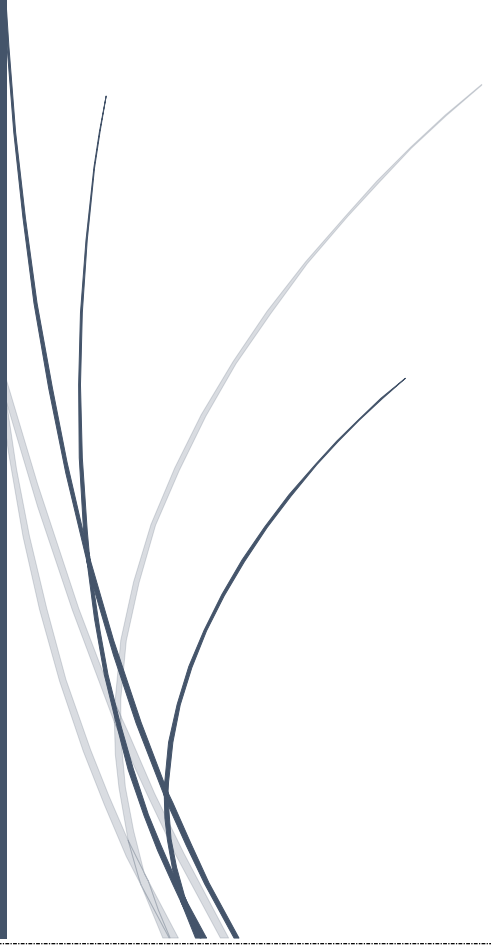
4/10/2023

Planning & Analysis

For Shelf Inventory Robot/Agent



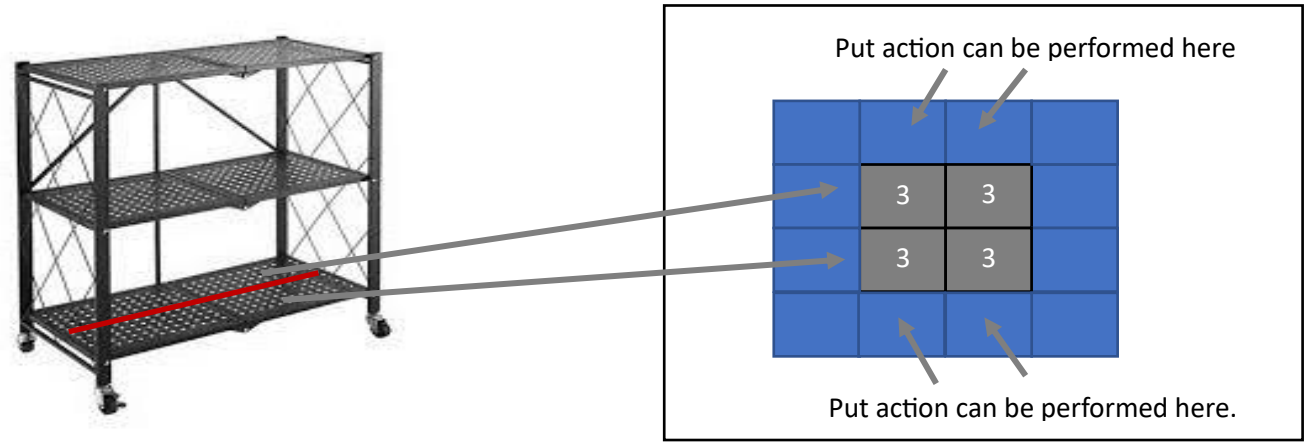
A. Problem Definition:

The shelf inventory problem describes a problem where we want to put boxes on inventory shelves. The objective here is to minimize the cost. The cost is represented in terms of blocks in world space where each block has the same cost (uniform cost).

Assumptions are:

- The problem is concerned only with storage not retrieving to apply search algorithms.
- Each Shelf has three levels, and all are identical.
- For the sake of simplicity, all 3 levels will be treated in the same way, for example putting a box in level 3 is the same as level 1.
- Also, for sake of simplicity rather than defining actions like putting a box in level 1, putting a box in level 2, or putting a box in level 3. It will be replaced with a general action put box where the agent needs to find the first empty place to put the box in (Assuming there is an empty level)
- Any put action updates the database (World grid).
- Each level has 2 horizontal sides & each horizontal side has 2 free empty slots (in summary each level has 4 empty quarters).
- To perform put action, the agent must be on top or below the cell only as shown in figure.

Illustration for mapping 1 shelf to the logic base architecture in the figure below, where grey color represents the shelf and blue color represents the road for agents.



B. Environment Type:

The environment will consist of two shelves stack vertically with two agents. Environment Properties:

- **Accessible:** Agents can obtain complete and accurate information about the shelves.
- **Deterministic & Fully Observable:** Based on the agent's current state and the selected action, we can determine the next state of the shelves. Therefore, there is no need to worry about any uncertainty.
- **Discrete:** There are 5 actions that can be performed within the shelves. The five actions are: Move Up – Move Down – Move Right – Move Left – Put action. The agents can stop and think about the next action to take to reach the goal state.
- **Static:** each agent knows what the other agent is changing in the environment based on the common DB.
- **Episodic/Sequential:** The agent starts in a particular state, performs a sequence of actions, and then reaches a goal state.

C. Agent Type:

Both agents will follow the logic-based architecture that best fits our problem as mentioned before where the agents must collaborate and coordinate with each other to reach the goal state. Between the agent there is:

- **Collaborate:** How to accomplish the delegated objective? (I.e., It is about the agents working together in general as a team)
- **Coordinate:** Who does what? (I.e., It is about organizing each agent's effort within the team to reach the goal state)

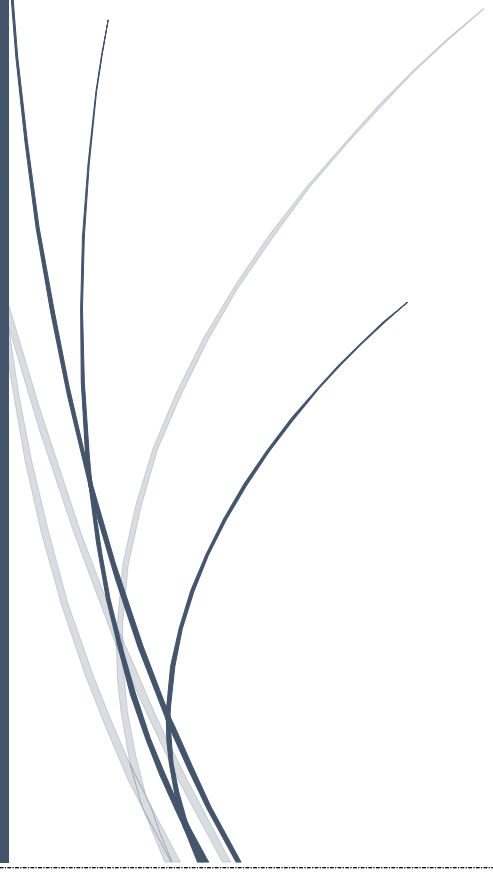
D. PEAS (Performance measure, Environment, Actuators, Sensors).

- **Performance measure:** putting the item in an empty slot with the minimum cost & time (closest empty slot to an agent).
- **Environment:** the shelf will be the Environment for our two agents.
- **Actuators:** three-level lifters (for carrying items), and 4 wheels (for agents' movement).
- **Sensors:** camera for detecting empty slots & Radar.

