IAT 1- Solutions

1. **Explain the interaction between agents and their environments in the context of AI.**

An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**. This simple idea is illustrated in Figure 2.1.
A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators. A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators. A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

We use the term **percept** to refer to the agent's perceptual inputs at any given instant. An agent's **percept sequence** is the complete history of everything the agent has ever perceived. In general, *an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.* By specifying the agent's choice of action for every possible percept sequence, we have said more or less everything there is to say about the agent. Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action.

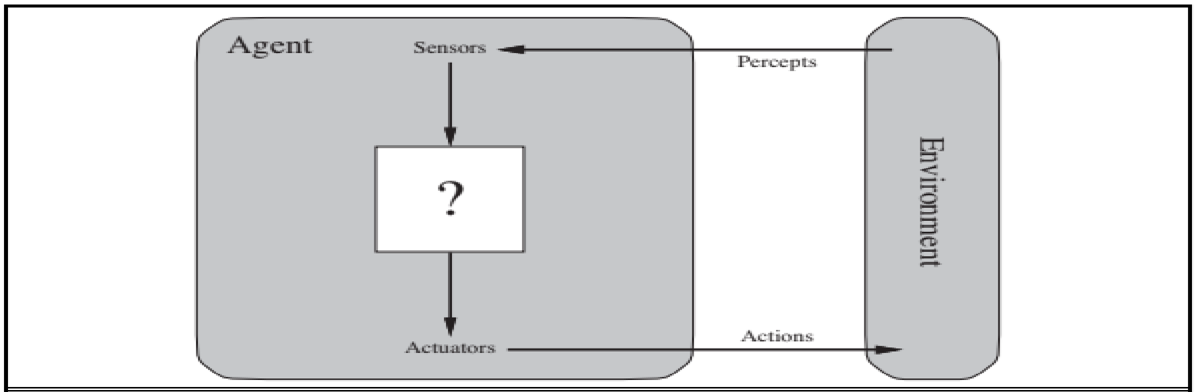


Figure 2.1 Agents interact with environments through sensors and actuators.

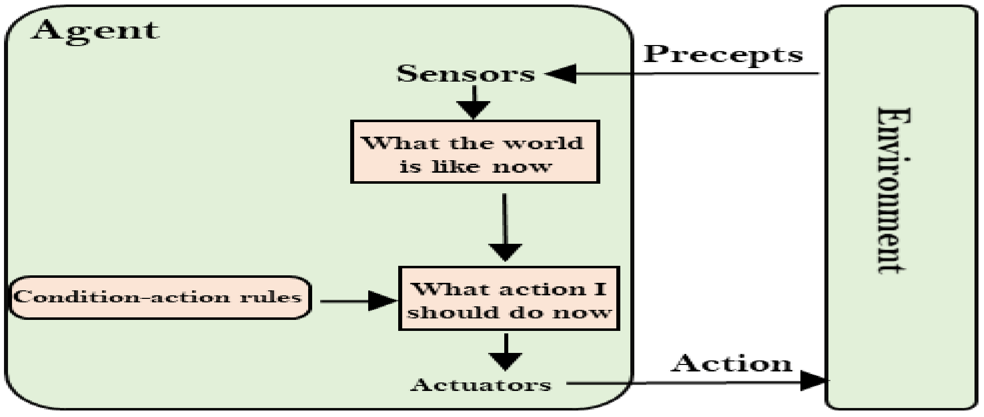
Internally, the agent function for an artificial agent will be implemented by an **agent program**. It is important to keep these two ideas distinct. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

To illustrate these ideas, we use a very simple example—the vacuum-cleaner world shown in Figure 2.2. This world is so simple that we can describe everything that happens; it's also a made-up world, so we can invent many variations. This particular world has just two locations: squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, suck up the dirt, or do nothing. One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square.

2. **Differentiate between simple reflex agents, model-based agents, goal-based agents, and utility-based agents.**

