

**UNIT-I**

**Artificial Intelligence:**

Artificial Intelligence is composed of two words. i.e. Artificial and Intelligence. Where Artificial define as a “**manmade**” and intelligence defines “**Thinking power**”. AI means a manmade thinking power.

**Definition:**

Artificial Intelligence is the study of man-made computational devices and systems which can be made to act in a manner which we would be inclined to call intelligent.

**OR**

"It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions".

**Q). AI Tasks:**

General Tasks	Formal Tasks	Expert Tasks
<b>Perception -</b>	<b>Games</b>	<b>Engineering</b>
Vision and Speech	Chess, Backgammon, Checkers	Design, Fault finding, Manufacturing Planning
<b>Natural Language -</b>	<b>Mathematics</b>	<b>Scientific Analysis</b>
Understanding, Generation, Translation	Geometry, Logic, Integral Calculus, Proving Properties of Program	<b>Medical Diagnosis</b>
<b>Common Sense Reasoning and Robot Control</b>		<b>Financial Analysis</b>

**Q). Applications of AI**

**Gaming:** AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens' problems and etc.

**Natural Language Processing:** Interact with the computer that understands natural language spoken by humans.

**Expert Systems:** Machine or software provide explanation and advice to the users.

**Vision Systems :** Systems understand, explain, and describe visual input on the computer.

**Speech Recognition:** There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it.

For example: Google assistant.

**Handwriting Recognition:** The Handwriting Recognition software reads the text written on paper and recognizes the shape of letters and convert it into editable text.

### **Problems of AI:**

Intelligence does not imply perfect understanding; every intelligent being has limited perception, memory and computation. Many points on the spectrum of intelligence versus cost are viable, from insects to humans. AI seeks to understand the computations required from intelligent behaviour and to produce computer systems that exhibit intelligence. Aspects of intelligence studied by AI include perception, communication using human languages, reasoning, planning, learning and memory.

The following questions are to be considered before we can step forward:

1. What are the underlying assumptions about intelligence?
2. What kinds of techniques will be useful for solving AI problems?
3. At what level human intelligence can be modelled?
4. When will it be realized when an intelligent program has been built?

### **Q). what is AI Technique:**

AI problems span a very broad spectrum. They appear to have very little in common except that they are hard.

One of the hard and results to come out of the first three decades of AI research is that intelligence requires knowledge. Knowledge processes some less, desirable properties including.

- It is Voluminous
- It is hard to characterize accurately
- It is constantly changing
- It differs from data by being organized in a way that corresponds to the ways it will be used.

An AI Technique is a method that exploits knowledge that should be represented in such a way that.

- ❖ The knowledge captures generalization.
- ❖ It can be understood by people who must provide although for many AI domains.
- ❖ It can easily be modified to correct errors and to reflect changes in the world and in our world view.
- ❖ It can be used in a great many situation even if it is not totally accurate or complete.

**LEVEL OF THE AI MODEL**

‘What is our goal in trying to produce programs that do the intelligent things that people do?’

Are we trying to produce programs that do the tasks the same way that people do?

OR

Are we trying to produce programs that simply do the tasks the easiest way that is possible?

Programs in the first class attempt to solve problems that a computer can easily solve and do not usually use AI techniques. AI techniques usually include a search, as no direct method is available, the use of knowledge about the objects involved in the problem area and abstraction on which allows an element of pruning to occur, and to enable a solution to be found in real time; otherwise, the data could explode in size. Examples of these trivial problems in the first class, which are now of interest only to psychologists are EPAM (Elementary Perceiver and Memorizer) which memorized garbage syllables.

The second class of problems attempts to solve problems that are non-trivial for a computer and use AI techniques. We wish to model human performance on these:

1. To test psychological theories of human performance. Ex. PARRY [Colby, 1975] – a program to simulate the conversational behavior of a paranoid person.
2. To enable computers to understand human reasoning – for example, programs that answer questions based upon newspaper articles indicating human behavior.
3. To enable people to understand computer reasoning. Some people are reluctant to accept computer results unless they understand the mechanisms involved in arriving at the results.
4. To exploit the knowledge gained by people who are best at gathering information. This persuaded the earlier workers to simulate human behavior in the SB part of AISB simulated behavior. Examples of this type of approach led to GPS (General Problem Solver).

### 1.5 CRITERIA FOR SUCCESS

One of the most important questions to answer in any scientific or engineering research project is “How will we know if we have succeeded?” Artificial intelligence is no exception. How will we know if we have constructed a machine that is intelligent? That question is at least as hard as the unanswerable question “What is intelligence?” But can we do anything to measure our progress?

In 1950, Alan Turing proposed the following method for determining whether a machine can think. His method has since become known as the *Turing Test*. To conduct this test, we need two people and the machine to be evaluated. One person plays the role of the interrogator, who is in a separate room from the computer and the other person. The interrogator can ask questions of either the person or the computer by typing questions and receiving typed responses. However, the interrogator knows them only as A and B and aims to determine which is the person and which is the machine. The goal of the machine is to fool the interrogator into believing that it is the person. If the machine succeeds at this, then we will conclude that the machine can think. The machine is allowed to do whatever it can to fool the interrogator. So, for example, if asked the question “How much is 12,324 times 73,981?” it could wait several minutes and then respond with the wrong answer [Turing, 1963].

The more serious issue, though, is the amount of knowledge that a machine would need to pass the Turing test. Turing gives the following example of the sort of dialogue a machine would have to be capable of:

Interrogator:	In the first line of your sonnet which reads “Shall I compare thee to a summer’s day,” would not “a spring day” do as well or better?
A:	It wouldn’t scan.
Interrogator:	How about “a winter’s day.” That would scan all right.
A:	Yes, but nobody wants to be compared to a winter’s day.
Interrogator:	Would you say Mr. Pickwick reminded you of Christmas?
A:	In a way.

For many everyday tasks, though, it may be even harder to measure a program’s performance. Suppose, for example, we ask a program to paraphrase a newspaper story. For problems such as this, the best test is usually just whether the program responded in a way that a person could have.

If our goal in writing a program is to simulate human performance at a task, then the measure of success is the extent to which the program’s behavior corresponds to that performance, as measured by various kinds of experiments and protocol analyses. In this we do not simply want a program that does as well as possible. We want one that fails when people do. Various techniques developed by psychologists for comparing individuals and for testing models can be used to do this analysis.

#### Q). Advantages of Artificial Intelligence:

- ❖ **High Accuracy with less errors:** AI machines or system are less errors and high accuracy as it takes decisions as per pre –experience or information
- ❖ **High Speed:** AI system can be of very high speed and fast decision making , because of that AI systems can beat a chess champion in the chess game.
- ❖ **High Reliability:** AI machines are highly reliable and can perform the same action multiple time with high accuracy.
- ❖ **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploiting the ocean float. Where to employ a human can be risky.
- ❖ **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-Commerce website to show the products as per customer requirement.



- ❖ **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle free, facial recognition for security purpose, natural language processing to communicate with the human in human language, etc.

### Problems, Problem Spaces, and search

#### Problems, Problem Spaces, and search:

##### Introduction:

To build a system to solve a particular problem, we need to do four things:

1. **Define the problem precisely:** This definition must include precise specification of what the initial situations will be as well as what final situations constitute acceptable solutions to the problem.
2. **Analyze the problem:** a few very important features can have and have an immense impact on the appropriateness of various possible techniques for solving the problem.
3. **Isolate and represent** convert these important features into knowledge representation.
4. **Choose the best problem-solving technique(s):** choose the best technique and apply it to particular problem.

#### Q). Defining the problem as a state space search:

The problems can be solved by defining the problem as a state space search. for example, consider the following water jug problem, and it can be solved by using the production rules.

##### Water jug Problem:

you are given two jugs, a 4 -gallon one and a 3- gallon one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug?

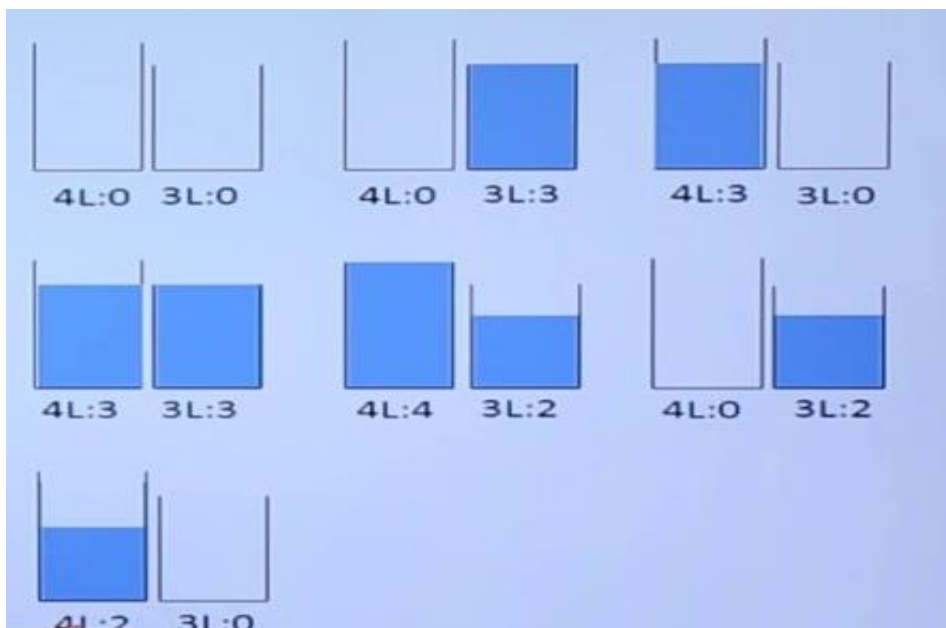
##### Production rules for the water jug problem:

1	$\{x,y\}$ is $X < 4 \rightarrow \{4,Y\}$	Fill the 4-gallon jug
2	$\{x,y\}$ if $Y < 3 \rightarrow \{x,3\}$	Fill the 3-gallon jug
3	$\{x,y\}$ if $x > 0 \rightarrow \{x-d,Y\}$	Pour some water out of the 4-gallon jug.
4	$\{x,y\}$ if $Y > 0 \rightarrow \{x,Y-d\}$	Pour some water out of 3-gallon jug.
5	$\{x,y\}$ if $x > 0 \rightarrow \{0,y\}$	Empty the 4-gallon jug on the ground
6	$\{x,y\}$ if $y > 0 \rightarrow \{x,0\}$	Empty the 3-gallon jug on the ground
7	$\{x,y\}$ if $X+Y \geq 4$ and $y > 0 \rightarrow \{4,y-(4-x)\}$	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8	$\{x,y\}$ if $X+Y \geq 3$ and $x > 0 \rightarrow \{x-(3-y),3\}$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
9	$\{x,y\}$ if $X+Y \leq 4$ and $y > 0 \rightarrow \{x+y,0\}$	Pour all the water from the 3-gallon jug into the 4-gallon jug.
10	$\{x,y\}$ if $X+Y \leq 3$ and $x > 0 \rightarrow \{0,x+y\}$	Pour all the water from the 4-gallon jug into the 3-gallon jug.
11	$\{0,2\} \rightarrow \{2,0\}$	Pour the 2-gallon water from 3-gallon jug into the 4-gallon jug.
12	$\{2,Y\} \rightarrow \{0,y\}$	Empty the 2-gallon in the 4-gallon jug on the ground.

**Solution to the water jug problem:**

Gallons in the 4-gallon jug	Gallons in the 3-gallon jug	Rule Applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

One solution to the water jug problem.

**Graphical Representation:****Q). Production Systems**

A Production System consists of:

- ❖ A set of rules, each consisting of left side a pattern that determines the applicability of the rule and right side that describes the operation to be performed if the rule is applied.
- ❖ One or more Knowledge/ database that contain whatever information is appropriate for the particular task.
- ❖ A control strategy that specifies the order in which the rules will be compared to the database and a way of resolving the conflicts that arise when several rules match at once.

❖ A rule applier.



### Control Strategies:

TO decide which rule to apply next during the process of search for a solution to a problem, the control strategies are applied. The control strategies have the following requirements.

✓ The first requirement of a good control strategy is that it cause motion.

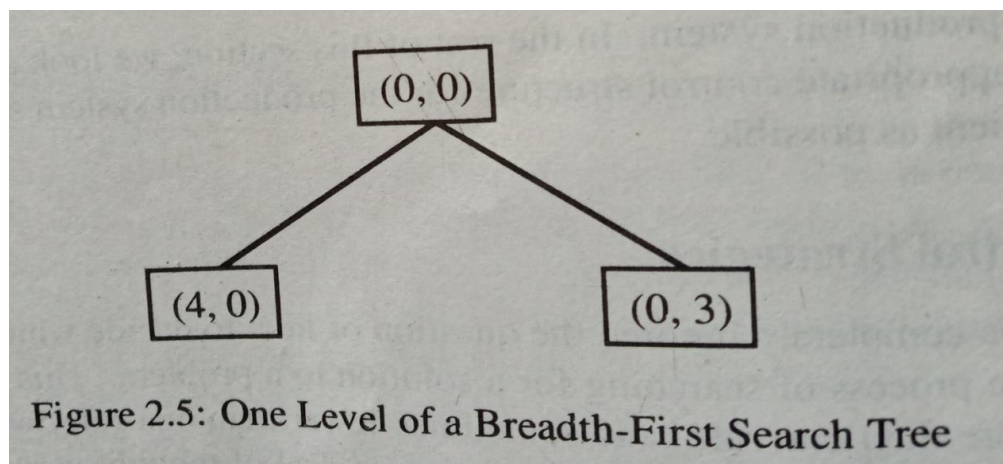
For example : consider the water jug problem ,the simple control strategy of starting each time at the top of the list of rules and choosing the first applicable one.

✓ The second requirement of a good control strategy is that it be systematic.

One systematic control strategy for the water jug problem is the Following .

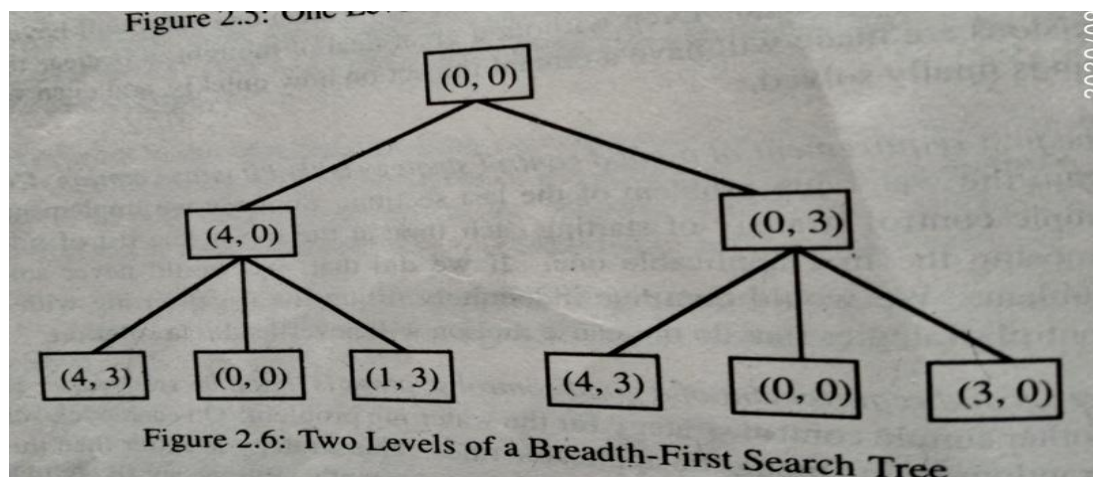
Construct a tree with the initial state as its root. Generate all the offspring of the root by applying each of the applicable for each

initial state as a Figure 2.5.



For each leaf node, generate all its successors by applying each of the Rules that are appropriate as figure 2.6. Continue this process until some rule produces a goal state. This process, called

### Breadth-First Search(BFS).



**For example:**

control strategy for solving water jug problem is **Breadth-First –Search(BFS)** Technique and algorithm is given below.

