

state(X,0).

state(X,Y):-

X+Y>=4,

Y>0,

Z=Y-(4-X),

not(rstate(4,Z)),

assert(rstate(4,Z)),

write("\n Rule 5 => (4,",Z,")"),

state(4,Z).

state(X,Y):-

X+Y>=3,

X>0,

Z=Y-(3-X),

not(rstate(Z,3)),

assert(rstate(Z,3)),

write("\n Rule 6 => (",Z,",3)"),

state(Z,3).

state(X,Y):-

X+Y<=4,

Y>0,

Z=X+Y,

not(rstate(Z,0)),

assert(rstate(Z,0)),

write("\n Rule 7 => (",Z,",0)"),

state(Z,0).

state(X,Y):-

X+Y<=3,

X>0,

Z=X+Y,

not(rstate(0,Z)),

assert(rstate(0,Z)),

write("\n Rule 8 => (0,",Z,")"),

state(0,Z).

GOAL

write("Initial State(0,0)");

state(0,0),

save("output.txt").

* Output:-