4/10/2023

Design

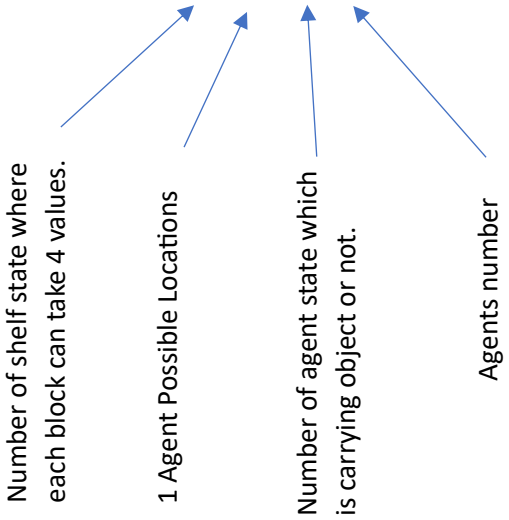
For Shelf Inventory Robot/Agent

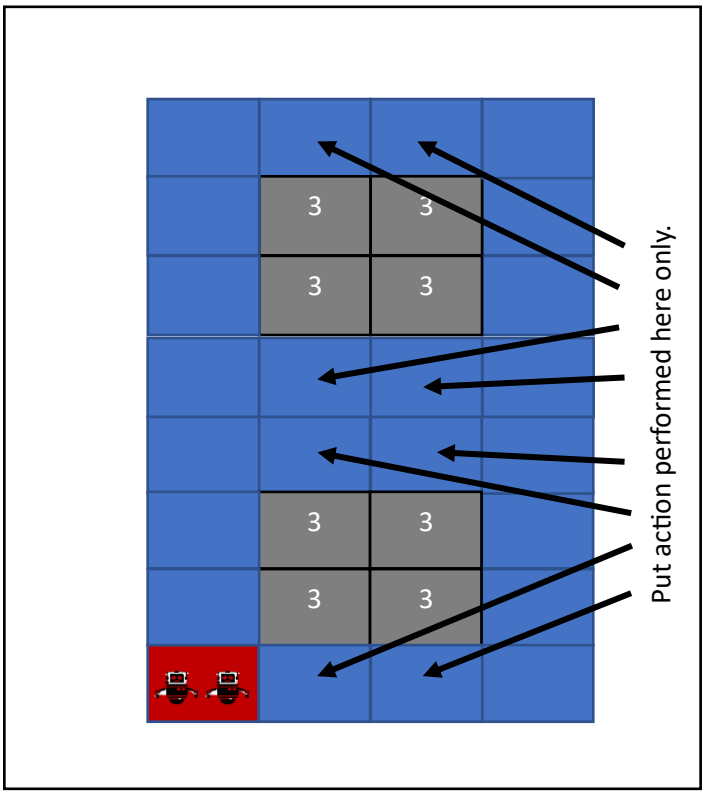


A-Problem State Space

State space refers to the collection of all possible states that the system can be in at any given time including the system's internal state and its interactions with the environment. It provides a formal framework for modeling the system's behavior, reasoning about its dynamics, and developing control strategies and algorithms to achieve desired outcomes.

Problem State = $4^8 * 24 * 2 * 2 = 6291456$





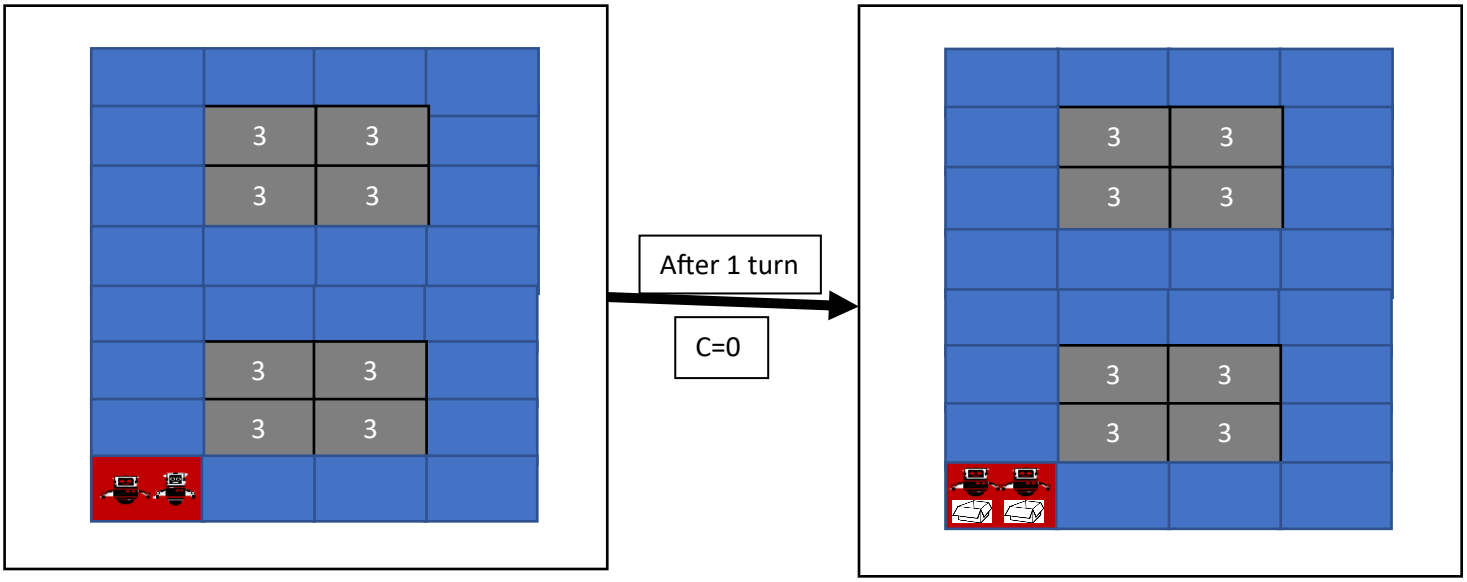
More Assumptions:

- Both agents can be on the same block.
- Each agent must know other agent intension on order to avoid conflict.
- Each agent can carry 1 box at time
- Red block represents the block where agent can get the next box if it doesn't carry one.
- Agent wasted 1 turn to load carry box at (1,1) but it not counted into cost since it concerned with distance.
- Put action(P) is the only action that doesn't increment the accumulated cost and available only in some states. While taking any movement action increases the accumulated cost by 1.
- 1 agent has higher priority than the other as it selects the minimum path first.
- If two paths have the same cost agent 1 pick it and increase it cost by 1. Then the other agent picks the path with minimum cost which will be the other one (idea for that is to prove way to tell other agent how long, it will take to put his box since they can't perform put action in same time. Not it handles cases like if costs are 2 & 4 as it forces the second agent to take first path and wait 1 turn – details will be provided later in next phase)

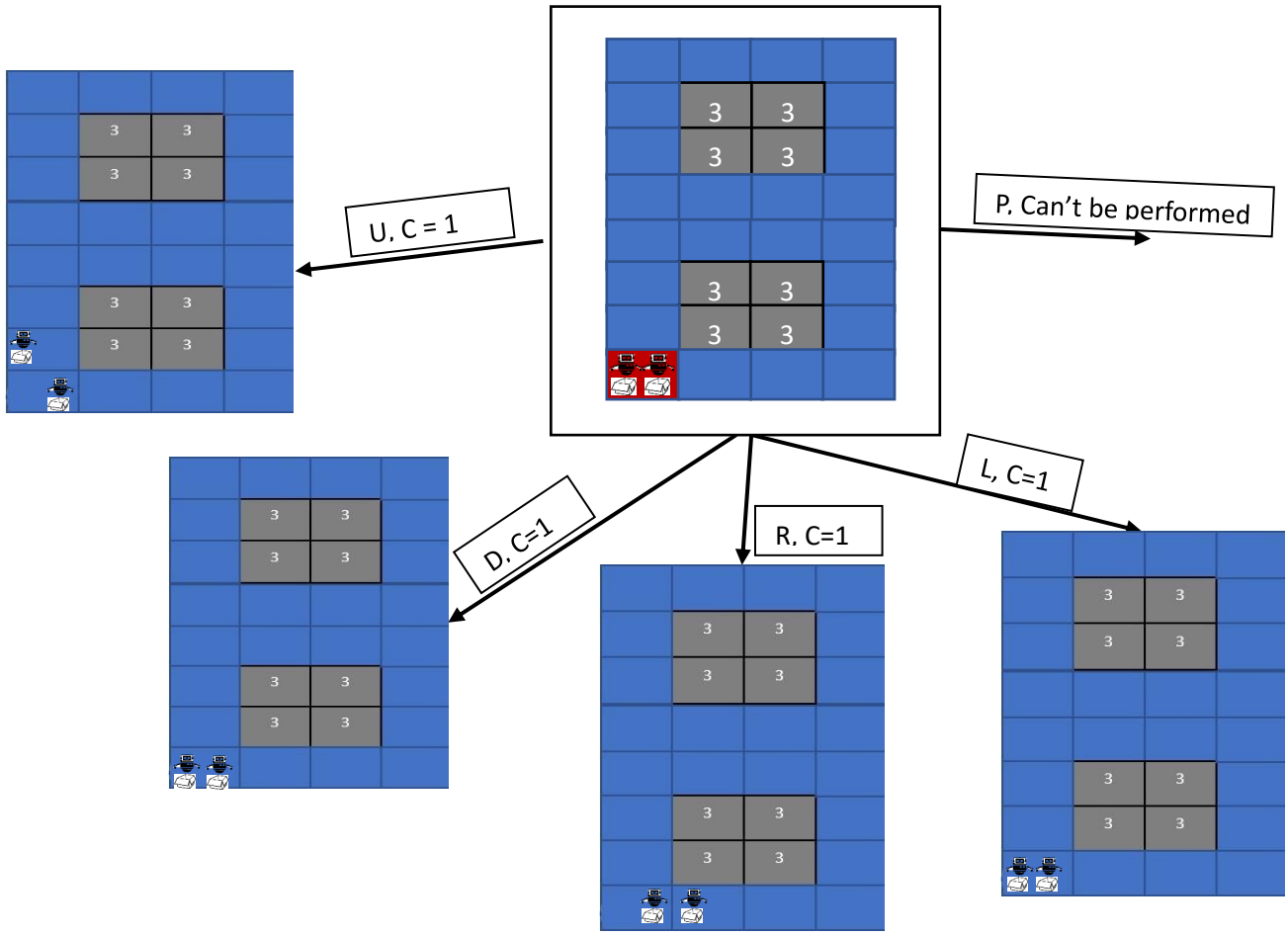
Initial state represented in where each agent doesn't carry box and every shelf is empty. Like previous image

