gradeup

Multi Agent Systems

Expert system

Content:

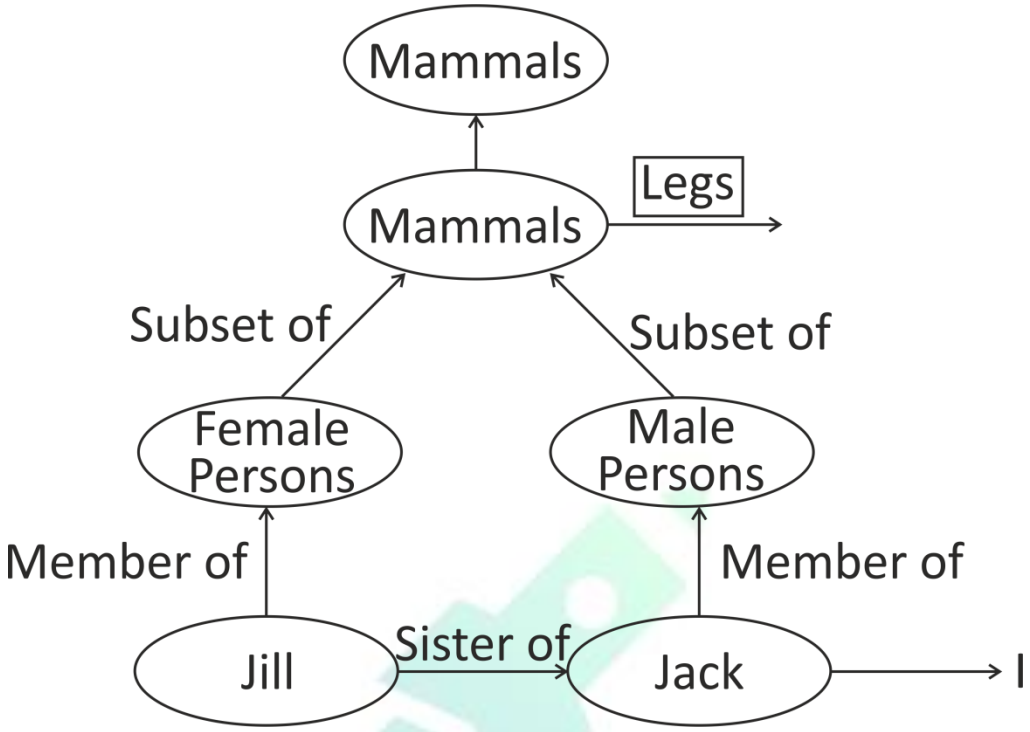
1. **Various Knowledge Representation Schemes**
2. **Forward and Backward Chaining Rule Based Systems**
3. **Expert System Shell**
4. **Examples of Expert System**
5. **Intelligent Agents**
6. **Rational Agent and Rationality**
7. **Different types of environments**
8. **Applications**
9. **Types of Agent**
10. **Learning Of Agents**

There are various Knowledge Representation Schemes:

1. **Semantic Network**: Semantic net (for semantic network is a knowledge representation technique used for propositional information. Therefore, it is also called a propositional net. Semantic nets convey meaning Semantic nets are two dimensional representations of knowledge. Mathematically a semantic net can be defined as a labeled directed graph.

   Semantic nets consist of nodes, links (edges) and link labels. In the semantic network diagram, nodes appear as circles or ellipses or rectangles to represent objects such as physical objects, concepts or situations. Links appear as arrows to express the relationships between objects, and link labels specify particular relations. Relationships provide the basic structure for organizing knowledge.

   The objects and relations involved need not be so concrete. As nodes are associated with other nodes semantic nets are also referred to as associative nets.



all the objects are within ovals and connected using labelled arcs. There is a link between Jill and Female Persons with label Member Of. Similarly there is a Member Of link between Jack and Male Persons and Sister of link between Jill and jack. The member of link between Jill and female persons indicates that Jill belongs to the category of female persons.

Disadvantage of semantic nets: One of the drawbacks of semantic network is that the links between the objects represent only binary relations For example, the sentence Run (Chennai Express, Chennai, Bangalore,Today) cannot be asserted directly.

Advantages of semantic nets

a) Semantic nets have the ability to represent default values for categories In fig (6.1) Jack has one leg while he is a person and all persons have two legs.

So persons have two legs has only default status which can be overridden by a specific value.

b) Semantic nets convey some meaning in a transparent manner.

c) Semantic nets are simple and easy to understand.

d) Semantic nets are easy to translate into PROLOG.

As information in semantic network is clustered together through relational links the knowledge required for the performance of some task is generally available within short spatial span of the semantic network. This type of knowledge organization in some way, resembles the way knowledge is stored and retrieved by human beings.

2. **Frames**: Frames are a variant of semantic networks that are one of the popular ways of representing non-procedural knowledge in an expert system inn frame, all the information relevant to a particular concept is stored in e singe complex entity, called a frame. Frames look like, the data structure record. Frames support inheritance. They are often used to about typical objects or events, such as a car, or even a mathematical object like rectangle. A frame is a structured object and different names like Schema, Script Prototype, and even Object are used instead of frame, in computer science literature.

YEAR : 2011
MONTH:
DAY:

- The is a relation is in fact the subset relation.
- The instance relation is in fact element of.
- The is a attribute possesses a transitivity property, This implies: Robert- howely is a Back and a Back la a Rugby Player who in tum is an Adult-mile and a Person.

But in case of multiple inheritance, i.e. in case of an object having more than one parent class, we have to decide which parent to inherit from. For example, a lion may inherit from "wild animals" or "circus animals". In general, both the slots and slot values may themselves be frames and so on.

Frame systems are pretty complex and sophisticated knowledge representation tools. This representation has become so popular that special high level frame based representation languages have been developed. Most of these languages use LISP as the host language. It is also possible to represent frame-like structures using object oriented programming languages, extensions to the programming language LISP.

3. **Proposition and predicate logic:**Predicate logic builds heavily upon the ideas of proposition logic to provide a more powerful system for expression and reasoning. A predicate is just a function with a range of two values, say false and true. We use predicates routinely in programminge.g. in conditional statements of the form

   if(p(…args..))