**Algorithm for BFS:**

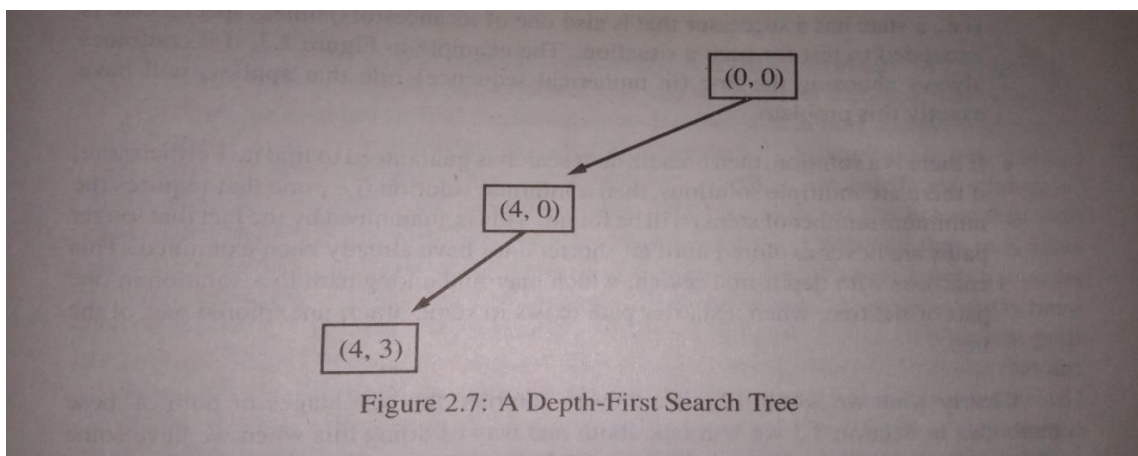
1. Create a variable called NODE-LIST and set it to the initial state
2. Until a goal state is found or NODE-LIST is empty do:
  - a) Remove the first element from NODE-LIST and call it E. if NODE-LIST was empty, quit.
  - b) For each that each rule can match the state described in E do:
    - I. Apply the rule to generate a new state.
    - II. If the new state is a goal state, quit and return this state.
    - III. Otherwise, add the new state to the end of NODE-LIST.

Another control strategy for solving water jug problem is **Depth-FirstSearch(DFS)** Technique and algorithm is given below.

**Algorithm for DFS:**

1. If the initial state is a goal state, quit and return success.
2. Other wise, do the following until success or failure is signaled:
  - a) Generate a successor, E of the initial state. If there are no more successors, signal failure.
  - b) Call DFS with E as the initial state.
  - c) If success is returns, signal success. Other wise continue in this loop.

Figure 2.7 shows a snapshot of a DFS for the water jug problem.





### Q). Advantage and Disadvantage of Production Systems

#### Advantages of Production System:

Some of the **advantages** of Production system in artificial intelligence are:

- ❖ Provides **excellent tools** for structuring AI programs
- ❖ The system is highly **modular** because individual rules can be added, removed or modified independently.
- ❖ Separation of **knowledge** and **Control-Recognises Act Cycle**
- ❖ A natural **mapping** onto **state-space research** data or goal-driven
- ❖ The system uses pattern directed control which is more **flexible** than algorithmic control.
- ❖ Provides opportunities for **heuristic control** of the search.
- ❖ A good way to model the **state-driven nature** of intelligent machines
- ❖ Quite helpful in a **real-time** environment and applications.

#### Disadvantages of Production Systems:

- ✓ It is very **difficult** to analyze the flow of control within a production system
- ✓ It describes the operations that can be performed in a search for a solution to the problem.
- ✓ There is an **absence of learning** due to a rule-based production system that does not store the result of the problem for future use.

### Q). Production System Characteristics:

- A set of production rules, which are of the form  $A \rightarrow B$ . Each rule consists of left hand side constituent that represent the current problem state and a right-hand side that represent an output state. A rule is applicable if its left-hand side matches with the current problem state.
- A database, which contains all the appropriate information for the particular task. Some part of the database may be permanent while some part of this may pertain only to the solution of the current problem.
- A control strategy that specifies order in which the rules will be compared to the database of rules and a way of resolving the conflicts that arise when several rules match simultaneously.
- A rule applier, which checks the capability of rule by matching the content state with the left hand side of the rule and finds the appropriate rule from database of rules.

The important roles played by production systems include a powerful knowledge representation scheme. A production system not only represents knowledge but also action.

Production system provides a language in which the representation of expert knowledge is very natural. We can represent knowledge in a production system as a set of rules of the form

### **If (condition) THEN (condition)**

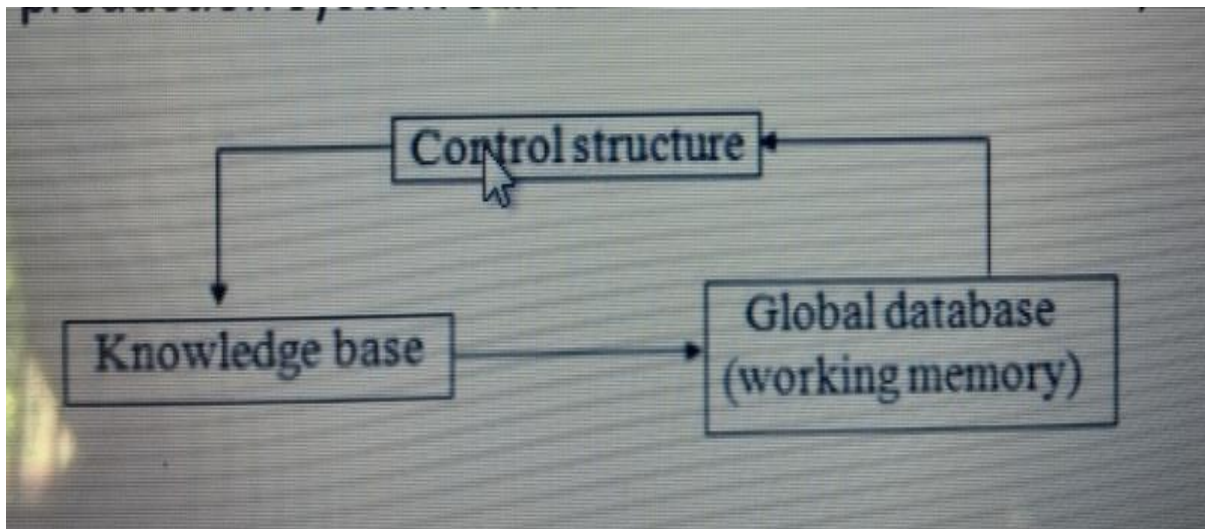
along with a control system and a database. The control system serves as a rule interpreter and sequencer. The database acts as a context buffer, which records the conditions evaluated by the rules and information on which the rules act. The production rules are also known as condition – action, feedback – result pairs.

### **For example:**

If (you have an exam tomorrow)

THEN (study the whole night)

The production system can be classified as monotonic, non-monotonic, partially commutative and commutative.



### **Q. PROBLEM CHARACTERSTICS**

In order to choose the most appropriate method for a particular problem, it is necessary to analyze the problem along several key dimensions:

- ❖ Is the problem decomposable into a set of independent smaller or easier subproblems?
- ❖ Can solution steps be ignored or at least undone if they prove unwise?
- ❖ the problem universally predictable?
- ❖ Is a good solution to the problem obvious without comparison to all the possible solutions?
- ❖ Is the desire solution a state of world or a path to a state?
- ❖ Is a large amount of knowledge absolutely required to solve the problem?
- ❖ Will the solution of the problem require interaction between the computer and the person?

The above characteristics of a problem are called as 7-problem characteristics under which the solution must take place

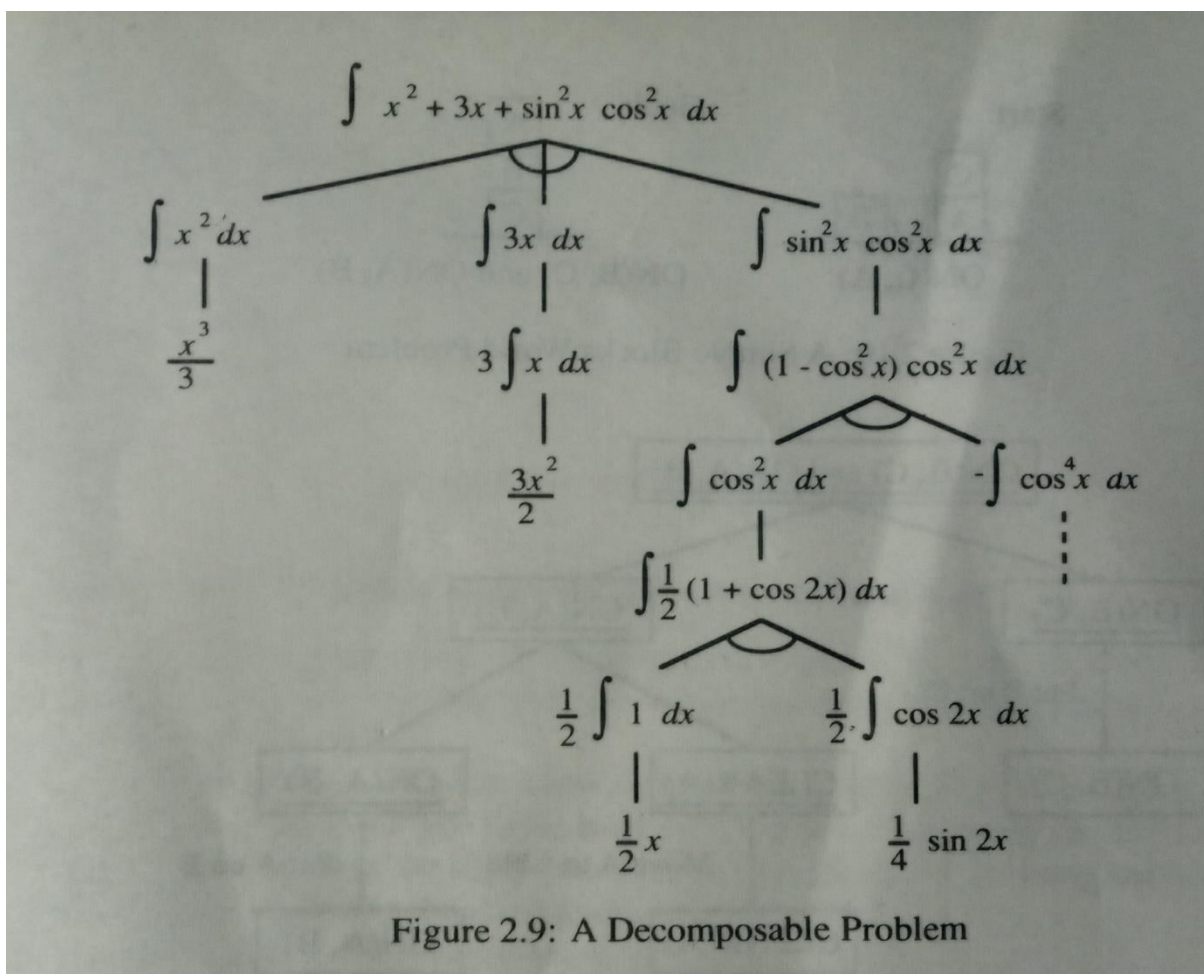
### Is the problem decomposable?

If the given problem is decomposed into small collection, then we can easily solve that problem.

For example, consider the following problem.

$$\int (x^2 + 3x + \sin^2 x \cdot \cos^2 x) dx$$

The above problem tree that will be generated by the process of problem decomposition as it can be exploited by a simple recursive integration program that works as follows.



### Can solution steps ignored or undone?

To solve a particular problem sometimes solution steps be ignored or undone. There are three different classes of problems are used to get the solution. They are:

- **Ignorable**(eg: theorem proving), in which solution steps can be ignored.
- **Recoverable**(eg: 8-Puzzle)in which solution steps can be undone.

➤ Irrecoverable(eg :chess) in which solution steps can be undone

### Is the Universe Predictable?

To solve a particular problem, we have to consider whether the solution of the problem is universe predictable or particular individual only.

### Is a Good Solution Absolute or Relative?

we have to consider whether the solution of the given problem is Absolute or Relative.

**For example:** consider the following problem of some simple facts.

1. **Marcus was a man**
2. **Marcus was a Pompeian**
3. **Marcus was born in 40 A.D**
4. **All men are mortal.**
5. **All Pompeians died when the volcano erupted in 79 A.D.**
6. **No mortal lives longer than 150 years.**
7. **It is now 1991 A.D.**

The Question is “**Is Marcus alive?**”

To Solve the above problem with the use of given facts the following steps are used.

	Justification
1. Marcus was a man.	axiom 1
4. All men are mortal.	axiom 4
8. Marcus is mortal.	1, 4
3. Marcus was born in 40 A.D.	axiom 3
7. It is now 1991 A.D.	axiom 7
9. Marcus' age is 1951 years.	3, 7
6. No mortal lives longer than 150 years.	axiom 6
10. Marcus is dead.	8, 6, 9
OR	
7. It is now 1991 A.D.	axiom 7
5. All Pompeians died in 79 A.D.	axiom 5
11. All Pompeians are dead now.	7, 5
2. Marcus was a Pompeian.	axiom 2
12. Marcus is dead.	11, 2

Figure 2.13: Two Ways of Deciding That Marcus Is Dead

Therefore the solution of the above problem is “**Marcus is Dead**”.

## Q). Heuristic Search

A heuristic is a technique that improves the efficiency of a search process. Heuristics are like tour guides. They are good to the extent that they point in generally interesting directions; they are bad to the extent that they may miss points of interest to particular heuristics that are useful in a wide variety of problem domains.

For example, applying it to the traveling salesman problem, we produce the following procedure:

- Arbitrary select a starting city.
- To select the next city, look at all cities not yet visited, and select the one closest to the current city. Go to it next.
- Repeat step 2 until all cities have been visited.

### Heuristic Function:

The purpose of the heuristic function is to guide the search process in the most profitable direction by suggesting which path to follow when more than one is available.

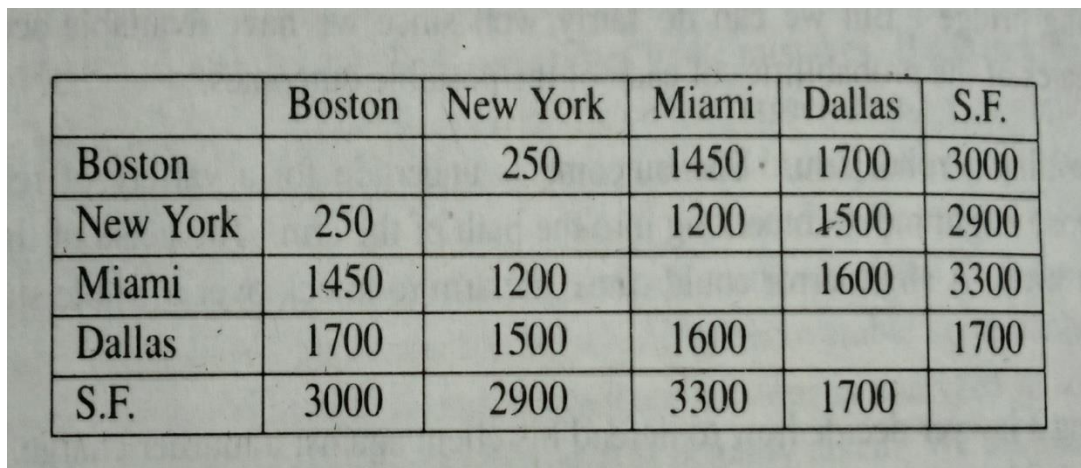
The more accurately heuristic function estimates the true merits of each node in the search tree. It would be possible to compute a perfect heuristic function by doing a complete search from the node in question and determining whether it leads to a good solution.

### Some simple heuristic functions:

Chess : the material advantage of our side over the Opponent.

Travelling Salesperson: the sum of the distances so far.

For example: Consider an instance of Travelling Salesperson (TSP) Problem. Our goal is to find the shortest route that visits each city exactly once. Suppose the cities to be visited and distances between them are as shown in figure.

