COEN 166 Artificial Intelligence

Lab Assignment #2: Vacuum Cleaner Agent

Name: Marianne Fuyu Yamazaki Dorr ID: 1606975

Explanation of the defined functions:

is_goal(state): Using the current state of the vacuum cleaner agent, the function will check if both squares are “Clean”. If so, the function will return True. If not, it will return False.

is_dirty(state): Using the current state of the vacuum cleaner agent, the function will check if the square the vacuum cleaner is currently at is “Dirty”. If it is, it will return True. If not, it will return False.

cleanup(state): Using the current state of the vacuum cleaner agent, the function will change the status of the location the vacuum cleaner is at to “Clean” and return the action “Suck”.

move(state): Using the current state of the vacuum cleaner agent, the function will move to vacuum cleaner to the left if it is in the right square and return the action “Left”. Otherwise, it will move the vacuum cleaner to the right if it was originally in the left square and return the action “Right”.

Explanation of the test case:

test_case(state, action_seq): This function will receive the initial state of the vacuum cleaner agent and an empty list of action sequence. It will initialize a variable totalCost to keep track of the cost of actions of the vacuum cleaner agent to 0. It will also make a copy of the initial state into the variable original_state. The function will then go into a while loop that will end when the value of is_goal returns True, meaning that the state has reached the goal state of both squares being “Clean”. In the loop, the function will check if the current location of the vacuum cleaner is “Dirty” using the is_dirty() function. If it is, it will invoke the function cleanup() and add the action returned to the action sequence list. Otherwise, the function will invoke the move() function and append the action taken to the action sequence list. Each action increments the totalCost by 1.

Once the while loop is done running, the function will print out the sequence of actions taken by the vacuum cleaner agent and the total cost of that sequence. It will then check if the results are correct by checking every possible input and its corresponding output to the output given by the function. If the output was correct, a “Correct Outputs!” message will be printed. Otherwise, a “Wrong Output!” message will be printed.

Appendix:

Code:

```
"""
Created on Tue Sep 27 13:43:21 2022
Lab 2 - Vacuum Cleaner Agent
COEN 166L - Wednesday 2:15PM
```

@author: Marianne Fuyu Yamazaki Dorr

Requirements:

• Define the “state” as a list of three elements: the 1st element specifies the status of the left square (“Clean” or “Dirty”), the 2nd element specifies the status of the right square (“Clean” or “Dirty”), and the 3rd element specifies the current location of the agent (0 for left square, 1 for right square).

• The candidate actions are “Suck”, “Left” (moving to the left square), “Right” (moving to the right square). Each action costs 1 point.

• The agent will take an action based on the current state. The agent program can update the state immediately after an action is taken. The program terminates when both squares are clean.

• Create a test case to verify your code. The test case takes two inputs: the current state, and an empty list that will store the actions; then it will generate two outputs: the sequence of actions taken by the agent to achieve the goal state, and the corresponding total cost.

• Your test case should be able to take any possible inputs, generate the corresponding outputs, and verify whether the outputs are correct or not. For example, if the outputs are correct, then your test code can print a message: “Correct Outputs!” Otherwise, it can print: “Wrong Outputs!”

• Explain in detail how your code works (for example, what each function does, and how the test case works).