   Here we are using the two possibilities for the return value of p. (true or false). We also use the propositional operators to combine predicates, such as in:

   if((p(....) && (!q(...) !! r (....)))

   Predicate logic deals with the combination of predicates using the propositional operators. It adds one more interesting element, the "quantifiers".

   The meaning of predicate logic expressions is suggested by the following:

   Expression + Interpretation + Assignment = Truth Value

Now we explain this equation.

An interpretation for a predicate logic expression consists of:

- a domain for each variable in the expression
- à predicate for each predicate symbol in the expression
- a function for each function symbol in the expression

The propositional operators are not counted as function symbols in the case of predicate logic, even though they represent functions. The reason for this is that we do not wish to subject them to interpretations other function.However, we distinguish them in predicate logic so as to separate predicates, which have truth values used by propositional operators, from functions that operate on arbitrary domains. Propositional Logic is concerned with propositions and their inter-relationships. The notion of proposition here cannot be define precisely.

First Order Predicate Logic (FOPL) is an extension of propositional logic, which was developed to extend the expressiveness of propositional logic. In addition to just propositions of propositional logic, the predicate logic uses predicates, functions, and variables together with variable quantifiers (Universal and Existential quantifiers) to express knowledge.

The inference rules of PL including Modus Ponens, Chain Rule and Rule of Transposition are valid in FOPL also after suitable modifications by which formulae of PL are replaced by formulae of FOPL.

In addition to these inference rules, the following four inference rules of FOPL, that will be called Q_r, Q_2, Q_3 andQ_4have no corresponding rules in PL. In the following F denotes a predicate and x a variable/parameter:

$$Q_1 : \frac{\sim(\exists x)F(x)}{(\forall x)\sim F(x)} \text{ and } \frac{(\forall x)\sim F(x)}{\sim(\exists x)f(x)}$$

The first of the above rules under Q_r says :

From negation of there exists x F(x), we can infer for all x not of F(x)

$Q_1 : \frac{\sim(\forall x)F(x)}{(\exists x)\sim F(x)}$ and $\frac{(\exists x)\sim F(x)}{\sim(\forall x)f(x)}$

The first of the above rules under Q_r says :

From negation of for all x F(x), we can infer there exists x such that not of F(x)

$Q_3 : \frac{(\forall x)F(x)}{F(a)}$, where a is (any) arbitrary element of the domain of F

The rule Q_3 is called universal instantiation. The rule is called universal generation

$Q_3 : \text{and} \frac{F(a) \text{for arbirary a}}{(\forall x)f(x)}$

The rule is called universal generation

$Q_4 : \frac{(\exists x)F(x)}{F(a)}$, where a is a particular (not arbitrary) constant.

This rule is also called existential instantiation :

$Q_4 : \frac{F(a) \text{for some a}}{(\exists x)F(x)}$

The rule is called existential generalization.

Steps for using Predicate Calculus as a Language for Representing Knowledge

Step 1: Conceptualization: All the relevant entities and the relations that exist between these entities are explicitly enumerated. Some of the implicit facts like 'a person dead once is dead forever' have to be explicated.

Step 2: Nomenclature and Translation: Giving appropriate names to the objects and relations. And then translate the given sentences in English to formulae in FOPL.

Appropriate names are essential in order to guide a reasoning system based on FOPL. It is well-established that no reasoning system is complete. In other words, a reasoning system may need help in arriving at desired conclusion.

4. **Route Based Systems:** Rather than representing Knowledge in a declarative and somewhat static way (as a set of statements, each of which is true), rule-based systems rep sent knowledge in terms of a set of rules each of which specifies the conclusion that could be reached or derived under given conditions or in different situations. A rule-based system consists of

a. Rule base, which is a set of IF-THEN rules,

b. A bunch of facts, and

c. Some interpreter of the facts and rules which is a mechanism which decides which rule to apply based on the set of available facts. The interpreter also initiates the action suggested by the rule selected for application:

A Rule-base may be of the form:

R : If A is an animal and A barks, than A is a dog

Fi: Rocky is an animal

F2: Rocky Barks

F3: rocky is a dog.

There are two broad kinds of rule-based systems:

Forward chaining systems and backward chaining systems.

In a forward chaining system we start with the initial facts, and keep using the rules to draw new intermediate conclusions (or take certain actions) given those facts. The process terminates when the final conclusion is established. In a backward chaining system, we star with some goal statements, which are intended to be established and keep looking for rules that would allow us to conclude, setting new sub-goals in the process of reaching the ultimate goal.

Backward Chaining System: This approach is most useful when we know all the material facts, but do not have much idea what the conclusion might be.

If we DO know, what the conclusion might be, or have some specific hypothesis to test forward chaining systems may be inefficient. We COULD keep on forward chaining until no more rules apply or we have added our hypothesis to the working memory. However, in the process the system is likely to do a lot of irrelevant work, adding uninteresting conclusions to working memory.

This can be done by backward chaining from the goal state (or on some hypothesized state that we are interested in). This is essentially, what Prolog does, so it should be familiar to you by now. Given a goal state to try to prove (e.g., (bad-mood Alison)) the system will first check to see if the goal matches the initial facts given. If it does, then that goal succeeds. If it does not the system will look for rules whose conclusions (previously referred to as actions)-match the goal. One such rule will chosen, and the system will then try to prove any facts in the preconditions of the rule using the same procedure, setting these as new goals to prove. Note that a backward chaining system does NOT need to update a working memory.

Instead, it needs to keep track of what goals it needs to prove to prove its main hypothesis.

In principle, we can use the same set of rules for both forward and backward chaining. However, in practice we may choose to write the rules slightly differently if we are going to be using them for backward chaining. In backward chaining, we are concerned with matching the conclusion of a rule against some goal that we are trying to prove. So the then part of the rule is usually not expressed as an action to take (e.g., add/delete), but as a state which will be true if the premises are true.