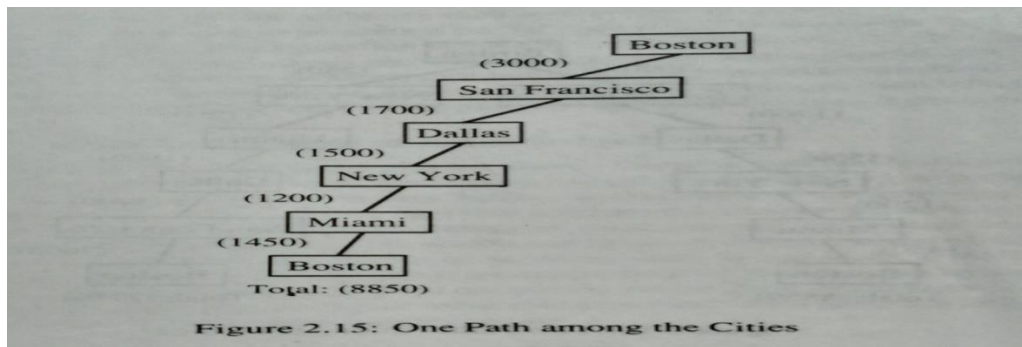


	Boston	New York	Miami	Dallas	S.F.
Boston		250	1450	1700	3000
New York	250		1200	1500	2900
Miami	1450	1200		1600	3300
Dallas	1700	1500	1600		1700
S.F.	3000	2900	3300	1700	

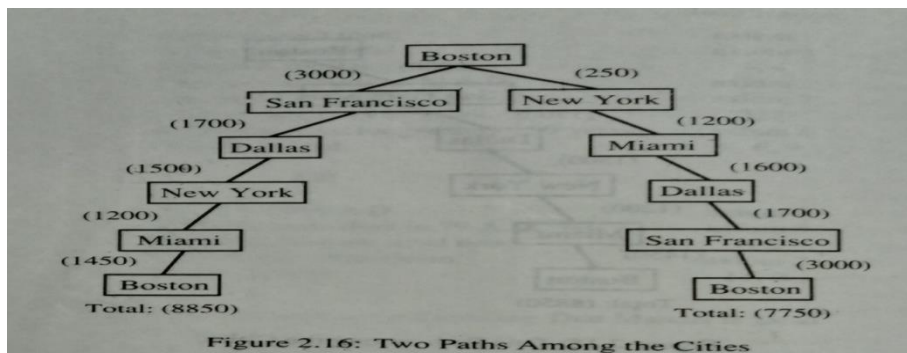
Fig 2.14: An Instance of the TSP

One place the salesman could start Boston. In that case, one path that mapping be followed is the one shown in figure 2.15, which is 8850 miles long.





The answer is that we cannot be sure unless we also try all other paths to make sure that none of them is shorter. In this case, as can be seen from Figure 2.16, the first path is definitely not the solution to the salesman's problem.



These two examples illustrate the difference between any path Problems. Best path problems are, in general, computationally harder than any path problem. Any path problems can often be solved in reasonable amount of time by using heuristic that suggest good paths to explore.

### Q). Heuristic Search Techniques

The following are the different types of heuristic Search Techniques.

- Generate-and-test
- Hill Climbing
- Best – first search
- Problem reduction
- Constraint Satisfaction
- Means-ends analysis
- Depth-First Search
- Breadth – First Search

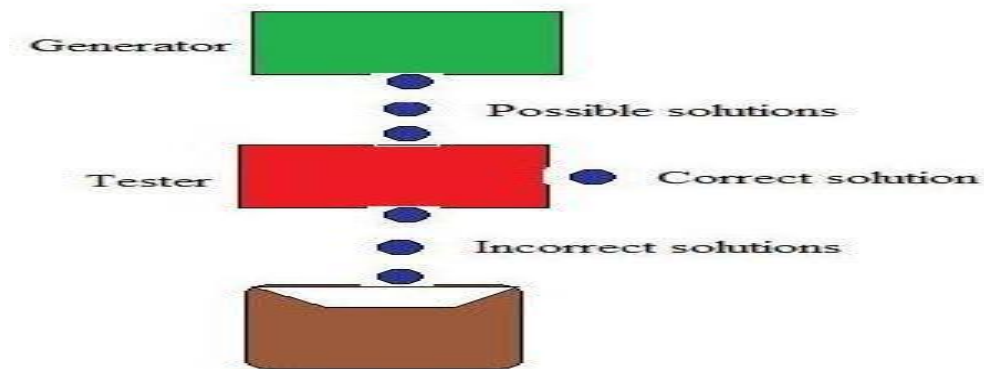
### Q). Generate-and-test:

Generate-and-test search algorithm is a very simple algorithm that guarantees to find a solution if done systematically and there exists a solution.

### Algorithm: Generate-and-test

1. Generate a possible solution.
2. Test to see if this is the expected solution.
3. If the solution has been found quit, else go to step 1.

Potential solution that needs to be generated vary depending on the Kinds of problems. For some problems the possible solutions may be Particular points in the problem space and for some problems, paths from the start state. As figure shown given below.



Generate and test like depth first search, requires that complete solution be generated for testing.

### Q). Hill Climbing:

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal maximum.

In the above definition, **mathematical optimization problems** imply that hill-climbing solves the problems where we need to maximize or minimize a given real function by choosing values from the given inputs.

Example-**Travelling salesman problem** where we need to minimize the distance traveled by the salesman.

### Features of Hill Climbing:

#### ➤ Variant of generate and test algorithm:

It is a variant of generate and test algorithm. The generate and test algorithm is as follows:

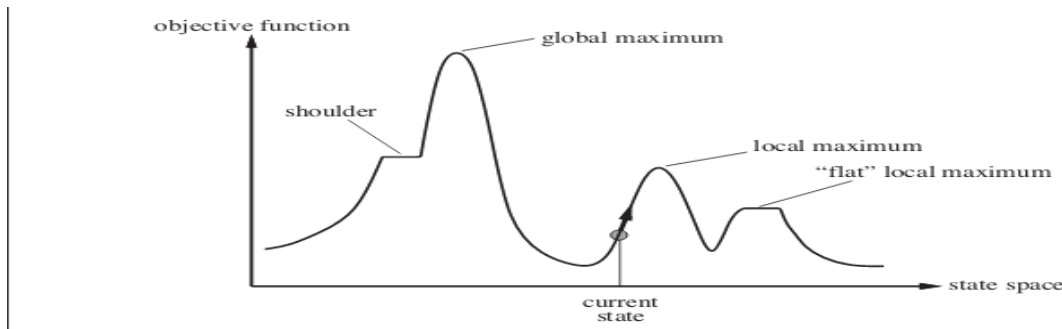
1. Generate possible solutions.
2. Test to see if this is the expected solution.
3. If the solution has been found quite else go to step 1.

Hence, we call Hill climbing as a variant of generate and test algorithm as it takes the feedback from the test procedure. Then this feedback is utilized by the generator in deciding the next move in search space.

- **Uses the Greedy approach:** At any point in state space, the search moves in that direction only which optimizes the cost of function with the hope of finding the optimal solution at the end.
- **State Space diagram for Hill Climbing**

State space diagram is a graphical representation of the set of states our search algorithm can reach vs the value of our objective function (the function which we wish to maximize).

**X-axis:** denotes the state space ie states or configuration our algorithm may reach.  
**Y-axis:** denotes the values of objective function corresponding to a particular state. The best solution will be that state space where objective function has maximum value(global maximum).



Following are the different regions in the State Space Diagram

❖ **Local maximum:**

It is a state which is better than its neighboring state however there exists a state which is better than it(global maximum). This state is better because here the value of the objective function is higher than its neighbors.

**Global maximum:**

It is the best possible state in the state space diagram. This because at this state, objective function has highest value.

**Plateau/flat local maximum:**

It is a flat region of state space where neighboring states have the same value.

**Ridge:**

It is region which is higher than its neighbors but itself has a slope. It is a special kind of local maximum.

**Current state:**

The region of state space diagram where we are currently present during the search.

**Shoulder:**

It is a plateau that has an uphill edge.

**Q). Types of Hill Climbing**

The following are the different types of Hill Climbing Techniques.

1. **Simple Hill Climbing**
2. **Steepest-Ascent Hill climbing**
3. **Simulated Annealing**

### 1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill-climbing algorithm. It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state.

It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.

This algorithm has the following features:

- ✓ Less time consuming
- ✓ Less optimal solution
- ✓ The solution is not guaranteed

#### Algorithm for Simple Hill Climbing:

**Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.

**Step 2:** Loop Until a solution is found or there is no new operator left to apply.

**Step 3:** Select and apply an operator to the current state.

**Step 4:** Check new state:

- I. If it is goal state, then return success and quit.
- II. else if it is better than the current state then assigns, new state as a current state.
- III. else if not better than the current state, then return to step 2.

**Step 5:** Exit.

### 2. Steepest-Ascent hill climbing

The steepest-Ascent algorithm is a variation of the simple hill-climbing algorithm.

This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state.

This algorithm consumes more time as it searches for multiple neighbors.

#### Algorithm for Steepest-Ascent hill climbing

**Step 1:** Evaluate the initial state, if it is goal state then return it and quit.

Otherwise, continue with the initial state as the current state.

**Step 2:** Loop until a solution is found or the current state does not change.

- I. Let **S** be a state such that any successor of the current state will be better than it.
- II. For each operator that applies to the current state;
  - Apply the new operator and generate a new state.
  - Evaluate the new state.
  - If it is goal state, then return it and quit, else compare it to the **S**.
  - If it is better than **S**, then set new state as **S**.
  - If the **S** is better than the current state, then set the current state to **S**.

**Step 5:** Exit.

Both basic and steepest- ascent hill climbing may fail to find a solution. Either algorithm may terminate not by finding a goal state but by getting to a state from which no better state can be generated. This will happen if the program has reached either a local maximum, a plateau, or a ridge.

**3. Simulated Annealing**

Simulated Annealing is a variation of hill climbing in which, at the beginning of the process, some downhill moves may be made.

Annealing is a process in a metallurgy where metals are slowly cooled to make them reach a state of low energy where there are strong. But there is some probability that a transition to a higher energy state will occur. This probability is given by the function.

$$p=e^{-\Delta E/kT}$$

Where  $\Delta E$  is the positive change in the energy level,  $T$  is the temperature, and  $k$  is **Boltzmann's constant**.

This should lower the chances of getting caught at a local maximum, a plateau or a ridge.

**Algorithm for Simulated Annealing**

**Step 1:** Evaluate the initial state, if it is goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

**Step 2:** Initialize BEST –SO- FAR to the current state.

**Step 3:** Initialize  $T$  according to the annealing schedule.

**Step 4:** Loop until a solution is found or until there are no new operator left to be applied in the current state.

a) . Select an operator that has not yet been applied to the current state and apply it to produce a new state.

b). Evaluate the new state. Compute

  $E = (\text{value of current}) - (\text{value of new state})$

- If the new state is a goal state, then return it and quit
- If it is not a goal state but is better than the current state, then make it the current state. Also set BEST-SO-FAR to this new state.
- If it is not better than the current state, then make it the current state with probability  $p'$  as defined above.
- Revise  $T$  as necessary according to the annealing schedule.

**Step 5:** Return BEST –SO- FAR as the answer.

**Q).Best-First Search:**

Best-first search, which is a way of combining the advantages of both **Depth First Search (DFS)** and **Breadth First Search (BFS)** into a Single Unit.

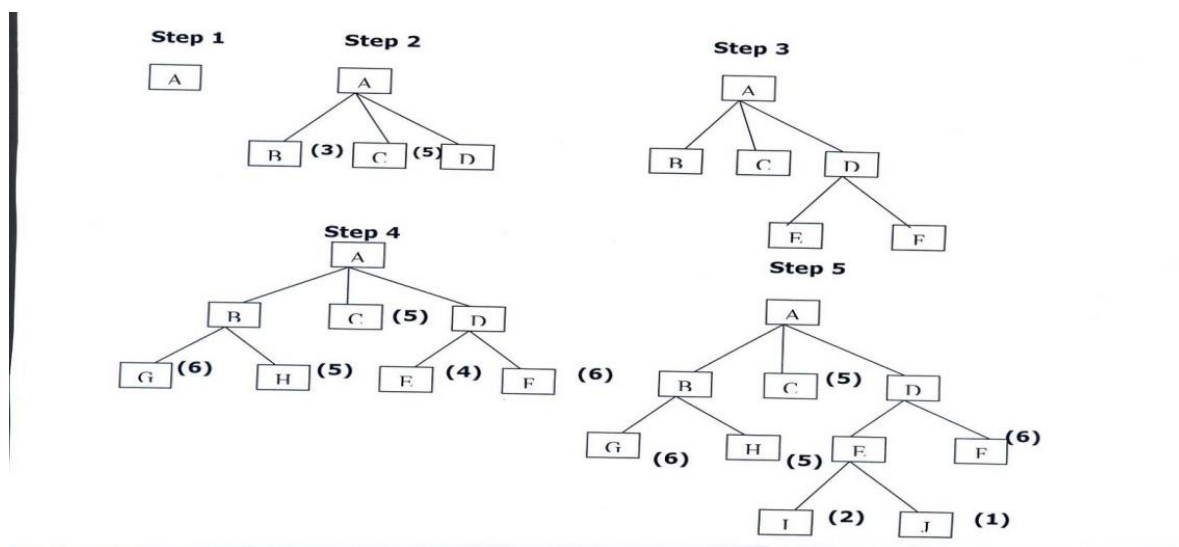
**OR –Graph:**



At each step of the best-first search process, we select the most promising of the nodes we have generated so far. This is done by applying an appropriate heuristic function to each of them. We then expand the chosen node by using the rules to generate its successors. If one of them is a solution, we can quit. If not, all those new nodes are added to the set of nodes generated so far. Again, the most promising node is selected, and the process continues.

fig 3.3 shows the beginning of a best-first search procedure. Initially, there is only one node, so it will be expanded to the three new nodes. The heuristic function, which, in this example, is an estimate of the cost of getting to a solution from a given node, is applied to each of these new nodes.

Since node D is the most promising. It is expanded next, producing two successor nodes, E and F. But then the heuristic function is applied to them. Now another path, that going through node B, looks more promising, so it is pursued, generating nodes G and H. But again, when these new nodes are evaluated, they look less promising than another path, attention is returned to the path through D to E. E is then expanded. Yielding nodes I and J. At the next step, J will be expanded, since it is the most promising. This process can continue until a solution is found.

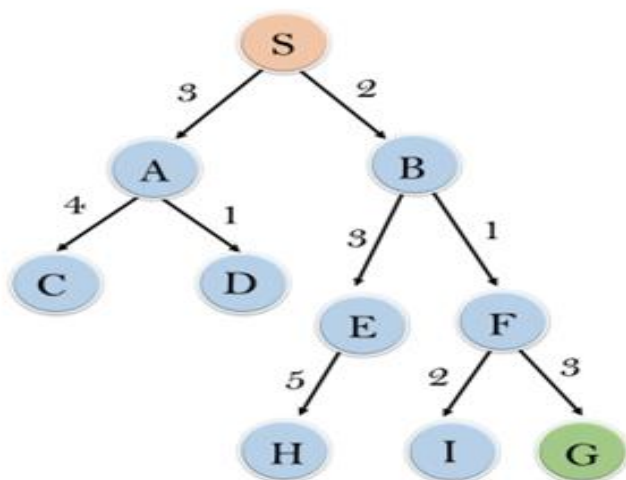


### Algorithm: Best – First Search

1. Start with OPEN containing just the initial state.
2. Until a goal is found or there are no nodes left on OPEN do:
  - a. Pick the best node on OPEN
  - b. Generate its Successors.
    - A. For each successor do
      1. If has not been generated before, evaluate it, add it to OPEN, and record its parent.
      2. It has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successor that this node may already have.

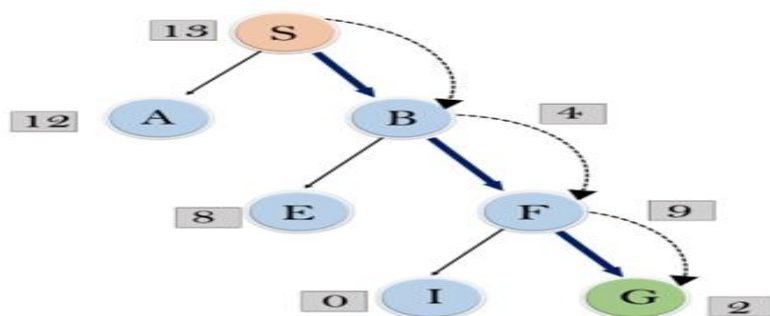
### Example:

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function  $f(n) = \min(h(n))$ , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



G is Zero(0) and I is 2 value.

**Expand the nodes of S and put in the CLOSED list**

**Initialization:** Open [A, B], Closed [S]

**Iteration 1:** Open [A], Closed [S, B]

**Iteration 2:** Open [E, F, A], Closed [S, B]  
:Open [E, A], Closed [S, B, F]

**Iteration 3:** Open [I, G, E, A], Closed [S, B, F]  
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F-----> G**

**Time Complexity:** The worst case time complexity of Greedy best first search is  $O(b^m)$ .

**Space Complexity:** The worst case space complexity of Greedy best first search is  $O(b^m)$ . Where, m is the maximum depth of the search space.

**Q). Problem Reduction**

The divide and conquer strategy, a solution to a problem can be obtained by decomposing it into smaller sub-problems. Each of this sub-problem can then be solved to get its sub solution. These sub solutions can then be recombined to get a solution as a whole. That is called is **Problem Reduction**. This method generates arc which is called as **AND** arcs. One AND arc may point to any number of successor nodes, all of which must be solved for an arc to point to a solution.