1. Simple Reflex agent:

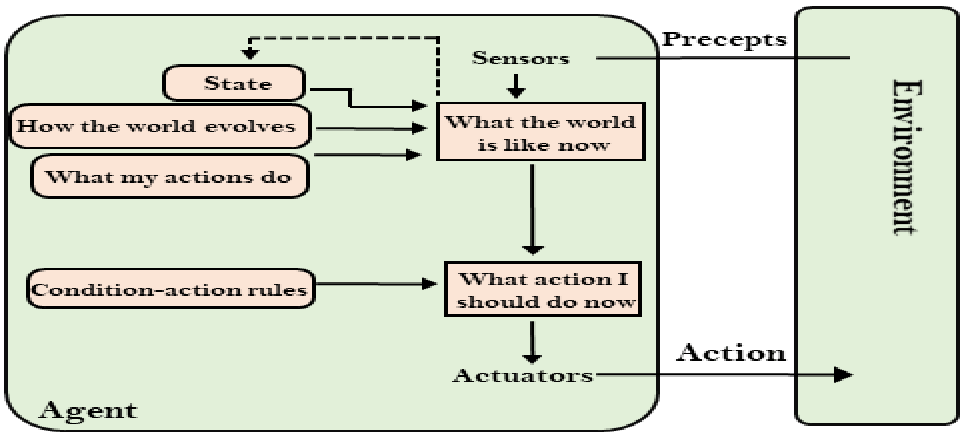
- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
- These agents only succeed in the fully observable environment.
- The Simple reflex agent does not consider any part of percepts history during their decision and action process.
- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:
    - They have very limited intelligence
    - They do not have knowledge of non-perceptual parts of the current state
    - Mostly too big to generate and to store.
    - Not adaptive to changes in the environment.



2. Model-based reflex agent

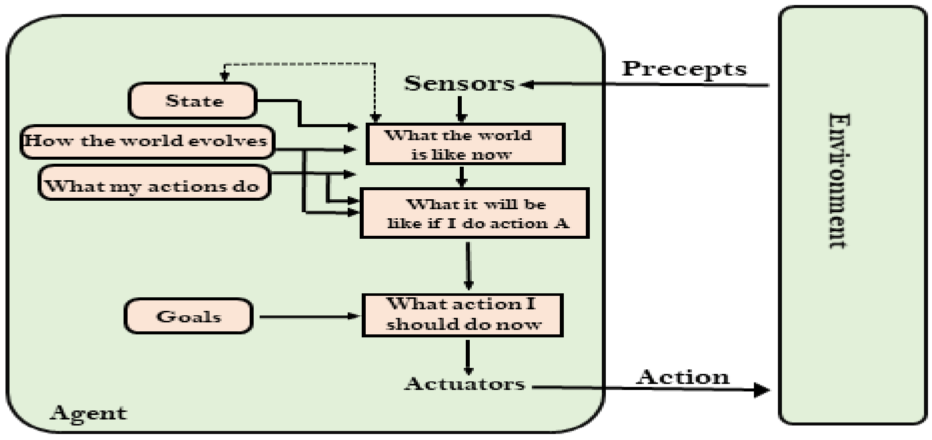
- The Model-based agent can work in a partially observable environment, and track the situation.
- A model-based agent has two important factors:

    1. Model: It is knowledge about "how things happen in the world," so it is called a Model-based agent.
    2. Internal State: It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
    1. How the world evolves
    2. How the agent's action affects the world.



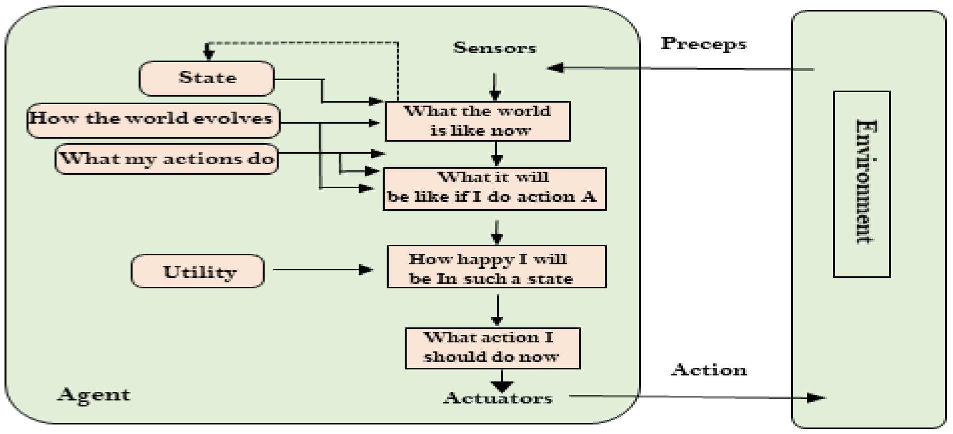
3. Goal-based agents

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose an action, so that they can achieve the goal.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.



4. Utility-based agents

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals**.**


State
How the world evolves
What my actions do
Utility
Sensors
What the world is like now
What it will be like if I do action A
How happy I will be In such a state
What action I should do now
Actuators
Agent
Preceps
Action
Enviroment

3. Explain how model-based knowledge provides the crucial robustness needed to make probabilistic systems feasible in the real world considering the example below.

