Team 49 Report

Method:

Our Approach to this search problem is to create a fluent that starts with the goal state and backtracks to the initial state, and create helper predicates for the successor state axiom.

Grid Representation:

The initial state of the search problem is represented in 5 predicates:-

-The grid predicate specifies the size of the grid. It takes two arguments: the number of rows and the number of columns. For example, if the grid is 3x3, grid(3,3) would be called.

-The agent_loc predicate specifies the initial location of the agent on the grid. It takes two arguments: the row and column indices of the agent's location. For example, if the agent is initially located at row 0, column 1, agent_loc(0,1) would be called.

-The ships_loc predicate specifies the locations of the ships on the grid. It takes a list of pairs of row and column indices, representing the locations of the ships. For example, if there are two ships located at row 2, column 2 and row 1, column 2, ships_loc([[2,2],[1,2]]) would be called.

-The station predicate specifies the location of the station on the grid. It takes two arguments: the row and column indices of the station's location. For example, if the station is located at row 1, column 1, station(1,1) would be called.

-The capacity/1 predicate specifies the maximum capacity of the agent. It takes a single argument: the maximum number of passengers that the agent can carry. For example, if the agent has a capacity of 1, capacity(1) would be called.

Fluent Implementation:

Our fluent name is goalHelper.

State Representation

goalHelper(CGX,CGY, Ships, Capacity, S):-

Parameters are:
CGX, CGY: the x and y position of the coast guard
Ships: list of ship locations
Capacity: current capacity of coast guard
S: current state

Base Case:-

As mentioned in our method, the base case in the initial state, which is also the first state in our knowledge base. It is mainly identified by the empty ship list, and start location of the ships, and the capacity. Since in our approach we assign the original ship list to the first goal helper call, but then delete ships everytime pick up is actually called.

```
goalHelper(CGX,CGY, [], Cap, s0):-
        agent_loc(CGX,CGY),
        capacity(Cap).
```

Successor State Axiom:-

The successor state axiom in this case is considered the content of the fluent goalHelper, and the helper predicates help identify the values in every single state.

```
goalHelper(CGX,CGY, CurrShips, Capacity, result(Action, S)):-
Action = pickup, pickup(CGX,CGY,CurrShips,NewShips, Capacity, NewCapacity), goalHelper(CGX,CGY, NewShips, NewCapacity, S);
Action = drop, Capacity>0, drop(CGX,CGY, NewCapacity), goalHelper(CGX,CGY, CurrShips, NewCapacity, S);
Action = up, down(CGX,CGXnew), goalHelper(CGXnew,CGY, CurrShips, Capacity, S);
Action = right, left(CGY, CGYnew), goalHelper(CGX,CGYnew, CurrShips, Capacity, S);
Action = down, up(CGX,CGXnew), goalHelper(CGXnew,CGY, CurrShips, Capacity, S);
Action = left, right(CGY, CGYnew), goalHelper(CGX,CGYnew, CurrShips, Capacity, S).
```

Helper predicates for successor state:-

Helper predicates were implemented for each action to help make sure each situation meets the requirements for our backtracking.

-The **up/2** predicate moves the agent one space up on the grid. It takes two arguments: the current row index of the agent and the new row index after the move. The move is only valid if the new row index is greater than 0 (since the agent cannot move outside of the grid). For example, if the agent is currently at row 0, calling up(0, CGXnew) would bind CGXnew to -1.

-The **down/2** predicate moves the agent one space down on the grid. It takes two arguments: the current row index of the agent and the new row index after the move. The move is only valid if the new row index is less than the number of rows in the grid (since the agent cannot move outside of the grid). For example, if the grid is 3x3 and the agent is currently at row 2, calling down(2, CGXnew) would bind CGXnew to 3.