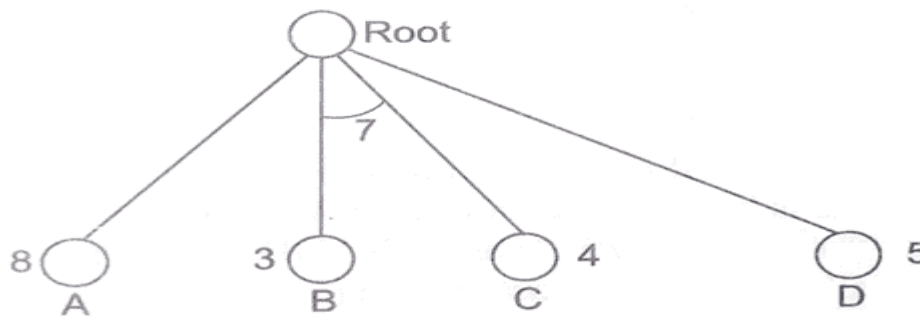


Fig: AND / OR Tree



Fig: AND / OR Graph

**Problem Reduction algorithm:**

1. Initialize the graph to the starting node.
2. Loop until the starting node is labelled **SOLVED** or until its cost goes above **FUTILITY**:
  - (i) Traverse the graph, starting at the initial node and following the current best path and accumulate the set of nodes that are on that path and have not yet been expanded.
  - (ii) Pick one of these unexpanded nodes and expand it. If there are no successors, assign **FUTILITY** as the value of this node. Otherwise, add its successors to the graph and for each of them compute  $f'(n)$ . If  $f'(n)$  of any node is 0, mark that node as **SOLVED**.
  - (iii) Change the  $f'(n)$  estimate of the newly expanded node to reflect the new information provided by its successors. Propagate this change backwards through the graph. If any node contains a successor arc whose descendants are all solved, label the node itself as **SOLVED**.

### Q).Means-Ends Analysis

- We have studied the strategies which can reason either in forward or backward, but a mixture of the two directions is appropriate for solving a complex and large problem. Such a mixed strategy, make it possible that first to solve the major part of a problem and then go back and solve the small problems arise during combining the big parts of the problem. Such a technique is called **Means-Ends Analysis**.
- Means-Ends Analysis is problem-solving techniques used in Artificial intelligence for limiting search in AI programs.
- It is a mixture of Backward and forward search technique.
- The MEA technique was first introduced in 1961 by Allen Newell, and Herbert A. Simon in their problem-solving computer program, which was named as General Problem Solver (GPS).
- The MEA analysis process centered on the evaluation of the difference between the current state and goal state.

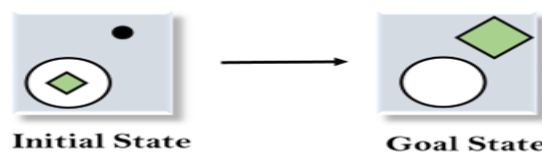
#### Algorithm :

- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
  - a. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
  - b. Attempt to apply operator O to CURRENT. Make a description of two states.
    - i) O-Start, a state in which O's preconditions are satisfied.
    - ii) O-Result, the state that would result if O were applied In O-start.
  - c. If  
(**First-Part** <----- MEA (CURRENT, O-START)  
And  
(**LAST-Part** <----- MEA (O-Result, GOAL), are successful, then signal Success and return the result of combining FIRST-PART, O, and LAST-PART.

The above-discussed algorithm is more suitable for a simple problem and not adequate for solving complex problems.

Example of Mean-Ends Analysis:

Let's take an example where we know the initial state and goal state as given below. In this problem, we need to get the goal state by finding differences between the initial state and goal state and applying operators.



Solution:

To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

- **Move**
- **Delete**
- **Expand**

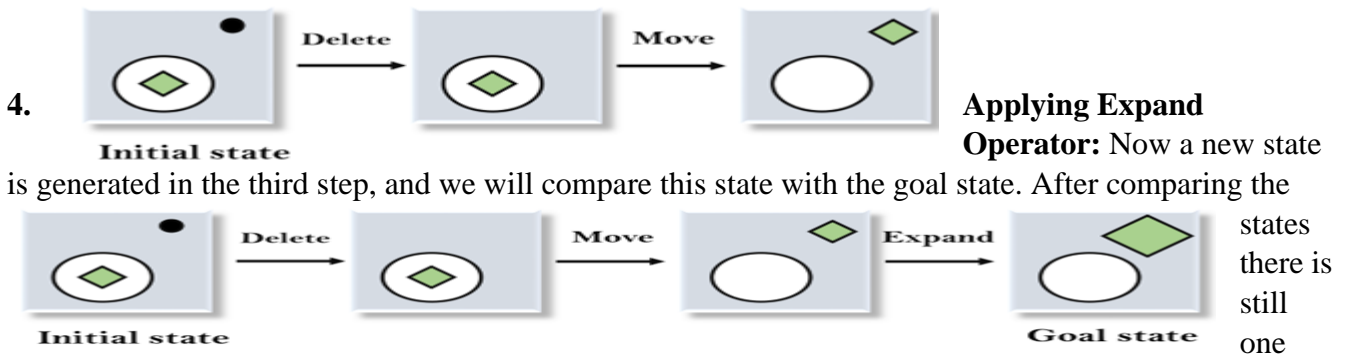
**1. Evaluating the initial state:** In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.



**2. Applying Delete operator:** As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



**3. Applying Move Operator:** After applying the Delete operator, the new state occurs which we will again compare with goal state. After comparing these states, there is another difference that is the square is outside the circle, so, we will apply the **Move Operator**.



difference which is the size of the square, so, we will apply **Expand operator**, and finally, it will generate the goal state.

#### Q). Constraint Satisfaction:

A Constraint Satisfaction Problem (CSP) is a program that requires its solution within some limitations or conditions that is called Constraints. It consists of:

1. A finite set of variables which store the solution  $V = \{V_1, V_2, V_3, \dots, V_n\}$
2. A set of discrete values known as Domain Form which the solution is picked.  $D = \{D_1, D_2, D_3, \dots, D_n\}$



3. A finite set of constraints  $C=\{C_1,C_2,C_3,\dots,C_n\}$ .

**Constraint Satisfaction Algorithm:**

1. Until a complete solution is found or until all path has led to dead ends, do:

- (i) Select an unexpected node of the search graph.
- (ii) Apply the constraints inference rules to the selected node to generate all possible new constraints.
- (iii) If the set of constraints contains a contradiction then report that this path is a dead-end.
- (iv) If the set of constraints describes a complete solution then report success.
- (v) If neither a contradiction nor a complete solution has been found then apply the problem space rules to generate new partial solutions that are consistent with the current set of constraints. Insert these partial solutions into the search graph.

## UNIT-II

### What is knowledge representation?

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning.**

Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

**Knowledge:** Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Types of knowledge

Following are the various types of knowledge:



1.

### Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarativesentences.

- It is simpler than procedural language.

## **2. Procedural Knowledge**

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

## **3. Meta-knowledge:**

- Knowledge about the other types of knowledge is called Meta-knowledge.

## **4. Heuristic knowledge:**

- Heuristic knowledge is representing knowledge of some experts in a field or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

## **5. Structural knowledge:**

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

## **Q).Approaches to knowledge representation:**

There are mainly four approaches to knowledge representation, which are given below:

### **1. Simple relational knowledge:**

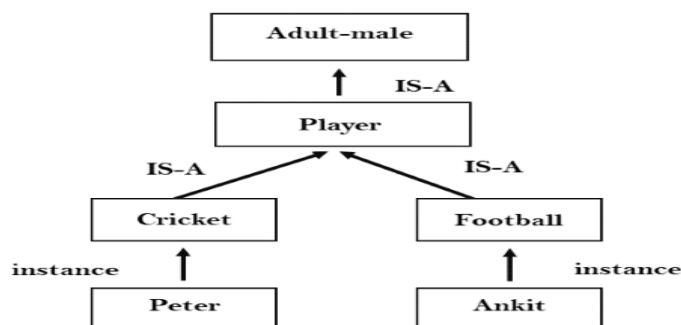
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

**Example: The following is the simple relational knowledge representation.**

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

## 2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.
- **Example:**



## 3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
  - a. Marcus is a man

- b. All men are mortal  
Then it can represent as;

**man(Marcus)**

**$\forall x = \text{man}(x) \text{ -----} \rightarrow \text{mortal}(x)$**

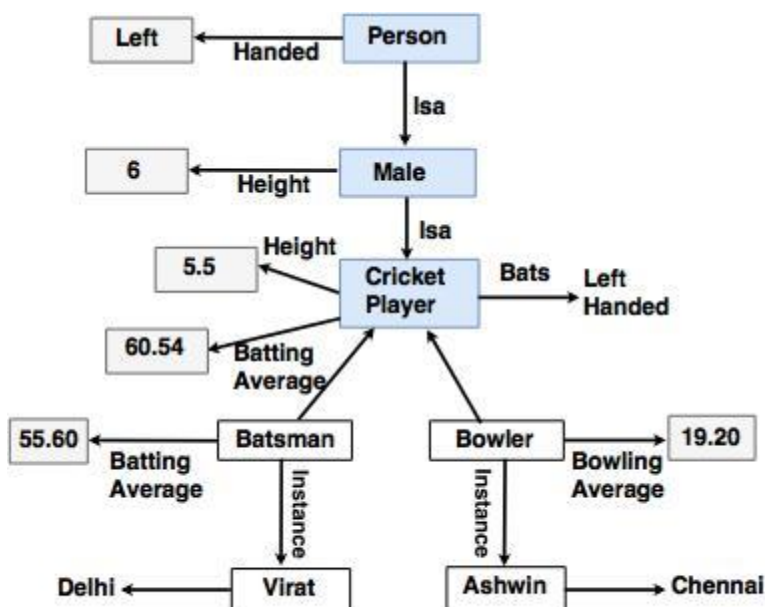
#### 4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is **If-Then rule**.
- In this knowledge, we can use various coding languages such as **LISP language** and **Prolog language**.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

#### Q).Issues in knowledge representation

The main objective of knowledge representation is to draw the conclusions from the knowledge, but there are many issues associated with the use of knowledge representation techniques.

Some of them are listed below:



**Fig: Inheratable Knowledge Representation**

Refer to the above diagram to refer to the following issues.



### 1. Important attributes

There are two attributes shown in the diagram, **instance** and **isa**. Since these attributes support property of inheritance, they are of prime importance.

### 2. Relationships among attributes

Basically, the attributes used to describe objects are nothing but the entities. However, the attributes of an object do not depend on the encoded specific knowledge.

### 3. Choosing the granularity of representation

While deciding the granularity of representation, it is necessary to know the following:

- i. What are the primitives and at what level should the knowledge be represented?
- ii. What should be the number (small or large) of low-level primitives or high-level facts?
- iii. High-level facts may be insufficient to draw the conclusion while Low-level primitives may require a lot of storage.

### 4. Representing sets of objects.

There are some properties of objects which satisfy the condition of a set together but not as individual;

### 5. Finding the right structure as needed

To describe a particular situation, it is always important to find the access of right structure. This can be done by selecting an initial structure and then revising the choice.

While selecting and reversing the right structure, it is necessary to solve following problem statements.

**They include the process on how to:**

- Select an initial appropriate structure.
- Fill the necessary details from the current situations.
- Determine a better structure if the initially selected structure is not appropriate to fulfill other conditions.
- Find the solution if none of the available structures is appropriate.
- Create and remember a new structure for the given condition.
- There is no specific way to solve these problems, but some of the effective knowledge representation techniques have the potential to solve them.

### Q).Representations and Mappings

- In order to solve complex problems encountered in artificial intelligence, one needs both a large amount of knowledge and some mechanism for manipulating that knowledge to create solutions.
- Knowledge and Representation are two distinct entities. They play central but distinguishable roles in intelligent system.
- Knowledge is a description of the world. It determines a system's competence by what it knows.
- Representation is the way knowledge is encoded. It defines a system's performance in doing something.
- Different types of knowledge require different kinds of representation.

The knowledge Representation models/mechanisms are often based on:

- Logic
- Rules
- Frames
- Semantic Net

Knowledge is categorized into two major types.

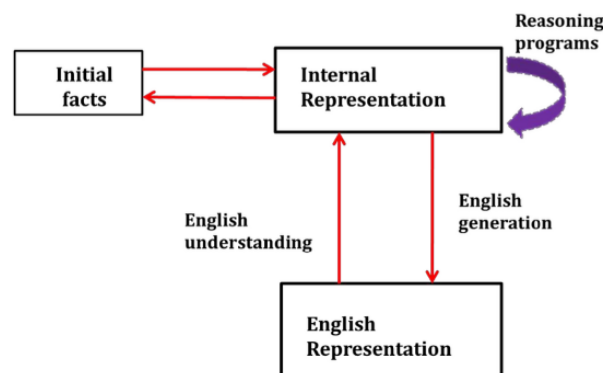
A variety of ways of representing knowledge have been exploited in AI programs.

There are two different kinds of entities, we are dealing with.

1. Facts: Truth in some relevant world. Things we want to represent.
2. Representation of facts in some chosen formalism. Things we will actually be able to manipulate.

These entities are structured at two levels:

1. The knowledge level, at which facts are described.
2. The symbol level, at which representation of objects are defined in terms of symbols that can be manipulated by programs.



### Chapter-II Using Predicate Logic

**Representation of simple facts using logics :**

- Propositional logic is useful because it is simple to deal with and a decision procedure for it exists.
- Propositional logic is appealing because it is simple to deal with and a decision procedure for it exists.
- We can easily represent real world facts as logical preposition written as well formed formulas(wff's) in prepositional logic.

It is raining.  
 RAINING  
 It is sunny.  
 SUNNY  
 It is windy.  
 WINDY  
 If it is raining, then it is not sunny.  
 $\text{RAINING} \rightarrow \neg \text{SUNNY}$

Fig: Some Simple Facts in Propositional Logic

Above Fig using these propositions, we could for example , conclude from the fact that it is raining the fact that it is not sunny.

- The facts described by these sentence can be represented as a set of wff's in predicate logic as follows

1. Marcus is a man.  
 $\text{man}(\text{Marcus})$
2. Plato is a man.  
 $\text{man}(\text{Plato})$
3. All men are mortal.  
 $\text{mortal}(\text{men})$
4. All Pompeian's were Romans  
 $\forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x)$
5. All Romans were either loyal to Caesar or hated him  
 $\forall x: \text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar})$
6. Everyone is loyal to someone.  
 $\forall x: \exists y: \text{loyalto}(x, y)$
7. Marcus tried to assassinate Caesar.  
 $\text{Tryassassinate}(\text{Marcus}, \text{Caesar})$
8. People only try to assassinate rulers they are not loyal to.  
 $\forall x: \forall y: \text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y)$

### Representing Instance and Isa Relationships

- ✚ Specific attributes **instance** and **isa** play important role particularly in a useful form of reasoning called property inheritance.
- ✚ The predicates "instance" and "isa" explicitly captured the relationships they are used to express, namely class membership and class inclusion.
- ✚ Below figure shows the first five sentences of the last section represented in logic in three different ways.
- ✚ The first part of the figure contains the representations we have already discussed. In these representations, class membership is represented with unary predicates (such as Roman), each of which corresponds to a class.
- ✚ Asserting that  $P(x)$  is true is equivalent to asserting that  $x$  is an instance (or element) of  $P$ .

- ✚ The second part of the figure contains representations that use the **instance** predicate explicitly.

1. <b>Man(Marcus).</b> 2. <b>Pompeian(Marcus).</b> 3. $\forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x).$ 4. <b>ruler(Caesar).</b> 5. $\forall x: \text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}).$
1. <b>instance(Marcus, man).</b> 2. <b>instance(Marcus, Pompeian).</b> 3. $\forall x: \text{instance}(x, \text{Pompeian}) \rightarrow \text{instance}(x, \text{Roman}).$ 4. <b>instance(Caesar, ruler).</b> 5. $\forall x: \text{instance}(x, \text{Roman}). \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}).$
1. <b>instance(Marcus, man).</b> 2. <b>instance(Marcus, Pompeian).</b> 3. <b>isa(Pompeian, Roman)</b> 4. <b>instance(Caesar, ruler).</b> 5. $\forall x: \text{instance}(x, \text{Roman}). \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}).$ 6. $\forall x: \forall y: \forall z: \text{instance}(x, y) \wedge \text{isa}(y, z) \rightarrow \text{instance}(x, z).$

- ✚ The predicate **instance** is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs.
- ✚ But these representations do not use an explicit **isa** predicate.
- ✚ Instead, subclass relationships, such as that between Pompeians and Romans, are described as shown in sentence 3.
- ✚ The implication rule states that if an object is an instance of the subclass Pompeian then it is an instance of the superclass Roman.
- ✚ Note that this rule is equivalent to the standard set-theoretic definition of the subclass-superclass relationship.
- ✚ The third part contains representations that use both the **instance** and **isa** predicates explicitly.
- ✚ The use of the **isa** predicate simplifies the representation of sentence 3, but it requires that one additional axiom (shown here as number 6) be provided.