So, suppose we have the following rules:

7. IF (lecturing X)

AND (marking-practical's X)

THEN (overworked X)

8. IF (month february)

THEN (lecturing alison)

9. IF (month february)

THEN (marking-practicalsalison)

10. IF (overworked X)

THEN (bad-mood X)

11. IF (slept-badly X)

THEN (bad-mood X)

12. IF (month february)

THEN (weather cold)

13. IF (year 1993)

THEN (economy bad) and initial Facts:

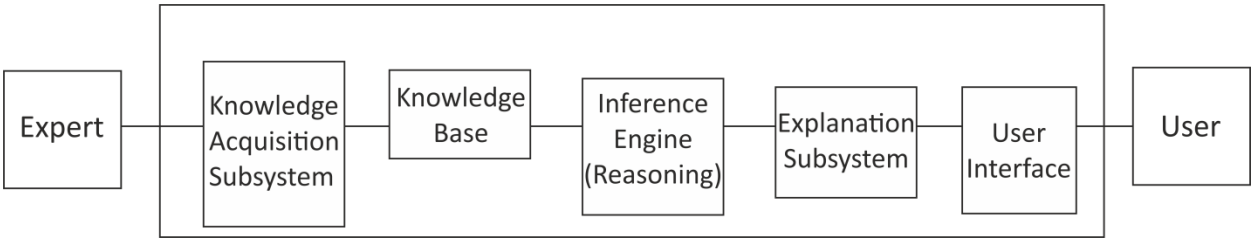
(monthfebruary)

(year-1993)

and we're trying to prove:

(bad-moodalison)

First we check whether the goal state is in the initial facts. As it isn't there, we try matching it against the conclusions of the rules. It matches rules 4 and 5. Let us assume the rule 4 is chosen first - it will try to prove (overworked alison) Rule 1 can be used, and the system will try to prove (lecturing alison) and (marking practicalsalison). Trying to prove the first goal, it will mat rule 2 and try to prove (month february). This is in the set of initial facts. We still have prove (marking practicalsalison). Rule 3 can be used, and we have proved the original goal (b mood alison).



the shell includes the inference engine, a knowledge acquisition subsystem, an explanation subsystem and a user interface. When faced with a new problem in any given domain, we can find a shell which can provide the right support for that problem, so all we need is the knowledge of an expert. There are many commercial expert system shells available now, each one adequate for a different range of problems. Taking help of expert system shells to develop expert systems greatly reduces the cost and the time of development as compared to developing the expert system from the scratch.

The components of a generic expert system shell are below:

- Knowledge Base
- Knowledge Acquisition
- Inference Engine
- Explanation Subsystem
- User Interface

Knowledge Base: It contains facts and heuristic knowledge. Developers try to use a uniform representation of knowledge as far as possible. There are many knowledge representation schemes for expressing knowledge about the application domain and some advance expert system shells use both frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

Knowledge Acquisition Sub system: The process of capturing and transformation of potentially useful information for a problem from any knowledge source (which may be a human expert) to a program in the format required by that program is the job of a knowledge acquisition subsystem. So we can say that these subsystem to help experts build knowledge bases.

As an expert may not be computer literate, so capturing information includes interviewing preparing questionnaires etc. which is a very slow and time consuming process. So collecting knowledge needed to solve problems and build the knowledge base has always been the biggest bottleneck in developing expert systems.

Some of the reasons behind the difficulty in collecting information are given below

- The facts and rules or principles of different domains cannot easily Be converted into a mathematical or a deterministic model, the properties. of which are known. For example a teacher knows how to motivate the students but putting down on the paper, the exact reasons, causes and other factor affecting students is not easy as they vary with individual students.
- Different domains have their own terminology and it is very difficult for experts to communicate exactly their knowledge in a normal language.
- Capturing the facts and principles alone is not sufficient to solve problems. For example experts In particular domains which information is important for specific judgments, which information sources are reliable and how problems can be simplified, which is based on personal experience. Capturing such knowledge is very difficult.
- Commonsense knowledge found In humans continues to be very difficult to capture, although systems are being made to capture it.

Knowledge acquisition in COMPASS: COMPASS is an expert system which was build for proper maintenance and troubleshooting of telephone company's switches.

For knowledge acquisition, knowledge from a human expert is elicited. An expert explains the problem solving technique and a knowledge engineers then converts it into and if-then-rule. The human expert then checks if the rule has the correct logic and if a change is needed then the knowledge engineer reformulates the rule.

Sometimes, it is easier to troubleshoot the rules with pencil and paper i.e., hand simulation), at least the first time than directly implementing the rule and changing them again and again.

Inference Engine: An inference engine is used to perform reasoning with both the expert knowledge which is extracted from an expert and most commonly a human expert) and data which is specific to the problem being solved. Expert knowledge is mostly in the form of a set of IF-THEN rules. The case specific data includes the data provided by the user and also partial conclusions (along with their certainty factors) based on this data. In a normal forward chaining rule-based system, the case specific data is the elements in the working memory.

Developing expert systems involve knowing how knowledge is accessed and used during the search for a solution. Knowledge about what is known and, when and how to use it is commonly called meta-knowledge. In solving problems, a certain level of planning, scheduling and controlling is required regarding what questions to be asked and when, what is to be checked and so on.

Different strategies for using domain-specific knowledge have great effects on the performance characteristics of programs, and also on theway in which a program finds or searches a solution among possible alternatives Most knowledge representation schemes are used under a variety reasoning methods and research going on in this area.

Explanation Subsystem (Example MYCIN): An explanation sub system allows the program to explain its reasoning to the use. The explanation can range from how the final or intermediate solutions were arrived at to justifying the need for additional data.

Explanation subsystems are important from the following points of view:

1. Proper use of knowledge: There must be some for the satisfaction of knowledge is applied properly even at the time of development of a prototype.
2. Correctness of conclusions: user's need to satisfy themselves that the conclusions produced by the system are correct.
3. Knowledge trace: in order to judge that the knowledge elicitation is proceeding smoothly and successfully, a complete trace of program execution is required.

4. Suitability of reasoning approach: explanation sub systems are necessary to ensure that reasoning technique applied is suitable to the particular domain.

Explanation in expert systems deals with the issue of control because the reasoning steps used by the programs will depend on how it searches for a solution.

