UNIT 2 INTELLIGENT AGENTS

Structure **Page Nos.**

2.0 INTRODUCTION

Since the time immemorial, we, the human beings, have always toyed with the idea of having some sort of slaves or agents, which would act as per our command, irrespective of the shape they take, as long as they do the job for which they have been designed or acquired. With the passage of time, human beings have developed different kinds of machines, where each machine has been intended to perform specific operation or a set of operations. However, ***with the development of the computer, human aspirations have increased manifolds*** *as it has allowed us to think of and actually implement non-human agents, which would show some level of independence and intelligence.* ***Robot, one of the most popular non-human agents,*** is a machine capable of perceiving the environment it is in, and further capable of taking some action or of performing some job either on its own or after taking some command.

Despite their perceived benefits, the dominant public image of the artificially embodied intelligent machines is more as potentially dangerous than potentially beneficial machines to the human race. The mankind is worried about the potentially dangerous capabilities of the robots to be designed and developed in the future.

Actually, any technology is a double-edged sword. For example, the Internet along with World Wide Web, on one hand, allows us to acquire information at the click of a button, but, at the same time, the Internet provides an environment in which a number of children (and, of course, the adults also) become addicts to downloading pornographic material. Similarly, **the development of non-human agents might not be without its tradeoffs. For example,** the more intelligent the robots are designed and developed, the more are the chances of a robot pursuing its own agenda than its master's, and more are the chances of a robot even attempting to destroy others to become more successful. Some intellectuals even think that, not in very distant future, there might be robots capable of enslaving the human beings, though designed and developed by the human beings themselves. Such concerns are not baseless. However, the (software) agents developed till today and the ones to be developed in the near future are expected to have very limited capabilities to match the kind of intelligence required for such behaviour.

In respect of the design and development of intelligent agents, with the passage of time, the momentum seems to have shifted from hardware to software, the letter being thought of as a major source of intelligence. But, obviously, some sort of hardware is essentially needed as a home to the intelligent agent.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- define the concept of an agent;
- explain the concepts of a 'rational agent' and 'rationality';
- tell about the various task environments, for which the use of agents is appropriate to solve problems from;
- explain the structure of an agent, and
- discuss various forms of used for agents to learn.

2.2 DEFINITIONS

An **agent** may be thought of as an entity that acts, generally on behalf of someone else. More precisely, an **agent** is an entity that ***perceives*** its environment through ***sensors*** and ***acts*** on the environment through ***actuators***. Some experts in the field require an agent to be additionally autonomous and goal directed also.

A **percept** may be thought of as an input to the agent through its censors, over a unit of time, sufficient enough to make some sense from the input.

Percept sequence is a sequence of percepts, generally long enough to allow the agent to initiate some action.

In order to further have an idea about what a *computer agent is*, let us consider one of the first definitions of agent, which was coined by John McCarthy and his friends at MIT.

A software agent is a system which, when given a goal to be achieved, could carry out the details of the appropriate (computer) operations and further, in case it gets stuck, it can ask for advice and can receive it from humans, may even evaluate the appropriateness of the advice and then act suitably.

Essentially, a computer agent is a computer software that additionally has the following attributes:

(i) it has autonomous control i.e., it operates under its own control
(ii) it is perceptive, i.e., it is capable of perceiving its own environment
(iii) it persists over a long period of time
(iv) it is adaptive to changes in the environment and
(v) it is capable of taking over others' goals.

As the concept of (software) agent is of relatively recent origin, different pioneers and other experts have been conceiving and using the term in different ways. There are two distinct but related approaches for defining an agent. The **first approach** treats an agent as an **ascription** i.e., the perception of a person (which includes expectations and points of view) whereas the **other approach** defines an agent on the basis of the **description** of the properties that the agent to be designed is expected to possess.

Let us first discuss the definition of agent according to first approach. Among the people who consider *an agent as an* ascription, a popular slogan is "**Agent is that agent does**". In everyday context, **an agent is expected to act on behalf of someone** to carry out a particular task, which has been delegated to it. But to perform its task successfully, the agent must have knowledge about the domain in which it is operating and also about the properties of its current user in question. In the course of normal life, we hire different agents for different jobs based on the required expertise for each job. Similarly, a **non-human intelligent agent also is imbedded with required expertise of the domain as per requirements of the job under consideration**. For example, ***a football-playing agent would be different from an email-managing agent,*** although both will have the common attribute of modeling their user.

According to the second approach, an agent is defined as an entity, which functions continuously and autonomously, in a particular environment, which may have other agents also. **By continuity and autonomy of an agent, it is meant** that the agent must be able to carry out its job in a flexible and intelligent fashion and further is expected to adapt to the changes in its environment without requiring *constant* human guidance or intervention. Ideally, an agent that functions continuously in an environment over a long period of time would also **learn** from its experience. In addition, we expect an agent, which lives in a multi-agent environment, to be able to **communicate and cooperate** with them, and perhaps move from place to place in doing so.

According to the **second approach** to defining agent, **an agent is supposed to possess some or all of the following properties:**

- **Reactivity:** The ability of sensing the environment and then acting accordingly.
- **Autonomy:** The ability of moving towards its goal, changing its moves or strategy, if required, without much human intervention.
- **Communicating ability:** *T*he ability to communicate with other agents and humans.
- **Ability to coexist by cooperating:** The ability to work in a multi-agent environment to achieve a common goal.
- **Ability to adapt to a new situation:** Ability to learn, change and adapt to the situations in the world around it.
- **Ability to draw inferences:** The ability to infer or conclude facts, which may be useful, but are not available directly.
- **Temporal continuity:** The ability to work over long periods of time.
- **Personality:** Ability to impersonate or simulate someone, on whose behalf the agent is acting.
- **Mobility:** Ability to move from one environment to another.

2.3 AGENTS AND RATIONALITY

Further, a **rational agent** is an agent that acts in a manner that achieves best outcome in an environment with certain outcomes. In an uncertain environment, a rational agent through its actions attempts the best-expected outcome.

It may be noted that *correct inferencing* is one of the several possible mechanisms for achieving *rationality*. However, sometimes a rational action is also possible without inferencing. For example, removing hand when a very hot utensil is touched *unintentionally* is an example of rationality based on *reflex action* instead of based on *inferencing*.

We discuss the concepts of **rationality** and **rational agent** in some more detail.

Attempting to take always the correct action, possibly but not necessarily involving logical reasoning, is only one part of being rational. Further, if a perfectly correct action or inference is not possible then taking an approximately correct, but, optimal action under the circumstances is a part of rationality.

Like other attributes, we need some *performance measure* (i.e., the criteria to judge the performance or success in respect of the task to be performed) to judge rationality. A good **performance measure for rationality** must be:

- Objective in nature
- It must be measurable or observable and
- It must be decided by the designer of the agent keeping in mind the problem or the set of problems for handling which the agent is designed.

In summary, rationality depends on:

- The performance measure chosen to judge the agent's activities.
- The agent's knowledge of the current environment or of the world in which it exists. The better the knowledge of the environment, the more will be probability of the agent taking an appropriate action.
- The length of the percept sequence of the agent. In the case of a longer percept sequence, the agent can take advantage of its earlier decisions or actions for similar kind of situations.
- The set of actions available to the agent.

From the previous discussion, we know that a **rational agent** should take an action, which **would correct its performance measure** on the basis of its knowledge of the world around it and the percept sequence.

By the ***rationality of an agent, we do not mean it to be always*** **successful or it to be omniscient**. ***Rationality is concerned with*** the agent's capabilities for **information gathering, exploration** and **learning** from its environment and experience and is also concerned with the **autonomy** of the agent.

2.3.1 Rationality vs. Omniscience

The basic difference between being rational and being omniscient is that **rationality** deals with trying to maximize the output on the basis of current input, environmental conditions, available actions and past experience whereas being **omniscient** means having knowledge of *everything*, including knowledge of the future i.e., what will be the output or outcome of its action. Obviously being omniscient is next to impossible.

In this context, let us consider the following scenario: Sohan is going to the nearby grocery shop, but unfortunately when Sohan is passing through the crossing suddenly a police party comes at that place chasing a terrorist and attempts to shoot the terrorist but unfortunately the bullet hits Sohan and he is injured.