Since the number of states is 6,000,000+. So, for the sake of simplicity in plot the state diagram. we assume that our subgoal state space is that each agent should deliver 2 boxes from (1,1) cell (the red cell) the southern west cell of the shelf (bottom left grey cell).

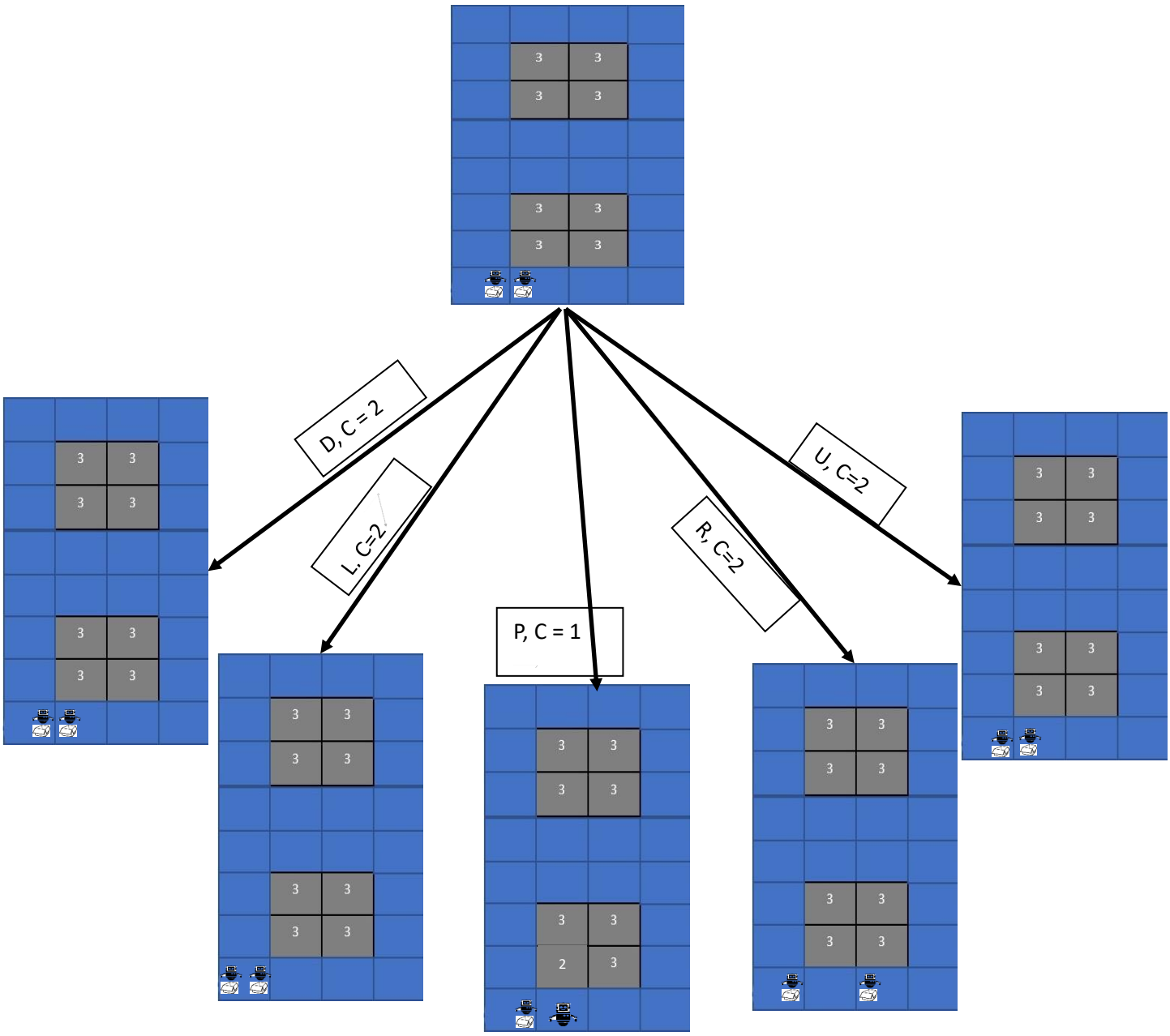
State subspace for Both Agent at turn = 0:



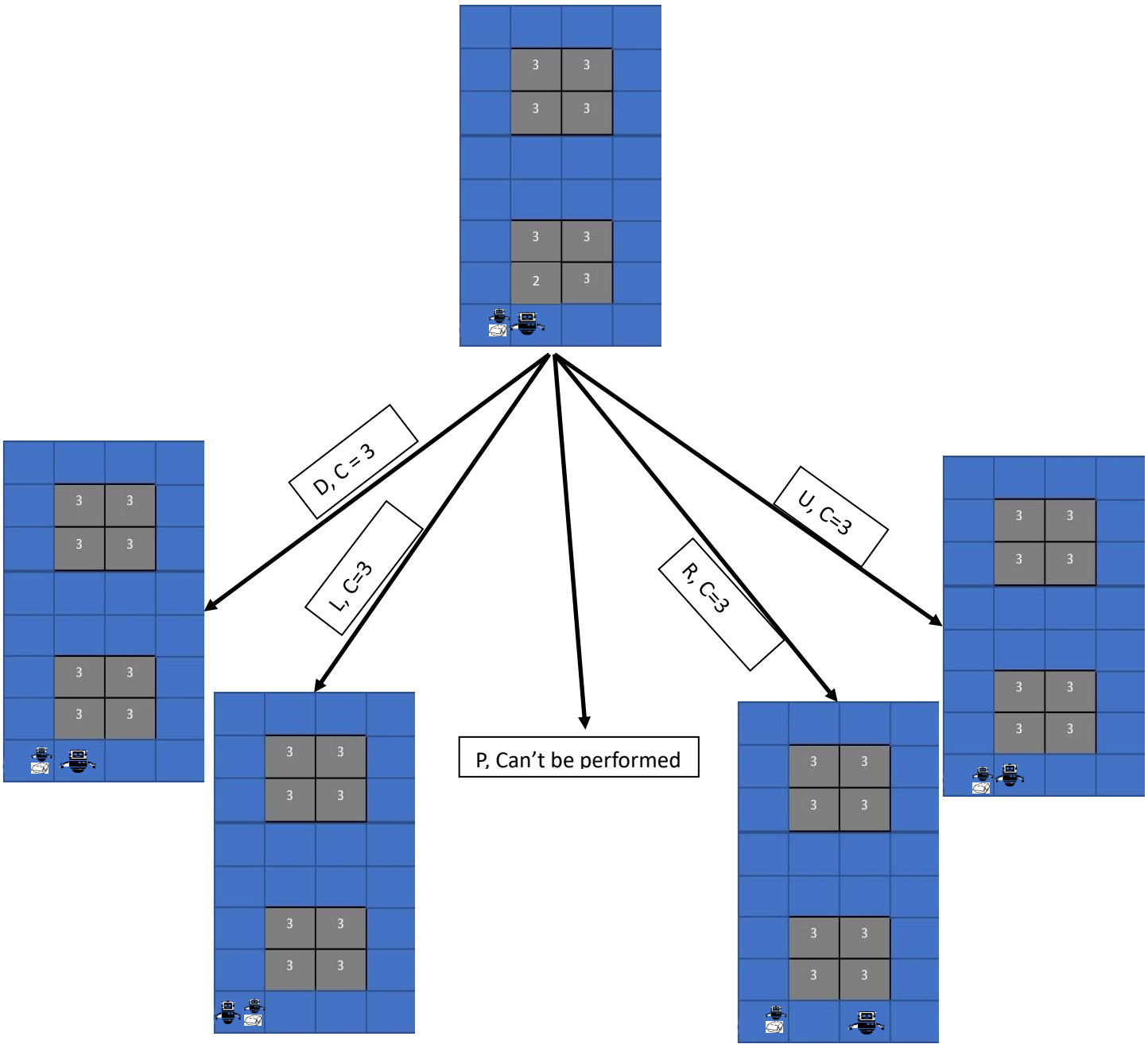
State subspace for agent 1:



Algorithm (BFS) Suggest point (2,1) for agent 1 so we will expand it:



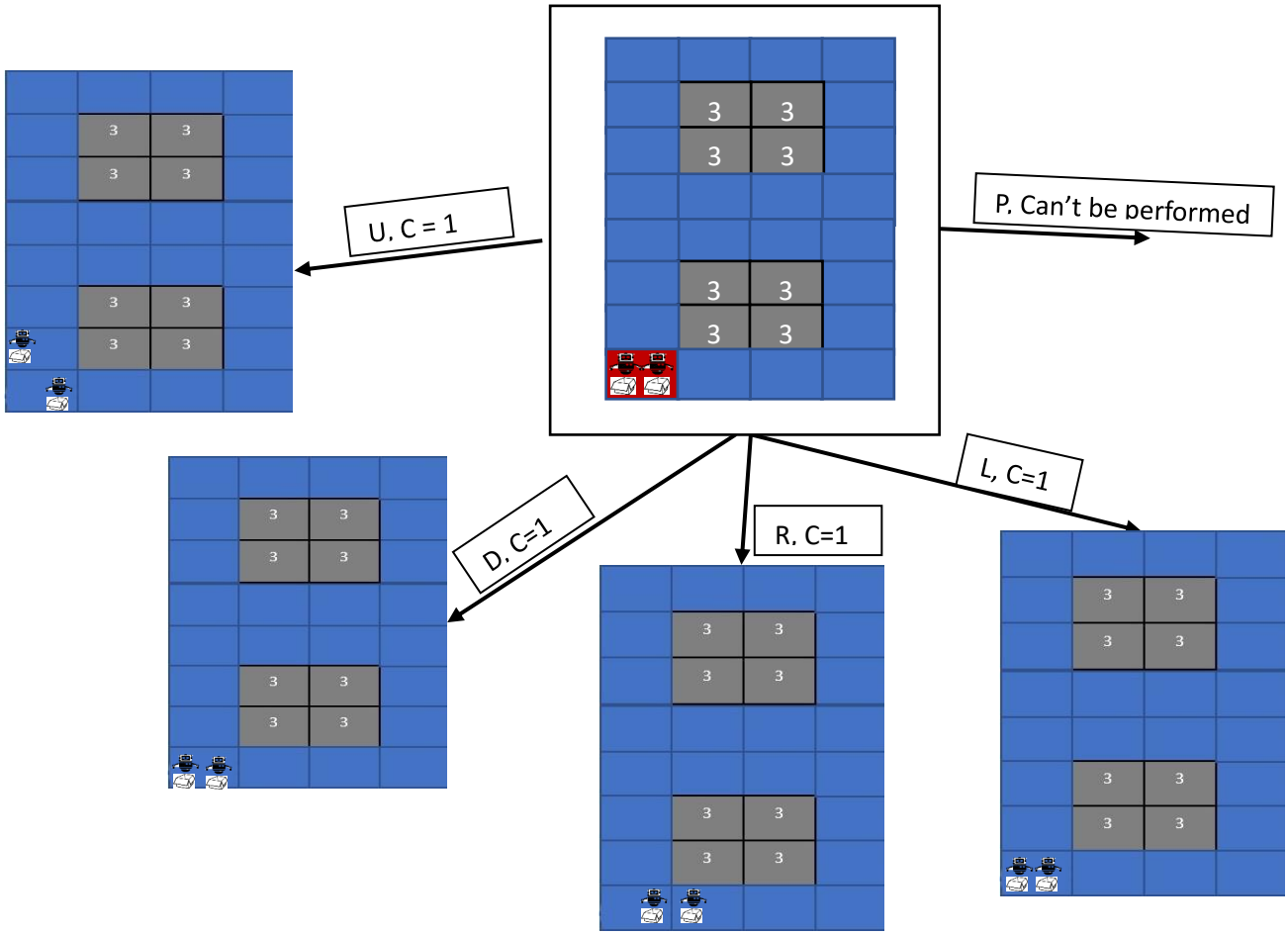
Since the put action is performed agent now must return to (1,1) to get another box.



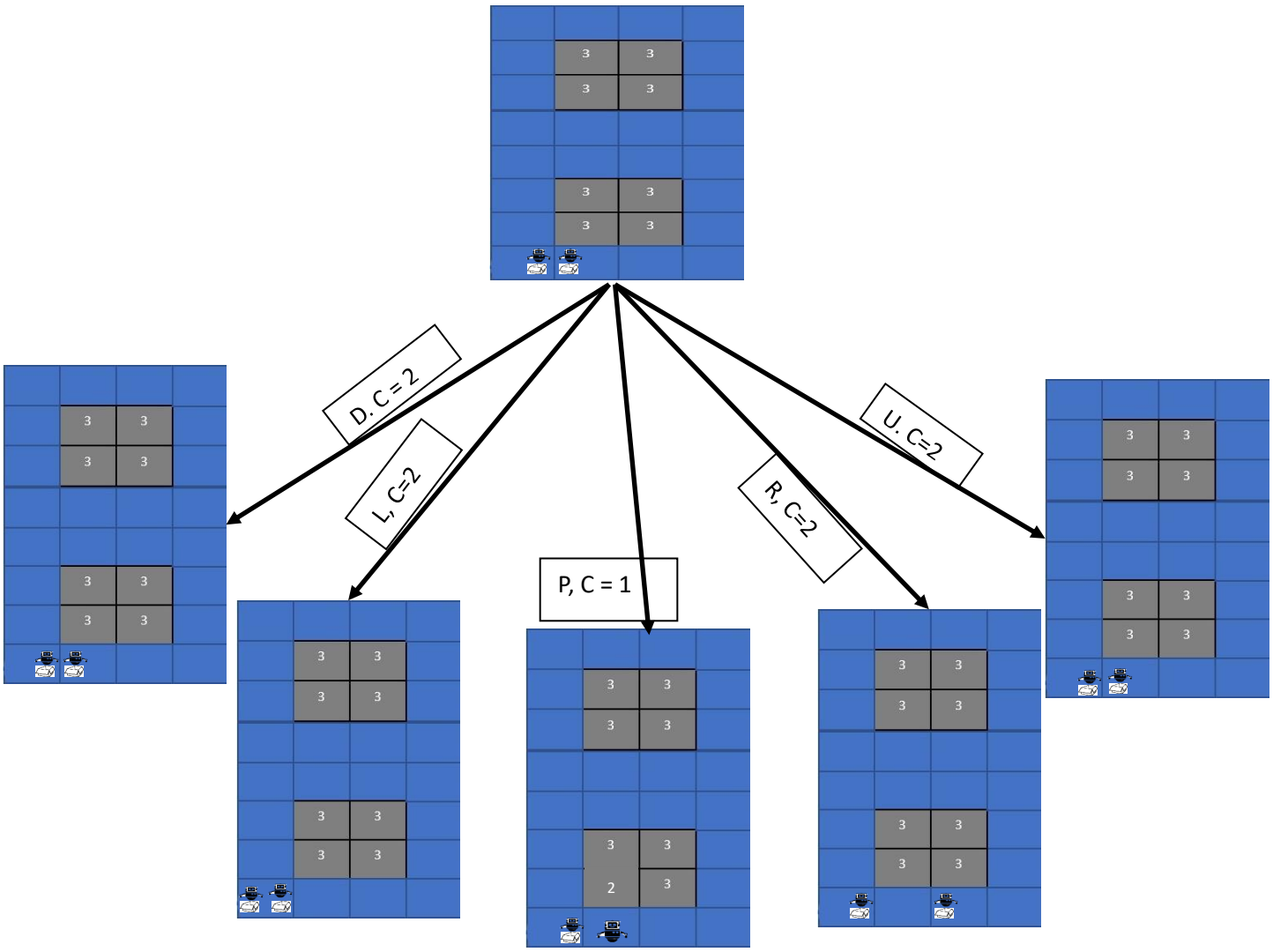
Now it back to (1,1) and put the box after 3 turn – 2 for movement & 1 for loading the first box.