Explanation subsystems are also related to evaluation as by checking the outputs produced by a system and after examining the trace of its actions performed while reasoning, it can be decided that whether or not a system k producing the right answer for right reasons. Without a good explanation subsystem, an expert will be unable to judge the system's performance or will be unable to find ways to improve it.

User interface: It is used to communicate with the user. The user interface is generally not a part of the expert system technology, and was not given much attention in the past. However, it is now widely accepted that the user interface can make a critical difference in the utility of a system regardless of the system's performance.

MYCIN (An expert system):MYCIN, one of the earliest designed expert systems in Stanford University in 1970s.MYCIN job was to diagnose and recommend treatment for certain blood infections. To do the proper diagnosis, it is required to grow cultures of the infecting organism which is a very time consuming process and patient is in critical state. So, doctors have to come up with quick guesses about likely problems from the available data, and use these guesses to provide a treatment where drugs are given which should deal with any type of problem.

MYCIN represented its knowledge as a set of IF-THEN rules with certainty factors. One of MYCIN rule could be like:

IF infection is primary-bacteremia AND the site of the culture is one of the sterile sites AND the suspected portal of entry is the gastrointestinal tract.THEN there is suggestive evidence (0.8) that bacteroid infection occurred.

The 0.8 is the certainty that the conclusion will be true given the evidence. If the evidence is uncertain the certainties of the pieces of evidence will be combined with the certainty of the rule to give the certainty of the conclusion.

MYCIN has been written in LISP, and its rules are formally represented as LISP expressions. The action part of the rule could just be a conclusion about the problem being solved, or it could be another LISP MYCIN is a goal-directed system, using the backward chaining However, to increase it reasoning power and efficiency MYCIN also uses various heuristics to control the search for a solution.

One of the strategies used by MYCIN is to first ask the user a of predefined questions that are most common and which allow the system to rule out totally unlikely diagnoses. Once these questions have been asked, the system can then focus on particular and more specific possible blood disorders. It then uses backward chaining approach to try and prove each one. This strategy a lot of unnecessary search, and is similar to the way doctor tries to diagnose a patient.

MYCIN has been popular in expert system's research, but it also had a number of problems or shortcomings because of which a number of its derivatives like NEOMYCIN developed.

COMPASS (An expert system):COMPASS is an expert system which checks error messages derived from the switch's self testroutines, look for open circuits, reduces the time of operation of components etc. To find the cause of a switch problem, it looks at a series of such messages and then uses it expertise. The system can suggest the running of additional tests, or the replacement of a particular component, for example, a relay or printed circuit board.

As expertise in this area was scarce, so it was a fit case for taking help an expert system like COMPASS.

DENDRAL: DENDRAL was developed at-Standard University in the late 1960. It was designed to analyze mass spectra. Based on the mass of fragments seen in the spectra, it would be possible to make inferences as to the nature of molecule tested,

identifying functional groups or even the entire molecule. In the hab this can be a very difficult process. DENDRAL did contain rules, but it worked differently from most expert systems. DENDRAL used heuristic knowledge obtained from experienced chemists to help constrain the problem and thereby reduce the search space. During tests, DENDRAL discovered a no. of structures previously unknown to climate experts.

DENDRAL has the following organizational features:

1. **Knowledge representation:** Production rules and algorithms the generating graph structures are constructed by META-DENDRAL, METADENDRAL is a program which was learning techniques to construct rules for an expert systems automatically.
2. **Reasoning:**Forward chaining (Data driver).
3. **Heuristic:** DENDRAL uses a variation of depth first search called generate and test where all hypothesis are generated and tested and their tested against the available evidence.
4. **Dialogue/explanation:** The dialogue uses mixed control. Theuser can supply info and the system can request into as required.

EMYCIN provides a domain-independent framework or template for constructing and running any consultation programs. EMYCIN stands for "Empty MYCIN" or "Essential MYCIN" because it basically constitutes a MYCIN system minus its domain-specific medical knowledge. However, EMYCIN is something more than this, as it offers a number of software ex tools for helping expert system designers in building and debugging performance programs.

Characteristics of EMYCIN are:

• It constitutes an abbreviated rule language, which uses ALGOL-like notation and which is easier than LISP and is more concise than the English subset used by MYCIN.

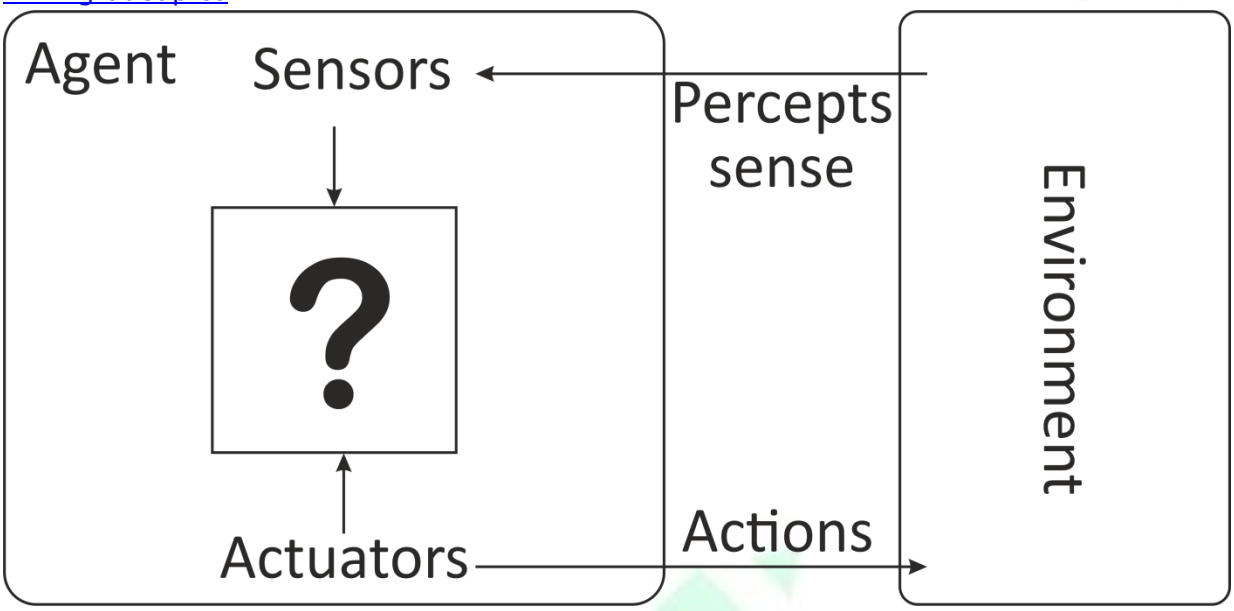
• It uses backward chaining which is similar to MYCIN.

