AI Project 2 Report

1 Syntax and Semantics

1.1 helper_goal

All the actions are implemented under the predicate of ***'helper_goal'***
The predicate symbols are

1. **Ex**: denotes the X position of Ethan.
2. **Ey**: denotes the Y position of Ethan.
3. **L**: denotes the List of positions of soldiers.
4. **C**: denotes the remaining Capacity of Ethan.
5. **S**: denotes the current State of taking actions.
6. **G**: denotes the Goal state.

The predicate is implemented 7 times. The semantics is explained in Sucessor State Axiom section,

1.2 has_var

This predicate is to check if the input S is a variable or contains a variable. It takes a variable S and keeps calling itself and checking with var predicate.

1.3 call_with_incremental_goal

This predicate is to call the helper_goal starting from depth 1 until it hits the right depth.
If the current depth does not give an output it increases it by 2.
It takes the same inputs as helper_goal plus variable D which denotes the depth.

1.4 is_there_soldier

This predicate is to check if there is a soldier in the current cell. If there is one, it removes it.

2 Successor state axiom

We only used one SSA which is represented by the helper_goal predicate.
As the SSA is implemented 7 times each one is to satisfy certain constraints to execute the actions.

1. **Goal State**: is when Ethan in Submarine, no soldiers left, ethan dropped all the soldiers in the submarine "C==Cp".
2. **move up**: decreasing Ex by one if Ex not equal to 0.
3. **move down**: increasing Ex by one if Ex not equal to 3.
4. **move left**: decreasing Ey by one if Ey not equal to 0.
5. **move right**: increasing Ey by one if Ey not equal to 3.
6. **carry**: is when C>0 and there is soldier in the current cell.
7. **drop**: is when Ethan in submarine and Ethan is carrying soldiers.

So, it checks these constraints, if it satisfies one of them so it implements it.

3 goal(S)

In the predicate we extract the Knowledge from KB.pl and then calling ***'helper_goal'***
with the following parameters.

1. Ex
2. Ey
3. L
4. C
5. s0 -> which denotes S in helper_goal
6. S -> which denotes G in helper_goal

This predicate is implemented twice.

3.1 If S has a var

Which means that the user wants a plan. So it call the predicate of call_with_incremental_goal starting with depth 1 until it finds a solution.

3.2 If S does not have a var

So in this case we call the helper_goal directly with the predicate of call_with_depth_limit with fixed depth which is here 15[1].
We divided it into 2 because It would run infinitely to find a solution unless it bounds to a specific depth.

4 Running Examples

4.1 First one

- **KB**
  - ethan_loc(0,3).
  - members_loc([[1,0],[2,1]]).
  - submarine(3,2).
  - capacity(2).
- Result: S = result(drop, result(right, result(down, result(carry, result(right, result(down, result(carry, result(left, result(left, result(left, result(down, s0))))))))))

4.2 Second one

- **KB**
  - ethan_loc(1,1).
  - members_loc([[2,2]]).
  - submarine(3,3).
  - capacity(1).
- Result: S = result(drop, result(right, result(down, result(carry, result(right, result(down, s0))))))

[1]Actually I have set it to 100 and it runs very quickly, but after I shut down the computer and ran it again the terminal takes much more time, I do not know why, so I set it to 15 to test it