What is the probability that a patient has diseases meningitis with a stiff neck?
Given Data:A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:
The Known probability that a patient has meningitis disease is 1/30,000.
The Known probability that a patient has a stiff neck is 2%.

BAYES' RULE AND ITS USE

Product rule an actually be written in two forms:
$P(a \wedge b) = P(a \mid b)P(b)$ and $P(a \wedge b) = P(b \mid a)P(a)$.

Equating the two right-hand sides and dividing by P(a), we get

This equation is known as **Bayes' rule** (also Bayes' law or Bayes' theorem). This simple equation underlies most modern AI systems for probabilistic inference.

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)} . \tag{13.12}$$

The more general case of Bayes' rule for multivalued variables can be written in the P notation as follows:

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(X \mid Y)\mathbf{P}(Y)}{\mathbf{P}(X)} ,$$

As before, this is to be taken as representing a set of equations, each dealing with specific values of the variables. We will also have occasion to use a more general version conditionalized on some background evidence e:

$$\mathbf{P}(Y \mid X, \mathbf{e}) = \frac{\mathbf{P}(X \mid Y, \mathbf{e})\mathbf{P}(Y \mid \mathbf{e})}{\mathbf{P}(X \mid \mathbf{e})} . \tag{13.13}$$

Applying Bayes' rule: The simple case
On the surface, Bayes' rule does not seem very useful. It allows us to compute the single term P(b | a) in terms of three terms: P(a | b), P(b), and P(a). That seems like two steps backwards, but Bayes' rule is useful in practice because there are many cases where we do have good probability

estimates for these three numbers and need to compute the fourth. Often, we perceive as evidence the effect of some unknown cause and we would like to determine that cause. In that case, Bayes' rule becomes

$$P(\mathit{cause} \mid \mathit{effect}) = \frac{P(\mathit{effect} \mid \mathit{cause})P(\mathit{cause})}{P(\mathit{effect})} .$$

The conditional probability P(effect | cause) quantifies the relationship in the **causal** direction, whereas P(cause | effect) describes the **diagnostic** direction. In a task such as medical diagnosis, we often have conditional probabilities on causal relationships (that is, the doctor knows P(symptoms | disease)) and want to derive a diagnosis, P(disease | symptoms). For example, a doctor knows that the disease meningitis causes the patient to have a stiff neck, say, 70% of the time. The doctor also knows some unconditional facts: the prior probability that a patient has meningitis is 1/50,000, and the prior probability that any patient has a stiff neck is 1%. Letting s be the proposition that the patient has a stiff neck and m be the proposition that the patient has meningitis, we have

$$\begin{aligned} P(s \mid m) &= 0.7 \\ P(m) &= 1/50000 \\ P(s) &= 0.01 \\ P(m \mid s) &= \frac{P(s \mid m)P(m)}{P(s)} = \frac{0.7 \times 1/50000}{0.01} = 0.0014 . \end{aligned} \tag{13.14}$$

That is, we expect less than 1 in 700 patients with a stiff neck to have meningitis. Notice that even though a stiff neck is quite strongly indicated by meningitis (with probability 0.7), the probability of meningitis in the patient remains small. This is because the prior probability of stiff necks is much higher than that of meningitis.

One can avoid assessing the prior probability of the evidence (here, P(s)) by instead computing a posterior probability for each value of the query variable (here, m and ¬m) and then normalizing the results. The same process can be applied when using Bayes' rule. We have

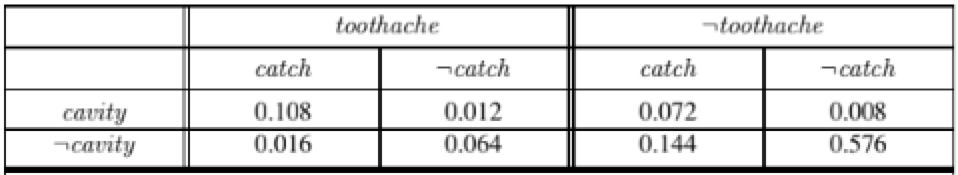
$$\mathbf{P}(M \mid s) = \alpha \, \langle P(s \mid m)P(m), P(s \mid \neg m)P(\neg m) \rangle .$$

Thus, to use this approach we need to estimate P(s | ¬m) instead of P(s). There is no free lunch—sometimes this is easier, sometimes it is harder. The general form of Bayes' rule with normalization is

$$\mathbf{P}(Y \mid X) = \alpha\, \mathbf{P}(X \mid Y)\mathbf{P}(Y)\ , \qquad (13.15)$$

where α is the normalization constant needed to make the entries in P(Y | X) sum to 1.