• It indexes rules, in-turn Organising them into groups, based on the parameters which are being referenced.

• It also has a user interface which allows the user to communicate with the system smoothly.

Intelligent agents

An agent is anything that can be viewed as perceiving its environment through sensor and acting upon that environment through actuators. This simple idea is illustrated in Fig. 9.1. A human agent has eyes, ears and other organs for sensors and hands, legs, mouth and other body parts 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 will make the general assumption that every agent can perceive its own action (but not always the effects). We use the term percept to refer to the agent's perceptual inputs at any giver 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. If we can specify the agent's choice of action for every possible percept sequence, then we have said more or less everything there is to say about the agent. Mathematically speaking, we say that an agent's behaviour is describe by the agent function that maps any given percept sequence to an action.



We can imagine tabulating the agent function that describes any given agent; for most agents, this would be a very large table-infinite, in fact, unless we place a bound on the length of percept sequences we want to consider. Given an agent to experiment with, we can, in principle, construct this table by trying out all possible percept sequences, and recording which actions the agent does in response. The table is, of course, an external characterization of the agent. 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 program is a concrete implementation running on the agent architecture.

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:

- It has autonomous control, i.e. it operates under its own control.
- It is perceptive i.e., it is capable of perceiving its own environment.
- It persists over a long period of time.

- It is adaptive to changes in the environment, and
- 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.

The definition of agent according to first approach is, 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 outits 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:** The 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.

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.

The 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.

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.

Rationality depends on:

- The performance measure chosen to judge the agent's activities.
- The agent's knowledge of the current environment, or the world in which 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.

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 near 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.

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 1.e, what will be the output or outcome of its action Obviously being omniscient is next to impossible.

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 cannot be said to be irrational.

Autonomy Capability Learning Capability

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 it it is capable offering from its experience and has hot 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 agentis 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 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.

kinds of Task Environments an agent

A task environment is a theoretical construct that is supposed to supply the goal relevant consequences of allowable moves in performing the task. Moves or actions considered allowable, if they can possibly advance an agent closer to his or her goal. People really operate in much long the broader activity spaces than task environments. We can pretend that they can view as role players, where their only concerns are those to do with performance of their task narrowly conceived. However, this is an idealization that eliminates from consideration many of the surprising ways people have of using the environment to help them control activity.

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 wholesome idea is happening around before taking any action. The agent in a fully observable environment will have a completed idea of the world around it at all times, and hence, it needed not maintain an internals state to remember and store what is going around it. Fully observable environment are found rarely in reality.
2. **Static Dynamic Environment**: it 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 it. So, the time taken by an agent to perform its on the environment because the environment is static. The task environment of the “boundary following robot" is because changes are possible in the environment around it, irrespective of how long the robot might 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 continuously.

3. **Episodic vs. Sequential Environment**: In an episodic environment, be agent's experience is divided into episodes. An agents actions during an episode depend only on the 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 affect 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.

4. **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 agents 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 stochastic environment.

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, le, to perform a collective action. Such a type of environment is called cooperative multi-agent environment.

6. **Discrete vs. Continuous Environment**: The word discrete and continuous and related to the way time is treated, Whether lime is divided into slots (l., discrete quantities), or It in treated aux 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 the 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 reserved values are treated as a stream of continuous data.

Some examples of task environment are:

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. In this case, we should 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.

Crossword Puzzle: 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, 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.

Automobile Driver Agent: The one subclass of a general driver agent, 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.

This is one of the most complex environments, 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.

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.

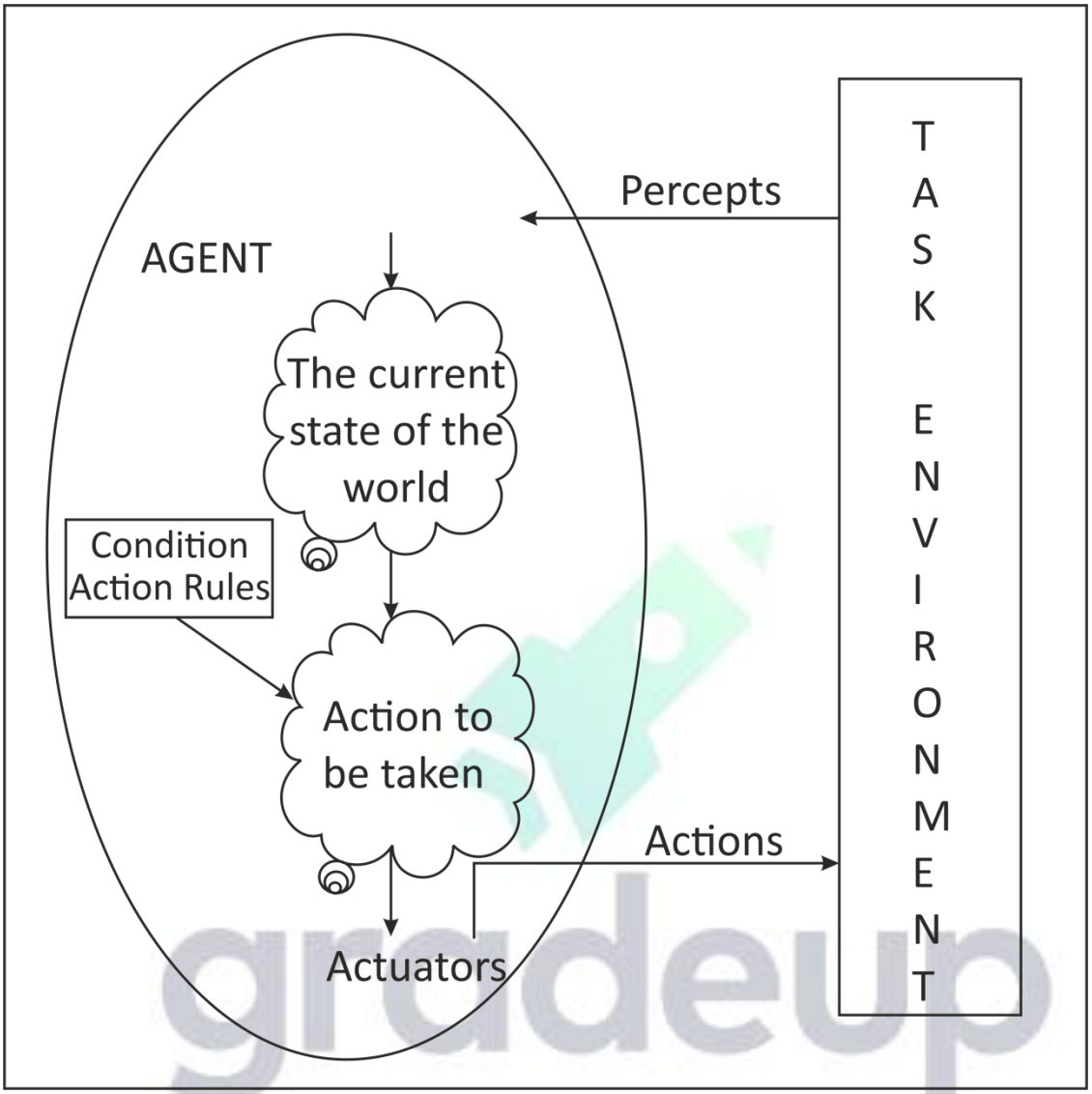
AI agent is a system which when given a goal to be achieved, could carry out the details of the appropriate (computer) operations and further, in case it gets stuck, it can ask for advice and can receive it from humans, may eyen evaluate the appropriateness of the advice and then act suitably.

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

- It has autonomous control, i.e., it operates under its own control,
- It is perceptive, i.e. it is capable of perceiving its own environment.
- It is adaptive to changes in the environment, and
- It is capable of taking over others goals.

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. Correct inferencing is one of the several possible mechanisms for achieving rationality.

Simple Reflex (SR) Agents: These are the agents or machines that have no internal state (i.e., do not 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 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.

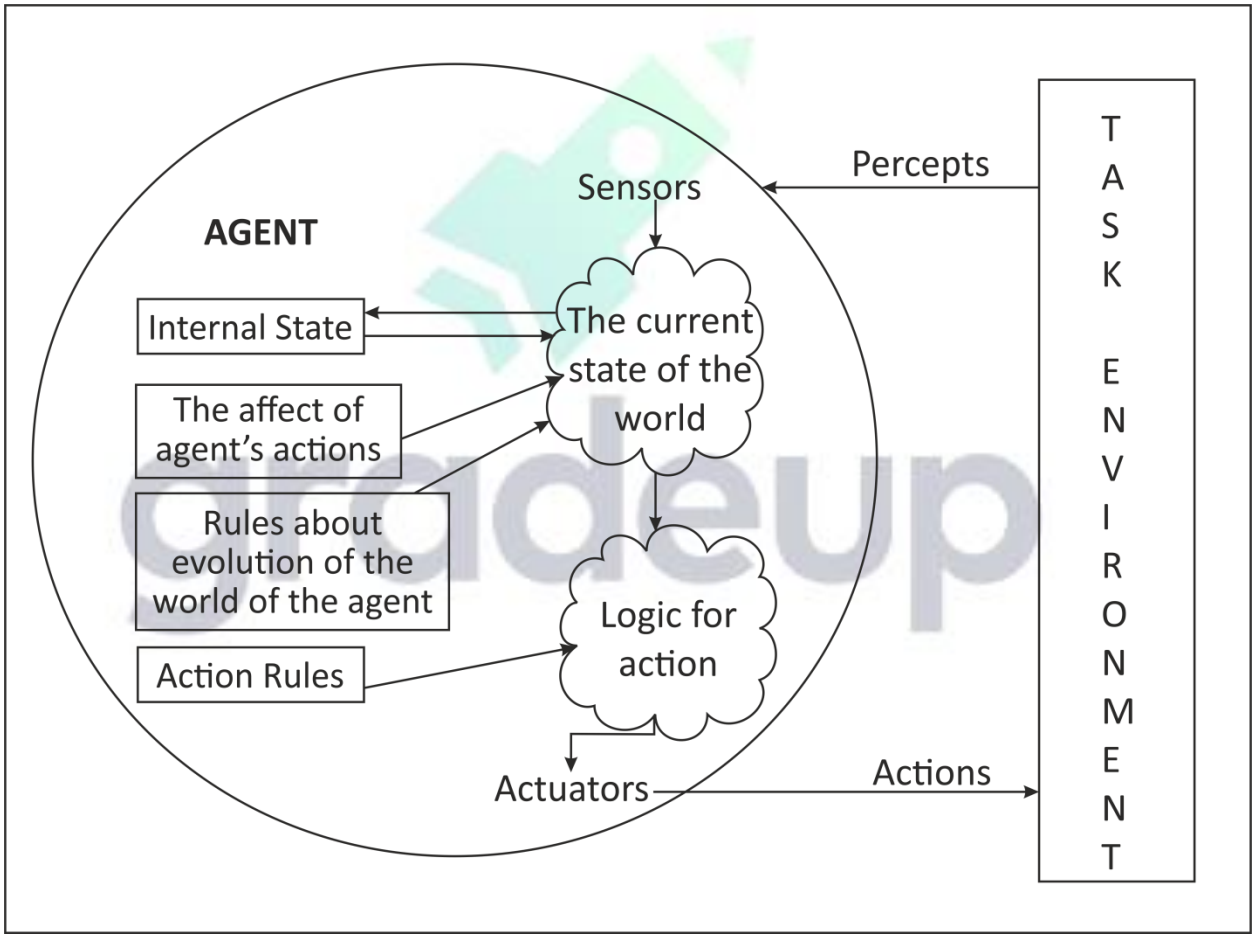


SR (Simple Reflex) Agent

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:

1. the process by which the task environment evolves independent of the agent and
2. 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.



A model Based reflex agent

Goal Based Agents: 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

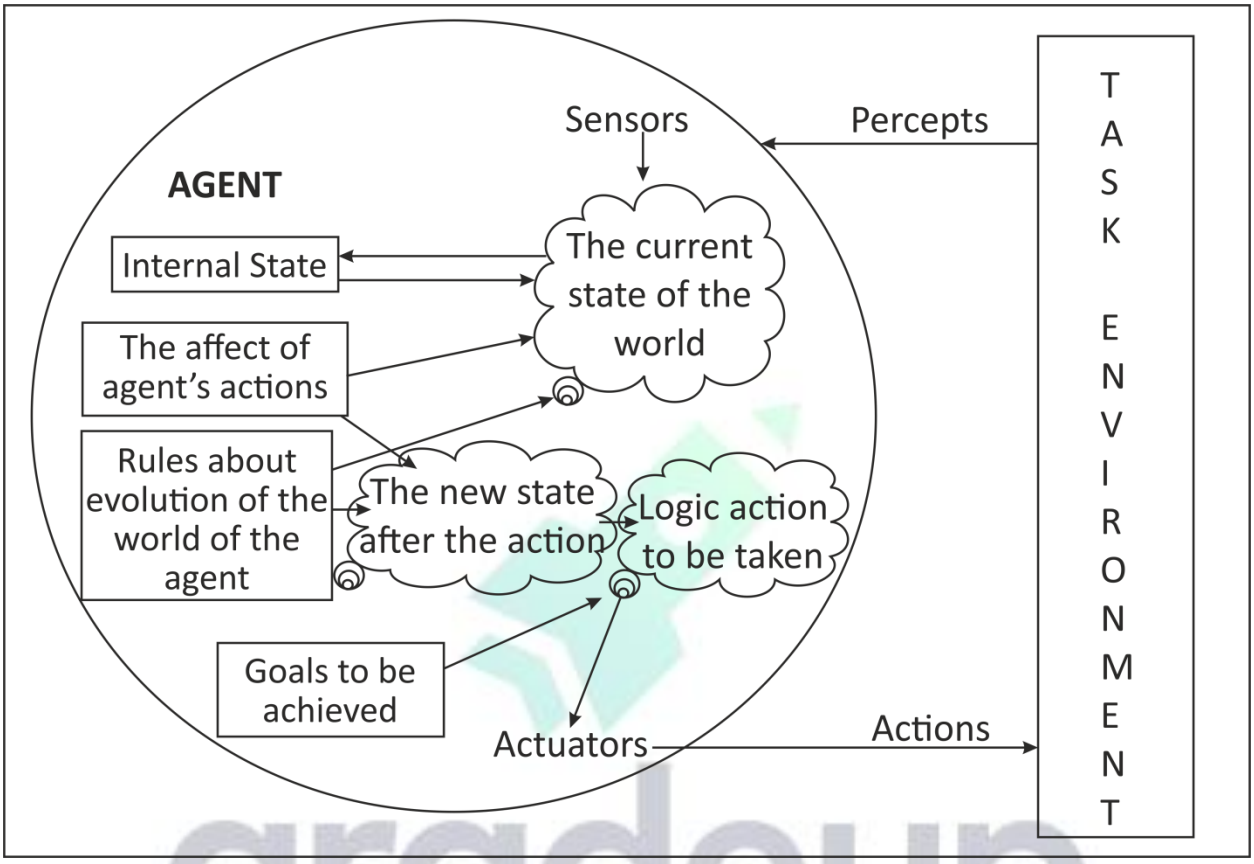
the current state description, and it describes everything that is desirable to achieve the goal. An example of a goal-based agent is an agent, that is required to find the path to reach a city. In such a case, if the agent is an automobile driver agent, and if the road is splitting ahead into two roads, 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 towards 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 instead 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 canbe 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.

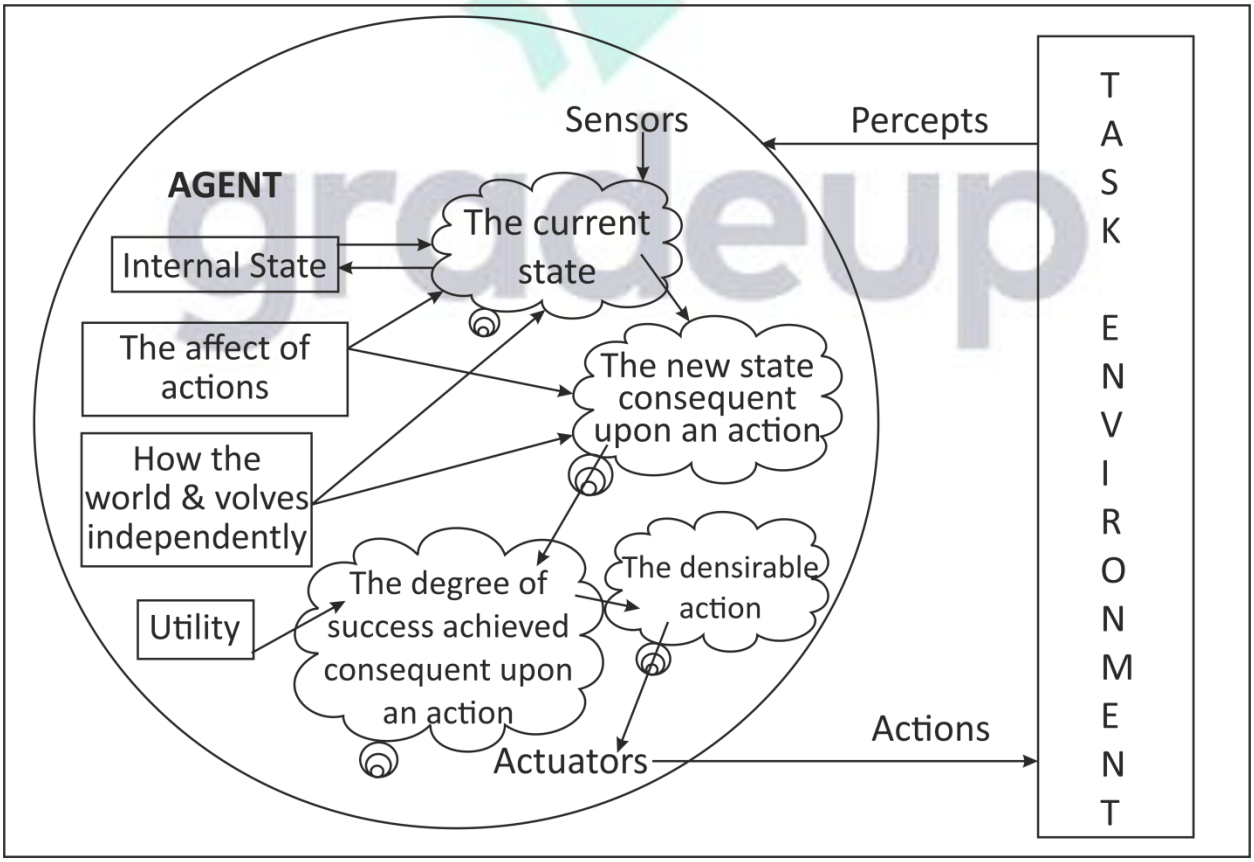


A Global Based Agent

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 if an agent may be through assigning to each

state a number a n 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 goat 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.



A Utility Based Agent

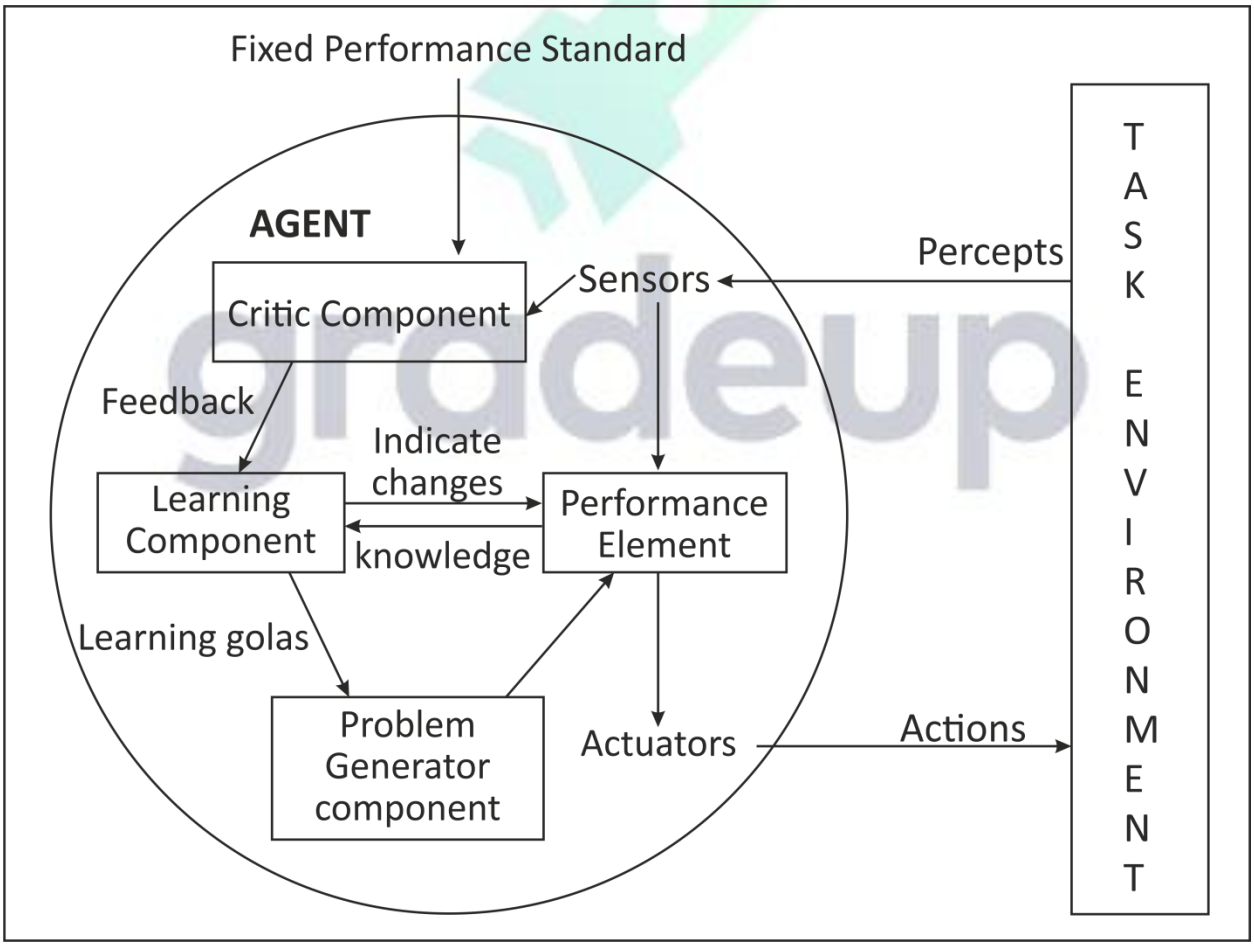
Learning Agents: It is not possible to encode all the knowledge in Advance, required by a rational agent for optimal performance during its lifetime. This is specially true of the real life, and not just theoretical, environments. These environments are dynamic in the sense that the environmental conditions change, hot 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:

- **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.
- **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:
- **Critic Component:** This component finds out how well the agent is doing with respect to a certain fixed performance standard, and it also responsible for any future modifications in the performance components. 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 task.

- **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.



There are five forms of learning:

- Rote learning or memorization
- Learning through instruction
- Learning by analogy
- Learning by deduction and
- Learning by induction.

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.

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 fearing 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.

Deductive learning which is based on deductive inference - an inevitable 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.

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 color, two horns symmetrically placed on the head etc.

Inductive reasoning, though, usually leads to correct conclusions, vet the conclusions may not be irrefutable.

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.

Various types of agents

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 (1.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.

Example An Automated Public Road Transport Driver Agent): 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 t0 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 agentis 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 ie, 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 (6PS) 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.

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co