Now the question is: *Is Sohan irrational in moving through that place.* The answer is no, because the human agent *Sohan* has no idea, nor is expected to have an idea, of what is going to happen at that place in the near future. Obviously, Sohan is *not omniscient* but, from this incident can *not be said to be irrational.*

2.3.2 Autonomy and Learning Capability of the Agent

Autonomy means the dependence of the agent own its on percepts (what it perceives or receives from the environment through senses) rather than the prior knowledge of its designer. In other words, an agent is **autonomous** if it is capable of learning from its experience and has not to depend upon its prior knowledge which may either be not complete or be not correct or both. Greater the autonomy more flexible and more rational the agent is expected to be. So we can say that a rational agent should be autonomous because it should be able to learn to compensate for the incomplete or incorrect prior knowledge provided by the designer. In the initial stage of its operations, the agent may not, rather should not, have *complete autonomy.* This is desirable, in view of the fact that in the initial stage, the agent is yet to acquire knowledge of the environment should use the experience and knowledge of the designer. But, as the agent gains experience of its environment, its behavior should become more and more independent of its prior knowledge provided to it by its designer.

Learning means the agent can update its knowledge based on its experience and changes in the environment, for the purpose of taking better actions in future. Although the designer might feed some prior knowledge of the environment in the agent but, as mentioned earlier, as the environment might change with the passage of time, therefore, feeding complete knowledge in the agent, at the beginning of the operations is neither possible nor desirable. Obviously, there could be some extreme cases in which environment is static, i.e., does not change with time, but such cases are rare. In such rare cases, however, giving complete information about the environment may be desirable. So, in general, in order to update itself, the agent should have the learning capability to update its knowledge with the passage of time.

2.3.3 Example (A boundary following robot)

Let us consider the first **example of an agent**, which is ***a robot that follows a boundary wall.***

The definition of the problem, to solve, which an agent is to be designed, consists in giving details of

(i) The perception capabilities that are to be available to the agent
(ii) The actions that the agent would be able to perform
(iii) The environment in which the agent is expected to perform the task or solve the problem.

Environment and tasks

The robot is assumed to be enclosed in a room, which is perceived as a two-dimensional grid-space. The room is enclosed on all sides by walls high enough not to allow the robot to escape. The room may have some immovable objects. For the purpose of movement of the robot, the room is supposed to be divided into cells, in

the sense that at one time, the robot occupies only one cell and does not have its parts spread over two or more cells. To be capable of boundary-following behavior, the robot must be ***able to sense*** whether or not certain cells are free for it to possibly occupy in the next move. The ***only action*** it is able to take is ***to move from the currently occupied cell to*** any one of the unoccupied surrounding cells, e.g., to c_1, c_2,….c_8 as shown in the following diagram.

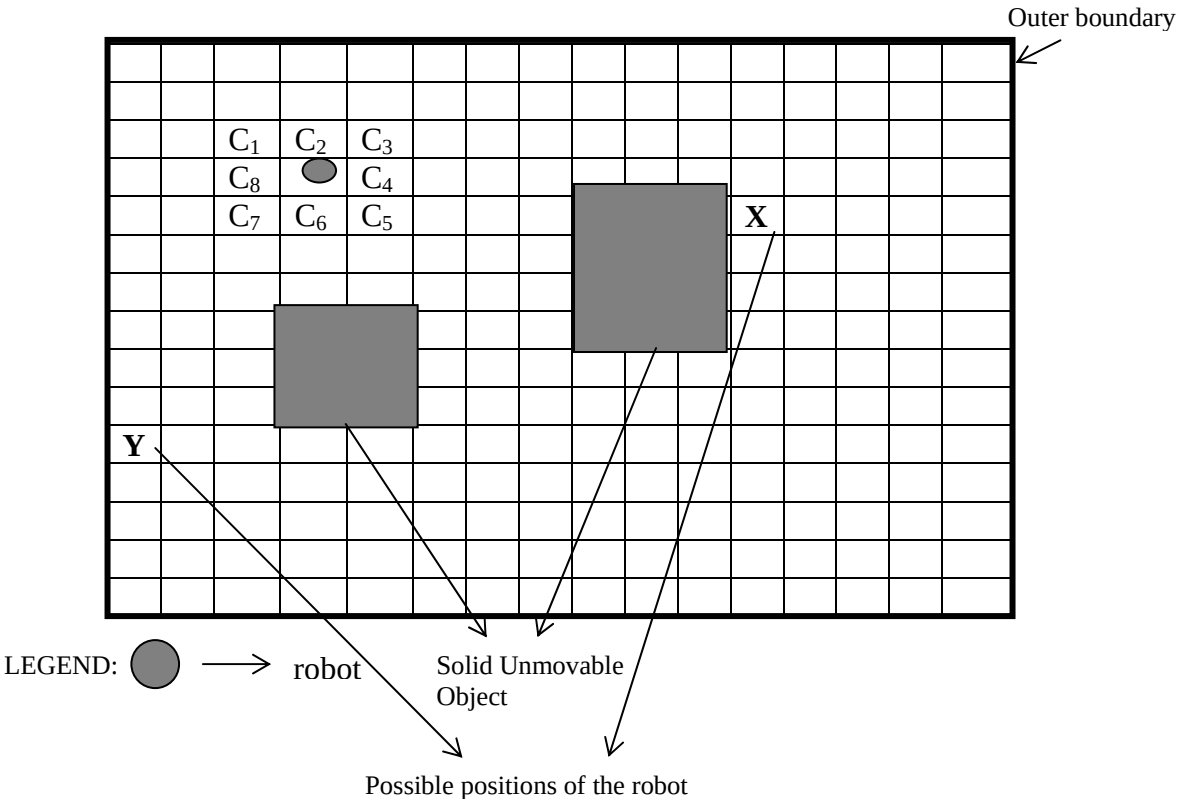


Figure 2.1: A Boundary Following Robot

The sensory input about the surrounding cells can be represented in the form of an 8-tuple vector say <c1,c2,c3,c4,c5,c6,c7,c8> where each ci is a binary number i.e., ci is either a 0 for representing a **free cell** or a 1 for representing the fact that the cell is **occupied** by some object. If, at some stage, the robot is in the cell represented by **X,** as shown in *Figure 2.1*, then the sense vector would be <0,0,0,0,0,0,0,1,1>. It may be noted that the corner and the boundary cells may not have exact eight surrounding cells. However, for such cells, the missing neighbouring cell may be treated as occupied. For example, for the cell y of *Figure 2.1*, the neighbouring cells to the left of the cell y do not exist. However, *we assume these cells exist but are occupied.* Thus, if the robot is in position **Y** of *Figure 2.1* then the sense vector for the robot is <1,0,0,0,0,0,1,1>.

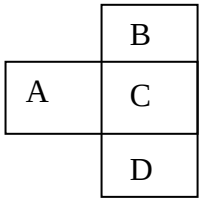
Also, the robot performs only one type of action i.e., it can move in one of the free cells adjacent to it. In case the robot attempts to move to a cell, which is not free, the action will have no effect i.e., there will be no action.

On the basis of the properties of the environment in which the robot is operating, **it is the job of the designer to develop an algorithm for the process that converts robot's precept sequences** (which in this case is the sensory input c1 to c8) **into corresponding actions**. Let us divide the task of determining the action the agent should take on the basis of the sensory inputs into two phases, viz., **perception** phase and **action** phase.

Perception Phase:

There is an enclosed space, which we call a room. The room is thought of as divided into cells. A cell has just enough space for the robot to be accommodated. Further, at any time, robot, except during transition, can not lie across two cells.

- Placement of the agent in some cell of the room is essentially random.
- Agent is to perform boundary following.
- The sensory input for the robot consists of the binary values c1, c2,…,c8 in the eight cells adjacent to the position occupied by the robot. Because there are 8 adjacent cells, so the possible combinations of these values is 2^8 i.e., 256 combinations. The movement of the robot to any one of the unoccupied/free surrounding cells depends upon the robot's environmental conditions. For example, if the (traffic) signal to the right is green and some cells on the right of the cell currently occupied by the robots, are free then move to the uppermost cell of free cells to the right. For example,



if A is the cell currently occupied by the robot and the cell B is occupied by some object, but cells C and D are free. And, further, signal on the right is green, then the robot moves to cell C. However, if both B and C are occupied, than the robot moves to cell D. These environmental conditions are determined in terms of what we call *features.* For the current problem, let us assume that there are 4 binary valued features of the agent, namely f1, f2, f3 and f4 and these features are used to calculate the action to be taken by the agent.

The rules for assigning binary values to features may be assumed to be as given below:

IF c2 = 1 OR c3 = 0 THEN f1 = 1 ELSE f1 = 0.

IF c4 = 0 OR c5 = 1 THEN f2 = 1 ELSE f2 = 0.

IF c6 = 1 OR c7 = 1 THEN f3 = 1 ELSE f3 = 0.

