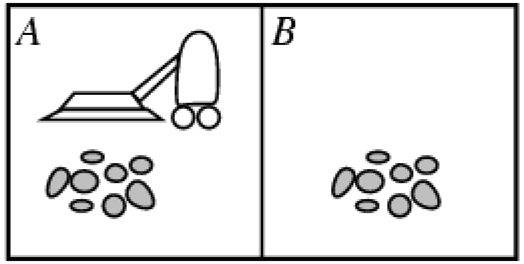
Problem Solving using
State Space Representations

Problem solving components

- **Initial state:** The agent knows itself to be in
- **Operator:** description of an action
- **State space:** all states reachable from the initial state by any sequence action
- **Path**: sequence of actions leading from one state to another
- **Goal test:** which the agent can apply to a single state description to determine if it is a goal state
- **Path cost function**: assign a cost to a path which the sum of the costs of the individual actions along the path.

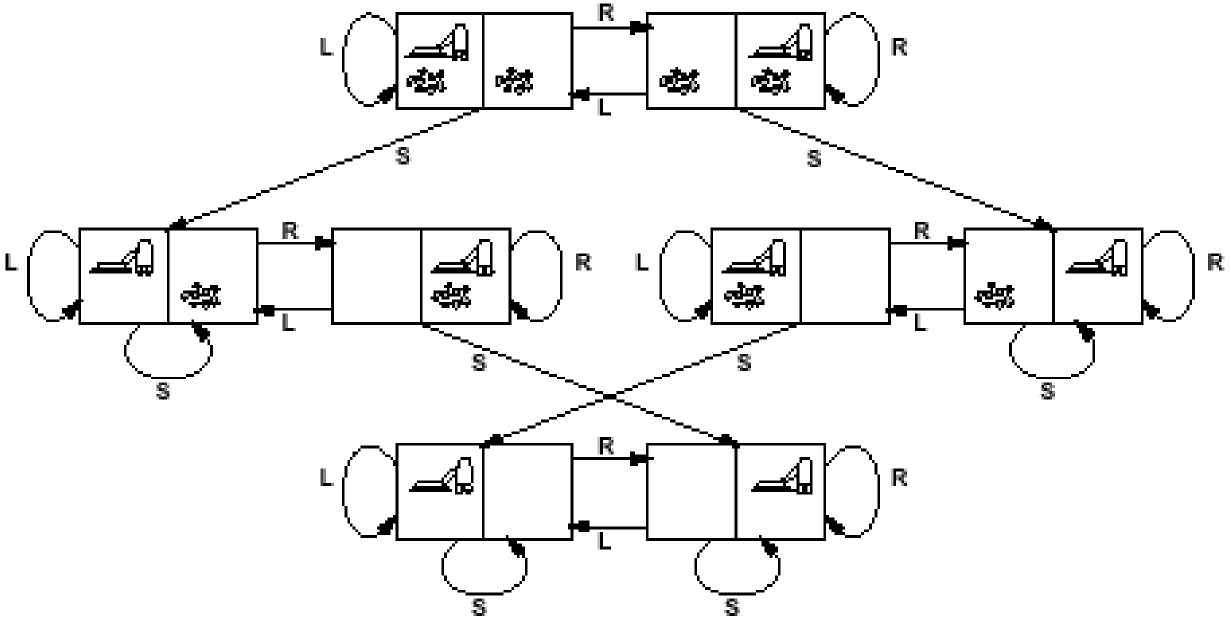
Vacuum-cleaner world



- Percepts: location and state of the environment, e.g., [A,Dirty], [A,Clean], [B,Dirty]

- Actions: *Left*, *Right*, *Suck*, *NoOp*

Vacuum-cleaner world



Contd...

- **States:** S_1 , S_2 , S_3 , S_4 , S_5 , S_6 , S_7 , S_8
- **Operators:** Go Left , Go Right , Suck
- **Goal test**: no dirt left in both squares
- **Path Cost:** each action costs 1.