### Computable Functions and Predicates:

- To express simple facts, such as the following greater-than and less-than relationships:  
 $\text{gt}(1,0) \text{ It}(0,1) \text{ gt}(2,1) \text{ It}(1,2) \text{ gt}(3,2) \text{ It}(2,3)$
- It is often also useful to have computable functions as well as computable predicates.  
Thus we might want to be able to evaluate the truth of  $\text{gt}(2 + 3,1)$
- To do so requires that we first compute the value of the plus function given the arguments 2 and 3, and then send the arguments 5 and 1 to gt.

Consider the following set of facts, again involving Marcus:

- Marcus was a man.  
 $\text{man}(\text{Marcus})$
- Marcus was a Pompeian.  
 $\text{Pompeian}(\text{Marcus})$
- Marcus was born in 40 A.D.  
 $\text{born}(\text{Marcus}, 40)$
- All men are mortal.  
 $x: \text{man}(x) \rightarrow \text{mortal}(x)$
- All Pompeians died when the volcano erupted in 79 A.D.  
 $\text{erupted}(\text{volcano}, 79) \wedge \forall x : [\text{Pompeian}(x) \rightarrow \text{died}(x, 79)]$
- No mortal lives longer than 150 years.

$$\forall x: \forall t_1: \forall t_2: \text{mortal}(x) \wedge \text{born}(x, t_1) \wedge \text{gt}(t_2 - t_1, 150) \rightarrow \text{died}(x, t_2)$$

7. It is now 1991.

$$\text{now} = 1991$$

So, Above example shows how these ideas of computable functions and predicates can be useful. It also makes use of the notion of equality and allows equal objects to be substituted for each other whenever it appears helpful to do so during a proof.

So, Now suppose we want to answer the question “Is Marcus alive?”

- The statements suggested here, there may be two ways of deducing an answer.
- Either we can show that Marcus is dead because he was killed by the volcano or we can show that he must be dead because he would otherwise be more than 150 years old, which we know is not possible.
- Also, As soon as we attempt to follow either of those paths rigorously, however, we discover, just as we did in the last example, that we need some additional knowledge. For example, our statements talk about dying, but they say nothing that relates to being alive, which is what the question is asking.

➤ So we add the following facts:

8. Alive means not dead.

$$\forall x: \forall t: [\text{alive}(x, t) \rightarrow \neg \text{dead}(x, t)] \wedge [\neg \text{dead}(x, t) \rightarrow \text{alive}(x, t)]$$

9. If someone dies, then he is dead at all later times.

$$\forall x: \forall t_1: \forall t_2: \text{died}(x, t_1) \wedge \text{gt}(t_2, t_1) \rightarrow \text{dead}(x, t_2)$$

So, Now let's attempt to answer the question “Is Marcus alive?”

by proving:  $\neg \text{alive}(\text{Marcus}, \text{now})$

## REPRESENTING KNOWLEDGE USING RULES IN AI

The classic methods of representing knowledge use either rules or logic. they are derived from data using a machine learning or data mining algorithm. Rules use a logic-based form for reasoning. Logic is the use of symbolic and mathematical techniques for deductive reasoning.

Knowledge representation logic is used heavily in AI follows

- Propositional logic
- predicate logic
- rules
- semantic nets
- frame
- object

### Procedural vs Declarative Knowledge:

#### Procedural Knowledge:

➤ A representation in which the control information that is necessary to use the knowledge is embedded in the knowledge itself for e.g. computer programs, directions, and recipes; these indicate specific use or implementation;

- The real difference between declarative and procedural views of knowledge lies in where control information reside.

For example, consider the following

Man (Marcus)

Man (Caesar)

Person (Cleopatra)

$\forall x: \text{Man}(x) \rightarrow \text{Person}(x)$

Now, try to answer the question.  $? \text{Person}(y)$

The knowledge base justifies any of the following answers.

$Y = \text{Marcus}$

$Y = \text{Caesar}$

$Y = \text{Cleopatra}$

- We get more than one value that satisfies the predicate.
- If only one value needed, then the answer to the question will depend on the order in which the assertions examined during the search for a response.
- If the assertions declarative then they do not themselves say anything about how they will be examined. In case of procedural representation, they say how they will examine.

### **Declarative Knowledge :**

- A statement in which knowledge specified, but the use to which that knowledge is to be put is not given.
- For example, laws, people's name; these are the facts which can stand alone, not dependent on other knowledge;
- So to use declarative representation, we must have a program that explains what is to do with the knowledge and how.
- For example, a set of logical assertions can combine with a resolution theorem prove to give a complete program for solving problems but in some cases, the logical assertions can view as a program rather than data to a program.
- Hence the implication statements define the legitimate reasoning paths and automatic assertions provide the starting points of those paths.
- These paths define the execution paths which is similar to the 'if then else' in traditional programming.
- So logical assertions can view as a procedural representation of knowledge.

### **Logic Programming – Representing Knowledge Using Rules**

- ✚ Logic programming is a programming paradigm in which logical assertions viewed as programs.
- ✚ These are several logic programming systems, PROLOG is one of them.
- ✚ A PROLOG program consists of several logical assertions where each is a horn clause i.e. a clause with at most one positive literal.
- ✚ Ex :  $P, P \vee Q, P \rightarrow Q$

For example the PROLOG clause  $P(x) :- Q(x, y)$  is equal to logical expression  $\forall x: \exists y: Q(x, y) \rightarrow P(x)$ .

- ✚ The difference between the logic and PROLOG representation is that the PROLOG interpretation has a fixed control strategy. And so, the assertions in the PROLOG program define a particular search path to answer any question.



- ✚ But, the logical assertions define only the set of answers but not about how to choose among those answers if there is more than one.

Consider the following example:

1. Logical representation
$$\forall x : \text{pet}(x) \wedge \text{small}(x) \rightarrow \text{apartmentpet}(x)$$
$$\forall x : \text{cat}(x) \wedge \text{dog}(x) \rightarrow \text{pet}(x)$$
$$\forall x : \text{poodle}(x) \rightarrow \text{dog}(x) \wedge \text{small}(x)$$

poodle (fluffy)
2. Prolog representation
$$\text{apartmentpet}(x) :- \text{pet}(x), \text{small}(x)$$
$$\text{pet}(x) :- \text{cat}(x)$$
$$\text{pet}(x) :- \text{dog}(x)$$
$$\text{dog}(x) :- \text{poodle}(x)$$
$$\text{small}(x) :- \text{poodle}(x)$$

poodle (fluffy)

Above example simple knowledge base represented in standard logical notation and then PROLOG. Both of these representations contain two types of statements, facts, which contain only constants and rules, which do contain variables. Facts represent statements about specific objects. Rules represent statements about classes of objects.

## **FARWORD VERSUS BACKWORD REASONING.**

**A search procedure must find a path between initial and goal states. There are two directions in which a search process could proceed.**

**(1) Reason forward from the initial states: Being form the root of the search tree. General the next level of the tree by finding all the rules whose left sides match the root node, and use their right sides to generate the siblings. Repeat the process until a configuration that matches the goal state is generated.**

**(2)Reason forward from the goal state(s): Begin building a search tree starting with the goal configuration(s) at the root. Generate the next level of the tree by finding all the rules whose right sides match with the root node. Use the left sides of the rules to generate the new nodes. Continue until a node that matches the start state is generated. This method of chaining backward from the desired final state is called goal directed reasoning or back tracing.**

**Selection of forward reasoning or backward reasoning depends on which direction offers less branching factor and justifies its reasoning process to the user. Most of the search techniques can be used to search either forward or backward.**

### **Forward chaining.**

1. It is also known as data driven inference technique.
2. Forward chaining matches the set of conditions and infer results from these conditions.  
Basically, forward chaining starts from a new data and aims for any conclusion.
3. It is bottom up reasoning.
4. It is a breadth first search.
5. It continues until no more rules can be applied or some cycle limit is met.

6. For example: If it is cold then I will wear a sweater. Here “it is cold is the data” and “I will wear a sweater” is a decision. It was already known that it is cold that is why it was decided to wear a sweater, This process is forward chaining.
7. It is mostly used in commercial applications i.e event driven systems are common example of forward chaining.
8. It can create an infinite number of possible conclusions.

### **Backward chaining.**

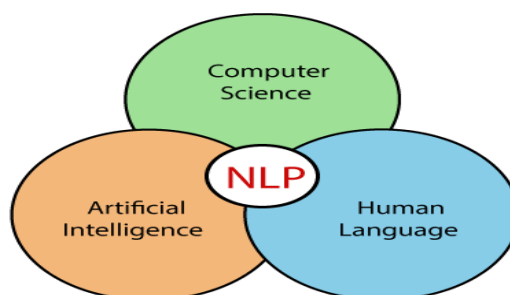
1. It is also called as goal driven inference technique.
2. It is a backward search from goal to the conditions used to get the goal. Basically it starts from possible conclusion or goal and aims for necessary data.
3. It is top down reasoning.
4. It is a depth first search.
5. It process operations in a backward direction from end to start, it will stop when the matching initial condition is met.
6. For example: If it is cold then I will wear a sweater. Here we have our possible conclusion “I will wear a sweater”. If I am wearing a sweater then it can be stated that it is cold that is why I am wearing a sweater. Hence it was derived in a backward direction so it is the process of backward chaining.
7. It is used in interrogative commercial applications i.e finding items that fulfill possible goals.
8. Number of possible final answers is reasonable.

## **UNIT 3**

### **NATURAL LANGUAGE PROCESSING**

#### **What is NLP?**

NLP stands for **Natural Language Processing**, which is a part of **Computer Science**, **Human language**, and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation**.



#### **Advantages of NLP**

- NLP helps users to ask questions about any subject and get a direct response within seconds.

- NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
- NLP helps computers to communicate with humans in their languages.
- It is very time efficient.
- Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

### Disadvantages of NLP

A list of disadvantages of NLP is given below:

- NLP may not show context.
- NLP is unpredictable
- NLP may require more keystrokes.
- NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

### Q).OVERVIEW OF LINGUISTICS:

- For the study of natural language, linguistics is not prerequisite, but grammar basics are important.
- Sentences are basic language elements.

### Levels of knowledge used in language understanding:

- To understand natural language, we need following knowledge.
  - PHONOLOGICAL** – It is relation between sound and word, by listening sound, word would be built or framed.
  - MORPHOLOGICAL** – It constructs words from basic units called Morphemes.  
Ex: FRIENDLY – 2 Morphemes (FRIEND-LY) and UNBREAKABLE – 3 Morphemes (UN-BREAK-ABLE)
  - SYNTACTIC** – It says about grammatically correct sentence formation.
  - SEMANTIC** – It says about constructing meaningful sentence by combining meanings of words and phrases.
  - PRAGMATIC** – It says about usage of sentences in different contexts which affects the meaning of sentences.
  - WORLD** – To communicate with opposite person, we must have knowledge on his beliefs and goals.

### General Approaches to Natural Language understanding:

- The use of keyword and pattern matching.
- Combined syntactic and semantic analysis.
- Comparing and matching input to real world situations.

**Q). GRAMMARS AND LANGUAGES:**

- We define grammar G as  $G=(V_n, V_t, S, P)$
- Where “ $V_n$ ” is Set of Non Terminal symbols, “ $V_t$ ” is Set of Terminal symbols, “S” is set of starting symbol and “P” is finite set of productions or rewrite rules.
- Production rule “P” has the form :  $xyz \rightarrow xwz$ .
- The rule here is “Y” should be rewritten as “W” in the context of x to z.
- We define simple grammar G with vocabulary Q such as:  
QN={S, NP, N, VP, V, ART} which is Non Terminal set.  
QT={boy, kiwi, frog, ate, kissed, flew, the, a} which is Terminal set.  
P: S  $\rightarrow$  NP VP  
NP  $\rightarrow$  ART N  
VP  $\rightarrow$  V NP  
N  $\rightarrow$  boy|kiwi|frog (Number of choices)  
V  $\rightarrow$  ate|kissed|flew (Number of choices)  
ART  $\rightarrow$  the|a

S - Initial Symbol (Sentence here)

NP – Noun Phrase, VP – Verb Phrase, N – Noun, V – Verb, ART – Article

- With the above grammar, we can write following sentences:
  - ➔ The boy ate a Kiwi
  - ➔ The frog kissed a boy
  - ➔ A boy ate the frog
- To generate a sentence, the rules from “P” are applied sequentially starting with “S” and proceeds until all non – terminal symbols are eliminated as shown below:  
S  $\rightarrow$  NP VP  
ART N VP  
The N VP  
The boy VP  
The boy V NP  
The boy ate NP  
The boy ate ART N  
The boy ate a N  
The boy ate a Kiwi

**The Chomsky Hierarchy of Generative Grammars:**

Chomsky Hierarchy represents the class of languages that are accepted by the different machine. The category of language in Chomsky's Hierarchy is as given below:

1. Type 0 known as Unrestricted Grammar.
2. Type 1 known as Context Sensitive Grammar.
3. Type 2 known as Context Free Grammar.

## 4. Type 3 Regular Grammar.

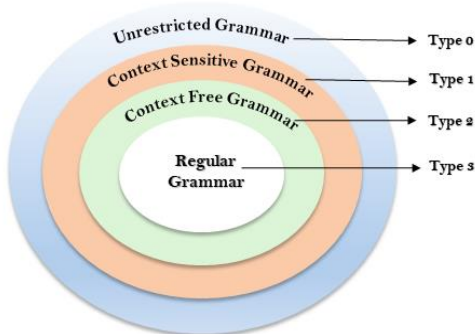


Fig: Chomsky Hierarchy

This is a hierarchy. Therefore every language of type 3 is also of type 2, 1 and 0. Similarly, every language of type 2 is also of type 1 and type 0, etc.

## Type 0 Grammar:

Type 0 grammar is known as Unrestricted grammar. There is no restriction on the grammar rules of these types of languages. These languages can be efficiently modeled by Turing machines.

- $bAa \rightarrow aa$
- $S \rightarrow s$

## Type 1 Grammar:

Type 1 grammar is known as Context Sensitive Grammar. The context sensitive grammar is used to represent context sensitive language. The context sensitive grammar follows the following rules:

- The context sensitive grammar may have more than one symbol on the left hand side of their production rules.
- The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
- The rule of the form  $A \rightarrow \varepsilon$  is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.
- The Type 1 grammar should be Type 0. In type 1, Production is in the form of  $V \rightarrow T$

Where the count of symbol in V is less than or equal to T.

type 1 grammar takes the forms as below:

- $S \rightarrow aS$
- $S \rightarrow aAB$
- $AB \rightarrow BA$
- $aA \rightarrow ab$
- $aA \rightarrow aa$

Here UPPERCASE letters are Non – Terminals and lowercase letters are Terminals.

Type 2 Grammar:

Type 2 Grammar is known as Context Free Grammar. Context free languages are the languages which can be represented by the context free grammar (CFG). Type 2 should be type 1. The production rule is of the form

- Type 2 Grammar (Context Free) – The rule is,
    - $\langle \text{Symbol} \rangle \rightarrow \langle \text{Symbol} 1 \rangle \dots \dots \dots \langle \text{Symbol } K \rangle ; K \geq 1$
- LHS is single non terminal symbol.
- Therefore,  $A \rightarrow XYZ$ ; where A is single non-terminal.

Productions for this type are:

$S \rightarrow aS$   
 $S \rightarrow aSb$   
 $S \rightarrow aB$   
 $S \rightarrow aAB$   
 $A \rightarrow a$   
 $B \rightarrow b$

Ex: Formal Programming Languages

Type 3 Grammar:

Type 3 Grammar is known as Regular Grammar. Regular languages are those languages which can be described using regular expressions. These languages can be modeled by NFA or DFA.

Type 3 is most restricted form of grammar. The Type 3 grammar should be Type 2 and Type 1. Type 3 should be in the form of

- Type 3 Grammar (Regular) – This is also called finite state or regular grammar whose rules are Characterized by forms.

$A \rightarrow aB$   
 $A \rightarrow a$

Ex: Formal Programming Languages

## Q).BASIC PARSING TECHNIQUES

Parsing and generation are sub-divisions of NLP dealing respectively with taking language apart and putting it together. To parse a sentence, it is necessary to find a way in which that sentence could have been generated from the start symbol. Parsing uses knowledge about word and the word meanings (lexicon) based upon the reasoning processes. It exploits the pre-defined legal structures (grammar) i.e. set of rules as shown in figure 6.