IF c8 = 1 OR c1 = 1 THEN f4 = 1 ELSE f4 = 0.

Action Phase:

After calculating the values of the feature set of the robot, a *pre-defined function* is used to determine boundary following action of the robot (i.e., the movement to one of the surrounding cell). In addition to specified actions determined by the features and the pre-defined function, there may be a default action of movement to a surrounding free cell. The default action may be executed when no action is possible.

In order to illustrate the type of possible actions, let us consider an example of specifying the rules for different actions to be performed by the robot:

Action 1: IF f1 = 0 AND f2 = 0 AND f3 = 0 AND f4 = 0
THEN move up to the right-most free surrounding cell. If no such cell is free then Attempt Action 3.

Action 2: IF f1 = 1 AND f2 = 0

THEN move topmost free surrounding cell on the right by one cell. If no such cell is free, then stop.

Action 3: IF f2 = 1 AND f3 = 0
THEN move down to the left-most free cell. If no such cell is free then attempt Action 2.

Action 4: IF f3 = 1 AND f4 = 0
THEN move to the bottom-most free cell on the left. If no such surrounding cell is free then move to action 1.

We can see that the combinations of the features values serve as conditions under which the robot would perform its actions.

Next, we enumerate below some of the possible parameters for performance measures in case of such a simple agent.

- The number of times agent has to reverse its earlier courses of actions (more times indicates lower performance).
- The maximum of the average of distances from the walls (the less the maximum, higher the performance).
- The boundary distance traversed in particular time period (the more the distance covered, better the performance).

In the next section, we discuss some of the commonly used terms in context of the Agents.

Exercise 1

On the basis of the figure given below, find the sense vector of the robot if:
(a) It starts from the location L_1
(b) It starts from the location

The robot can sense whether the 8 cells adjacent to it are free for it to move into one of them. If the location of the robot is such that some of the surrounding cells do not exist, then these cells are, assumed to exist and further assumed to be occupied

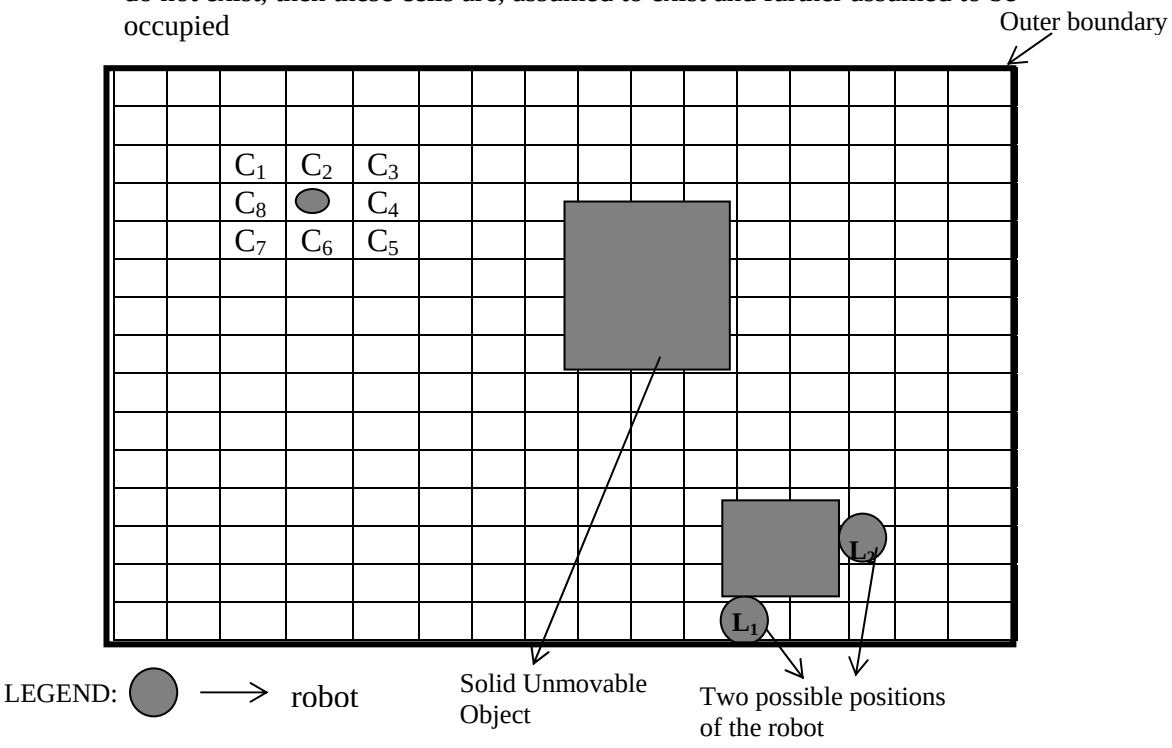


Figure 2.2 Environment of the robot

2.4 TASK ENVIRONMENT OF AGENTS

Task environments or problem environments are the environments, which include all the elements involved in the problems for which agents are thought of as solutions. Task environments will vary with every new task or problem for which an agent is being designed. Specifying the task environment is a long process which involves looking at different measures or parameters. Next we discuss a standard set of measures or parameters for *specifying* a task environment under the heading **PEAS**.

2.4.1 PEAS (Performance, Environment, Actuators, Sensors)

For designing an agent, the first requirement is to specify the task environment to the maximum extent possible. The task environment for an agent to solve one type of problems, may be described by the four major parameters namely, ***performance*** (which is actually the expected performance), ***environment*** (i.e., the world around the agent), ***actuators*** (which include entities through which the agent may perform actions) and ***sensors*** (which describes the different entities through which the agent will gather information about the environment).

The four parameters may be collectively called as PEAS.

We explain these parameters further, **through an example of an automated agent**, which we will preferably call ***automated public road transport driver***. This is a much more complex agent than the simple ***boundary following robot*** which we have already discussed.

2.4.2 Example (An Automated Public Road Transport Driver Agent)

We describe the task environment of the agent on the basis of PEAS.
Performance Measures:

Some of the performance measures which can easily be perceived of an

automated public road transport driver would be :

- Maximizing safety of passengers
- Maximizing comfort of passengers
- Ability to reach correct destination
- Ability to minimize the time to reach the destination
- Obeying traffic rules
- Causing minimum discomfort or disturbance to other agents
- Minimizing costs, etc.

Environment (or the world around the agent)

We must remember that the environment or the world around the agent is extremely uncertain or open ended. There are unlimited combinations of possibilities of the environment situations, which such an agent could face. Let us enumerate some of the possibilities or circumstances which an agent might face:

- Variety of roads e.g., from 12-lane express-ways, freeways to dusty rural bumpy roads; different road rules including the ones requiring left-hand drive in some parts of the world and right-hand drive in other parts.
- The degree of knowledge of various places through which and to which driving is to be done.

- Various kinds of passengers, including high cultured to almost ruffians etc.
- All kind of other traffic possibly including heavy vehicles, ultra modern cars, three-wheelers and even bullock carts.

Actuators:

These include the following:

- Handling steering wheel, brakes, gears and accelerator
- Understanding the display screen
- A device or devices for all communication required

Sensors:

The agent acting as automated public road transport driver must have some way of sensing the world around it i.e., the traffic around it, the distance between the automobile and the automobiles ahead of it and its speed, the speeds of neighbouring vehicles, the condition of the road, any turn ahead etc. *It may use sensors like odometer, speedometer, sensors telling the different parameters of the engine, Global Positioning System (GPS)* to understand its current location and the path ahead. Also, there should be some sort of sensors to calculate its distance from other vehicles etc.

We must remember that the agent example *the automated public road transport driver*, which we have considered above, is quite difficult to implement. However, there are many other agents, which operate in comparatively simpler and less dynamic environments, e.g., a game playing robot, an assembly line robot control, and an image processing agent etc.

2.4.3 Different Types of Task Environments

Next, we discuss some of the important characteristics of the environments of the problems which are generally considered for solution through agent technology.