Real-world problems

- Routine finding
  - Routing in computer networks
  - Automated travel advisory system
  - Airline travel planning system
  - Goal: the best path between the origin and the destination
- Travelling Salesperson problem (TSP)
  - Is a famous touring problem in which each city must be visited exactly once.
  - Goal: shortest tour

Example: Traveling in Romania

- On holiday in Romania; currently in Arad
- Flight leaves tomorrow from Bucharest

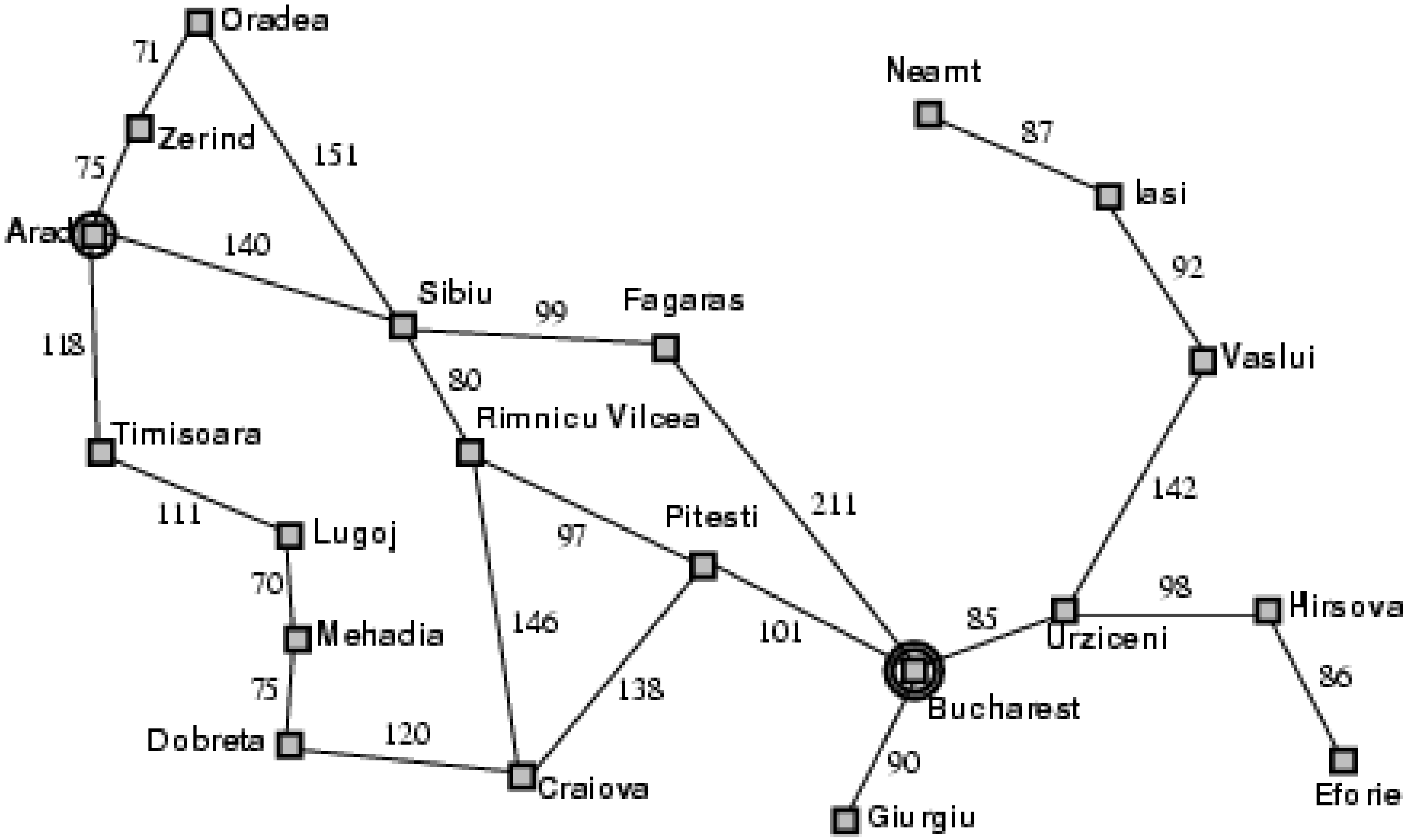
- Formulate goal:
  - be in Bucharest

- Formulate problem:
  - states: various cities
  - actions/operators: drive between cities

