# COEN 166 Artificial Intelligence

# Lab Assignment #2: Vacuum Cleaner Agent - Report

Name: Tanner Kaczmarek  ID: 00001443463

Explanation of defined functions:

In Vacuum.py… I make a class called Vacuum that can do 3 actions. It can move left, right, or suck. It finishes when both left and right are clean.

- The **function action** takes itself, a state and a location. It first checks to see if the location is dirty or clean. If it is dirty it will Suck and make the current state Clean as well. The next if else statement moves to the right as long as the current location is the left location. It will make its current location 1 or the Right. The Else statement, it the current location is Clean and is the right location, it will then move to the left and make its current location 0 or the Left. All three of these if/else if/ else statements will add one item to the cost and returns back to function tester the action, its current state, and location.
- The **function runAtion** takes itself, state, location and actions and only stops the while loop and returns the sequence of actions and the cost when both states are Clean. The function also runs the function action and appends the results of that action onto a sequence named actions.

AGENT.PY

```
class Vacuum:

    def runAction(self, state, location):
        cost = 0
        actions = []
        while True:  #While loop until returns the sequence of actions
as well as the cost
            if state == ["Clean", "Clean"]:
                print("The cost of Cleaning was ", cost, "\n")
                print("The actions were ", actions, "\n")
                print("--------------------------------\n")
                return actions, cost
            action, state, location = self.action(state, location)
            cost += 1  #action happened last time so time to add the
cost
            actions.append(action)

    def action(self, state, location):
        if state[location] == "Dirty":  #dirty so suck and then set
location to clean
            action = "Suck"
            state[location] = "Clean"
            return action, state, location
        elif location == 0:  #left side is clean so move to the left
            action = "Right"
            location = 1
            return action, state, location
        elif location == 1:   #right side is clean so move to the
right
            action = "Left"
            location = 0
            return action, state, location
```

Explanation of the Test Case:

To test I imported the handy unittest and use that to make a class and test all possible different 8 states the vacuum cleaner can be in. I chose my test cases because I thought to be safe, why not test all test cases possible for the correct solution.

- Test_1 tests both spots dirty and starts vacuum at the left.
- Test_2 tests both spots dirty and starts the vacuum at the right.
- Test_3 tests left dirty and right clean and starts the vacuum at the left.
- Test_4 tests left dirty and right clean and starts the vacuum at the right.
- Test_5 tests left clean and right dirty and starts the vacuum at the left.
- Test_6 tests left clean and right dirty and starts the vacuum at the right.
- Test_7 tests both clean and starts the vacuum at the left.
- Test_8 tests both clean and starts the vacuum at the right.

TESTAGENT.PY

```
#Runs through all the test cases
class Unit_Test(unittest.TestCase):
    def test_1(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Dirty", "Dirty"], 0),
(["Suck", "Right", "Suck"], 3))
    def test_2(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Dirty", "Dirty"], 1),
(["Suck", "Left", "Suck"], 3))
    def test_3(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Dirty", "Clean"], 0),
(["Suck"], 1))
    def test_4(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Dirty", "Clean"], 1),
(["Left", "Suck"], 2))
    def test_5(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Clean", "Dirty"], 0),
(["Right", "Suck"], 2))
    def test_6(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Clean", "Dirty"], 1),
(["Suck"], 1))
    def test_7(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Clean", "Clean"], 0),
([], 0))
    def test_8(self):
        vacuuming1 = Vacuum()
        self.assertEqual(vacuuming1.runAction(["Clean", "Clean"], 1),
([], 0))

if __name__ == '__main__':
    unittest.main()
```