"""

```
# Lab 2 Vaccuum Bot
# Actions = {"Suck", "Left", "Right"}
# state is list ("Clean", "Dirty", 0 or 1) 0 = left and 1 = right

#Checks if all squares are clean, if yes returns True, if not returns False
def is_goal(state):
    if (state[0] == "Clean") & (state[1] == "Clean"):
        return True
    else:
        return False

#Checks if square at the vaccuum bot's location is dirty. If yes, returns True, False otherwise
def is_dirty(state):
    if(state[2] == 0):
        if(state[0] == "Dirty"):
            return True
        else:
            return False
    else:
        if(state[1] == "Dirty"):
            return True
```

```
        else:
            return False

#Changes the state of the square where the vaccuum is at to "Clean" and returns the action "Suck"
def cleanup(state):
    if (state[2] == 0):
        state[0] = "Clean"
    else:
        state[1] = "Clean"
    return "Suck"

#Moves the location of the vaccuum bot and returns the action of the movement made either "Right" or
"Left"
def move(state):
    if(state[2] == 0):
        state[2] = 1
        return "Right"
    else:
        state[2] = 0
        return "Left"

#Tests the vaccuum bot code and checks if it returns the correct output
def test_case(state, action_seq):

    totalCost = 0
    original_state = state.copy()

    #Generate sequence of actions taken by vaccuum bot
    while(is_goal(state) == False):
        if (is_dirty(state) == True):
            action_seq.append(cleanup(state))
            totalCost += 1
        else:
            action_seq.append(move(state))
            totalCost += 1

    #Print sequence of actions and total cost of vaccuum bot
    print("The sequence of actions taken by the vacuum bot is: " + str(action_seq))
    print("The total cost of the sequence is: " + str(totalCost))
    print()

    #Check if outputs are correct for all possible inputs
    correct = None
    if (original_state == ['Dirty', 'Dirty', 0]):
        if (action_seq == ["Suck", "Right", "Suck"]):
            correct = True
        else:
            correct = False
    if (original_state == ['Dirty', 'Dirty', 1]):
        if (action_seq == ["Suck", "Left", "Suck"]):
            correct = True
```

```
        else:
            correct = False

    if (original_state == ['Clean', 'Dirty', 0]):
        if (action_seq == ["Right", "Suck"]):
            correct = True
        else:
            correct = False
    if (original_state == ['Clean', 'Dirty', 1]):
        if (action_seq == ["Suck"]):
            correct = True
        else:
            correct = False
    if (original_state == ['Dirty', 'Clean', 0]):
        if (action_seq == ["Suck"]):
            correct = True
        else:
            correct = False
    if (original_state == ['Dirty', 'Clean', 1]):
        if (action_seq == ["Left", "Suck"]):
            correct = True
        else:
            correct = False
    if (is_goal(original_state) == True):
        if (action_seq == []):
            correct = True
        else:
            correct = False

    #Checks if both the action_seq and totalCost match up
    success = None
    if (correct == True) & (totalCost == len(action_seq)):
        print("Correct Outputs!")
        success = True
    else:
        print("Wrong Outputs!")
        success = False
    print()
    #returns the list of actions taken, the cost of those actions, and whether the test was successful or not
    return action_seq, totalCost, success

#Tests:
test1 = ["Dirty", "Dirty", 0]
test2 = ["Dirty", "Dirty", 1]
test3 = ["Clean", "Dirty", 0]
test4 = ["Clean", "Dirty", 1]
test5 = ["Dirty", "Clean", 0]
test6 = ["Dirty", "Clean", 1]
test7 = ["Clean", "Clean", 0]
test8 = ["Clean", "Clean", 1]
list_tests = [test1, test2, test3, test4, test5, test6, test7, test8]
```

```
score = 0
for test in list_tests:
    print("---------------------------------")
    print("Original state is: " + str(test))
    print()
    test_action_seq = []
    result = test_case(test, test_action_seq)
    if (result[2] == True):
        score += 1
    print("End state is: " + str(test))
    print("---------------------------------")

print()
print("###################")
print()
print("Score: " + str(score) + "/" + str(len(list_tests)))
print()
print("####################")
```

Output:

```
---------------------------------
Original state is: ['Dirty', 'Dirty', 0]

The sequence of actions taken by the vacuum bot is: ['Suck', 'Right', 'Suck']
The total cost of the sequence is: 3

Correct Outputs!

End state is: ['Clean', 'Clean', 1]
---------------------------------
---------------------------------
Original state is: ['Dirty', 'Dirty', 1]

The sequence of actions taken by the vacuum bot is: ['Suck', 'Left', 'Suck']
The total cost of the sequence is: 3

Correct Outputs!

End state is: ['Clean', 'Clean', 0]
---------------------------------
---------------------------------
Original state is: ['Clean', 'Dirty', 0]

The sequence of actions taken by the vacuum bot is: ['Right', 'Suck']
The total cost of the sequence is: 2

Correct Outputs!

End state is: ['Clean', 'Clean', 1]
---------------------------------
```

```
---------------------------------
Original state is: ['Clean', 'Dirty', 1]

The sequence of actions taken by the vacuum bot is: ['Suck']
The total cost of the sequence is: 1

Correct Outputs!

End state is: ['Clean', 'Clean', 1]
---------------------------------
---------------------------------
Original state is: ['Dirty', 'Clean', 0]

The sequence of actions taken by the vacuum bot is: ['Suck']
The total cost of the sequence is: 1

Correct Outputs!

End state is: ['Clean', 'Clean', 0]
---------------------------------
---------------------------------
Original state is: ['Dirty', 'Clean', 1]

The sequence of actions taken by the vacuum bot is: ['Left', 'Suck']
The total cost of the sequence is: 2

Correct Outputs!

End state is: ['Clean', 'Clean', 0]
---------------------------------
---------------------------------
Original state is: ['Clean', 'Clean', 0]

The sequence of actions taken by the vacuum bot is: []
The total cost of the sequence is: 0

Correct Outputs!

End state is: ['Clean', 'Clean', 0]
---------------------------------
---------------------------------
Original state is: ['Clean', 'Clean', 1]

The sequence of actions taken by the vacuum bot is: []
The total cost of the sequence is: 0

Correct Outputs!

End state is: ['Clean', 'Clean', 1]
---------------------------------
```

##################

Score: 8/8

###################