- Fully observable vs. partially observable
- Static vs. dynamic
- Deterministic vs. stochastic
- Episodic vs. sequential
- Single agent vs. multi-agent

2.4.3.1 Fully Observable vs. Partially Observable Environment

If the agent knows *everything* about its environment or the world in which it exists, through its sensors, then we say that the environment is ***fully observable***. It would be quite convenient for the agent if the environment is a fully observable one because the agent will have a wholesome idea of what all is happening around before taking any action. The agent in a fully observable environment will have a complete idea of the world around it at all times and hence it need not maintain an internal state to remember and store what is going around it. **Fully observable environments are found rarely in reality.**

A ***partially observable environment*** is that in which the agent can sense **some** of the aspects or features of the world around it, but not all because of any number of reasons including the limitations of its sensors. Normally, there are greater chances of finding a partially observable environment in reality. *In the first* ***example***, viz., **a boundary following robot**, which we have studied, the agent or robot has a limited

view of the environment around it, i.e., the agent has information only about the eight cells immediately adjacent to it in respect of whether each of these is occupied or free. It has no idea of how many non-moving objects are there within the room. Nor, it has an idea of its distance from the boundary wall. Also, *in the second* ***example***, viz., **an automated public road transport driver**, *the driver or the agent has a very restricted idea of the environment around it*. It operates in a very dynamic environment, i.e., an environment that changes very frequently. Further, it has no idea of the number of vehicles and their positions on the road, the action the driver of the vehicle ahead is expected to take, the location of various potholes it is going to face in the next one kilometer and so on. So we can see that in both of these examples, the environment is partially observable.

2.4.3.2 Static vs. Dynamic Environment

Static environment means an environment which does not change while the agent is operating. Implementing an agent in a static environment is relatively very simple, as the agent does not have to keep track of the changes in the environment around it. So the time taken by an agent to perform its operations has no effect on the environment because the environment is static. The **task environment of the "boundary following robot" is static** because no changes are possible in the environment around it, irrespective of how long the robot might operate. Similarly, the environment of an agent solving a crossword puzzle is static.

A dynamic environment on the other hand may change with time on its own or because of the agent's actions. Implementing an agent in a dynamic environment is quite complex as the agent has to keep track of the changing environment around it. The **task environment of the "Automated public road transport driver" is an example of a dynamic environment** because the whole environment around the agent keeps on changing continuously.

In case an environment does not change by itself but an agent's performance changes the environment, then the environment is said to be **semi-dynamic**.

2.4.3.3 Deterministic vs. Stochastic Environment

A deterministic environment means that the current state and the current set of actions performed by the agent will completely determine the next state of the agent's environment *otherwise* the environment is said to be **stochastic**. The decision in respect of an environment being stochastic or deterministic, is taken from the point of view of the agent. If the environment is *simple and fully observable* then there are greater chances of its being *deterministic*. However, if the environment is *partially observable and complex* then it may be *stochastic* to the agent. For example, the *boundary following robot* exists in a deterministic environment whereas *an automated road transport driver agent* exists in a stochastic environment. In the later case, the agent has no prior knowledge and also it cannot predict the behavior of the traffic around it.

2.4.3.4 Episodic vs. Sequential Environment

In an episodic environment, the agent's experience is divided into *episodes.* An agent's actions during an episode depend only on that episode because subsequent episodes do not depend on previous episodes. For example, an agent checking the tickets of persons who want to enter the cinema hall, works in an episodic environment, as the process of checking the tickets of every new person is an episode which is independent of the previous episode i.e., checking the ticket of the previous person, and also the current action taken by the ticket checking agent does not effect the next action to be taken in this regard. Also a robot while checking the seals on the

bottles on an assembly line, also works in an episodic environment as checking the seal of every new bottle is a new and independent episode.

In sequential environments there are no episodes and **the previous actions could affect the current decision or action and further, the current action could effect the future actions or decisions**. The task environment of "An automated public road transport driver" is an example of a sequential environment as any decision taken or action performed by the driver agent may have long consequences.

Comparatively, working in an episodic environment is much simpler for an agent than in a sequential environment, because, in the former case, previous actions do not effect the current actions of the agent. Also, while taking current action, the agent need not think of its future effects.

2.4.3.5 Single-agent vs. Multi-agent Environment

A single-agent environment is the simplest possible environment as the agent is the only active entity in the whole environment and it does not have to synchronize with the actions or activities of any other agent. Some of the examples of a single-agent environment are *boundary following robot, a crossword puzzle solving agent* etc.

If there is a possibility of other agents also existing in addition to the agent being described then the task environment is said to **be multi-agent environment**. In a multi-agent environment, the scenario becomes quite complex as the agent has to keep track of the behavior and the actions of the other agents also. There can be two general scenarios, one in which all the agents are working together to achieve some common goal, i.e., to perform a collective action. Such a type of environment is called **cooperative multi-agent environment**.

Also, it may happen that all or some of the agents existing in the environment are competing with each other to achieve something (for example, in a tic-tac-toe game or in a game of chess both of which are two agent games, each player tries to win by trying to predict the possible moves of the other agent), such an environment is called **competitive multi-agent environment**.

An **important issue in a multi-agent environment** is to determine whether other entities existing in the environment have to be treated as *passive objects or agents*. Normally, if an entity tries to optimize its performance which have an effect on the agent in question then that entity is treated as an agent, but there is no fixed rule to decide this fact. Also, **another very important issue** in a multi-agent environment is to decide how the **communication between different agents** will occur.

2.4.3.6 Discrete vs. Continuous Environment

The word discrete and continuous are related to the way time is treated, i.e., whether time is divided into slots (i.e., discrete quantities) or it is treated as a continuous quantity (continuous).

For example, the task environment of a "boundary following robot" or a "chess playing agent" is a discrete environment because there are finite number of discrete states in which an agent may find itself. On the other hand, *an automated public transport driver* agent's environment is a continuous one, because, the actions of the agent itself as well as the environment around it are changing continuously. In the case of a driver agent, the values received through the sensor may be at discrete intervals but generally the values are received so frequently that practically the received values are treated as a stream of continuous data.

2.4.4 Some Examples of Task Environments

(i) Planning crossword puzzle
(ii) Medical diagnosis
(iii) Playing tic-tac-toe
(iv) Playing chess
(v) Driving automobile

2.4.4.1 Crossword Puzzle

As we have already discussed the environment is ***single agent*** and ***static*** which makes it simple as there are no other players, i.e., agents and the environment does not change. But the environment is ***sequential*** in nature, as the current decision will affect all the future decisions also. On the simpler side, the environment of a crossword puzzle is ***fully observable*** as it does not require remembering of some facts or decisions taken earlier and also the environment is ***deterministic*** as the next state of the environment fully depends on the current state and the current action taken by the agent. Time can be divided into ***discrete quanta*** between moves and so we can say that the environment is discrete in nature.

2.4.2.2 Medical Diagnosis

The first problem in the task environment of an agent performing a medical diagnosis is to decide whether the environment is to be treated as a single-agent or multi-agent one. If the other entities like the patients or staff members also have to be viewed as agents (obviously based on whether they are trying to maximize some performance measure e.g., Profitability) then it will be a ***multi-agent*** environment ***otherwise*** it will be a ***single*** agent environment. To keep the discussion simple, in this case, let us choose the environment as ***a single agent*** one.

We can very well perceive that the task environment is ***partially observable*** as all the facts and information needed may not be available readily and hence, need to be remembered or retrieved. The environment is ***stochastic*** in nature, as the next state of the environment may not be fully determined only by the current state and current action. Diagnosing a disease is stochastic rather than deterministic.

Also the task environment is partially ***episodic*** in nature and partially ***sequential.*** The environment is episodic because, each patient is diagnosed, irrespective of the diagnoses of earlier patients. The environment is sequential, in view of the fact that for a particular patient, earlier treatment decisions are also taken into consideration in deciding further treatment. Furthermore, the environment may be treated as **continuous** as there seems to be no benefit of dividing the time in discrete slots.

2.4.4.3 Playing Tic-tac-toe

As it is a two-player game so obviously the environment is ***multi-agent***. Moreover, both the players try to maximize their chances of winning, so it is a ***competitive*** one. Further, the environment is a ***fully observable*** environment. The environment here is ***neither fully deterministic nor stochastic*** as the environment is deterministic except for one fact that action of the other agent is unpredictable, so the environment is ***strategic***. Obviously the environment is ***semi-dynamic*** as the environment itself does not change but the agents performance score does change with time. As time may be divided into discrete slots between moves so the environment may be viewed as ***discrete.*** Further, the current decisions while making a move affect all the future decisions also, therefore, the environment is ***sequential*** also.

2.4.4.4 Playing Chess

The environment of a chess-playing agent is also the same as is that of an agent-playing tic-tac-toe. It is also a two-player i.e., **multi-agent** game. **Some people treat the environment as fully observable** but actually for some of the rules require remembering the game history so the full environment is not observable from the current state of the chess board. Thus, **strictly speaking the environment is partially-observable**. Further, using the same arguments as given in case of tic-tac-toe, the environment is **strategic, semi-dynamic, discrete and sequential** in nature.

