Project Report

1. **Fluent Used**

takeAction(X,Y,C,PC,L,N,s).

The fluent explores all the possible actions that could be taken by the agent including movement in 4 directions, pickup and drop. The parameters: X → x locations of the agent, Y → y location of the agent, C → max capacity of the agent, PC → passengers carried by the agent, L → array of ships left to pick up from in this state, N → no. of ships left to pickup passengers from, s → current situation of the agent.

2. **Successor State Axioms**

- Base case unifies the agent location with the initial location of the agent from the KB, the capacity C that the agent can hold, all the available ships on the grid, and the length of all the available ships as it represents the total number of passengers that need to be saved. The state return from the base case is s0.
- The second case represents the recursive call to the previous state of the agent before action A was taken until base case is reached then it backtracks until a valid path is constructed. We check the previous state of the agent to see what action can be taken to produce the new state. If the agent's state is in the same location of the station and there are carried passengers, the action is 'drop' and we reset the number of carried passengers. If there is one passenger left on a ship and the agent location is at the same location of the ship with this one passenger and the max capacity of the agent is smaller than the number of passengers carried, the action is 'pickup' and this ship is removed from the list of remaining ships and the number of ships left to pickup passengers from is now 0. If there is two passengers left and the agent location is at the same location of one of the ships that contains a passenger and the max capacity of the agent is smaller than the number of passengers carried, the action is 'pickup' and this ship is removed from the list of remaining ships and the number of ships left to pick up passengers from is now 1. As for the remaining actions, the agent can move 'up', 'down', 'left' and 'right' and the new coordinates are set and then this new cell is checked to make sure its valid and within the boundaries of the grid. We created a helper predicate called validCell to check the new position of the agent. For the 4 movement actions, the number of passengers carried and the array of ships and the number of ships left to pickup passengers from remain the same as in the old state and only the coordinates are updated.

3. Test Cases

a. grid(4,4).
   agent_loc(3,3).
   ships_loc([[0,0], [3,1]]).
   station(0,2).
   capacity(2).

   S= result(drop, result(right, result(right, result(pickup, result(left, result(up, result(up, result(up, result(pickup, result(left, result(left, s0))))))))))).

   T = 1.896 seconds cpu time

b. grid(4,4).
   agent_loc(0,0).
   ships_loc([[3,0]]).
   station(0,3).
   capacity(1).

   S= result(drop, result(right, result(right, result(right, result(up, result(up, result(up, result(pickup, result(down, result(down, result(down, s0))))))))))).

   T= 1.198 seconds cpu time

c. grid(3,3).
   agent_loc(0,2).
   ships_loc([[2,0]]).
   station(2,2).
   capacity(1).

   S= result(drop, result(right, result(right, result(pickup, result(left, result(left, result(down, result(down, s0)))))))).

   T= 0.026 seconds cpu time

d. grid(3,3).
   agent_loc(1,1).
   ships_loc([[0,0], [2,2]]).

station(0,2).
capacity(2).

S= result(drop, result(up, result(up, result(pickup, result(right, result(right, result(down, result(down, result(pickup, result(left, result(up, s0))))))))))).

T=1.522 seconds cpu time

e. grid(3,3).
   agent_loc(1,1).
   ships_loc([[0,0], [2,2]]).
   station(0,2).
   capacity(2).

   goal(result(drop, result(right, result(right, result(pickup, result(up, result(left, result(left, result(up, result(pickup, result(right, result(down, s0)))))))))))).
   true.