Caden Kroonenberg
Problem Set 2
EECS 649
1-28-22

2.1

Agent Type	Performance Measure	Environment	Actuators	Sensors
Titan's subsurface ocean explorer	Careful, observant, maximize area explored, minimize fuel consumption, avoid obstacles, maintain communication with NASA	Subsurface oceans of liquid water with high concentrations of salt, potentially containing organic life, cold temperatures (-290 °F)	Steering, propellors, buoyancy control, mechanical arms, lights, sample collector, transmitter	Video cameras, thermometer, GPS, depth, pH, chemical sensor, speedometer, motion detection, photosensor, fuel level sensor
Online shopper for used AI books	Minimize cost, minimize shipping time, avoid duplicate book purchases, avoid books covering the same topic, maximize review score, do not spend more than budget covers, maximize book condition, only books in English	Reliable online vendors (no suspicious websites), online textbook vendors (Amazon, college bookstores), used item vendors (eBay, Facebook Marketplace)	Internet browsing control (query search engines), searching control (search vendors for specific books), purchase control	Cost, shipping time, review score, book condition, book title, book author, book topic, knowledge of previous purchases

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Subsurface oceans of Titan	Partially	Single	Nondeterministic	Sequential	Dynamic	Continuous
Online AI book shopper	Partially	Multiple	Deterministic	Sequential	Dynamic	Discrete

2.2

$|A| = 1, |P| = 2, |(S/R\ agents)| = 1^2 = 1$

$|A| = 2, |P| = 1, |(S/R\ agents)| = 2^1 = 2$

$|A| = 2, |P| = 3, |(S/R\ agents)| = 2^3 = 8$

$|A| = 3, |P| = 2, |(S/R\ agents)| = 3^2 = 9$

$|A| = 1, |P| = n, |(S/R\ agents)| = 1^n = 1$

$|A| = m, |P| = n, |(S/R\ agents)| = m^n$

In general, there are $|A|^{|P|}$ S/R agents.