2.4.4.5 Automobile Driver Agent

We have already discussed one subclass of a general driver agent in detail, viz., *an automated public road transport driver* which may include a bus driver, taxi driver or auto driver etc. A driver agent might also be christened as a *cruise control agent*, which may include other types of transport like water transport or air transport also with some variations in their environments.

Coming back to *an automated public road transport driver,* we can see that this is one of the most complex environments discussed so far. So the environment is ***multi-agent and partially observable*** as it is not possible to see and assume what all other agents i.e., other drivers are doing or thinking. Also the environment is fully **dynamic**, as it is changing all the time as the location of the agent changes and also the locations of the other agents change. The environment is **stochastic** in nature as anything unpredictable can happen, which may not be perceived exactly as a result of the current state and the actions of various agents. The environment is **sequential** as any decision taken by the driver might affect or change the whole future course of actions. Also the time is **continuous** in nature although the sensors of the driver agent might work in discrete mode.

2.5 THE STRUCTURE OF AGENTS

There are two parts of an agent or its structure:

- A (hardware) device with sensors and actuators in which that agent will reside, called the *architecture* of the agent.
- An agent program that will convert or map the percepts in to actions.

Also, the agent program and its architecture are related in the sense that for a different agent architecture a different type of agent program is required and vice-versa. For example, in case of a *boundary following robot*, if the robot does not have the capability of sensing adjacent cells to the right, then the agent program for the robot has to be changed.

Next, we discuss different categories of agents, which are differentiated from each other on the basis of their agent programs. Capability to write efficient agent programs is the key to the success for developing efficient rational agents. Although the table driven approach (in which an agent acts on the basis of the set of all possible percepts by storing these percepts in tables) to design agents is possible yet the approach of developing equivalent agent programs is found much more efficient.

Next we discuss some of the general categories of agents based on their agents programs:

- ***SR (Simple Reflex) agents***
- ***Model Based reflex agents***

- ***Goal-based agents***
- ***Utility based agents***

2.5.1 Simple Reflex (SR) Agents

These are the agents or machines that have no internal state (i.e., the don't remember anything) and simply react to the current percepts in their environments. An interesting set of agents can be built, the behaviour of the agents in which can be captured in the form of a simple set of functions of their sensory inputs. One of the earliest implemented agent of this category was called ***Machina Speculatrix.*** This was a device with wheels, motor, photo cells and vacuum tubes and was designed to move in the direction of light of less intensity and was designed to avoid the direction of the bright light. **A boundary following robot is also an SR agent**. For an automobile-driving agent also, some aspects of its behavior like applying brakes immediately on observing either the vehicle immediately ahead applying brakes or a human being coming just in front of the automobile suddenly, show the simple reflex capability of the agent. Such a simple reflex action in the agent program of the agent can be implemented with the help of simple condition-action rules.

For example : ***IF*** *a human being comes in front of the automobile suddenly*
THEN *apply breaks immediately*.

Although implementation of SR agents is simple **yet on the negative side** this type of agents have very limited intelligence because ***they do not store or remember anything.*** As a consequence they cannot make use of any previous experience. In summary, they do not learn. Also **they are capable of operating correctly only if the environment is fully observable.**

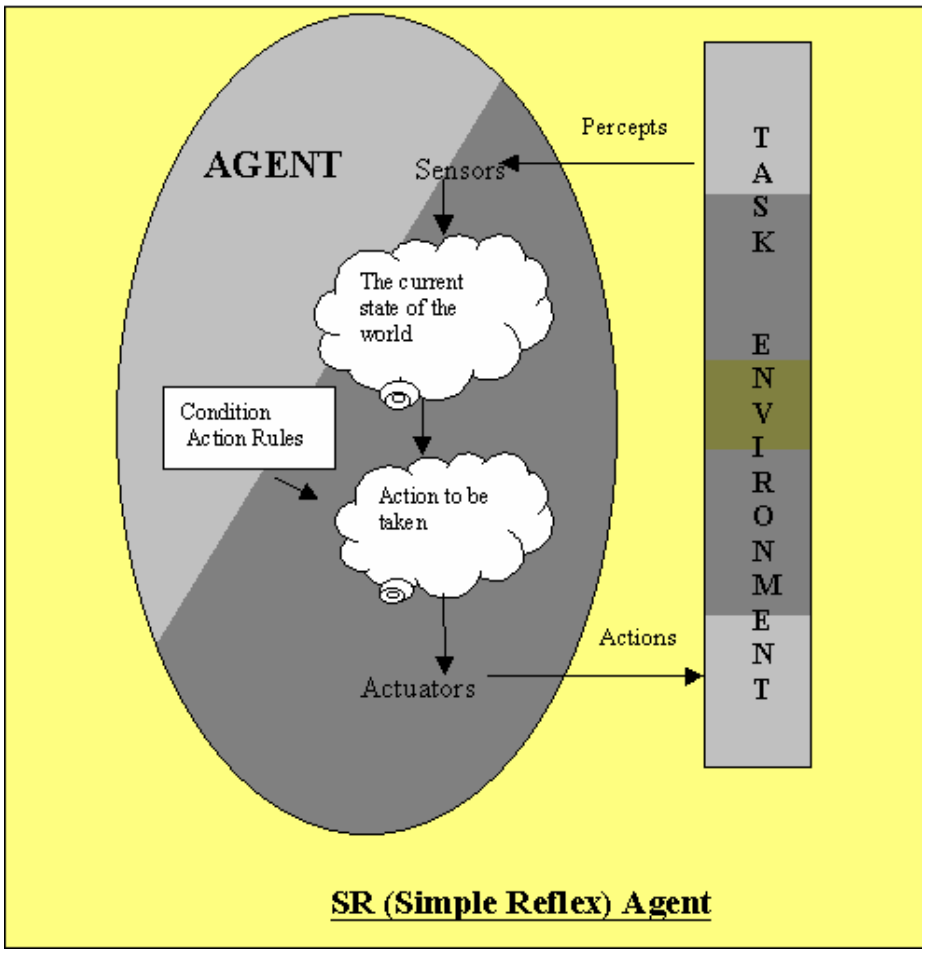


Figure 2.3: SR (Simple Reflex) Agent

2.5.2 Model Based Reflex agents

Simple Reflex agents are not capable of handling task environments that are not fully observable. In order to handle such environments properly, in addition to reflex capabilities, the agent should, maintain some sort of internal state in the form of a function of the sequence of percepts recovered up to the time of action by the agent. Using the percept sequence, the internal state is determined in such a manner that it reflects some of the aspects of the unobservable environment. Further, in order to reflect properly the unobserved environment, the agent is expected to have a model of the task environment encoded in the agent's program, where the model has the knowledge about–

(i) the process by which the task environment evolves independent of the agent and
(ii) effects of the actions of the agent have on the environment.

Thus, in order to handle properly the partial observability of the environment, the agent should have a model of the task environment in addition to reflex capabilities. Such agents are called **Model-based Reflex Agents**.