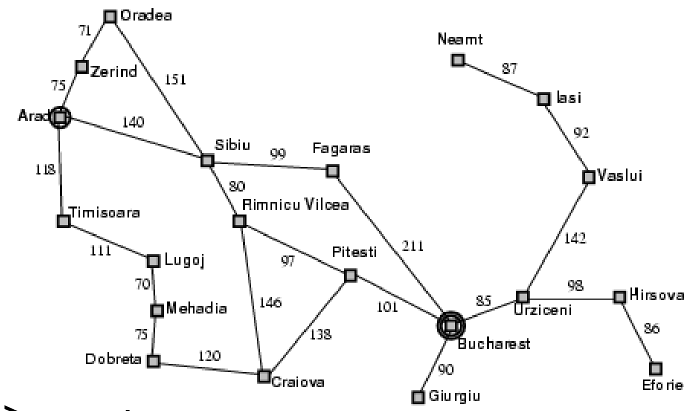
- Find solution
  - By searching through states to find a goal
  - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

- Execute states that lead to a solution

Example: Traveling in Romania



State-Space Problem Formulation



A **problem** is defined by four items:

1. **initial state** e.g., "at Arad"

2. **actions** or successor function

   S(x) = set of action–state pairs

   e.g., *S(Arad) = {<Arad → Zerind, Zerind>, … }*

3. **goal test** (or set of goal states)

   e.g., *x* = "at Bucharest", *Checkmate(x)*

4. **path cost** (additive)

   e.g., sum of distances, number of actions executed, etc.

   c(x,a,y) is the step cost, assumed to be ≥ 0

A **solution** is a sequence of actions leading from the initial state to a goal state

Example: Formulating the Navigation Problem

- Set of States
  - individual cities
  - e.g., Irvine, SF, Las Vegas, Reno, Boise, Phoenix, Denver

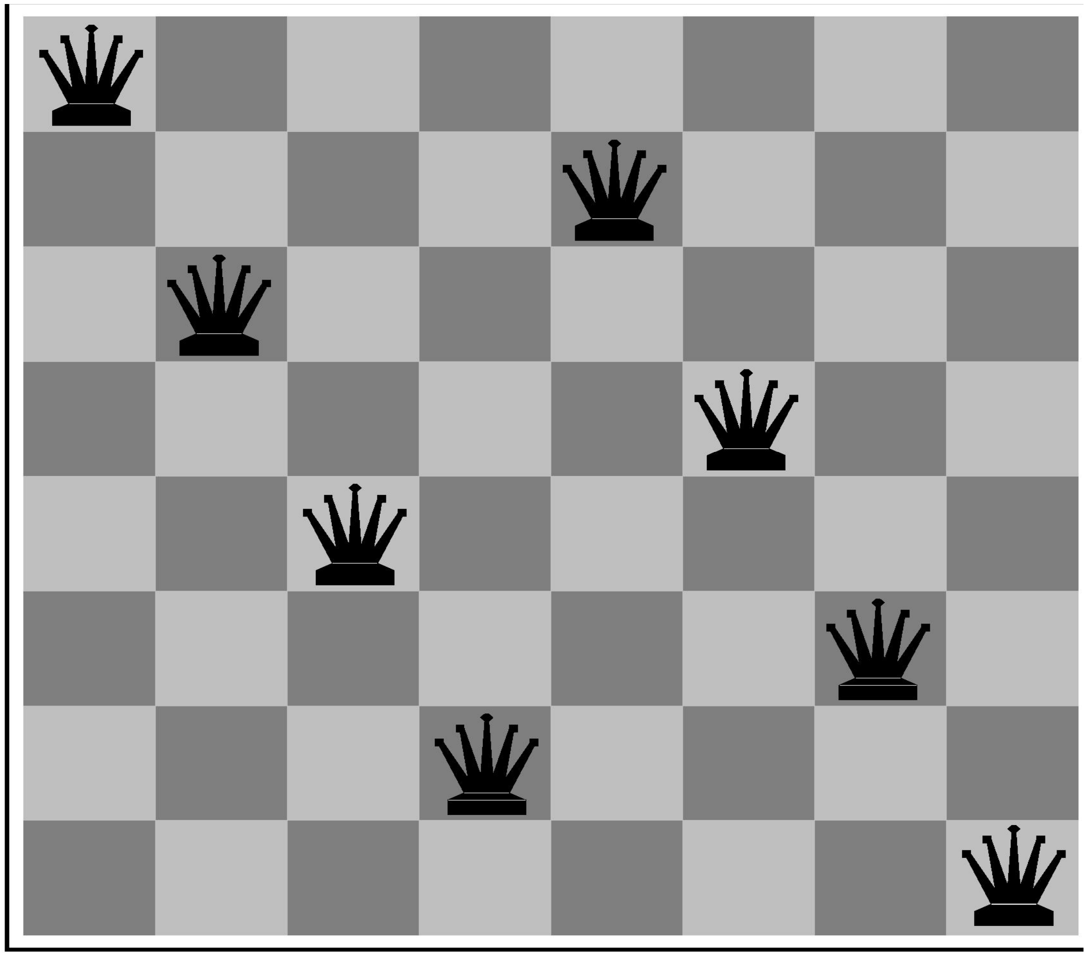
- Operators
  - freeway routes from one city to another
  - e.g., Irvine to SF via 5, SF to Seattle, etc

