Introduction to Artificial Intelligence

Project 2: ∃a, s[MissionCompleted(Ethan, Result(a, s))]

Report.

1- <u>Actions and Predicate Terms:</u>

a. Actions:

Actions in the program are represented as facts *(right,left,up,down,carry,drop)*. They are assigned accordingly in the state-handling Logic inside the Successor State Axiom.

b. Predicate Terms:

The following predicates are used in the program:

- **in_grid(X,Y):** Checks if the X and Y coordinates in the boundaries of the grid.
- **carried(X,Y,AgentsList,AgentsListUpdated):** assigns the carry status of the agent at position (X,Y) in AgentsList with carry status of 0, the value of 1, and returns the list AgentsListUpdated with the updated carry status of this agent.
- **dropped(X,Y,AgentsList,AgentsListUpdated):** assigns the carry status of the the agent at position (X,Y) with carry status of 1, the value of 2, and returns the list AgentsListUpdated with the updated carry status of this agent.
- **create_agents_array_with_carry(AgentsListKB,AgentsListWithCarry):** initializes a copy AgentsListWithCarry of the agents list in the Knowledge Base AgentsListKB, but with an additional parameter with each agent, C = 0, where C is the carry status of the agent (C=0 “Not Carried”, C=1 ”Carried”, C=2 “Dropped”).
- **create_agents_array_with_drop(AgentsListKB,AgentsListWithDrop):** initializes a copy AgentsListWithDrop of the agents list in the Knowledge Base AgentsListKB, but with an additional parameter with each agent, C = 2, where C is the carry status of the agent (C=0 “Not Carried”, C=1 ”Carried”, C=2 “Dropped”).
- **reverse2(L,R):** reverses the compound output L of the algorithm into R, to match the required format.

2- <u>Successor State Axiom:</u>

a. Variables:

The program has only <u>**One**</u> successor state axiom, called <u>*state*</u>, and it keeps track of the following:

- Ethan’s position: the X and Y coordinates of Ethan in the search problem Grid.
- Ethan’s current carry count: The number of agents Ethan is currently carrying.
- AgentsLeft: the number of agents left uncarried on the Grid.
- Agents: the List of the agents.

b. Logic:

The successor state axiom holds true for ethan to be in (X,Y), with a carry count C, if and only if:

- He can go right to (X,Y+1), if the next situation position is not out of the grid bounds.
- He can go left to (X,Y-1), if the next situation position is not out of the grid bounds.
- He can go up to (X-1,Y), if the next situation position is not out of the grid bounds.
- He can go down to (X+1,Y), if the next situation position is not out of the grid bounds.
- He can stay in (X,Y), and if there's an agent with a carry status 0 "Not Carried" in the same position, and he has room for more agents to carry "ethanCarry < Capacity", he can Carry.
- He can stay in (X,Y), and if (X,Y) is the position of the submarine, and has agents carried "ethanCarry > 0 and their carry status is 1"Carried" ", He can drop.

3- The Goal Predicate:

The program runs using the goal predicate *goal(S)*, that takes a situation S, and returns true if it's a valid goal situation in the search problem.

goal(S) calls the initial state from the KB. The successor state axiom "state" is then started to query, and has a base case that holds true when ethan is at the submarine, with all agents of carry status: 2 "Dropped".

4- Running Examples:

- **Working Example 1:**

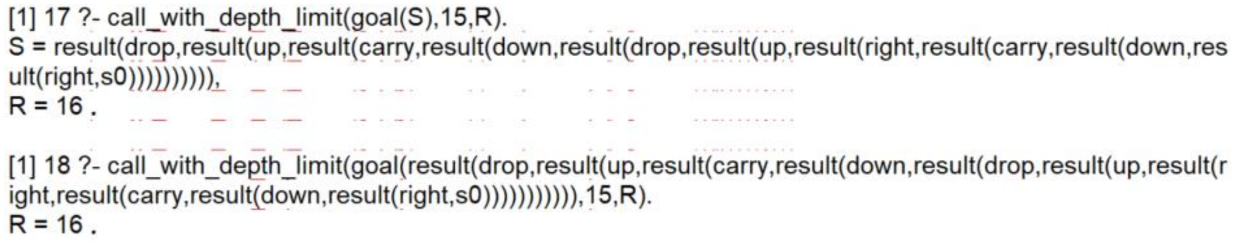
KB:

Ethan: [0,0]

Agents: [[1,1], [1,2]]

Submarine: [0,2]

Capacity: 1



```
[1] 17 ?- call_with_depth_limit(goal(S),15,R).
S = result(drop,result(up,result(carry,result(down,result(drop,result(up,result(right,result(carry,result(down,res
ult(right,s0)))))))))),
R = 16 .

[1] 18 ?- call_with_depth_limit(goal(result(drop,result(up,result(carry,result(down,result(drop,result(up,result(r
ight,result(carry,result(down,result(right,s0))))))))))),15,R).
R = 16 .
```

- **Working Example 2:**

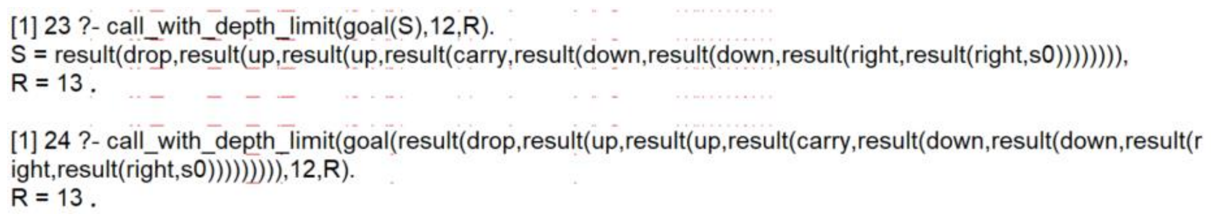
KB:

Ethan: [0,0]

Agents: [[2,2]]

Submarine: [0,2]

Capacity: 1



```
[1] 23 ?- call_with_depth_limit(goal(S),12,R).
S = result(drop,result(up,result(up,result(carry,result(down,result(down,result(right,result(right,s0)))))))),
R = 13 .

[1] 24 ?- call_with_depth_limit(goal(result(drop,result(up,result(up,result(carry,result(down,result(down,result(r
ight,result(right,s0))))))))),12,R).
R = 13 .
```