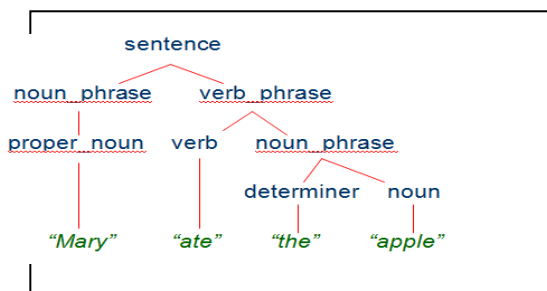


Fig 6: Parsing Tree

Parsing has two main components :

- *Grammar*: a declarative representation describing the syntactic structure of sentences in the language.
- *Parser*: an algorithm that analyzes the input and outputs its structural representation (its parse) consistent with the grammar specification. To construct the parsing tree the below mentioned techniques are used:

*Top-down Parsing*: It begins with start symbol and apply the grammar rules forward until the symbols at the terminals of the tree correspond to the components of the sentence being parsed as shown in figure 7. (i.e. Top-down parsing starts with the S symbol and tries to rewrite it into the sentence).

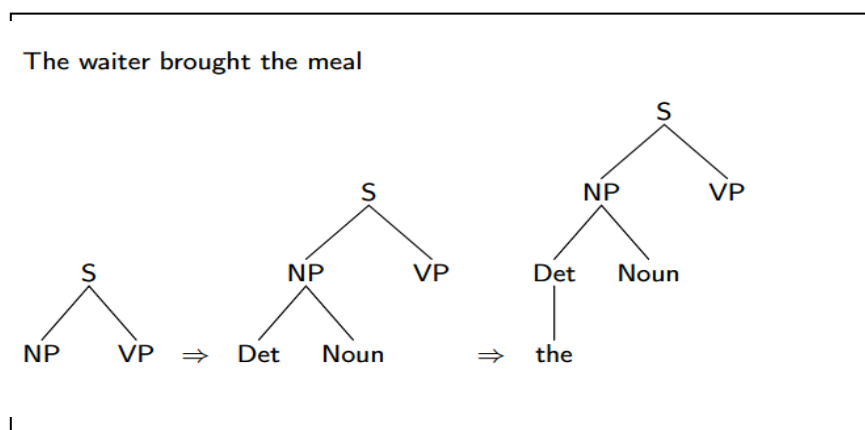


Fig 7: Top-down Parsing

- *Bottom-up parsing*: It begin with the sentence to be parsed and apply the grammar rules backward until a single tree whose terminals are the words of the sentence and whose top node is the start symbol has been produced as shown in figure 8. (i.e. Bottom-up parsing starts with the words and tries to find symbols that generate them).

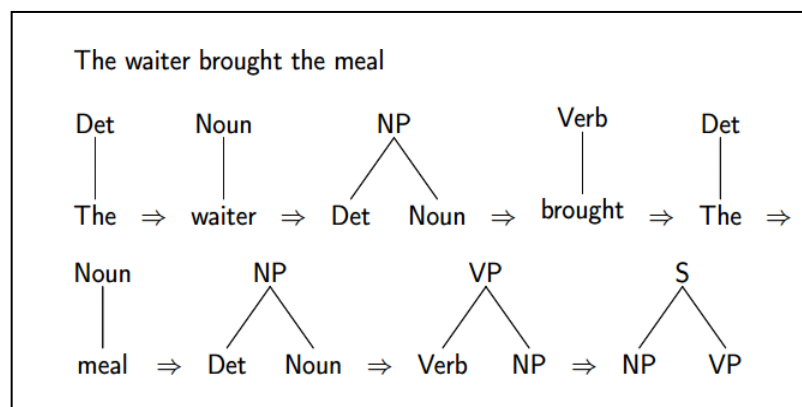


Fig 8: Bottom-up Parsing [12]

### Q). Semantic Analysis and Representation Structure:

Semantic Analysis is a subfield of Natural Language Processing (NLP) that attempts to understand the meaning of Natural Language. Understanding Natural Language might seem a straightforward process to us as humans. However, due to the vast complexity and subjectivity involved in human language, interpreting it is quite a complicated task for machines. Semantic Analysis of Natural Language captures the meaning of the given text while taking into account context, logical structuring of sentences and grammar roles.

#### Elements of Semantic Analysis

Followings are some important elements of semantic analysis –

##### Hyponymy :

It may be defined as the relationship between a generic term and instances of that generic term. Here the generic term is called hypernym and its instances are called hyponyms. For example, the word color is hypernym and the color blue, yellow etc. are hyponyms.

##### Homonymy:

It may be defined as the words having same spelling or same form but having different and unrelated meaning. For example, the word “Bat” is a homonymy word because bat can be an implement to hit a ball or bat is a nocturnal flying mammal also.

##### Polysemy:

Polysemy is a Greek word, which means “many signs”. It is a word or phrase with different but related sense. In other words, we can say that polysemy has the same spelling but different and related meaning. For example, the word “bank” is a polysemy word having the following meanings –

- A financial institution.
- The building in which such an institution is located.
- A synonym for “to rely on”.

#### Building Blocks of Semantic System

In word representation or representation of the meaning of the words, the following building blocks play an important role –

- **Entities** – It represents the individual such as a particular person, location etc. For example, Haryana, India, Ram all are entities.
- **Concepts** – It represents the general category of the individuals such as a person, city, etc.

- **Relations** – It represents the relationship between entities and concept. For example, Ram is a person.
- **Predicates** – It represents the verb structures. For example, semantic roles and case grammar are the examples of predicates.

### Approaches to Meaning Representations :

Semantic analysis uses the following approaches for the representation of meaning –

- First order predicate logic (FOPL)
- Semantic Nets
- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

### Q).Natural Language Generation (NLG):

Natural language generation (NLG) is the use of artificial intelligence (AI) programming to produce written or spoken narratives from a data set. NLG is related to human-to-machine and machine-to-human interaction, including computational linguistics, natural language processing (NLP) and natural language understanding (NLU).

#### How NLG works

NLG is a multi-stage process, with each step further refining the data being used to produce content with natural-sounding language. The six stages of NLG are as follows:

1. **Content analysis.** Data is filtered to determine what should be included in the content produced at the end of the process. This stage includes identifying the main topics in the source document and the relationships between them.
2. **Data understanding.** The data is interpreted, patterns are identified and it's put into context. Machine learning is often used at this stage.
3. **Document structuring.** A document plan is created and a narrative structure chosen based on the type of data being interpreted.
4. **Sentence aggregation.** Relevant sentences or parts of sentences are combined in ways that accurately summarize the topic.
5. **Grammatical structuring.** Grammatical rules are applied to generate natural-sounding text. The program deduces the syntactical structure of the sentence. It then uses this information to rewrite the sentence in a grammatically correct manner.
6. **Language presentation.** The final output is generated based on a template or format the user or programmer has selected.

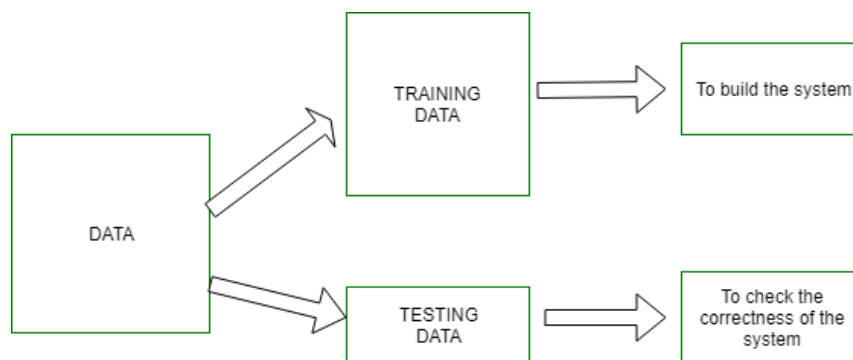
#### How is NLG used

Natural language generation is being used in an array of ways. Some of the many uses include the following:

- generating the responses of chatbots and voice assistants such as Google's Alexa and Apple's Siri;
- converting financial reports and other types of business data into easily understood content for employees and customers;
- automating lead nurturing email, messaging and chat responses;
- personalizing responses to customer emails and messages;
- generating and personalizing scripts used by customer service representatives;
- aggregating and summarizing news reports;
- reporting on the status of internet of things devices; and
- creating product descriptions for e-commerce webpages and customer messaging.

### Pattern Recognition

**Pattern recognition** is the process of recognizing patterns by using a machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of pattern recognition is its application potential.



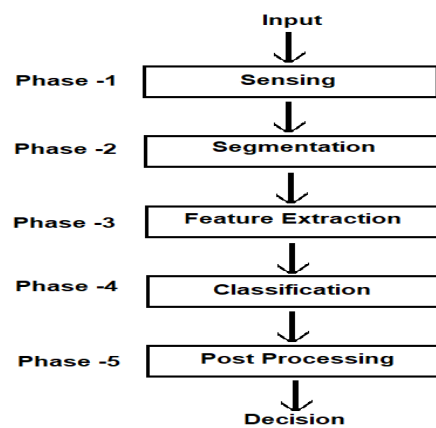
**Pattern recognition possesses the following features:**

- Pattern recognition system should recognize familiar patterns quickly and accurately
- Recognize and classify unfamiliar objects
- Accurately recognize shapes and objects from different angles
- Identify patterns and objects even when partly hidden
- Recognize patterns quickly with ease, and with automaticity.

**The Recognition and classification Process:**

Approaches for Pattern Recognition Systems can be represented by distinct phases, as Pattern Recognition Systems can be divided into the following components.

- **Phase 1:** Convert images or sounds or other inputs into signal data.
- **Phase 2:** Isolate the sensed objects from the background.
- **Phase 3:** Measure objects' properties that are useful for classification.
- **Phase 4:** Assign the sensed object to a category.
- **Phase 5:** Take other considerations to decide on appropriate action.



### Phases in Pattern Recognition

Problems solved by

these Phases are as follows:

1. **Sensing:** It deals with problem arises in the input such as its bandwidth, resolution, sensitivity, distortion, signal-to-noise ratio, latency, etc.
2. **Segmentation and Grouping:** Deepest problems in pattern recognition that deals with the problem of recognizing or grouping together the various parts of an object.
3. **Feature Extraction:** It deals with the characterization of an object so that it can be recognized easily by measurements. Those objects whose values are very similar for the objects are considered to be in the same category, while those whose values are quite different for the objects are placed in different categories.
4. **Classification:** It deals with assigning the object to their particular categories by using the feature vector provided by the feature extractor and determining the values of all of the features for a particular input.
5. **Post Processing:** It deals with action decision-making by using the output of the classifier. Action such as minimum-error-rate classification will minimize the total expected cost.

### Activities for designing the Pattern Recognition Systems

There are various sequences of activities that are used for designing the Pattern Recognition Systems. These activities are as follows:

- Data Collection
- Feature Choice
- Model Choice
- Training
- Evaluation

### Learning Classification Patterns:

Pattern recognition involves the classification and cluster of patterns.

- In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.
- Clustering generated a partition of the data which helps decision making, the specific decision-making activity of interest to us. Clustering is used in unsupervised learning.

**Features** may be represented as continuous, discrete, or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

**Example:** consider our face then eyes, ears, nose, etc are features of the face.

A set of features that are taken together, forms the **features vector**.

**Applications:**

- **Image processing, segmentation, and analysis :**  
Pattern recognition is used to give human recognition intelligence to machines that are required in image processing.
- **Computer vision :**  
Pattern recognition is used to extract meaningful features from given image/video samples and is used in computer vision for various applications like biological and biomedical imaging.
- **Seismic analysis :**  
The pattern recognition approach is used for the discovery, imaging, and interpretation of temporal patterns in seismic array recordings. Statistical pattern recognition is implemented and used in different types of seismic analysis models.
- **Radar signal classification/analysis:**  
Pattern recognition and signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.
- **Speech recognition:**  
The greatest success in speech recognition has been obtained using pattern recognition paradigms. It is used in various algorithms of speech recognition which tries to avoid the problems of using a phoneme level of description and treats larger units such as words as pattern
- **Fingerprint identification:**  
Fingerprint recognition technology is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches are widely used.

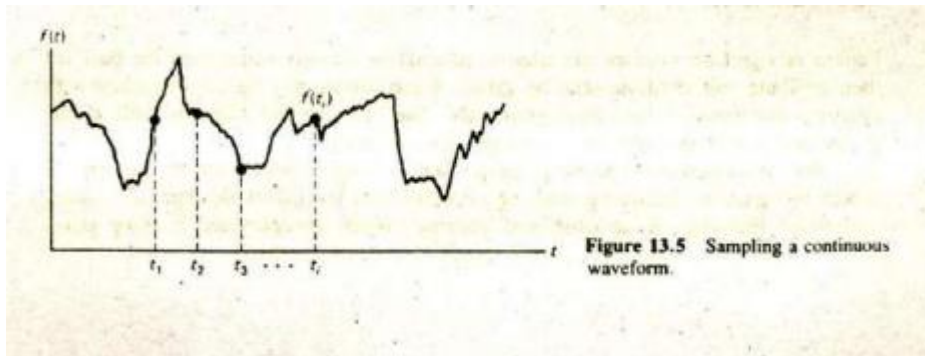
### **RECOGNIZING AND UNDERSTANDING SPEECH :**

Developing systems that understand speech has been a continuing goal of AI researchers. Speech is one of our most expedient and natural forms of communication, and so understandably, it is a capability we would like AI systems to possess. The ability to communicate directly with programs offers several advantages. It eliminates the need for keyboard entries and speeds up the interchange of information between user and system.

With speech as the communication medium, users are also free to perform other tasks concurrently with the computer interchange. And finally, more untrained personnel would be able to use computers in a variety of applications.

The recognition of continuous waveform patterns such as speech begins with sampling and digitizing the waveforms. In this case the feature values are the sampled points  $x_i = f(t_i)$  as illustrated in Figure. 13.5.





It is known from information theory that a sampling rate of twice the highest speech frequency is needed to capture the information content of the speech waveforms. Thus, sampling requirements will normally be equivalent to 20K to 30K bytes per second. While this rate of information in itself is not too difficult to handle, this, added to the subsequent processing, does place some heavy requirements on real time understanding of speech.

Following sample digitization, the signals are processed 'at different levels of abstraction. The lowest level deals with phones (the smallest unit of sound), allophones (variations of the phoneme as they actually occur in words), and syllables. Higher level processing deals with words, phrases, and sentences.

### Digital Image Processing System or Visual Image Understanding

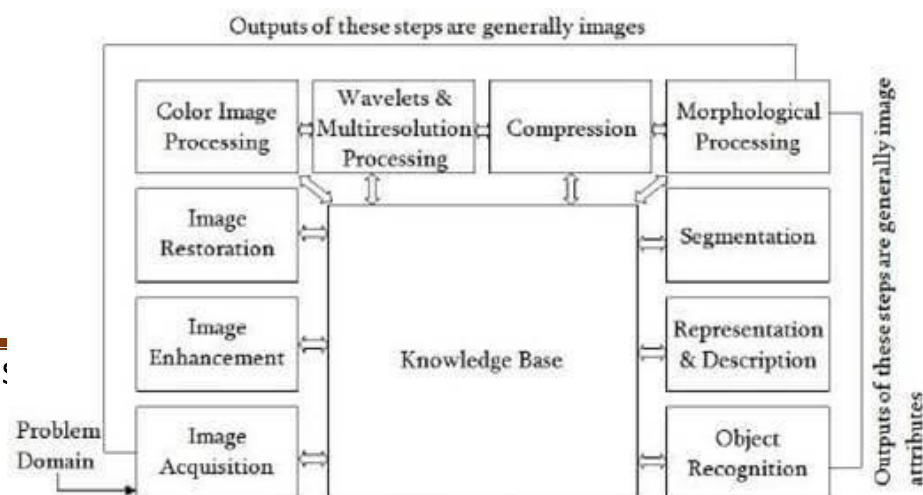
In computer science, digital image processing uses algorithms to perform image processing on digital images to extract some useful information. Digital image processing has many advantages as compared to analog image processing. Wide range of algorithms can be applied to input data which can avoid problems such as noise and signal distortion during processing. As we know, images are defined in two dimensions, so DIP can be modeled in multidimensional systems.

Purpose of Image processing

The main purpose of the DIP is divided into following 5 groups:

1. **Visualization:** The objects which are not visible, they are observed.
2. **Image sharpening and restoration:** It is used for better image resolution.
3. **Image retrieval:** An image of interest can be seen
4. **Measurement of pattern:** In an image, all the objects are measured.
5. **Image Recognition:** Each object in an image can be distinguished.

Following are Fundamental Steps of Digital Image Processing:



### 1. Image Acquisition

Image acquisition is the first step of the fundamental steps of DIP. In this stage, an image is given in the digital form. Generally, in this stage, pre-processing such as scaling is done.