4. Explain Independence with respect to quantifying uncertainty



	toothache		*¬toothache*	
	catch	*¬catch*	*catch*	*¬catch*
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

Let us expand the full joint distribution in Figure 13.3 by adding a fourth variable, Weather . The full joint distribution then becomes P(Toothache, Catch, Cavity,Weather), which has 2 × 2 × 2 × 4 = 32 entries. It contains four "editions" of the table shown in Figure 13.3, one for each kind of weather. What relationship do these editions have to each other and to the original three-variable table? For example, how are P(toothache, catch, cavity, cloudy) and P(toothache, catch, cavity) related? We can use the product rule:

P(toothache, catch, cavity, cloudy)
= P(cloudy |toothache, catch, cavity)P(toothache, catch, cavity) .

Now, unless one is in the deity business, one should not imagine that one's dental problems influence the weather. And for indoor dentistry, at least, it seems safe to say that the weather does not influence the dental variables. Therefore, the following assertion seems reasonable:

P(cloudy |toothache, catch, cavity) = P(cloudy) . (13.10)

From this, we can deduce P(toothache, catch, cavity, cloudy) = P(cloudy)P(toothache, catch, cavity) . A similar equation exists for every entry in P(Toothache, Catch, Cavity,Weather). In fact, we can write the general equation P(Toothache, Catch, Cavity, Weather) = P(Toothache, Catch, Cavity)P(Weather) . Thus, the 32-element table for four variables can be constructed from one 8-element table and one 4-element table. This decomposition is illustrated schematically in Figure 13.4(a).

The property we used in Equation (13.10) is called **independence** (also **marginal in-dependence and absolute independence**). In particular, the weather is independent of one's dental problems. Independence between propositions a and b can be written as

$P(a \mid b) = P(a)$ or $P(b \mid a) = P(b)$ or $P(a \wedge b) = P(a)P(b)$. (13.11)

All these forms are equivalent (Exercise 13.12). Independence between variables X and Y can be written as follows (again, these are all equivalent):