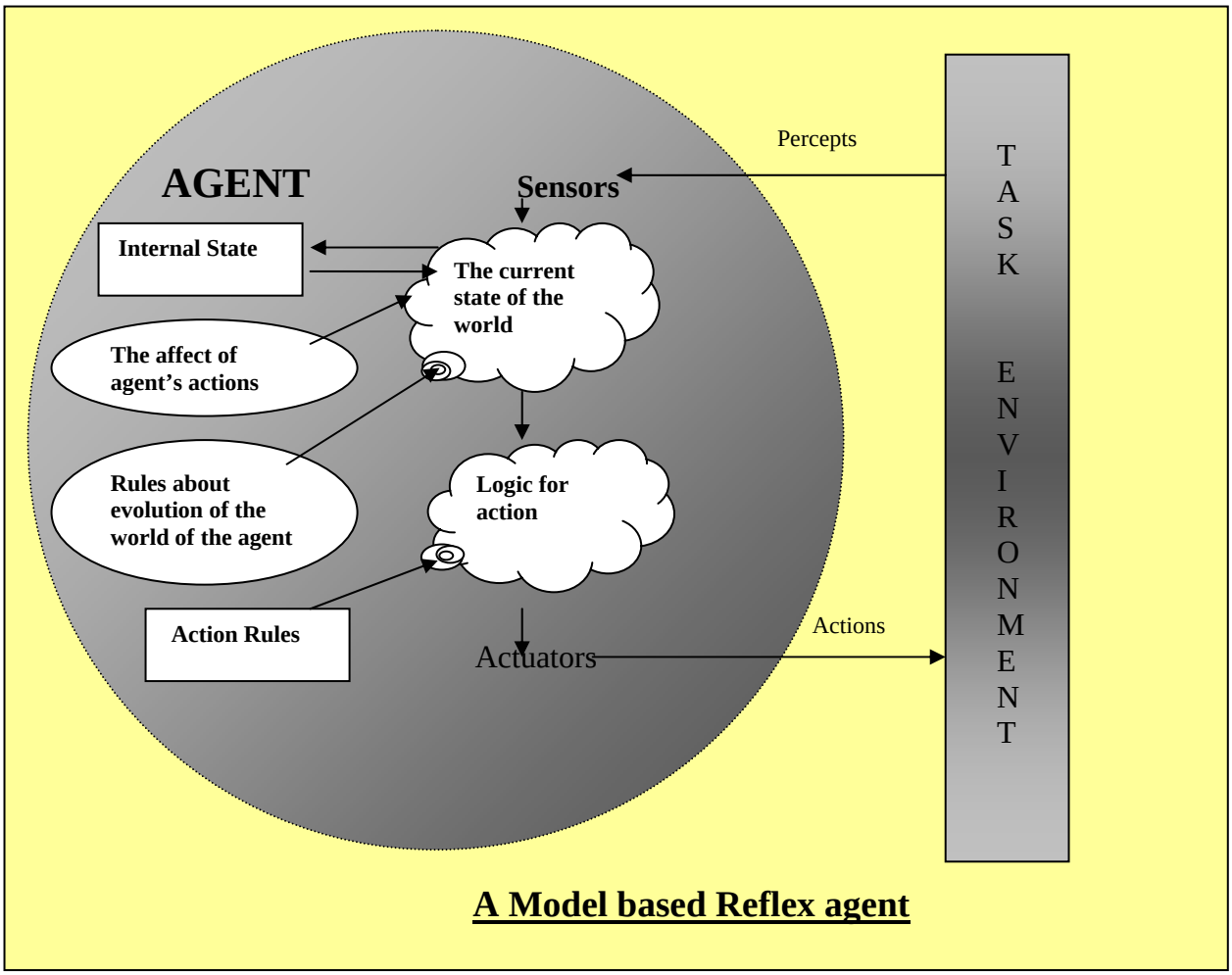


Figure 2.4: A Model based Relfex agent

2.5.3 Goal Based Agents

In order to design appropriate agent for a particular type of task, we know the nature of the task environment plays an important role. Also, it is desirable that the complexity of the agent should be minimum and just sufficient to handle the task in a particular environment. In this regard, first we discussed the simplest type of agents,

viz., Simple Reflex Agents. The action of this type of agent is decided by the current precept only. Next, we discussed the Model-Based Reflex Agents, for which an action is decided by taking into consideration not only the latest precept, but the whole precept history summarized in the form of internal state. Also, action for this type of agent is also decided by taking into consideration the knowledge of the task environment, represented by a model of the environment and encoded into the agent's program. However, in respect of a number of tasks, even this much knowledge may not be sufficient for appropriate action. For example, when we are going from city A to city B, in order to take appropriate action, it is not enough to know the summary of actions and path which has taken us to some city C between A and B. We also have to remember the goal of reaching to city B.

Goal based agents are **driven by the goal** they want to achieve, i.e., **their actions are based on the information regarding their goal, in addition to, of course, other information in the current state.** This goal information is also a part of the current state description and it describes everything that is desirable to achieve the goal. As mentioned earlier, an example of a goal-based agent is an agent that is required to find the path to reach a city. In such a case, if the agent is *an automobile driver agent*, and if the road is splitting ahead into two roads then the agent has to decide which way to go to achieve its goal of reaching its destination. Further, if there is a crossing ahead then the agent has to decide, whether to go straight, to go to the left or to go to the right. In order to achieve its goal, the agent needs some information regarding the goal which describes the desirable events and situations to reach the goal. The agent program would then use this goal information to decide the set of actions to take in order to reach its goal.

Another desirable capability which a good goal based agent should have is that if an agent finds that a part of the sequence of the previous steps has taken the agent away from its goal then it should be able to retract and start its actions from a point which may take the agent toward the goal.

In order to take appropriate action, decision-making process in goal-based agents may be simple or quite complex depending on the problem. Also, **the decision-making required by the agents of this kind needs some sort of looking into the future.** For example, it may analyze the possible outcome of a particular action before it actually performs that action. In other words, we can say that ***the agent would perform some sort of reasoning of if-then-else type***, e.g., an automobile driver agent having one of its goals as not to hit any vehicle in front of it, when finds the vehicle immediately ahead of it slowing down may not apply brakes with full force and in stead may apply brakes slowly so that the vehicles following it may not hit it.

As the goal-based agents may have to reason before they take an action, these agents might be slower than other types of agents but will be more flexible in taking actions as their decisions are based on the acquired knowledge which can be modified also. Hence, **as compared to SR agents** which may require rewriting of all the condition-action rules in case of change in the environment, the goal-based agents can adapt easily when there is any change in its goal.

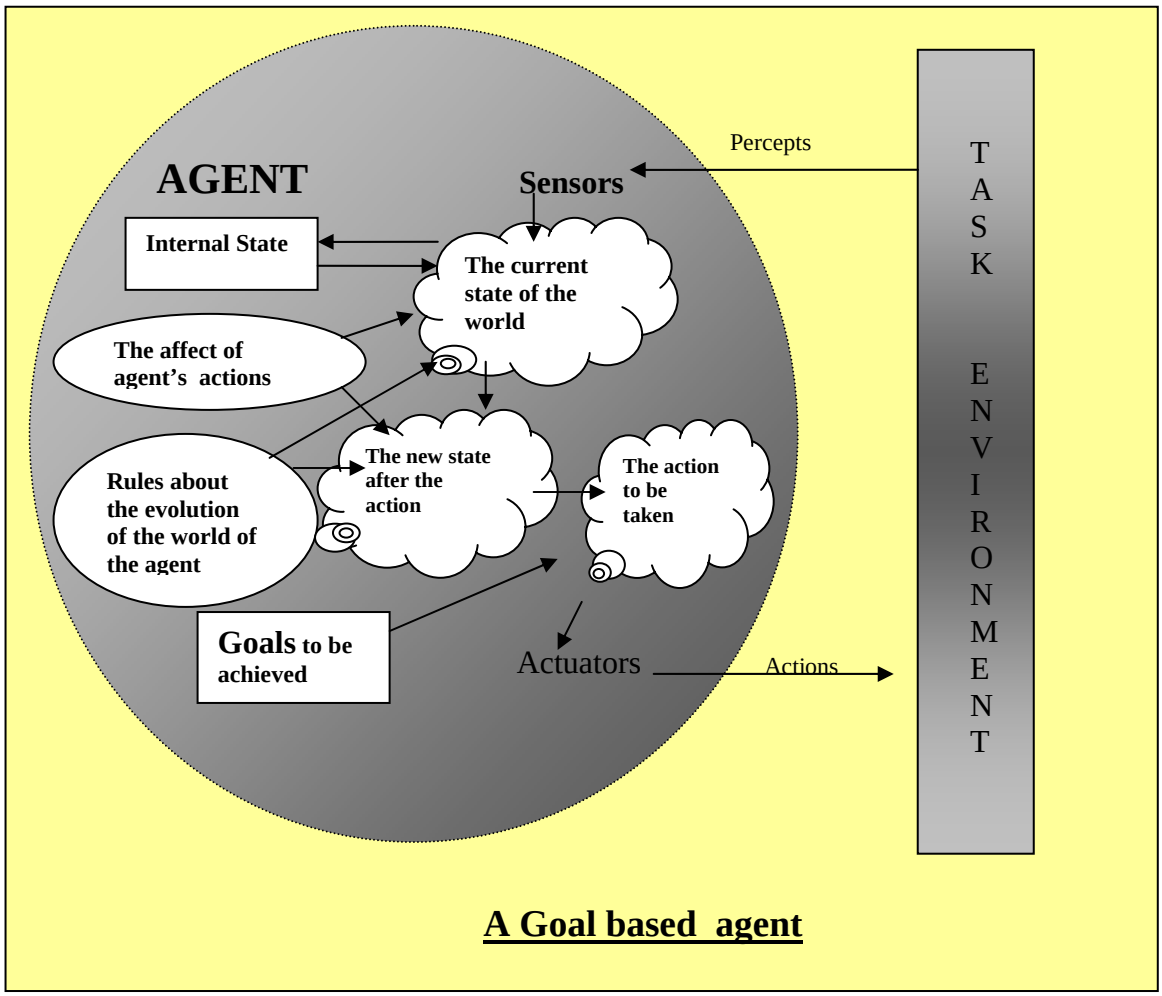


Figure 2.5: A Goal based agent

2.5.4 Utility Based Agents

Goal based agent's success or failure is judged in terms of its capability for achieving or not achieving its goal. A goal-based agent, for a given pair of environment state and possible input, only knows whether the pair will lead to the goal state or not. Such an agent will not be able to decide in which direction to proceed when there are more than one conflicting goals. Also, in a goal-based agent, there is no concept of partial success or somewhat satisfactory success. Further, if there are more than one methods of achieving a goal, then no mechanism is incorporated in a Goal-based agent of choosing or finding the method which is faster and more efficient one, out of the available ones, to reach its goal.