-The **right/2** predicate moves the agent one space to the right on the grid. It takes two arguments: the current column index of the agent and the new column index after the move. The move is only valid if the new column index is less than the number of columns in the grid (since the agent cannot move outside of the grid). For example, if the grid is 3x3 and the agent is currently at column 2, calling right(2, CGYnew) would bind CGYnew to 3.

-The **left/2** predicate moves the agent one space to the left on the grid. It takes two arguments: the current column index of the agent and the new column index after the move. The move is only valid if the new column index is greater than 0 (since the agent cannot move outside of the grid). For example, if the agent is currently at column 0, calling left(0, CGYnew) would bind CGYnew to -1.

-The **drop/3** predicate that updates the previous state in which a drop actually happened. It takes three arguments: the X, Y of the coast guard to check if it's on the same location of a station or not, and updates the new capacity, which is 0.

-The **pickup/6** predicate updates the previous state in which the agent was actually standing on a ship and had performed a pick up. It takes the location of the coast guard, and ship list,

and each ship that was picked up, was deleted from the list, since the base case is only reached if the state of the world doesn't include ships.

Test Cases:-

When we include KB1, the example queries all run perfectly fine except goal(S). Which displays a correct plan along with an incorrect plan (only missing a drop).

```
5 ?- goal(S).
S = result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
S = result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
S = result(up, result(left, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
S = result(up, result(left, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(right, result(down, result(down, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(down, result(right, result(down, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(down, result(down, result(right, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(left, result(up, result(pickup, result(right, result(down, result(down, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(left, result(up, result(pickup, result(down, result(right, result(down, s0))))))))) ;
S = result(left, result(pickup, result(right, result(drop, result(left, result(up, result(pickup, result(down, result(down, result(right, s0))))))))) ;
S = result(left, result(up, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
S = result(left, result(up, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
S = result(left, result(up, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
S = result(left, result(up, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
S = result(drop, result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))))) ;
S = result(drop, result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))))) 
```

However when we include KB2, all test cases run perfectly fine, and generate valid plans.

Time it takes to run:

KB1:

0.377

```
8 ?- time(goal(S)).
% 5,719,787 inferences, 0.375 CPU in 0.377 seconds (100% CPU, 15252765 Lips)
S = result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
% 93 inferences, 0.000 CPU in 0.000 seconds (0% CPU, Infinite Lips)
S = result(up, result(left, result(pickup, result(right, result(down, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
% 36,020 inferences, 0.016 CPU in 0.014 seconds (108% CPU, 2305280 Lips)
S = result(up, result(left, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(right, result(down, s0))))))))) ;
% 92 inferences, 0.000 CPU in 0.000 seconds (0% CPU, Infinite Lips)
S = result(up, result(left, result(pickup, result(down, result(right, result(drop, result(left, result(pickup, result(down, result(right, s0))))))))) ;
% 1,943,096 inferences, 0.156 CPU in 0.162 seconds (97% CPU, 12435814 Lips)
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(right, result(down, result(down, s0))))))))) ;
% 285 inferences, 0.000 CPU in 0.000 seconds (0% CPU, Infinite Lips)
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(down, result(right, result(down, s0))))))))) ;
% 92 inferences, 0.000 CPU in 0.000 seconds (0% CPU, Infinite Lips)
S = result(left, result(pickup, result(right, result(drop, result(up, result(left, result(pickup, result(down, result(down, result(right, s0))))))))) ;
% 13,300 inferences, 0.016 CPU in 0.007 seconds (225% CPU, 851200 Lips)
S = result(left, result(pickup, result(right, result(drop, result(left, result(up, result(pickup, result(right, result(down, result(down, s0))))))))) ;
```

KB2:

0.175 second

```
6 ?- time(goal(S)).
% 2,746,372 inferences, 0.172 CPU in 0.175 seconds (98% CPU, 15978892 Lips)
S = result(drop, result(up, result(up, result(left, result(pickup, result(down, result(down, result(pickup, result(down, s0))))))))) 
```