$P(X \mid Y) = P(X)$ or $P(Y \mid X) = P(Y)$ or $P(X, Y) = P(X)P(Y)$

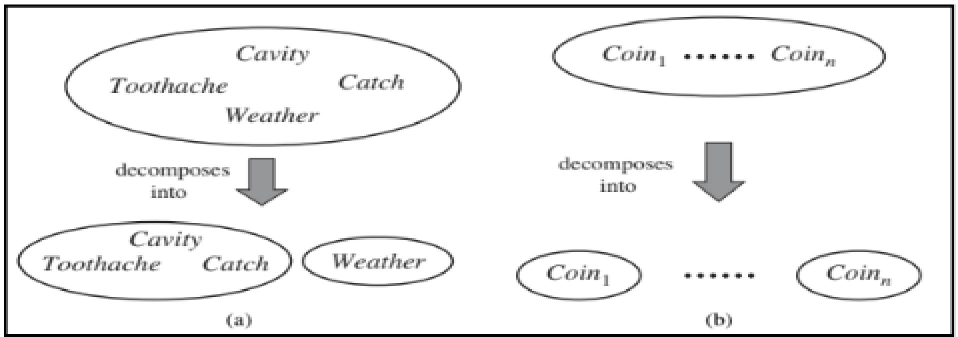


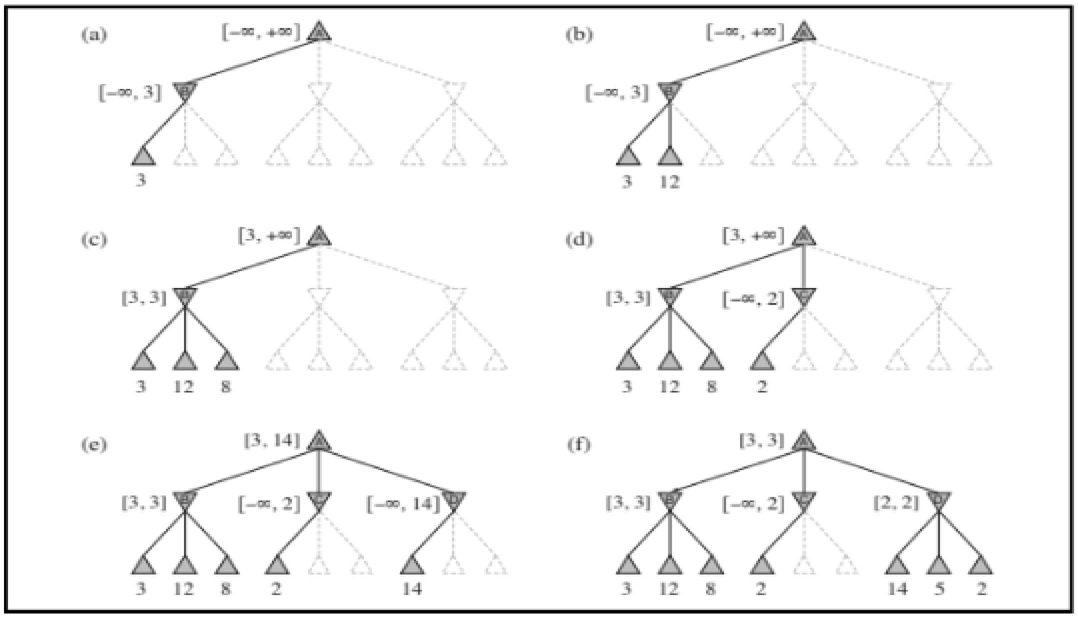
Figure 13.4 Two examples of factoring a large joint distribution into smaller distributions, using absolute independence. (a) Weather and dental problems are independent. (b) Coin flips are independent.

5. Nithin wants to build a game of checkers, which concept of AI is used to achieve this. Define the technique used to achieve this in detail by deriving the suitable algorithm being used and the time complexity achieved by this technique.

Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves. The general principle is this: consider a node n somewhere in the tree (see Figure 5.6), such that Player has a choice of moving to that node. If Player has a better choice m either at the parent node of n or at any choice point further up, then n will never be reached in actual play. So once we have found out enough about n (by examining some of its descendants) to reach this conclusion, we can prune it.

Alpha–beta search updates the values of α and β as it goes along and prunes the remaining branches at a node (i.e., terminates the recursive call) as soon as the value of the current node is known to be worse than the current α or β value for MAX or MIN, respectively.

Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves. The general principle is this: consider a node n somewhere in the tree (see Figure 5.6), such that Player has a choice of moving to that node. If Player has a better choice m either at the parent node of n or at any choice point further up, then n will never be reached in actual play. So once we have found out enough about n (by examining some of its descendants) to reach this conclusion, we can prune it.



5.3.1 Move ordering

The effectiveness of alpha–beta pruning is highly dependent on the order in which the states are examined. For example, in Figure 5.5(e) and (f), we could not prune any successors of D at all because the worst successors (from the point of view of MIN) were generated first. If the third successor of D had been generated first, we would have been able to prune the other two. This suggests that it might be worthwhile to try to examine first the successors that are likely to be best.

If this can be done, then it turns out that alpha–beta needs to examine only $O(b^{m/2})$ nodes to pick the best move, instead of $O(b^{m})$ for minimax. This means that the effective branching factor becomes $\sqrt{}$b instead of b—for chess, about 6 instead of 35. Put another way, alpha–beta can solve a tree roughly twice as deep as minimax in the same amount of time. If successors are examined in random order rather than best-first, the total number of nodes examined will be roughly $O(b^{3m/4})$ for moderate b. For chess, a fairly simple ordering function (such as trying captures first, then threats, then forward moves, and then backward moves) gets you to within about a factor of 2 of the best-case $O(^{bm/2})$ result.

5.3.1 Move ordering

The effectiveness of alpha–beta pruning is highly dependent on the order in which the states are examined. For example, in Figure 5.5(e) and (f), we could not prune any successors of D at all because the worst successors (from the point of view of MIN) were generated first. If the third successor of D had been generated first, we would have been able to prune the other two. This suggests that it might be worthwhile to try to examine first the successors that are likely to be best.

If this can be done, then it turns out that alpha–beta needs to examine only O($b^{m/2}$) nodes to pick the best move, instead of O(b^m) for minimax. This means that the effective branching factor becomes √b instead of b—for chess, about 6 instead of 35. Put another way, alpha–beta can solve a tree roughly twice as deep as minimax in the same amount of time. If successors are examined in random order rather than best-first, the total number of nodes examined will be roughly O($b^{3m/4}$) for moderate b. For chess, a fairly simple ordering function (such as trying captures first, then threats, then forward moves, and then backward moves) gets you to within about a factor of 2 of the best-case O($^{bm/2}$) result.