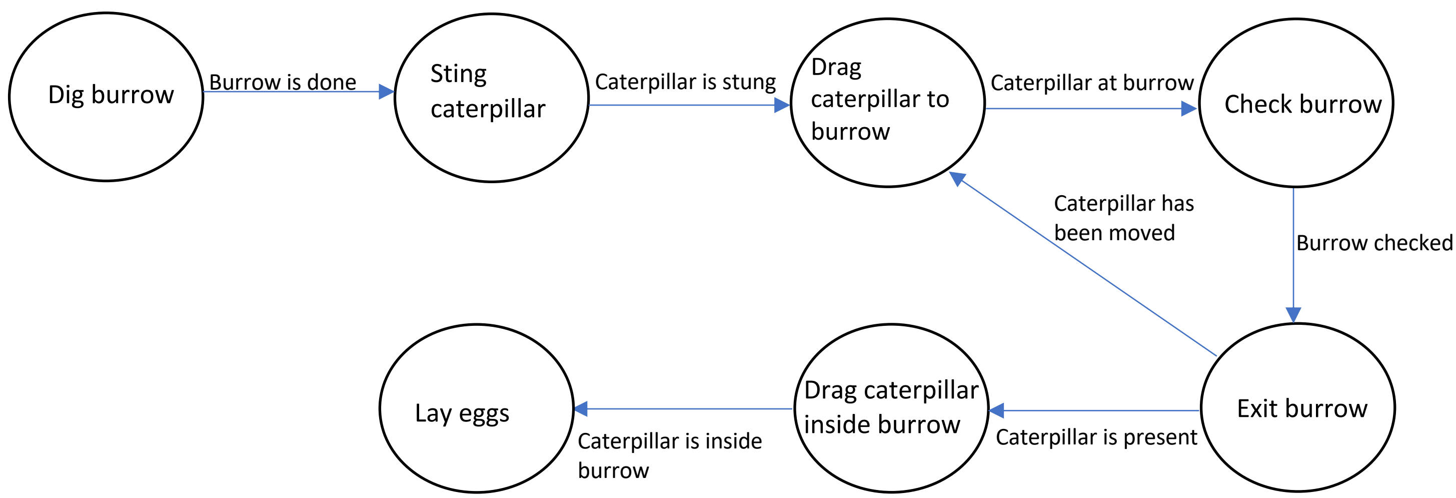
<u>Vacuum agent</u>

|A| = 3, A = {Left, Right, Suck}

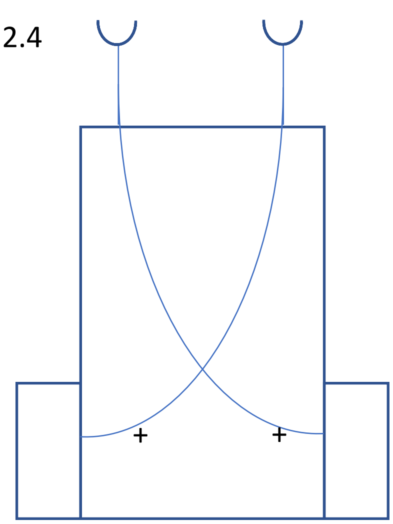
|P| = 4, P = {(A, Dirty), (A, Clean), (B, Dirty), (B, Clean)}

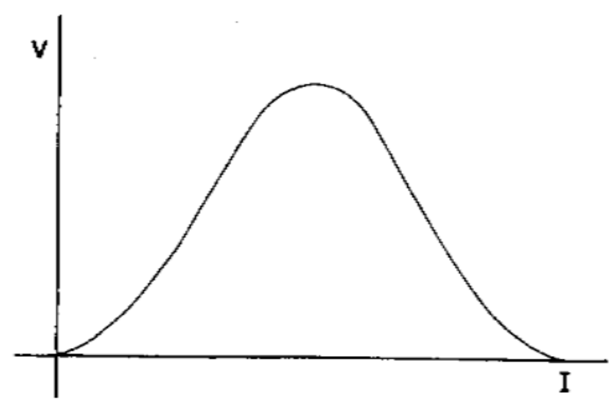
$|(S/R\ agents)| = |A|^{|P|} = 3^4 = 81$

2.3



2.4



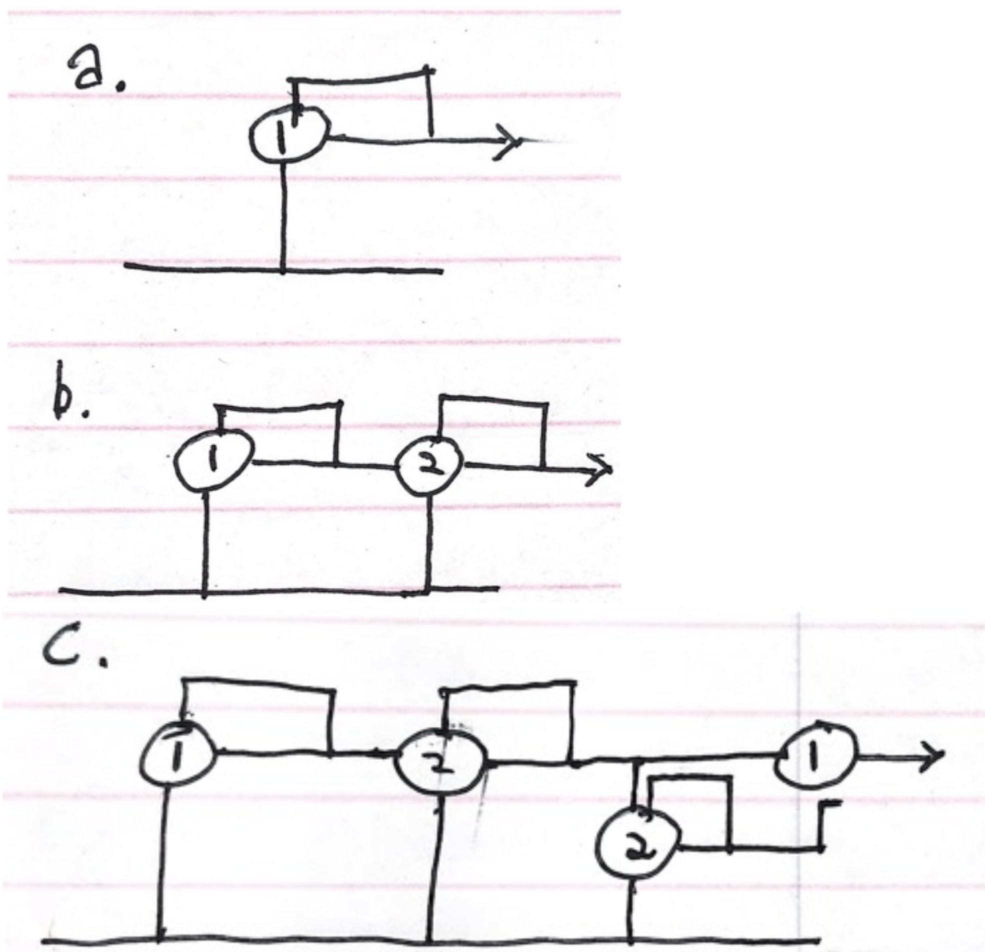


2.5

t	s	v	w	x	y	z
0	0	0	0	0	0	0
1	1	0	0	0	0	0
2	1	1	1	1	1	0
3	1	1	0	1	1	0
4	1	1	0	1	1	0
5	0	1	0	1	1	0
6	1	0	0	0	1	1