A more general way to judge the success or happiness of an agent may be, through assigning to each state a number as an approximate measure of its success in reaching the goal from the state. In case, the agent is embedded with such a capability of assigning such numbers to states, then it can choose, out of the reachable states in the next move, the state with the highest assigned number, out of the numbers assigned to various reachable states, indicating possibly the best chance of reaching the goal.

It will allow the goal to be achieved more efficiently. Such an agent will be more useful, i.e. will have more utility. A utility-based agent uses a **utility function**, which maps each of the world states of the agent to some degree of success. If it is possible

to define the utility function accurately, then the agent will be able to reach the goal quite efficiently. Also, a utility-based agent is *able to make decisions in case of conflicting goals*, generally choosing the goal with higher success rating or value. Further, in environments with multiple goals, the utility-based agent quite likely chooses the goal with least cost or higher utility goal out of multiple goals.

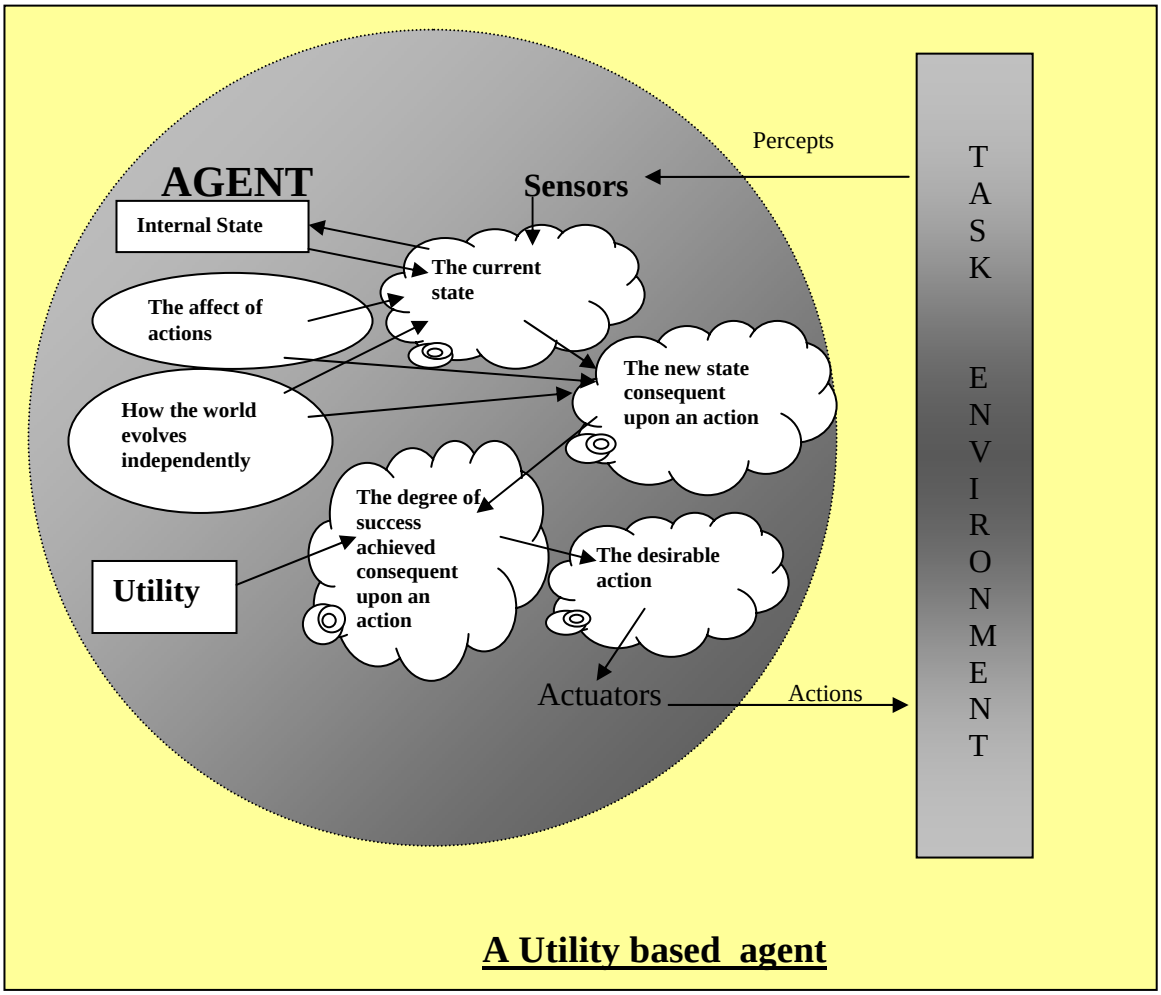


Figure 2.6: A Utility based agent

2.5.5 Learning Agents

It is not possible to encode all the knowledge in advance, required by a rational agent for optimal performance during its lifetime. This is specially true of the real life, and not just theoretical, environments. These environments are **dynamic** in the sense that the environmental conditions change, not only due to the actions of the agents under considerations, but due to other environmental factors also. For example, all of a sudden a pedestrian comes just in front of the moving vehicle, even when there is green signal for the vehicle. In a multi-agent environment, all the possible decisions and actions an agent is required to take, are generally unpredictable in view of the decisions taken and actions performed simultaneously by other agents. Hence, **the ability of an agent to succeed in an uncertain and unknown environment depends on its learning capability** i.e., its capability to change approximately its knowledge of the environment. For an agent with learning capability, some initial knowledge is coded in the agent program and after the agent starts operating, it learns from its actions the evolving environment, the actions of its competitors or adversaries etc. so as to improve its performance in ever-changing environment. If approximate learning component is incorporated in the agent, then the knowledge of the agent gradually

increases after each action starting from its initial knowledge which was manually coded into it at the start.

Conceptually the learning agent consists of four components:

***(i) Learning Component*:** It is the component of the agent, which on the basis of the percepts and the feedback from the environment, gradually improves the performance of the agent.

***(ii) Performance Component*:** It is the component from which all actions originate on the basis of external percepts and the knowledge provided by the learning component.

The design of learning component and the design of performance element are very much related to each other because a learning component is of no use unless the performance component can be designed to convert the newly acquired knowledge into better useful actions.

***(iii) Critic Component*:** This component finds out how well the agent is doing with respect to a certain fixed performance standard and it is also responsible for any future modifications in the performance component. ***The critic is necessary to judge the agent's success with respect to the chosen performance standard, specially in a dynamic environment.* For example**, in order to check whether a certain job is accomplished, the critic will not depend on external percepts only but it will also compare the current state to the state, which indicates the completion of that task.

(iv) Problem Generator Component: This component is responsible for suggesting actions (some of which may not be optimal) in order to gain some fresh and innovative experiences. Thus, ***this component allows the agent to experiment a little*** by traversing sometimes uncharted territories by choosing some new and suboptimal actions. This may be useful, because the actions which may seem suboptimal in a short run, may turn out to be much better in the long run.

In the case of *an automobile driver agent*, this agent would be of little use if it does not have learning capability, as the environment in which it has to operate is totally dynamic and unpredictable in nature. **Once the automobile driver agent starts operating, it keeps on learning from its experiences, both positive and negative**. If faced with a totally new and previously unknown situation, e.g., encountering a vehicle coming from the opposite direction on a one-way road, the problem generator component of the driver agent might suggest some innovative action to tackle this new situation. Moreover, the learning becomes more difficult in the case of an automobile driver agent, because the environment is only partially observable.