- Start State
  - current city where we are, Irvine

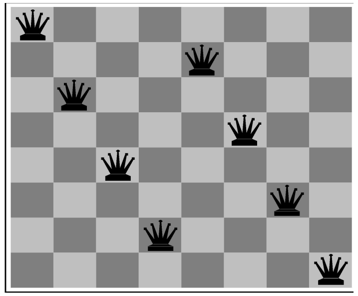
- Goal States
  - set of cities we would like to be in
  - e.g., cities which are closer than Irvine

- Solution
  - a specific goal city, e.g., Boise
  - a sequence of operators which get us there,
    - e.g., Irvine to SF via 5, SF to Reno via 80, etc

Example: 8-queens problem



State-Space problem formulation



- states? -any arrangement of n<=8 queens
  -*or* arrangements of n<=8 queens in leftmost n columns, 1 per column, such that no queen attacks any other.

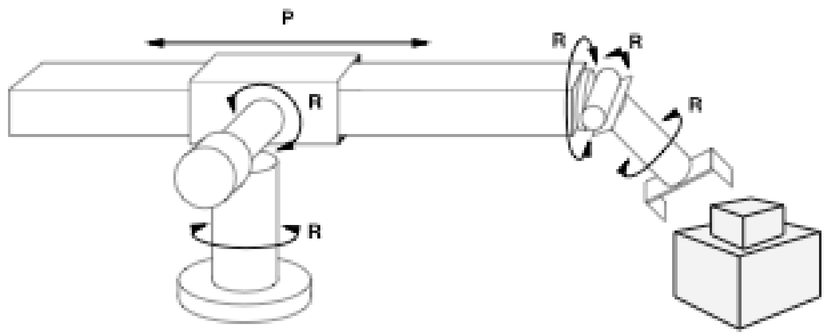
- initial state? no queens on the board

- actions? -add queen to any empty square
  -*or* add queen to leftmost empty square such that it is not attacked by other queens.

- goal test? 8 queens on the board, none attacked.

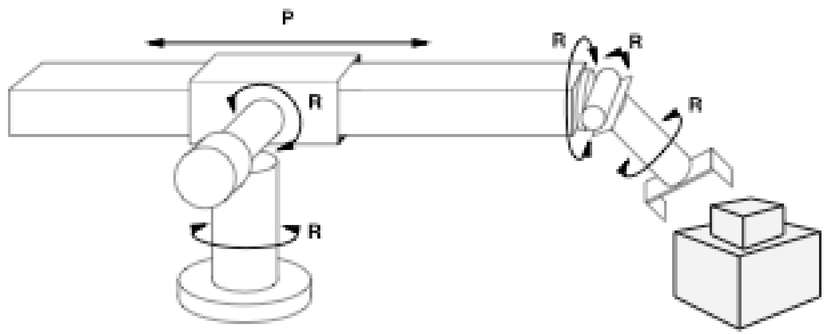
- path cost? 1 per move

Example: Robot Assembly



- States
- Initial state
- Actions
- Goal test
- Path Cost

Example: Robot Assembly



- States: configuration of robot (angles, positions) and object parts
- Initial state: any configuration of robot and object parts
- Actions: continuous motion of robot joints
- Goal test: object assembled?
- Path Cost: time-taken or number of actions