Also, it wastes 1 another turns to load box so total is 4 turns.

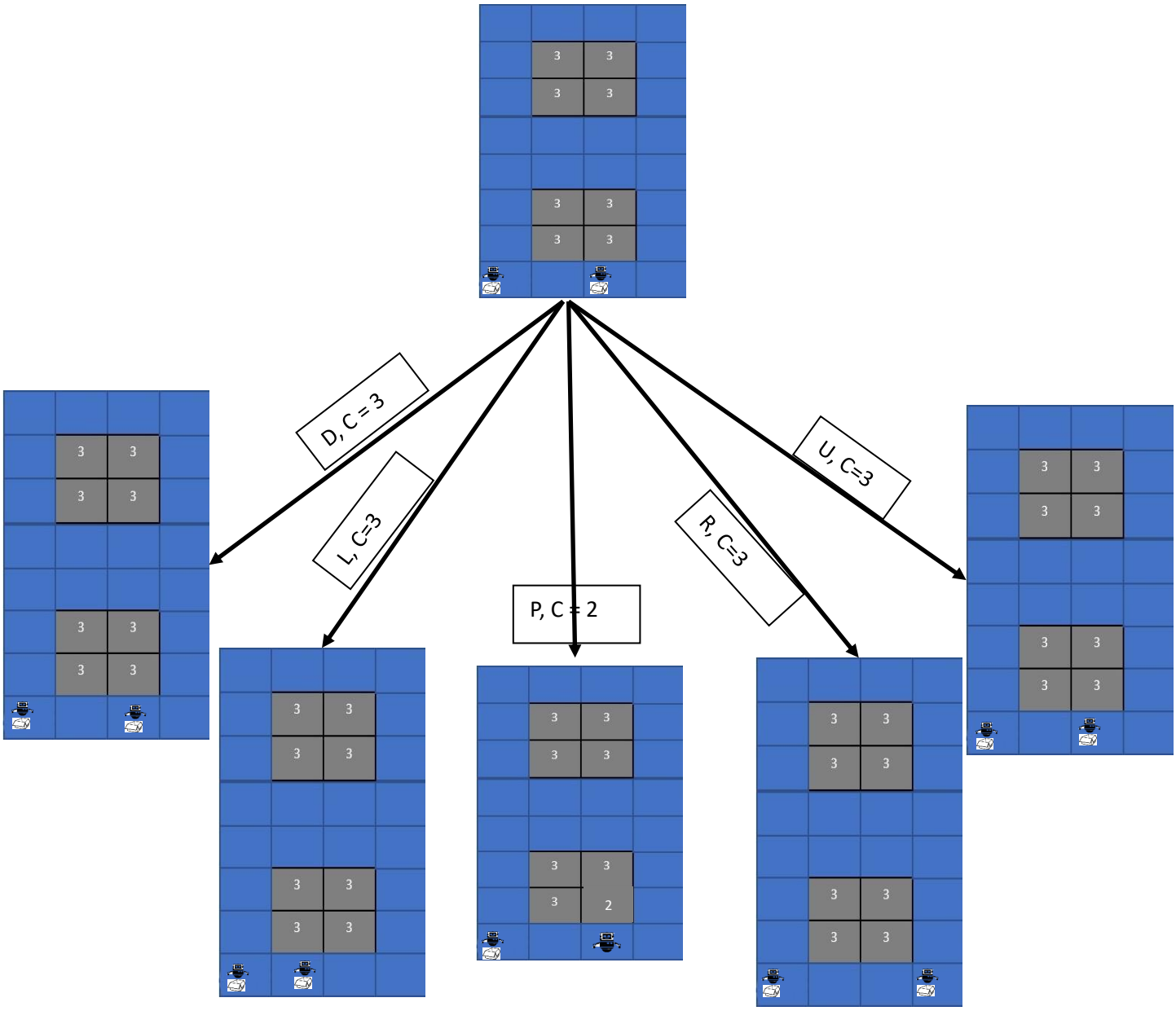
State subspace for agent 1 (4 Turns only):



Algorithm (BFS) Suggest point (3,1) for agent 1 so we will follow shortest path:



Although put action can be performed but algorithm assign agent 2 to perform put action in (3,1)



Since the put action is performed agent now must return to (1,1) to get another box. Simply It apply backtracking ong taken path (same was apply for agent 1)

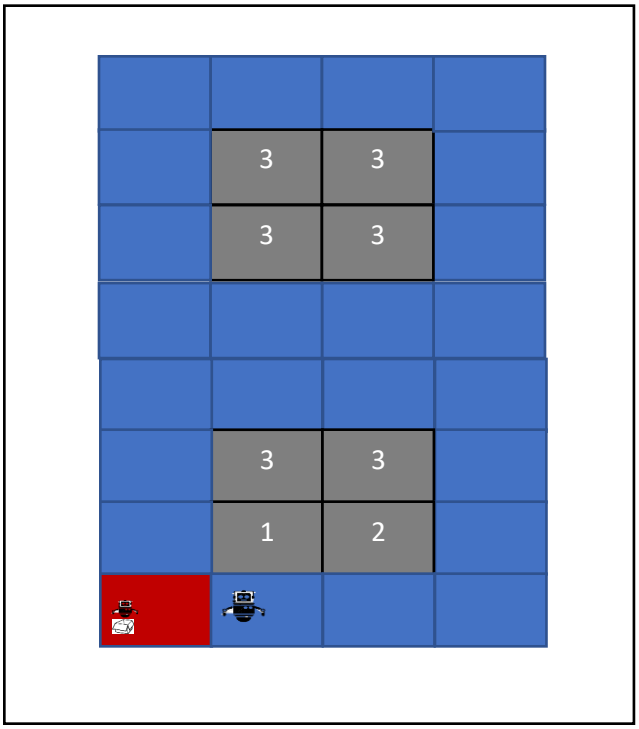
-So, in summary in 4 turns:

- Agent 1: put 1 box and stop at (1,1) with box.
- Agent 2: put 1 box and stop at (2,1) without box (empty hand)

B-Goal State

Given N box it must place with minimum cost & time as example if we have 3 box the result should be as follows.