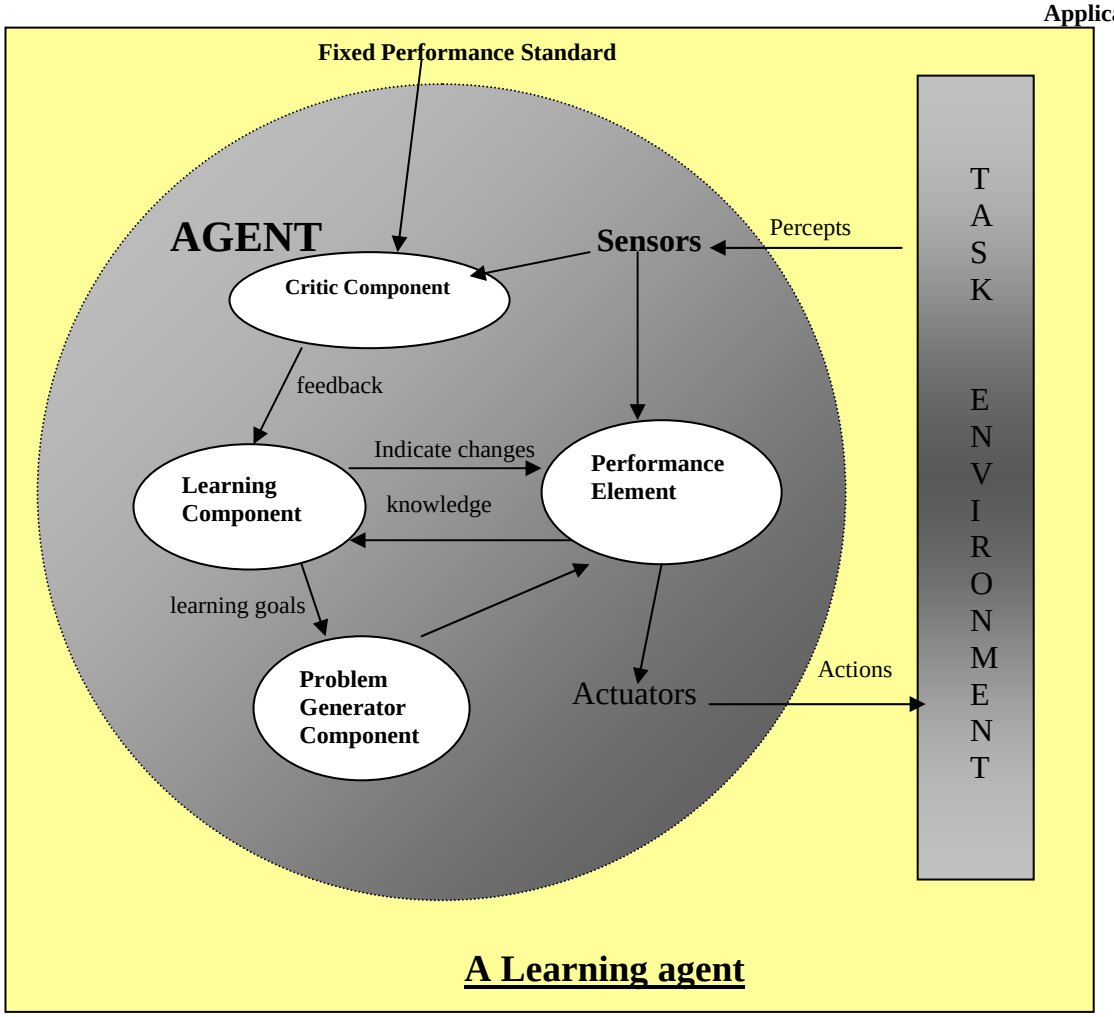


Figure 2.7: A Learning agent

2.6 Different Forms of Learning in Agents

The purpose of embedding learning capability in an agent is that it should not depend totally on the knowledge initially encoded in it and on the external percepts for its actions. The agent learns by evaluating its own decisions and/or making observations of new situations it encounters in the ever-changing environment.

There may be various criteria for developing learning taxonomies. The criteria may be based on –

- The type of knowledge learnt, e.g., concepts, problem-solving or game playing,
- The type of representation used, e.g., predicate calculus, rules or frames,
- The area of application, e.g., medical diagnosis, scheduling or prediction.

We mention below another classification, which is independent of the representation method and knowledge representation used. Under this classification scheme, there are five learning methods:

(i) Rote learning or memorization
(ii) Learning through instruction
(iii) Learning by analogy
(iv) Learning by induction and
(v) Learning by deduction.

Rote learning is the simplest form of learning, which involves least amount of inferencing. In this form of learning, the knowledge is simply copied in the knowledge base. This is the learning, which is involved in memorizing multiplication tables.

Next type of learning is *learning through instruction*. This type of learning involves more inferencing because the knowledge in order to be operational should be integrated in the existing knowledge base. This is the type of learning that is involved in the learning of a pupil from a teacher.

Learning by analogy involves development of new concepts through already known similar concepts. This type of learning is commonly adopted in textbooks, where some example problems are solved in the text and then exercises based on these solved examples are given to students to solve. Also, this type of learning is involved when, on the basis of experience of driving light vehicles, one attempts to drive heavy vehicles.

Learning by induction is the most frequently used form of learning employed by human being. This is a form of learning which involves inductive reasoning — a form of reasoning under which a conclusion is drawn on the basis of a large number of positive examples. For example, after seeing a large number of cows, we conclude a cow has four legs, white colour, two horns symmetrically placed on the head etc. Inductive reasoning, though usually leads to correct conclusions, yet the conclusions may not be *irrefutable*.

For instance, in the case of the concept of cow as discussed above, we may find a black cow, or we may find a three-legged cow who has lost one leg in an accident or a single-horn cow.

Finally, we discuss *deductive learning*, which is based on deductive inference — an *irrefutable* form of reasoning. By irrefutable form of reasoning we mean that the conclusion arrived at through deductive (i.e., any irrefutable) reasoning process is always correct, if the hypotheses (or given facts) are correct. This is the form of reasoning which is dominantly applied in Mathematics.

Inductive learning occupies an important place in designing the learning component of an agent. The learning in an agent is based on–

- The subject matter of learning, e.g., concepts, problem-solving or game playing etc.
- The representation used, predicate calculus, frame or script etc.
- The critic, which gives the feedback about the overall health of the agent.

Learning based on feedback is normally categorized as:

- Supervised learning
- Unsupervised learning
- Reinforcement Learning.

Supervised Learning: *It involves a learning function from the given examples of its inputs and outputs.* Some of the examples of this type of learning are:

- Generating useful properties of the world around from the percepts
- Mapping from conditions regarding the current state to actions
- Gathering information about the way world evolves.

Unsupervised Learning: In this type of learning, the pair of inputs and corresponding expected outputs are not available. Hence, the learning system, on its own, has to find

relevant properties from the otherwise unknown objects. For example, finding shortest path, without any prior knowledge, between two cities in a totally unknown country.

Reinforcement (Rewards) Learning: *In many problems, the task or the problem is only realized, but cannot be stated*. Further, the task may be an ongoing one. The user expresses his or her satisfaction or dissatisfaction regarding the performance of the agent by occasionally giving the agent positive or negative rewards (i.e., reinforcements). ***The job of the agent is to maximize the amount of reward (i.e., reinforcement) it receives***. In case of a simple goal achieving problem, the agent can be rewarded positively when it achieves the goal and punished every time it fails to do so.

In this kind of task environment, an action policy is needed to maximize reward. But *sometimes in case of ongoing and non-terminating tasks the future reward might be infinite, so it is difficult to decide how to maximize it.* In such a scenario, a method of progressing ahead is to discount future rewards beyond a certain factor. i.e., the agent may prefer rewards in the immediate future to those in the distant future.

Learning action policies in environments in which rewards depend on a sequence of earlier actions is called **delayed-reinforcement learning**.

2.7 SUMMARY

In this unit, the concept of 'Intelligent Agent' is introduced and further various issues about intelligent agents are discussed. Some definitions in this context are given in section 2.2. The next section discusses the concept of rationality in context of agents. Section 2.4 discusses various types of task environments for agents. Next section discusses structure of an agent. Finally, section 2.6 discusses various forms of learning in an agent.

2.8 SOLUTIONS/ANSWERS

Ex. 1) **For L_1**

The left upper cell is unoccupied, therefore, $C_1=0$, $C_2=0=C_3=C_4$ and $C_8=0$
However, as C_5, C_6 and C_7 do not exist hence are assumed to be occupied.
Thus, the sense-vector is (0, 0, 0, 0, 1, 1, 1, 0)

For L_2

$C_1=1=C_8$ and $C_2=C_3=C_4=C_5=C_6=C_7=0$
Thus, the sense-vector is (1, 0, 0, 0, 0, 0, 0, 1)

2.9 FURTHER READING

1. Russell S. & Norvig P, *Artificial Intelligence: A Modern Approach* (Second Edition), (Pearson Education, 2003).