Name : ABID ALI

Student_no : 2019380141

Assignment 2 and 3

Chapter-2-(PG-61)

Intelligent Agents

2.3 For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.
a. An agent that senses only partial information about the state cannot be perfectly rational.

b. There exist task environments in which no pure reflex agent can behave rationally.
c. There exists a task environment in which every agent is rational.
d. The input to an agent program is the same as the input to the agent function.
e. Every agent function is implementable by some program/machine combination.
f. Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.
g. It is possible for a given agent to be perfectly rational in two distinct task environments.
h. Every agent is rational in an unobservable environment.
i. A perfectly rational poker-playing agent never loses.

Solution:

a. **An agent that senses only partial information about the state cannot be perfectly rational.**
False. Perfect rationality refers to the ability to make good decisions given the sensor information received.

b. **There exist task environments in which no pure reflex agent can behave rationally.**
True. A pure reflex agent ignores previous percepts, so cannot obtain an optimal state estimate in a partially observable environment. For example, correspondence chess is played by sending moves; if the other player's move is the current percept, a reflex agent could not keep track of the board state and would have to respond to, say, "a4" in the same way regardless of the position in which it was played.

c. **There exists a task environment in which every agent is rational.**
True. For example, in an environment with a single state, such that all actions have the same reward, it doesn't matter which action is taken. More generally, any environment that is reward-invariant under permutation of the actions will satisfy this property.

d. **The input to an agent program is the same as the input to the agent function.**
False. The agent function, notionally speaking, takes as input the entire percept sequence up to that point, whereas the agent program takes the current percept only.

e. **Every agent function is implementable by some program/machine combination.**
False. For example, the environment may contain Turing machines and input tapes and the agent's job is to solve the halting problem; there is an agent function that specifies the right answers, but no agent program can implement it. Another example would be an agent function that requires solving intractable problem instances of arbitrary size in constant time.

f. **Suppose an agent selects its action uniformly at random from the set of possible actions.**
There exists a deterministic task environment in which this agent is rational.
True. This is a special case of (c); if it doesn't matter which action you take, selecting randomly is rational.

g. **It is possible for a given agent to be perfectly rational in two distinct task environments.**
True. For example, we can arbitrarily modify the parts of the environment that are unreachable by any optimal policy as long as they stay unreachable.
h. **Every agent is rational in an unobservable environment.**
False. Some actions are stupid—and the agent may know this if it has a model of the environment—even if one cannot perceive the environment state.

i. **A perfectly rational poker-playing agent never loses.**
False. Unless it draws the perfect hand, the agent can always lose if an opponent has better cards. This can happen for game after game. The correct statement is that the agent's expected winnings are nonnegative.

2.5 Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.

Solution:

The following are just some of the many possible definitions that can be written:

• **Agent:** an entity that perceives and acts; or, one that can be viewed as perceiving and acting. Essentially any object qualifies; the key point is the way the object implements an agent function. (Note: some authors restrict the term to programs that operate on behalf of a human, or to programs that can cause some or all of their code to run on other machines on a network, MOBILE AGENT as in **mobile agents**.)

• **Agent function**: a function that specifies the agent's action in response to every possible percept sequence.

• **Agent program:** that program which, combined with a machine architecture, implements an agent function. In our simple designs, the program takes a new percept on each invocation and returns an action.

• **Rationality**: a property of agents that choose actions that maximize their expected utility, given the percepts to date.

• **Autonomy:** a property of agents whose behavior is determined by their own experience rather than solely by their initial programming.

• **Reflex agent:** an agent whose action depends only on the current percept.

• **Model-based agent**: an agent whose action is derived directly from an internal model of the current world state that is updated over time.

• **Goal-based agent**: an agent that selects actions that it believes will achieve explicitly represented goals.

• **Utility-based agent:** an agent that selects actions that it believes will maximize the expected utility of the outcome state.

• **Learning agent:** an agent whose behavior improves over time based on its experience.

2.6 This exercise explores the differences between agent functions and agent programs.
a. Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.
b. Are there agent functions that cannot be implemented by any agent program?
c. Given a fixed machine architecture, does each agent program implement exactly one agent function?
d. Given an architecture with n bits of storage, how many different possible agent programs are there?
e. Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?