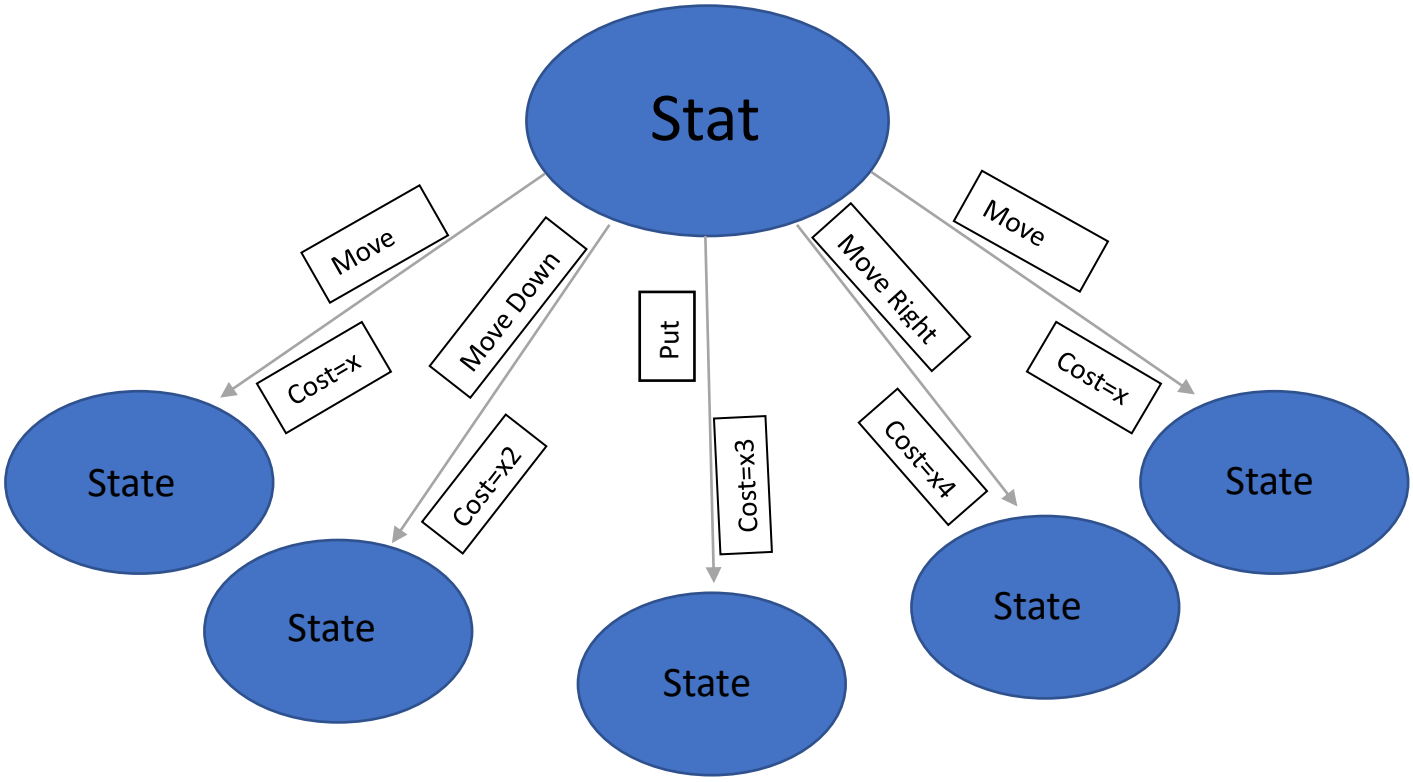
1) At turn = 0, both agents load the box.
2) At turn = 1, both agents at (2,1)
3) At turn = 2, agent 1 put the box & agent 2 go to (3,1)
4) At turn = 3, agent 1 back to (1,1) & agent 2 put the box
5) At turn =4, agent 1 waste turn to load the box, agent 2 back to (2,1)
6) At turn = 5, agent 1 go to (2,1) & agent 2 at (1,1)
7) At turn = 6, agent 1 put the box & agent 2 waste 1 turn to load the box
8) Since all boxes are put the goal has reached with this state



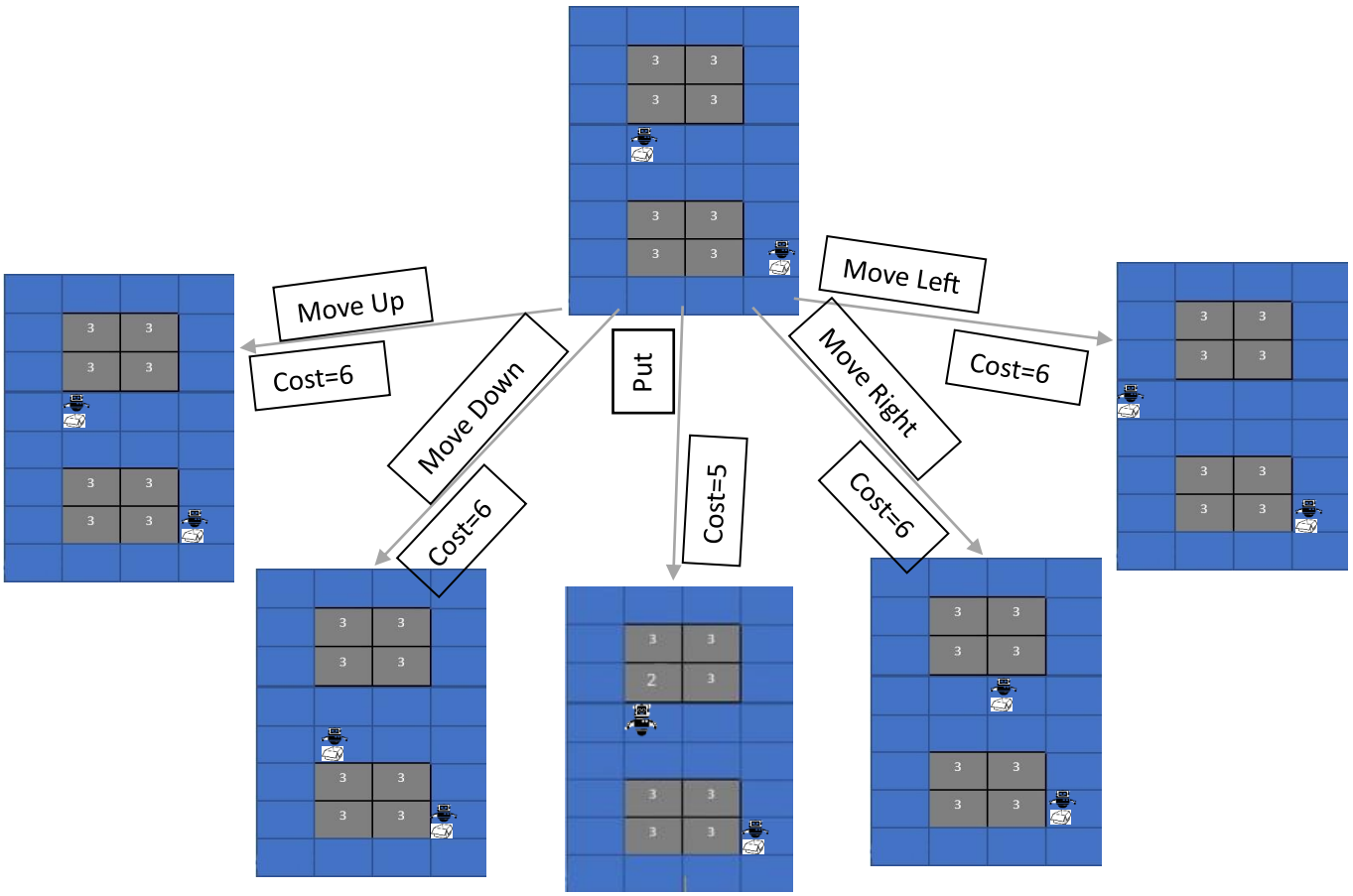
C-Successor function

The successor function is a way to define the possible next states that can be reached from the current state of a system. It is represented by Successor (S) and is used in various applications, including search algorithms and game theory.