### 2. Image Enhancement

Image enhancement is the simplest and most attractive area of DIP. In this stage details which are not known, or we can say that interesting features of an image is highlighted. Such as brightness, contrast, etc...

### 3. Image Restoration

Image restoration is the stage in which the appearance of an image is improved.

### 4. Color Image Processing

Color image processing is a famous area because it has increased the use of digital images on the internet. This includes color modeling, processing in a digital domain, etc....

### 5. Wavelets and Multi-Resolution Processing

In this stage, an image is represented in various degrees of resolution. Image is divided into smaller regions for data compression and for the pyramidal representation.

### 6. Compression

Compression is a technique which is used for reducing the requirement of storing an image. It is a very important stage because it is very necessary to compress data for internet use.

### 7. Morphological Processing

This stage deals with tools which are used for extracting the components of the image, which is useful in the representation and description of shape.

### 8. Segmentation

In this stage, an image is a partitioned into its objects. Segmentation is the most difficult tasks in DIP. It is a process which takes a lot of time for the successful solution of imaging problems which requires objects to identify individually.

### 9. Representation and Description

Representation and description follow the output of the segmentation stage. The output is a raw pixel data which has all points of the region itself. To transform the raw data, representation is the only solution. Whereas description is used for extracting information's to differentiate one class of objects from another.

### 10. Object recognition

In this stage, the label is assigned to the object, which is based on descriptors.

## 11. Knowledge Base

Knowledge is the last stage in DIP. In this stage, important information of the image is located, which limits the searching processes. The knowledge base is very complex when the image database has a high-resolution satellite.

### Image Transformation:

An image is obtained in spatial coordinates (x, y) or (x, y, z). There are many advantages if the spatial domain image is transformed into another domain. In which solution of any problem can be found easily.

### Following are two types of transformations:

#### 1. Fourier Transform

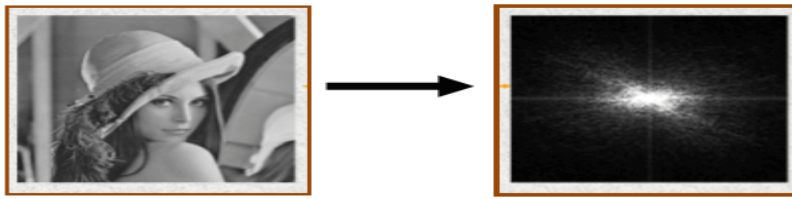
Fourier transform is mainly used for image processing. In the Fourier transform, the intensity of the image is transformed into frequency variation and then to the frequency domain. It is used for slow varying intensity images such as the background of a passport size photo can be represented as low-frequency components and the edges can be represented as high-frequency components. Low-frequency components can be removed using filters of FT domain. When an image is filtered in the FT domain, it contains only the edges of the image. And if we do inverse FT domain to spatial domain then also an image contains only edges. Fourier transform is the simplest technique in which edges of the image can be fined.

### Two Dimensional Fourier Transform

$$\left. \begin{aligned}
 v(k, l) &= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m, n) W_N^{km} W_N^{ln}, \quad W_N \triangleq \exp\left(\frac{-j2\pi}{N}\right) \\
 u(m, n) &= \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k, l) W_N^{-km} W_N^{-ln} \\
 v(k, l) &= \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m, n) W_N^{km} W_N^{ln} \\
 u(m, n) &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k, l) W_N^{-km} W_N^{-ln}
 \end{aligned} \right\} \text{Unitary DFT Pair}$$

### Properties of Fourier transformation are as follows:

- Symmetric Unitary
- Periodic Extension
- Sampled Fourier
- Fast
- Conjugate Symmetry
- Circular Convolution



**Fourier transformation of the image**

For Example:

Discrete Cosine Transformation (DCT)

In Discrete Cosine Transformation, coefficients carry information about the pixels of the image. Also, much information is contained using very few coefficients, and the remaining coefficient contains minimal information. These coefficients can be removed without losing information. By doing this, the file size is reduced in the DCT domain. DCT is used for lossy compression.

**One Dimension Discrete cosine transformation:**

$$\begin{aligned}
 C &= \{c(k, n)\} \\
 c(k, n) &= \begin{cases} \frac{1}{\sqrt{N}} & k = 0, 0 \leq n \leq N-1 \\ \frac{2}{\sqrt{N}} \cos\left(\frac{\pi(2n+1)k}{2N}\right) & 1 \leq k \leq N-1, 0 \leq n \leq N-1 \end{cases} \\
 v(k) &= \alpha(k) \sum_{n=0}^{N-1} u(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad 0 \leq k \leq N-1 \\
 \alpha(0) &\triangleq \frac{1}{\sqrt{N}}, \quad \alpha(k) \triangleq \frac{2}{\sqrt{N}}, \quad 1 \leq k \leq N-1 \\
 u(n) &= \sum_{k=0}^{N-1} \alpha(k) v(k) \cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad 0 \leq n \leq N-1
 \end{aligned}$$

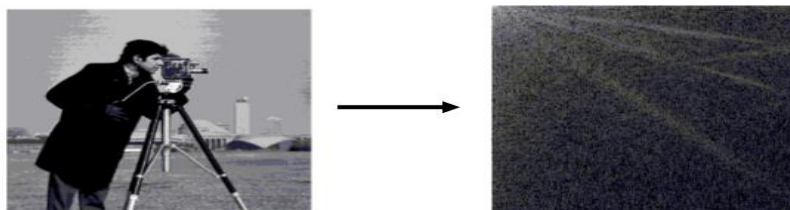
**Two Dimension Discrete cosine transformations:**

$$-A = A^* = C$$

**Properties of Discrete cosine transformation are as following:**

- Real and Orthogonal:  $C=C^* \rightarrow C^{-1}=C^T$
- Not! Real part of DFT
- Fast Transform
- Excellent Energy Compaction (Highly Correlated Data)

For Example:



**Applications of image transforms are as follows:**

- Fourier transform is used for Edge Detection.
- Discrete Cosine Transform is used for image compression.

**Levels of Image Processing:**

In general, there are three levels of processing or three types of processes in digital image processing namely: low, mid and high-level processes

### **Low-Level Image Processing**

- Low-Level processing operation involves tasks such as image preprocessing to reduce noise, contrast enhancement, image sharpening, etc.
- In the low-level process, both input and output are images.

### **Mid-Level Image Processing**

- Mid-Level processing involves tasks such as image segmentation, description of images, object recognition, etc.
- In the mid-level process, inputs are generally images but its outputs are generally image attributes.

### **High-Level Image Processing**

- High-Level processing involves making sense from a group of recognized objects.
- This process is normally associated with computer vision.

## **UNIT IV** **EXPERT SYSTEM ARCHITECTURE**

### **Introduction:**

An expert system is a set of programs that manipulate encoded knowledge to solve problems in a specialized domain that normally requires human expertise. An expert system's knowledge is obtained from expert sources and coded in a form suitable for the system to use in its inference or reasoning processes.

The expert knowledge must be obtained from specialists or other sources of expertise, such as texts, journal articles, and data bases. This type of knowledge usually requires much training and experience in some specialized field such as medicine, geology, system configuration, or engineering design. Once a sufficient body of expert knowledge has been acquired, it must be encoded in some form, loaded into a knowledge base, then tested, and refined continually throughout the life of the system.

### **RULE BASED SYSTEM ARCHITECTURE**

- The most common form of architecture used in expert and other types of knowledge based systems is the production system, also called the Rule based system.
- This type of system uses knowledge encoded in the form of production rules, that is if... then rules.
- The condition part resides at left hand side and conclusion or action part resides at right hand side.

IF: condition – 1 and condition – 2 and condition – 3  
THEN: Take Action – 4

IF: The temperature is greater than 200 degrees, and The water level is low  
THEN: Open the safety valve

$A \ \& \ B \ \& \ C \ \& \ D \rightarrow E \ \& \ F$

- The main components of a typical expert system is as shown below:

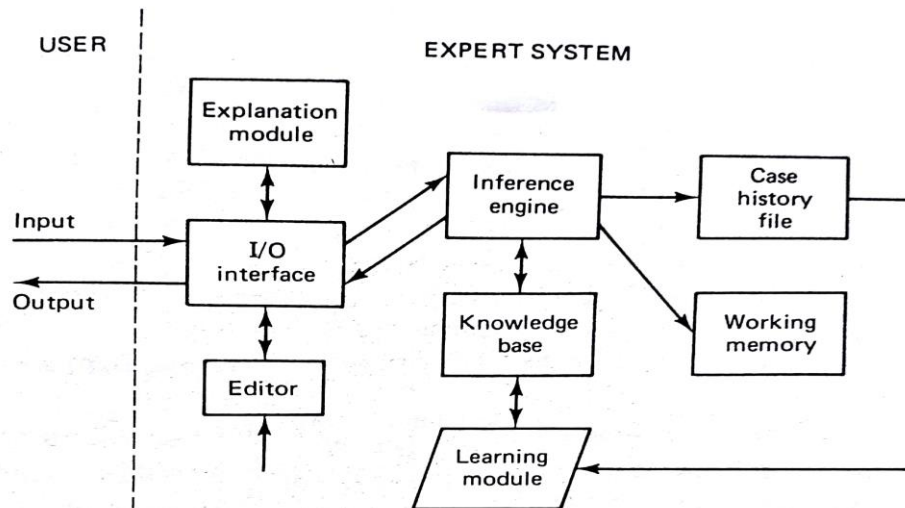


Figure 15.1 Components of a typical expert system.

### The Knowledge Base:

- The knowledge base contains some facts and rules about some specialized knowledge domain.
- A LISP (List Processing Programming Language) format example of simple knowledge base giving family relationships is as shown below:

```

((a1 (male bob))
 (a2 (female sue))
 (a3 (male sam))
 (a4 (male bill))
 (a5 (female pam))
 (r1 ((husband ?x ?y))
  →
  (male ?x))
 (r2 ((wife ?x ?y))
  →
  (female ?x))
 (r3 ((wife ?x ?y))
  →
  (husband ?y ?x))
 (r4 ((mother ?x ?y)
  (husband ?z ?x))
  →
  (father ?z ?y))
 (r5 ((father ?x ?y)
  (wife ?z ?x))
  →
  (mother ?z ?y))
 (r6 ((husband ?x ?y))
  →
  (wife ?y ?x))
 (r7 ((father ?x ?z)
  (mother ?y ?z))
  →
  (husband ?x ?y))
 (r8 ((father ?x ?z)
  (mother ?y ?z))
  →
  (wife ?y ?z))
 (r9 ((father ?x ?y)
  (father ?y ?z))
  →
  (grandfather ?x ?z)))

```

Figure 15.2 Facts and rules in a simple knowledge base.



- Each fact and rule is identified with a name (a1, a2, ..., r1, r2, ...).
- Variables are identified as a symbol preceded by a question mark.
- In PROLOG, rules are written as clauses with both head and body. For example, a rule about a patient's symptoms and corresponding diagnosis of hepatitis might read in English as the rule.

**IF:** The patient has a chronic disorder, and  
the sex of the patient is female, and  
the age of the patient is less than 30, and  
the patient shows condition A, and  
test B reveals biochemistry condition C

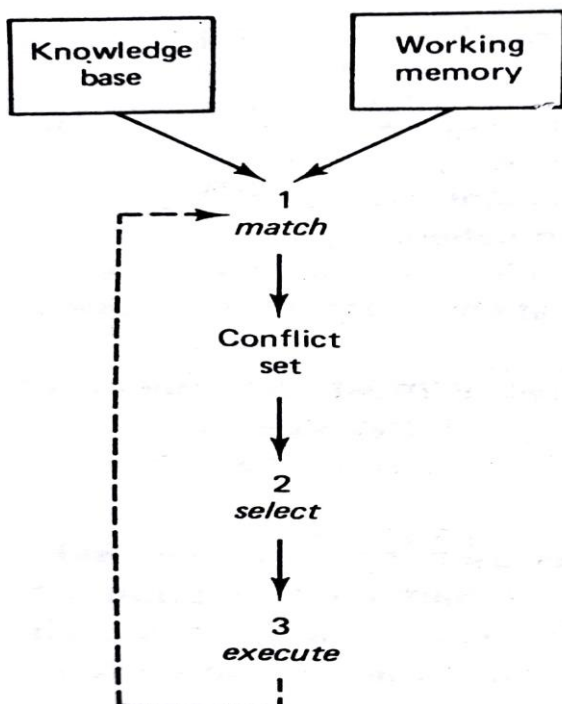
**THEN:** conclude the patient's diagnosis is autoimmune-chronic-hepatitis.

This rule could be written straightaway in PROLOG as

```
conclude(patient, diagnosis, autoimmune_chronic_hepatitis):-
    same(patient, disorder, chronic),
    same(patient, sex, female),
    lessthan(patient, age, 30),
    same(patient, symptom_a, value_a),
    same(patient, biochemistry, value_c).
```

Note that PROLOG rules have at most one conclusion clause.

### 1.1. The Inference Process:



**Figure 15.3** The production system inference cycle.

### **The Inference Process:**

The inference engine accepts user input queries and responses to questions through the I/O interface and uses this dynamic information together with the static knowledge (the rules and facts) stored in the knowledge base. The knowledge in the knowledge base is used to derive conclusions about the current case or situation as presented by the user's input.

The inferring process is carried out recursively in three stages: (1) match, (2) select, and (3) execute. During the match stage, the contents of working memory are compared to facts and rules contained in the knowledge base. When consistent matches are found, the corresponding rules are placed in a conflict set. To find an appropriate and consistent match, substitutions (instantiations) may be required.

Once all the matched rules have been added to the conflict set during a given cycle, one of the rules is selected for execution. The criteria for selection may be most recent use, rule condition specificity. (the number of conjuncts on the left), or simply the smallest rule number. The selected rule is then executed and the right-hand side or action part of the rule is then carried out.

### **NON-PRODUCTION SYSTEM ARCHITECTURES**

Instead of rules, these systems employ more structured representation schemes like associative or semantic networks, frame and rule structures, decision trees, or even specialized networks like neural networks.

#### **Associative or Semantic Network Architectures**

Associative network representations are especially useful in depicting hierarchical knowledge structures, where property inheritance is common. Objects belonging to a class of other objects may inherit many of the characteristics of the class. Inheritance can also be treated as a form of default reasoning. This facilitates the storage of information when shared by many objects as well as the interfacing process.

Associative network representations are not a popular form of representation for standard expert systems. More often, these network representations are used in natural language or computer vision systems or in conjunction with some other form of representation. One expert system based on the use of an associative network representation is CASNET (Causal Associational Network), which was developed at Rutgers University during the early 1970s (Weiss et al, 1978).

The network in CASNET is divided into three planes or types of knowledge as depicted in Figure 15.4.

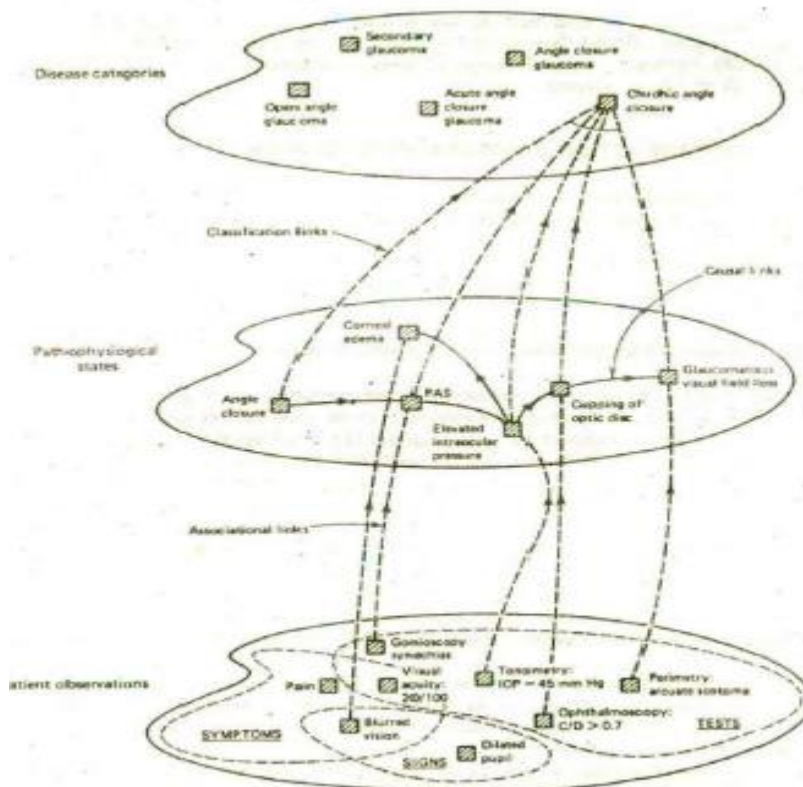


Figure 15.4 Levels of network description in CASNET. (From Artificial Intelligence Journal, Vol. II p. 148, 1978. By permission.)

### Frame Architectures

Frames are structured sets of closely related knowledge, such as an object or concept name, the object's main attributes and their corresponding values, and possibly some attached procedures (if-needed, if-added, if-removed procedures). The attribute, values, and procedures are stored in specified slots and slot facets of the frame. Individual frames are usually linked together as a network much like the nodes in an associative network. Thus, frames may have many of the features of associative networks, namely, property inheritance and default reasoning.

Several expert systems have been constructed with frame architectures, and a number of building tools which create and manipulate frame structured systems have been developed. The medical knowledge in PIP is organized in frame structures, where each frame is composed of categories of slots with names such as

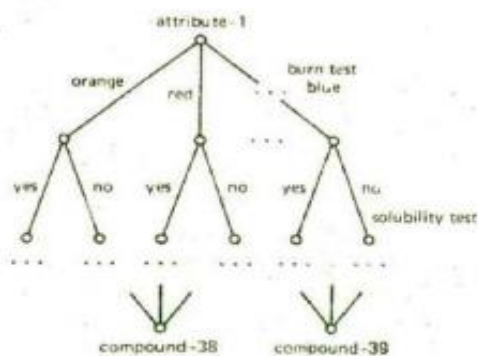
- Typical findings
- Logical decision criteria
- Complimentary relations to other frames
- Differential diagnosis
- Scoring

The patient findings are matched against frames, and when a close match is found, a trigger status occurs. A trigger is a finding that is so strongly related to a disorder that the system regards it as an active hypothesis, one to be pursued further. A special is-sufficient slot is used to confirm the presence of a disease when key findings correlate with the slot contents.

### Decision Tree Architectures

The knowledge base, which is the decision tree for an identification system, can be constructed with a special tree-building editor or with a learning module. In either case, a set of the most discriminating attributes for the class of objects being identified should be selected. Only those attributes that discriminate well among different objects need be used.

Permissible values for each of the attributes are grouped into separable sets, and each such set determines a branch from an attribute node to the next node. New nodes and branches can be added to the tree when additional attributes are needed to further discriminate among new objects.



15.5 A segment of a decision tree structure.

### Neural Network Architectures

Neural networks are large networks of simple processing elements or nodes which process information dynamically in response to external inputs. The nodes are simplified models of neurons. The knowledge in a neural network is distributed throughout the network in the form of inter node connections and weighted links which form the inputs to the nodes.

Neural networks were originally inspired as being models of the human nervous system. They are greatly simplified models to be sure (neurons are known to be fairly complex processors). Even so, they have been shown to exhibit many "intelligent" abilities, such as learning, generalization, and abstraction. A single node is illustrated in Figure 15.7.

The inputs to the node are the values  $x_1, x_2, \dots, x_n$ , which typically take on values of  $-1$  to  $0$ . The weights  $w_1, w_2, \dots, w_n$  correspond to the synaptic strengths of a neuron. They serve to increase or decrease the effects of the corresponding  $x_i$  input values. The sum of the products  $x_i w_i$ ,  $i = 1, 2, \dots, n$ , serve as the total combined input to the node.

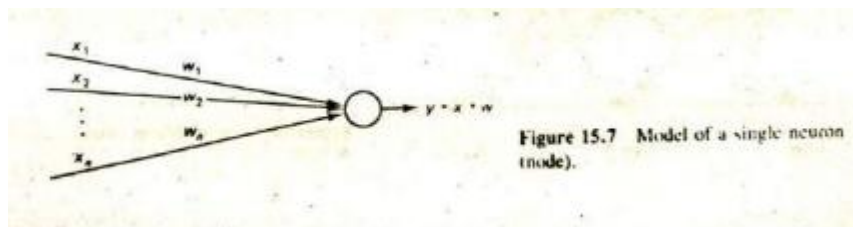


Figure 15.7 Model of a single neuron (node).

Figure 15.8 illustrates three layers of a number of interconnected nodes. The first layer serves as the input layer, receiving inputs from some set of stimuli. The second layer (called the hidden layer) receives inputs from the first layer and produces a pattern of inputs to the third layer, the output layer. The pattern of outputs from the final layer are the network's responses to the input stimuli patterns. Input links to layer = 1, 2, 3 have weights  $w$  for  $i = 1, 2, \dots, n$ .

A neural network can be thought of as a black box that transforms the input vector  $x$  to the output vector  $y$  where the transformation performed is the result of the pattern of connections and weights, that is, according to the values of the weight matrix  $W$ .

Consider the vector product

$$x \cdot w = \sum x_i w_i$$

There is a geometric interpretation for this product. It is equivalent to projecting one vector onto the other vector in  $n$ -dimensional space. This notion is depicted in Figure 15.9 for the two-dimensional case. The magnitude of the resultant vector is given by

$$x \cdot w = |x||w| \cos \theta$$

where  $|x|$  denotes the norm or length of the vector  $x$ . Note that this product is maximum when both vectors point in the same direction, that is, when  $\theta = 0$ .

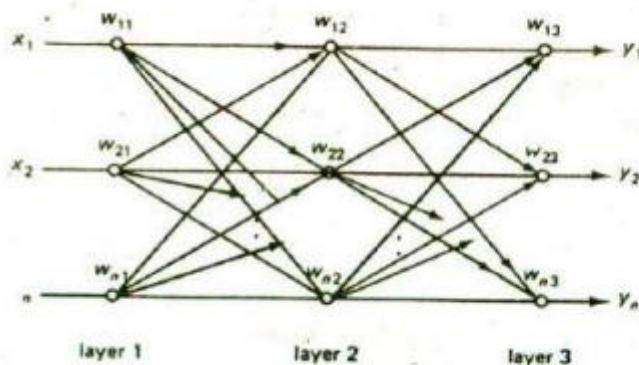


Figure 15.8 A multilayer neural network

## KNOWLEDGE ACQUISITION AND VALIDATION:

The knowledge elicitation process is depicted in Figure 15.11. To elicit the requisite knowledge, a knowledge engineer conducts extensive interviews with domain experts. During the interviews, the expert is asked to solve typical problems in the domain of interest and to explain his or her solutions.

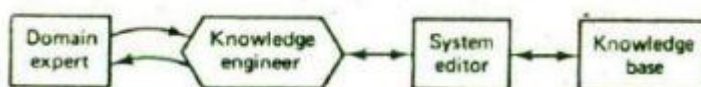


Figure 15.11 The knowledge acquisition process.

Using the knowledge gained from experts and other sources, the knowledge engineer codes the knowledge in the form of rules or some other representation scheme. This knowledge is then used to solve sample problems for review and validation by the experts. Errors and omissions are uncovered and corrected, and additional knowledge is added as needed. The process is repeated until a sufficient body of knowledge has been collected to solve a large class of problems in the chosen domain. The whole process may take as many as tens of person years.

Penny Nit, an experienced knowledge engineer at Stanford University, has described some useful practices to follow in solving acquisition problems through a sequence of heuristics she uses



You can't be your own expert. By examining the process of your own expertise our risk becoming like the Centipede who got tangled up in her own legs and stopped dead when she tried to figure out how she moved a hundred legs in harmony.

From the beginning, the knowledge engineer must count on throwing efforts away. Writers make drafts, painters make preliminary sketches; knowledge engineers are no different.

The problem must be well chosen. AI is a young field and isn't ready to take on every problem the world has to offer. Expert systems work best when the problem is well bounded, which is computer talk to describe a problem for which large amounts of specialized knowledge may be needed, but not a general knowledge of the world.

If you want to do any serious application you need to meet the expert more than half way; if he's had no exposure to computing, your job will be that much harder.

If none of the tools you normally use works, build a new one.

Dealing with anything but facts implies uncertainty. Heuristic knowledge is not hard and fast and cannot be treated as factual. A weighting procedure has to be built into the expert system to allow for expressions such as "I strongly believe that ..." or "The evidence suggests that....."

A high-performance program, or a program that will eventually be taken over by the expert for his own use, must have very easy ways of allowing the knowledge to be modified so that new information can be added and out-of-date information deleted.

The problem needs to be a useful, interesting one. There are knowledge-based programs to solve arcane puzzles, but who cares? More important, the user has to understand the system's real value to his work.

### **General Concepts in Knowledge Acquisition:**

Definition:

Knowledge acquisition is the process of adding new knowledge to a knowledge base and refining or otherwise improving knowledge that was previously acquired. Acquisition is usually associated with some purpose such as expanding the capabilities of a system or improving its performance at some specified task.

Therefore, we will think of acquisition as goal oriented creation and refinement of knowledge. We take a broad view of the definition here and include autonomous acquisition, contrary to many workers in the field who regard acquisition solely as the process of knowledge elicitation from experts.

Acquired knowledge may consist of facts, rules, concepts, procedures, heuristics, formulas, relationships, statistics, or other useful information. Sources of this knowledge may include one or more of the following

- Experts in the domain of interest
- Textbooks
- Technical papers
- Databases

- Reports
- The environment

Table 16.1 depicts several types of knowledge and possible representation structures which by now should be familiar. In building a knowledge base, it is necessary to create or modify structures such as these for subsequent use by a performance component (like a theorem prover or an inference engine).

**TABLE 16.1 TYPES OF KNOWLEDGE AND POSSIBLE STRUCTURES**

Type of Knowledge	Examples of Structures
Facts	(snow color white)
Relations	(father_of john bill)
Rules	(if (temperature > 200 degrees) (open relief_valve))
Concepts	((forall (x y) (if (and (male x) ((brother_of x) (or (father_of y) (mother_of y))) (uncle x y))
Procedures, Plans, etc.	.....

### TYPES OF LEARNING:

The five different learning methods under this taxonomy are

- Memorization (rote learning)
- Direct instruction (by being told)
- Analogy
- Induction
- Deduction

Learning by memorization is the simplest form of learning. It requires the least amount of inference and is accomplished by simply copying the knowledge in the same form that it will be used directly into the knowledge base. We use this type of learning when we memorize multiplication tables, for example.

A slightly more complex form of learning is by direct instruction. This type of learning requires more inference than rote learning since the knowledge must be transformed into an operational form before being integrated into the knowledge base. We use this type of learning when a teacher presents a number of facts directly to us in a well organized manner

The third type Listed, analogical learning, is the process of learning a new concept or solution through the use of similar known concepts or solutions. We use this type of learning when solving problems on an exam where previously learned examples serve as a guide or when we learn to drive a truck using our knowledge of car driving.

The fourth type of learning is also one that is used frequently by humans. It is a powerful form of learning which, like analogical learning, also requires more inferring than the first two methods. This form of learning requires the use of inductive inference, a form of invalid but useful



inference. We use inductive learning when we formulate a general concept after seeing a number of instances or examples of the concept.

The final type of acquisition is deductive learning. It is accomplished through a sequence of deductive inference steps using known facts.. From the known facts. new facts or relationships are logically derived.

### **KNOWLEDGE ACQUISITION IS DIFFICULT:**

Early expert systems initially had a knowledge base consisting of a few hundred rules. This is equivalent to less than  $10^6$  bits of knowledge. In contrast, the capacity of a mature human brain has been estimated at some  $10^{15}$  bits of knowledge .

If we expect to build expert systems that are highly competent and possess knowledge in more than a single narrow domain, the amount of knowledge required for such knowledge bases will be somewhere between these two extremes, perhaps as much as  $10^{10}$  bits.

If we were able to build such systems at even ten times the rate these early systems were built, it would still require on the order of  $10^4$  person years. This estimate is based on the assumption that the time required is directly proportional to the size of the knowledge base, a simplified assumption. since the complexity of the knowledge and the interdependencies grow more rapidly with the size of the knowledge base.

Clearly, this rate of acquisition is not acceptable. We must develop better acquisition and learning methods before we can implement such systems within a realistic time frame.

Even with the progress made in the past few years through the development of special editors and related tools, more significant breakthroughs are needed before truly large knowledge bases can be assembled and maintained.

Because of this, we expect the research interest in knowledge acquisition and machine learning to continue to grow at an accelerated rate for some years in the future.

It has been stated before that a system's performance is strongly dependent on the level and quality of its knowledge, and that "in knowledge lies power." If we accept this adage, we must also agree that the acquisition of knowledge is of paramount importance and, in fact, that 'the real power lies in the ability to acquire new knowledge efficiently."

To build a machine that can learn and continue to improve its performance has been a long time dream of mankind. The fulfillment of that dream now seems closer than ever before with the modest successes achieved by AI researchers over the past twenty years

. We will consider the complexity problem noted above again from a different point of view when we study the different learning paradigms.

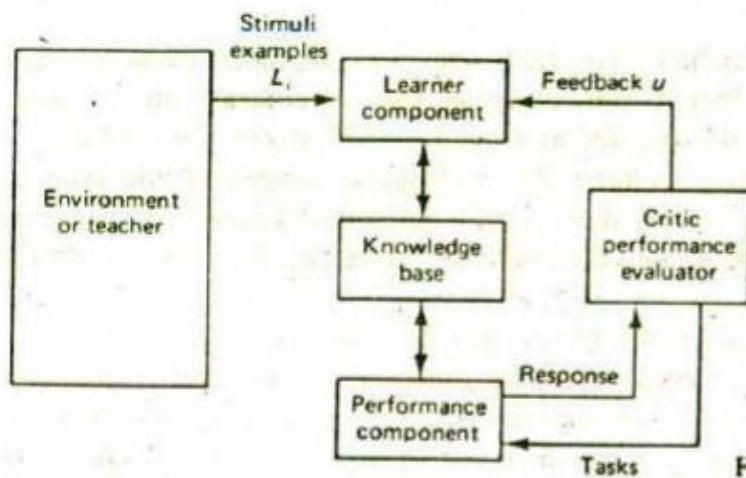
**GENERAL LEARNING MODEL:**

Figure 16.1 General learning model.

A general learning model is depicted in Figure 16.1 where the environment has been included as part of the overall learner system. The environment may be regarded as either a form of nature which produces random stimuli or as a more organized training source such as a teacher which provides carefully selected training examples for the learner component.

The actual form of environment used will depend on the particular learning paradigm. In any case, some representation language must be assumed for communication between the environment and the learner. The language may be the same representation scheme as that used in the knowledge base (such as a form of predicate calculus).

When they are chosen to be the same, we say the single representation trick is being used. This usually results in a simpler implementation since it is not necessary to transform between two or more different representations.

For some systems the environment may be a user working at a keyboard. Other systems will use program modules to simulate a particular environment. In even more realistic cases, the system will have real physical sensors which interface with some world environment.

Inputs to the learner component may be physical stimuli of some type or descriptive, symbolic training examples. The information conveyed to the learner component is used to create and modify knowledge structures in the knowledge base. This same knowledge is used by the performance component to carry out some tasks, such as solving a problem, playing a game, or classifying instances of some concept.

When given its task, the performance component produces a response describing its actions in performing the task. The critic module then evaluates this response relative to an optimal response.

Feedback, indicating whether or not the performance was acceptable, is then sent by the critic module to the learner component for its subsequent use in modifying the structures in the knowledge base. If proper learning was accomplished, the system's performance will have improved with the changes made to the knowledge base.

### PERFORMANCE MEASURES:

some relative performance characteristics among the different learning methods we will be considering.

**Generality:** One of the most important performance measures for learning methods is the generality or scope of the method. Generality is a measure of the ease with which the method can be adapted to different domains of application. A completely general algorithm is one which is a fixed or self adjusting configuration that can learn or adapt in any environment or application domain. At the other extreme are methods which function in a single domain only. Methods which have some degree of generality will function well in at least a few domains.

**Efficiency:** The efficiency of a method is a measure of the average time required to construct the target knowledge structures from some specified initial structures. Since this measure is often difficult to determine and is meaningless without some standard comparison time, a relative efficiency index can be used instead. For example, the relative efficiency of a method can be defined as the ratio of the time required for the given method to the time required for a purely random search to find the target structures.

**Robustness:** Robustness is the ability of a learning system to function with unreliable feedback and with a variety of training examples, including noisy ones. A robust system must be able to build tentative structures which are subject to modification or withdrawal if later found to be inconsistent with statistically sound structures.

**Efficacy:** The efficacy of a system is a measure of the overall power of the system. It is a combination of the factors generality, efficiency, and robustness. We say that system A is more efficacious than system B if system A is more efficient, robust, and general than B.

**Ease of implementation:** Ease of implementation relates to the complexity of the programs and data structures and the resources required to develop the given learning system. Lacking good complexity metrics, this measure will often be somewhat subjective.