Solution:

Although these questions are very simple, they hint at some very fundamental issues. Our answers are for the simple agent designs for static environments where nothing happens while the agent is deliberating; the issues get even more interesting for dynamic environments.

a. Yes; take any agent program and insert null statements that do not affect the output.

b. Yes; the agent function might specify that the agent print true when the percept is a

Turing machine program that halts, and false otherwise. (Note: in dynamic environments, for machines of less than infinite speed, the rational agent function may not be implementable; e.g., the agent function that always plays a winning move, if any, in a game of chess.)

c. Yes; the agent's behavior is fixed by the architecture and program.

d. There are 2n agent programs, although many of these will not run at all. (Note: Any given program can devote at most n bits to storage, so its internal state can distinguish among only 2n past histories. Because the agent function specifies actions based on percept histories, there will be many agent functions that cannot be implemented because of lack of memory in the machine.)

e. It depends on the program and the environment. If the environment is dynamic, speeding up the machine may mean choosing different (perhaps better) actions and/or acting sooner. If the environment is static and the program pays no attention to the passage of elapsed time, the agent function is unchanged.

Chapter-3-(PG-112)

Solving Problems by Searching

3.1 Explain why problem formulation must follow goal formulation.

Solution:

In goal formulation, we decide which aspects of the world we are interested in, and which can be ignored or abstracted away. Then in problem formulation we decide how to manipulate the important aspects (and ignore the others). If we did problem formulation first we would not know what to include and what to leave out. That said, it can happen that there is a cycle of iterations between goal formulation, problem formulation, and problem solving until one arrives at a sufficiently useful and efficient solution.

3.6 Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.
a. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.
b. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates.

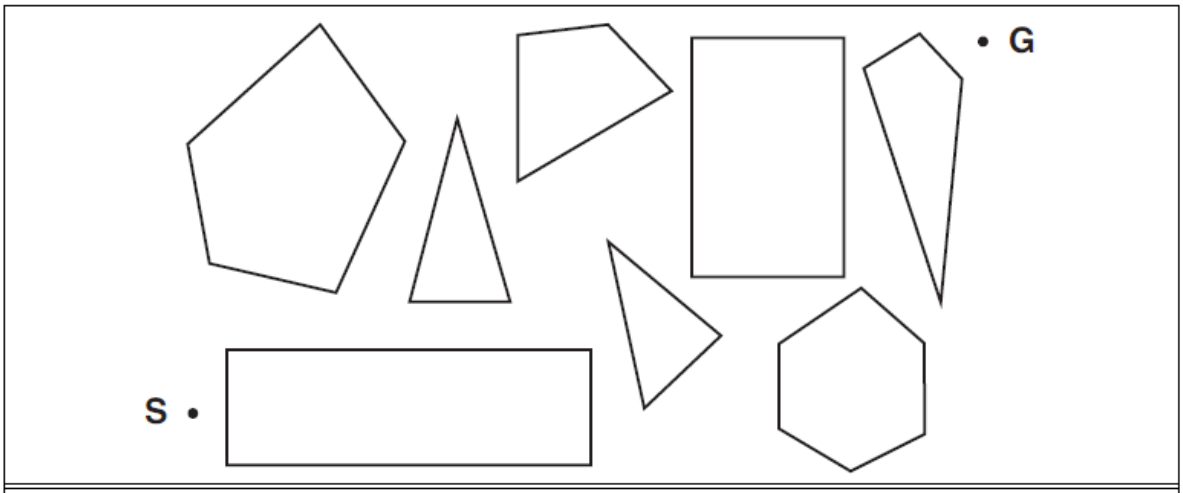

Figure 3.31 A scene with polygonal obstacles. S and G are the start and goal states.

c. You have a program that outputs the message "illegal input record" when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

d. You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

Solution:

a. Initial state: No regions colored.
Goal test: All regions colored, and no two adjacent regions have the same color.
Successor function: Assign a color to a region.
Cost function: Number of assignments.

b. Initial state: As described in the text.
Goal test: Monkey has bananas.
Successor function: Hop on crate; Hop off crate; Push crate from one spot to another; Walk from one spot to another; grab bananas (if standing on crate).
Cost function: Number of actions.

c. Initial state: considering all input records.
Goal test: considering a single record, and it gives "illegal input" message.
Successor function: run again on the first half of the records; run again on the second half of the records.
Cost function: Number of runs.
Note: This is a **contingency problem**; you need to see whether a run gives an error message or not to decide what to do next.

d. Initial state: jugs have values [0, 0, 0].
Successor function: given values [x, y, z], generate [12, y, z], [x, 8, z], [x, y, 3] (by filling); [0, y, z], [x, 0, z], [x, y, 0] (by emptying); or for any two jugs with current values x and y, pour y into x; this changes the jug with x to the minimum of x + y and the capacity of the jug, and decrements the jug with y by by the amount gained by the first jug.
Cost function: Number of actions.