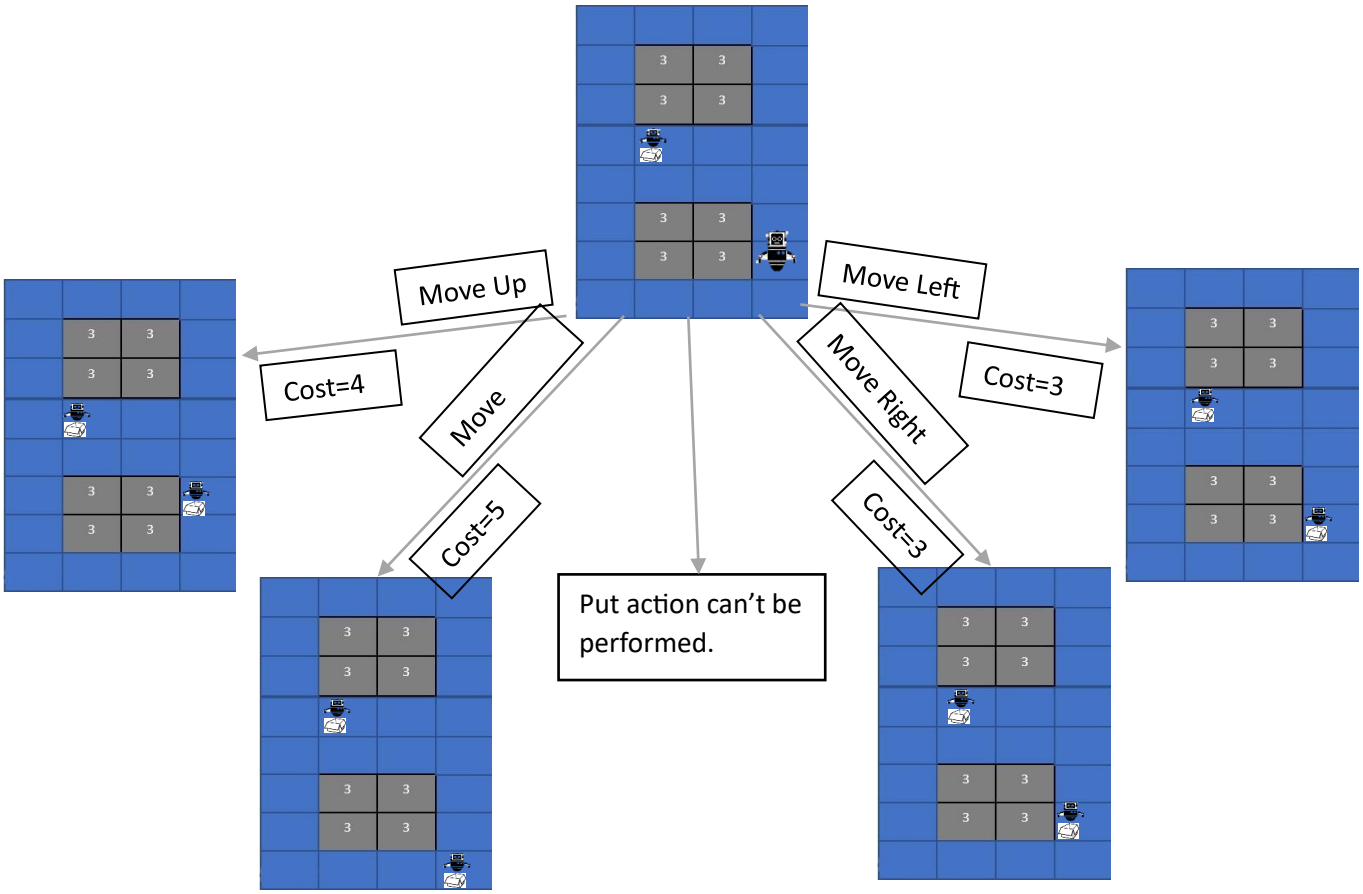
In the context of a system or game, there are typically a set of actions that can be taken to reach the next state. In this case, there are five actions: Move Up, Move Down, Move Right, Move Left, and Put. The cost of each action is measured by the accumulative path cost, which is the sum of all the costs incurred in reaching the current state.



-Example For agent 1 were going to block (2,5) cost 6:



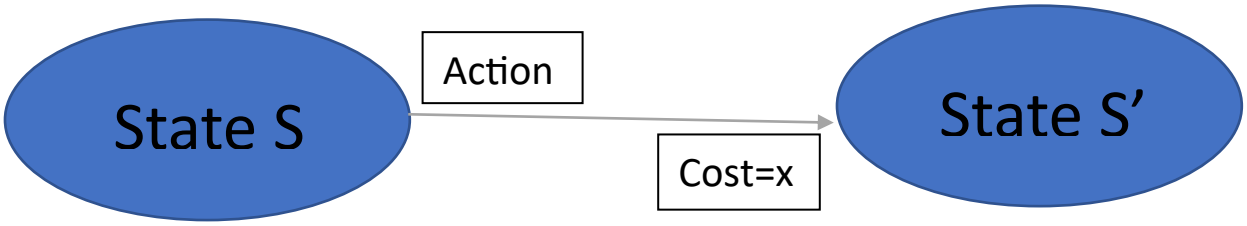
-For agent 2:



D-Transition Model

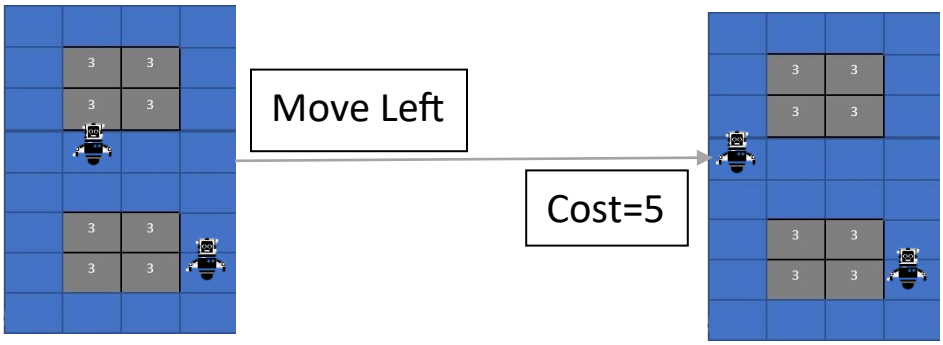
The transition model is a mathematical model that describes how a system or game moves from one state to another based on an action. It is represented by Transition (S, a), where S is the current state and a is the action taken.

In the context of a system or game, there are typically a set of actions that can be taken, such as Move Up, Move Down, Move Right, Move Left, and Put. The cost of each action is measured by the accumulative path cost, which is the sum of all the costs incurred in reaching the current state.

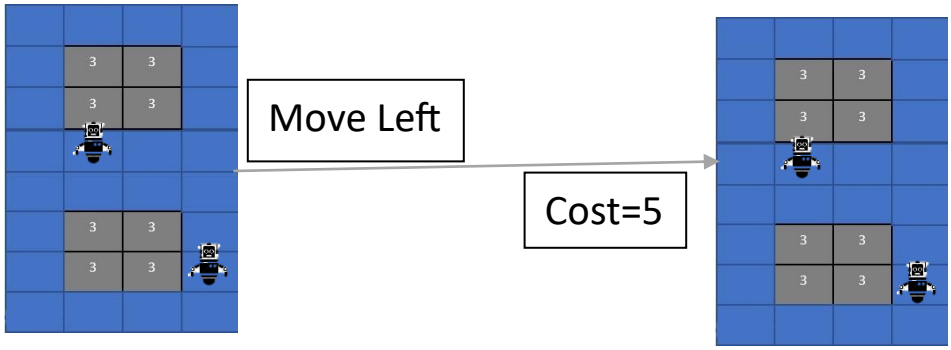


Example:

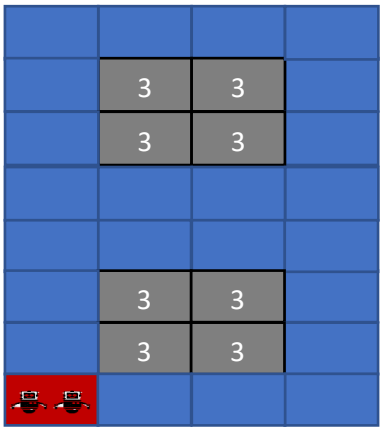
- Assuming an Intermediate state:
- Action: Move Left
  - For agent 1:



  - For agent 2:



E-Transformation Plan



First Step:

If there is at least 1 agent in (1,1)

For each agent:

If number of remaining boxes > 0 then Waste 1 Turn & Go to step 2.

Else Terminate agent.

If all agent terminated, then terminate the program.