Learning a spam email classifier

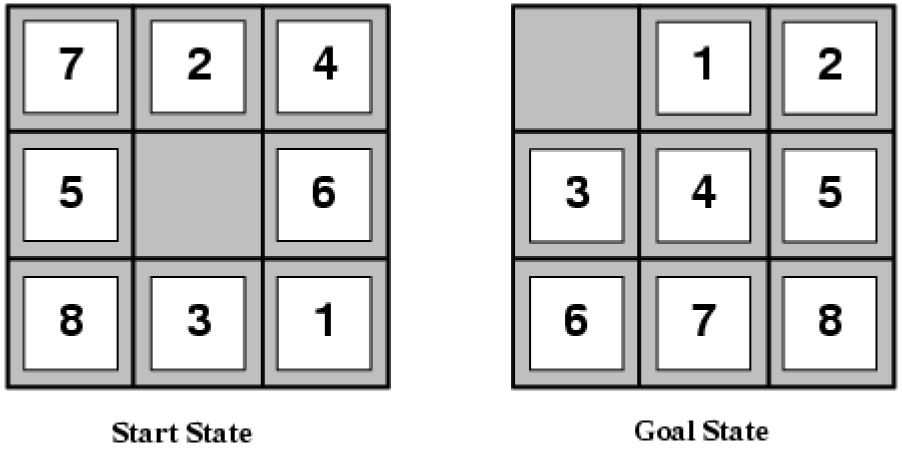
- States
- Initial state
- Actions
- Goal test
- Path Cost

Learning a spam email classifier

- States: settings of the parameters in our model
- Initial state: random parameter settings
- Actions: moving in parameter space
- Goal test: optimal accuracy on the training data
- Path Cost: time taken to find optimal parameters

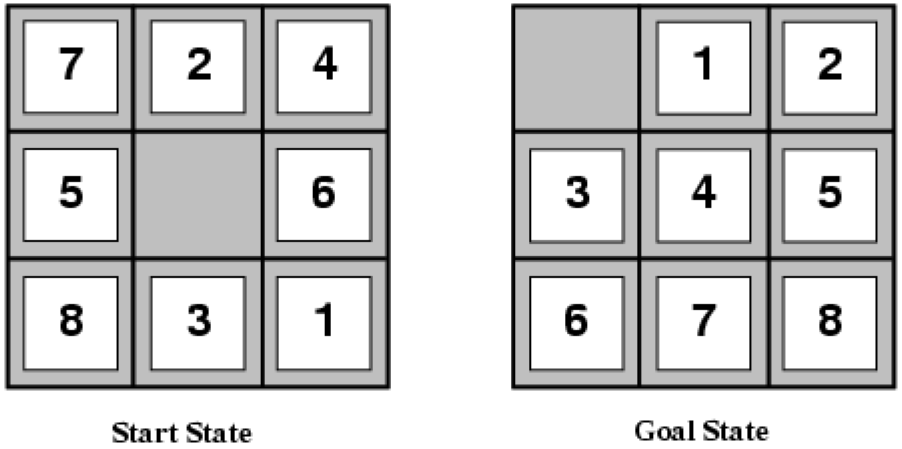
(Note: this is an optimization problem – many machine learning problems can be cast as optimization)

Example: 8-puzzle



- states?
- initial state?
- actions?
- goal test?
- path cost?

Example: 8-puzzle



- states? locations of tiles
- initial state? given
- actions? move blank left, right, up, down
- goal test? goal state (given)
- path cost? 1 per move

Crypt-Arithmetic puzzle

- Problem Statement:
  - Solve the following puzzle by assigning numeral (0-9) in such a way that each letter is assigned unique digit which satisfy the following addition.
  - Constraints : No two letters have the same value. (The constraints of arithmetic).

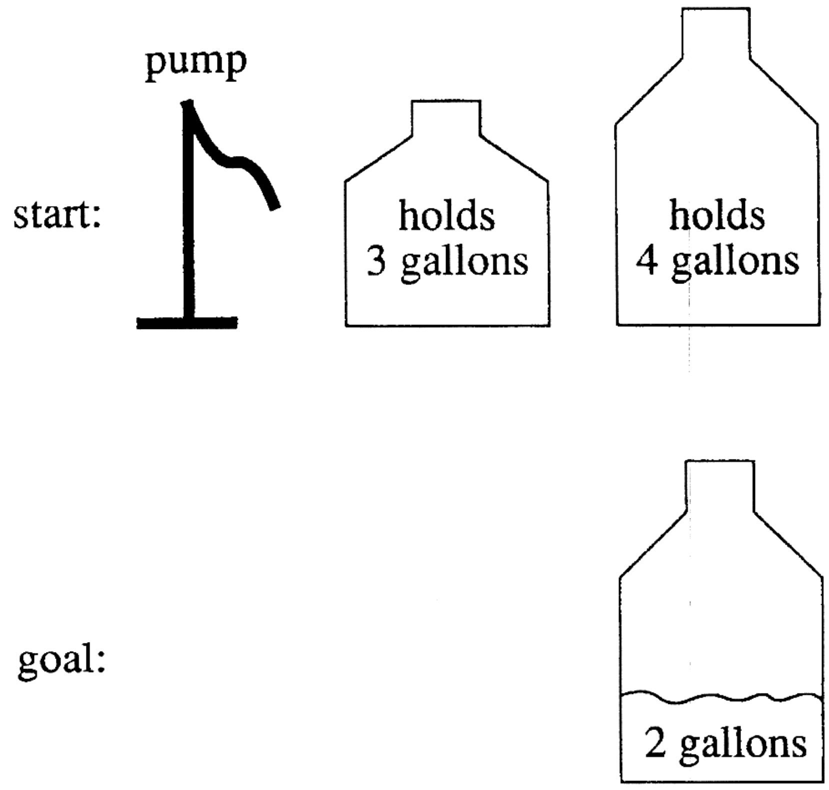
- CROSS + ROADS = DANGER

- SEND + MORE = MONEY

- Initial Problem State
  - S = ? ; E = ? ;N = ? ; D = ? ; M = ? ;O = ? ; R = ? ;Y = ?

A Water Jug Problem

- You have a 4-gallon and a 3-gallon water jug
- You have a faucet with an unlimited amount of water
- You need to get exactly 2 gallons in 4-gallon jug



Puzzle-solving as Search

- State representation**: (x, y)**
  - x: Contents of four gallon
  - y: Contents of three gallon
- Start state**: (0, 0)**
- Goal state **(2, n)**
- Operators
  - Fill 3-gallon from faucet, fill 4-gallon from faucet
  - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
  - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
  - Dump 3-gallon down drain, dump 4-gallon down drain

Production Rules for the Water Jug Problem

1 $(x,y) \rightarrow (4,y)$ if $x < 4$	Fill the 4-gallon jug
2 $(x,y) \rightarrow (x,3)$ if $y < 3$	Fill the 3-gallon jug
3 $(x,y) \rightarrow (x - d,y)$ if $x > 0$	Pour some water out of the 4-gallon jug
4 $(x,y) \rightarrow (x,y - d)$ if $x > 0$	Pour some water out of the 3-gallon jug
5 $(x,y) \rightarrow (0,y)$ if $x > 0$	Empty the 4-gallon jug on the ground
6 $(x,y) \rightarrow (x,0)$ if $y > 0$	Empty the 3-gallon jug on the ground
7 $(x,y) \rightarrow (4,y - (4 - x))$ if $x + y \geq 4$ and $y > 0$	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full

The Water Jug Problem (cont'd)

8 $(x,y) \rightarrow (x - (3 - y),3)$ if $x + y \geq 3$ and $x > 0$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full
9 $(x,y) \rightarrow (x + y, 0)$ if $x + y \leq 4$ and $y > 0$	Pour all the water from the 3-gallon jug into the 4-gallon jug
10 $(x,y) \rightarrow (0, x + y)$ if $x + y \leq 3$ and $x > 0$	Pour all the water from the 4-gallon jug into the 3-gallon jug

One Solution to the Water Jug Problem

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	