$v_{t+1} = [s_t \geq 1]$
$w_{t+1} = [s_t - v_t \geq 1]$
$x_{t+1} = [s_t + w_t \geq 2]$
$y_{t+1} = [s_t + x_t \geq 2]$
$z_{t+1} = [y_t - s_t \geq 1]$

2.6



2.7

S/R table for agent actions:

Percept	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck

My program consists of two classes which interact with each other: the environment and the agent. The environment class stores the state of the two locations (‘Dirty’ or ‘Clean’). It also has functions to:

- Return the status of the agent’s current location
- Execute the agent’s actions (update the agent’s location or update a location’s status)
- Return the agent’s starting position (either A or B).

The agent class stores the agent’s current location (A or B) and the performance score. It also has a function to return an action based on the percept at that point in time (given by the environment).

For each possible combination of location statuses and agent starting positions, 1000 time steps are completed. In each time step, the agent chooses an action based on the percept given by the environment. The environment class executes the action and updates either the status of the agent’s location (Clean or Dirty at A or B) or the agent’s location (A or B). For each clean location at the end of a time step, the agent’s performance score is increased by 1.

Performance score based on initial configuration of the environment:

Initial ‘A’ State	Initial ‘B’ State	Initial Agent Location	Score
Dirty	Dirty	A	1998
Dirty	Dirty	B	1998
Clean	Clean	A	2000
Clean	Clean	B	2000
Dirty	Clean	A	2000
Dirty	Clean	B	1999
Clean	Dirty	A	1999
Clean	Dirty	B	1999

The average score is 1999.25.

I used code from R&N's code repository: https://github.com/aimacode/aima-python/blob/master/vacuum_world.ipynb in my program. I used the TrivialVacuumEnvironment class as the basis for my environment class – I removed code in the the execute_action function to prevent Left and Right movements from decreasing the performance score of the agent. I also modified the environment class to use pre-defined location statuses and initial agent location rather than random choices for these variables.

2.8

a. A simple reflex agent cannot be perfectly rational because once both states A and B are clean, the agent will move Left and Right infinitely because it doesn't know that there is no more cleaning to be done. These actions are irrational since they can only lead to decreased performance score.

b. A reflex agent with state is almost perfectly rational in this environment. We can call the state 'Moved' and initially set to False. We update the value to True once a 'Left' or 'Right' action is taken by the agent. We would also need a new action, NoOp (No Operation) to execute if there is no rational action to take (like moving to the other state if the agent has already been there). This allows for a new S/R table for agent actions to be used to determine actions (see below). This prevents the agent from irrationally moving to the opposite state if the agent has already been there. Because the agent doesn't know the status of the location opposite to its starting location, it will always need to check it. It is possible for the agent to make an irrational action if the opposite location is initially clean.

S/R table for agent actions:

Percept	Action
[A, Clean] AND [Moved, False]	Right
[A, Dirty] AND [Moved, False]	Suck
[B, Clean] AND [Moved, False]	Left
[B, Dirty] AND [Moved, False]	Suck
[A, Clean] AND [Moved, True]	NoOp
[A, Dirty] AND [Moved, True]	Suck
[B, Clean] AND [Moved, True]	NoOp
[B, Dirty] AND [Moved, True]	Suck

c.

An agent with percepts that give it the status of both states in the environment can be perfectly rational because it will not irrationally move to the opposite state if the current state is clean like a simple agent reflex would unless it needs to. It is perfectly rational because every action it takes accomplishes something and is done to maximize its performance score.

Percept	Location	Action
[A, Clean], [B, Clean]	A	NoOp
[A, Dirty], [B, Clean]	A	Suck
[A, Clean], [B, Dirty]	A	Right
[A, Dirty], [B, Dirty]	A	Suck
[A, Clean], [B, Clean]	B	NoOp
[A, Dirty], [B, Clean]	B	Left
[A, Clean], [B, Dirty]	B	Suck
[A, Dirty] [B, Dirty	B	Suck