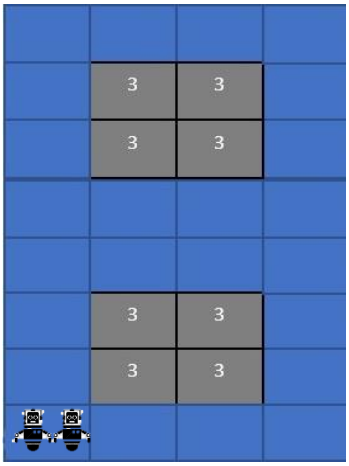
Second Step:

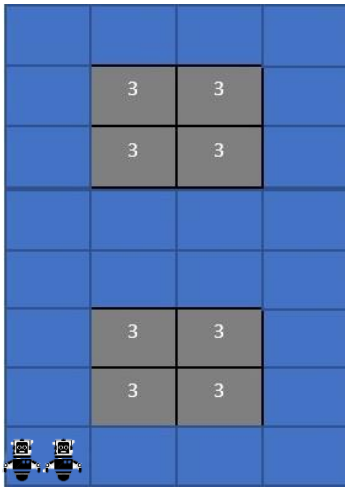
First BFS algorithm find all possible path given start point which is (1,1)

Then algorithm list all possible paths with lowest cost, now we want to get to nearest position where put action is possible.

As example let assume that is output from algorithm:

- (1,1) to (2,1) cost 1
- (1,1) to (3,1) cost 2
- (1,1) to (2,4) cost 4
- Etc.





Third Step:

Now there is 2 cases.

1-Both Agent At (1,1):

- Agent 1 chooses the minimum cost & set Number of movements = path cost
- Increase taken path cost by 1 (to avoid both, perform put action in same time)
- Go to step 4.

2-One Agent At (1,1):

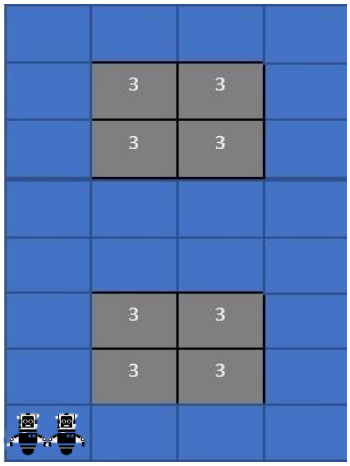
- Agent 1 chooses the minimum cost & set Number of movements = path cost
- Go to step 5.

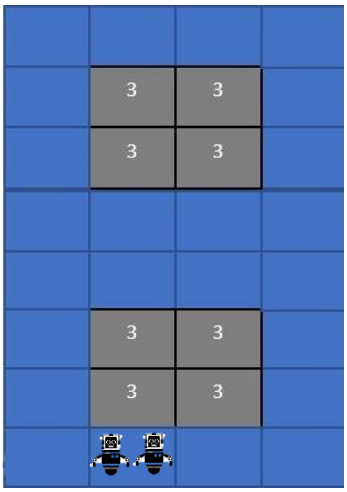
In both cases the accessed shelf gets decreased by 1 (3 (Empty) -> 2 -> 1 -> 0 (Full so Algorithm won't pick it again))

Fourth Step:

Now cost represent how many turns the other agent will need to put the box. If paths have equal cost, we can choose the further one (mentioned in assumption) so it will go to (3,1)

and set number of movements = 2 (Cost)





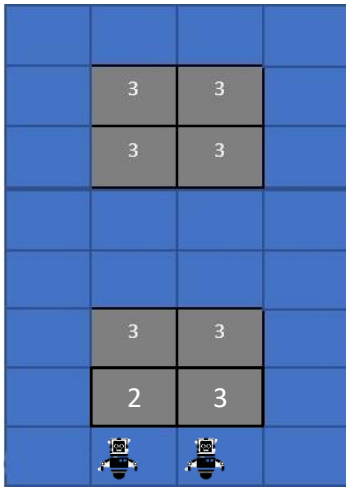
Fifth Step (Actual movement):

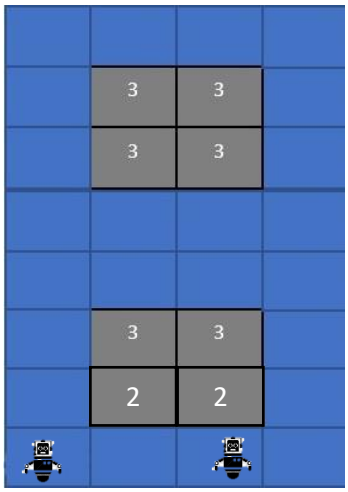
Both agents use the successor function to determine the needed action to follow the given path in case number of movement > 0

Every movement decrease number of movements by 1 and once it reaches 0 then it goes to next step.

Sixth Step:

put action is performed in next turn





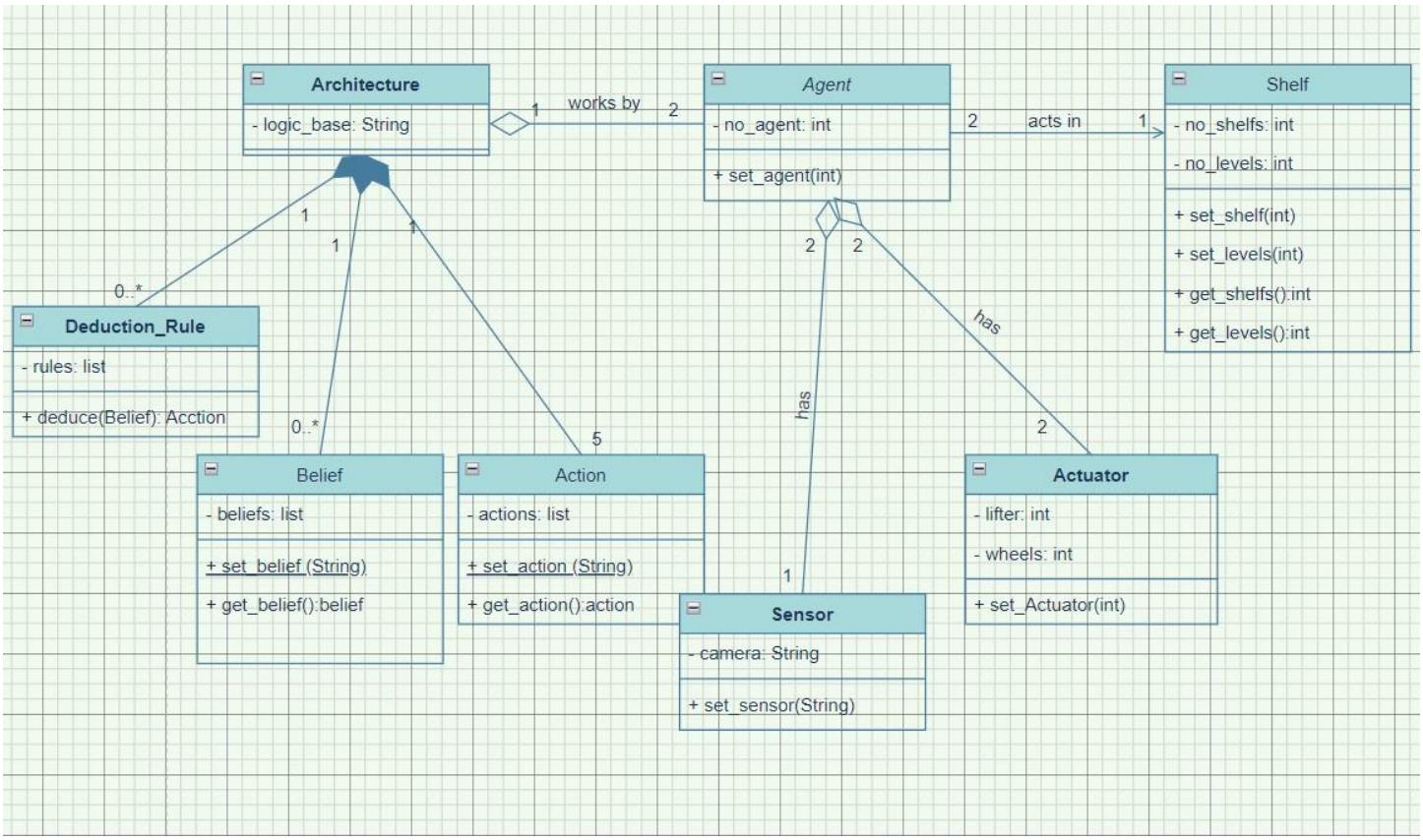
Seventh Step:

agent must backtrack his path to back to (1,1) point in minimum cost.

Once it reaches (1,1) it back to step 1

F- UML Diagram

Class Diagram:



Use case Diagram: