LECTURENOTES.IN

Artificial Intelligence

Contributed By:
Sahil Kumar

Disclaimer

This document may not contain any originality and should not be used as a substitute for prescribed textbook. The information present here is uploaded by contributors to help other users. Various sources may have been used/referred while preparing this material. Further, this document is not intended to be used for commercial purpose and neither the contributor(s) nor LectureNotes.in is accountable for any issues, legal or otherwise, arising out of use of this document. The contributor makes no representation or warranties with respect to the accuracy or completeness of the contents of this document and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. By proceeding further, you agree to LectureNotes.in Terms of Use. Sharing of this document is forbidden under LectureNotes Term of use. Sharing it will be meant as violation of LectureNotes Terms of Use.

This document was downloaded by: Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone number 9999234890 and email deepgargak2010@gmail.com on 02nd Nov, 2019. and it may not be used by anyone else.

At LectureNotes.in you can also find

1. Previous Year Questions for BPUT

2. Notes from best faculties for all subjects

3. Solution to Previous year Questions

www.lecturenotes.in

Artificial Intelligence

MCA - 405

Unit-I

Topics Covered:

1.) Introduction To Artificial Intelligence
2.) Historical Aspects of AI
3.) Applications of AI
4.) Components of AI system
5) Problem Representation in AI.
6) Introduction to Logic.
7.) Propopositional Logic
8.) Predicate Logic.
9.) Rules of Inference
10) Resolution
11.) Unification, Unification Algorithm.
12.) Knowledge Representation
13) Types of Knowledge Representation
14.) Schemes for Knowledge Representation
- Semantic Networks
- Conceptual Graphs
- Frames
- Scripts
- Conceptual Dependency

Nidhi Kalra
Assistant Professor
MCA Department.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence

Topic:

Artificial Intelligency

Contributed By:

Sahil Kumar

Artificial Intelligence

Unit-I

Intelligence: → It is the ability to reason, to trigger new thoughts, to perceive and to learn.

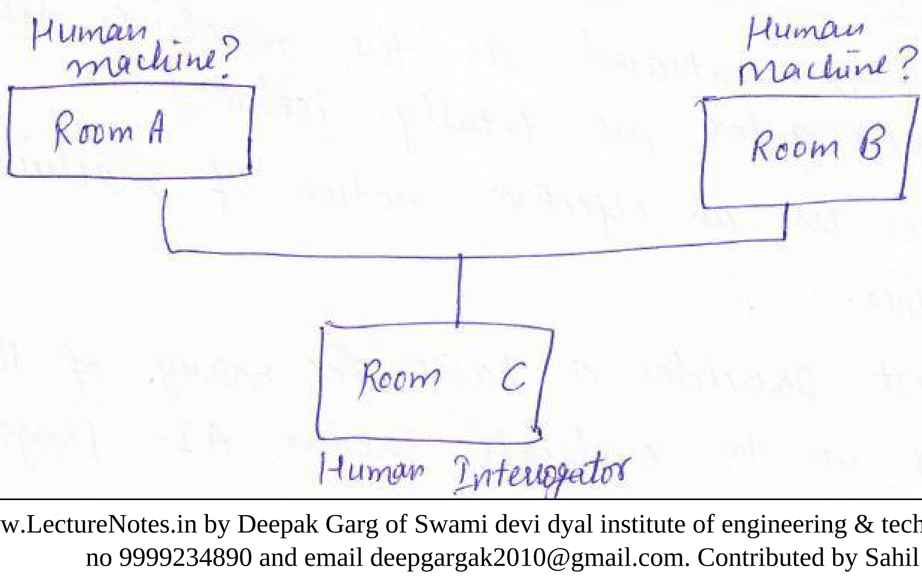
· **Artificial Intelligence:** → So AI refers as "ability to learn, to trigger new thoughts & to reason" artificially.

→ Hence AI can be defined as developing computer programs to solve problems by application of processes that are analogous to human reasoning process."

* **Turing Test:** → Turing in 1950, published an article, which triggered a controversial topic "Can a machine think?"

Turing proposed an 'imitation game' which was later modified to "Turing Test".

In the imitation game, the players are three humans – a male, a female and an interrogator. The interrogator who is shielded from the other two, asks questions to both of them and based on their typewritten answers determines who is male and who is female.



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

The aim of the male is to imitate the female and deceive the interrogator and the role of female is to provide replies that would inform the interrogator about her true sex.

The Turing Test is a variation of this imitation game where the human interrogator now communicates with the players via a teletype - one human and the other a machine.

Turing proposed that if the human interrogator in Room C is not able to identify who is in Room A or in Room B, then the machine possesses intelligence.

Turing Test has some criticism for the following reasons:

(1.) The computers can only do as they are told and can never perform intelligent actions on their own.

(2.) It is impossible to create a set of rules that will tell an individual exactly what to do under every possible set of circumstances.

Important Features of Turing Test:→

1. This test is unbiased as the machine, human and interrogator are totally isolated.

2. It gives us an objective notion of machine intelligence.

3. This test provides a basis for many of the schemes actually use to evaluate modern A.I. programs.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Conclusion :→ The computer can never imitate a human in each and every situation, it can only do for what it is programmed.

Historical Aspects of AI (History) :→

(1.) In 17th and 18th century the concept of A.I. begins with myths and stories and rumours that machines can have intelligence and conciousne

(2.) In the middle of 20th century, a handful of scientist begin to explore a new approach to their asian idea, based on their discoveries in neurology a new mathematical theory of information and understanding of control and communication called "Cybernitix".

(3.) In year 1950, a mathematician named Alan Turing gave after his own name "Turing Test" which began the notion of A.I. as machine intelligence.

(4.) The term artificial intelligence was first coined in year 1956 at Dartmouth conference.

(5.) In 1970, expert system comes into the picture. These expert systems had the potential to interpret, to formulate tools and to forecast stock market etc. In the same year many new methods were tested in the development of A.I. such as Minsky's Frame.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

6 Another development during this time was PROLOG language. Fuzzylogy first appeared in U.S. as the unique ability to make decision under certain conditions.

Applications in the field of Artificial Intelligence:→

1.) Pattern Recognition:→ Pattern Recognition in AI is the research area that studies the operation and design of systems that recognize patterns in data.
for eg.→ A vision program may try to match a pattern of eyes and nose in a scene in order to find a face.

(a.) Fraud Detection and Prevention:→ AI performs a really very good tasks for the bankers. If your card has been queried, its probably because more banks are now using artificial intelligence software to try to detect fraud.

(b.) Face Recognition:→ It is used to unlock the machine without the need to enter a password via the keyboard. This prevents others from using the computer because their faces are not likely to match the original user's stored face model.

(c.) Handwriting Recognition:→ It is one of the most promising methods of interacting with small portable computing devices, such as personal digital assistants, is the use of handwriting in AI.
In order to make this communication method more natural, they proposed to observe visually the writing process on an ordinary paper and to automatically recover the pen trajectory from numerical tablet sequences.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Conclusion :→ The computer can never imitate a human in each and every situation, it can only do for what it is programmed.

Historical Aspects of AI (History):→

(1.) In 17^{th} and 18^{th} century the concept of A.I. begins with myths and stories and rumours that machines can have intelligence and Conciousness.

(2.) In the middle of 20th century, a handful of scientist begin to explore a new approach to their asian idea, based on their discoveries in neurology a new mathematical theory of information and understanding of control and communication called "Cybernitix".

(3.) In year 1950, a mathematician named Alan Turing gave after his own name "Turing Test" which began the notion of A.I. as machine intelligence.

(4.) The term artificial intelligence was first coined in year 1956 at Dartmouth conference.

(5.) In 1970, expert system comes into the picture. These expert systems had the potential to interpret, to formulate tools and to forecast stock market etc. In the same year many new methods were tested in the development of A.I. such as Minsky's Frame.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

6. Another development during this time was PROLOG language. Fuzzylogy first appeared in U.S. as the unique ability to make decision under certain conditions.

Applications in the field of Artificial Intelligence:→

1.) Pattern Recognition:→ Pattern Recognition in AI is the research area that studies the operation and design of systems that recognize patterns in data. ~~when~~
For eg.→ A vision program may try to match a pattern of eyes and nose in a scene in order to find a face.

(a.) Fraud Detection and Prevention:→ AI performs a really very good tasks for the bankers. If your card has been queried, its probably because more banks are now using artificial intelligence software to try to detect fraud.

(b.) Face Recognition:→ It is used to unlock the machine without the need to enter a password via the computer keyboard. This prevents others from using the computer because their faces are not likely to match the original user's stored face model.

(c.) Handwriting Recognition:→ It is one of the most promising methods of interacting with small portable computing devices, such as personal digital assistants, is the use of handwriting in AI.
In order to make this communication method more natural, they proposed to observe visually the writing proc on an ordinary paper and to automatically recover the ... from numerical tablet sequences.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

2. Bio-informatics:→ AI provides several powerful algorithms and techniques for solving important problem in bioinformatics and chemo-informatics.
The goal in applying AI to bio informatics and chemo-informatics is to extract useful information from the wealth of available data by building good probabilistic models.

a.) Data Mining:→ Data mining is an AI powered tool that can discover useful information within a DB that can then be used to improve actions.

b) Bio-Medical Informatics:→ Bio-medical informatics in the field of AI is a combination of the expertise of medical informatics in developing clinical applications and the focused principles that have background guided bio-informatics could create a synergy between the two areas of applications.

3. Expert Systems:→ Expert System in AI is the knowledge-based applications of artificial intelligence have enhanced productivity in business, science, engineering and the military.

(a.) Diagonosis and Trouble-Shooting:→ It explains the development and testing of a condition-monitoring sub-module of an integrated plant maintenance management application based on AI techniques, mainly knowledge-based systems, having several modules, sub-modules and sections.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(b) Intelligent Decision Support Systems:→ These systems have the potential to transform human decision making by combining research in artificial intelligence, information technology and systems engineering.

(c.) Process Monitoring and Control:→ Process monitoring and control a generic AI architecture for intelligent monitoring and control, suitable for application in multiple domain like in the domain of patient monitoring in a surgical intensive care unit. (SICU).

d.) EIA (Environmental Impact Assessment):→ Expert Systems are promising technologies that manage information demands and provide required expertise.

Additional advantages of using expert systems for EIA are:

(1.) Expert systems help users cope with large volumes of EIA work.

2.) Expert Systems deliver EIA expertise to the non-expert.

3.) Expert systems enhance user accountability for decisions reached.

4.) Expert systems provide a structured approach to EIA.

(4.) Computer Vision:→ Vision involves both the acquistion and processing of visual information. AI vision technology has made possible such applications as image stabilization, 3D modelling, image synthesis, surgical navigation, handwritten document recognition and vision based

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

2. Bio-informatics:→ AI provides several powerful algorithms and techniques for solving important problems in bioinformatics and chemo-informatics.
The goal in applying AI to bio informatics and chemo-informatics is to extract useful information from the wealth of available data by building good probabilistic models.

a.) Data Mining:→ Data Mining is an AI powered tool that can discover useful information within a DB that can then be used to improve actions.

b) Bio-Medical Informatics:→ Bio-medical informatics in the field of AI is a combination of the expertise of medical informatics in developing clinical applications and the focused principles that have background guided bio-informatics could create a synergy between the two areas of applications.

3. Expert Systems:→ Expert System in AI is the knowledge-based applications of artificial intelligence have enhanced productivity in business, science, engineering and the military.

(a.) Diagonosis and Trouble-Shooting:→ It explains the development and testing of a condition-monitoring sub-module of an integrated plant maintenance management application based on AI techniques, mainly knowledge-based systems, having several modules, sub-modules and sections.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(b) Intelligent Decision Support Systems:→ These systems have the potential to transform human decision making by combining research in artificial intelligence, information technology and systems engineering.

(c) Process Monitoring and Control:→ Process monitoring and control a generic AI architecture for intelligent monitoring and control, suitable for application in multiple domain like in the domain of patient monitoring in a surgical intensive care unit (SICU).

d) EIA (Environmental Impact Assessment):→ Expert systems are promising technologies that manage information demands and provide required expertise.

Additional advantages of using expert systems for EIA are:

(1) Expert systems help users cope with large volumes of EIA work.

2) Expert Systems deliver EIA expertise to the non-expert.

3) Expert systems enhance user accountability for decisions reached.

4) Expert systems provide a structured approach to EIA.

(4) Computer Vision:→ Vision involves both the acquistion and processing of visual information. AI vision technology has made possible such applications as image stabilization, 3D modelling, image synthesis, surgical navigation, handwritten document recognition and vision based

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Computer interfaces.

5. Image Processing :→ In AI, applications include video phone, teleconferencing and multimedia database. Increasingly, this research has combined image or vision with audio or speech.

6. Robotics :→ a) Cybernetics
b) Commercial and Research Applications
c) Sensors in Robots.

In the area of robotics, computers are now widely used in assembly plants, but they are capable only of very limited tasks.
In AI, the action capability is physically interacting with the environment; two types of sensors have to be used in any robotic system:

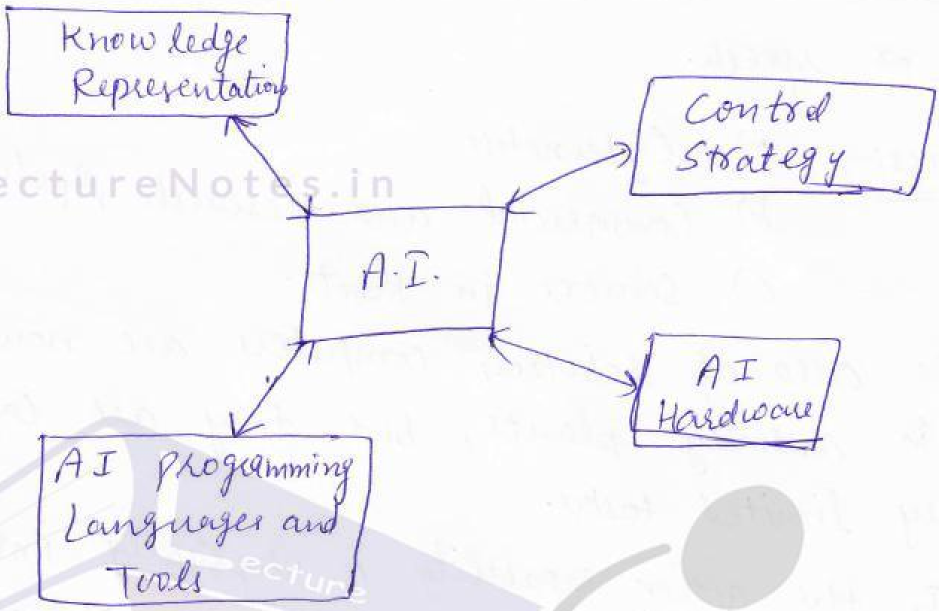
1. Proprio :→ Ceptors for the measurement of the robot's (internal) parameters.
2. Extero :→ ceptors for the measurement of its environmental (external) parameters.

7. Gaming :→ We can buy machines that can play master level chess for a few hundred dollars. There is some AI in them; ~~but they play with~~ Using AI, we can also beat world champion by brute-force and known reliable heuristics requires being able to look at 200 million positions per second.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Components of AI System:→

Any AI system has four major components which are explained as follows.



(1.) **Knowledge Representation:→** It is most vital Component of AI systems because the quality of an intelligent system depend upon how much the knowledge the system has?, how this knowledgebase executes and what are the knowledge representations?

2. **Control Strategy or Infrence Process:→** It contains mainly heuristic search procedures. It is the operational task that characterise the AI programs. Every intelligent program depends on the Control strategy to perform its prescribe function.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

3. Programming Language :→ AI programming language provides the basic function for AI tools. LISP an PROLOG are basic AI programming languages.

4. A.I. Hardware :→ The AI programs can be execute on various hardware platforms like uniprocessors and array processors. (multiprocessors).

* Two main Goals of A.I. :→

1.) Two understand human intelligence better.

2.) To create useful smart, programs which are able to do the task. themselves that could normally require a human intelligence.

* Problem Representation in A.I. :→ Before a solution can be found for a problem, the prime condition is that the problem must be very precisely definite. By defining it, one converts the abstract problem into real workable states that are really understood. The states are operated on by set of operators and the decision of which operator to be applied, when and where is dictated by the overall control strategy.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

The most common methods of problem representation in A.I. are:

1. State Space Representation
2. Problem Reduction

① **State Space Representation** :→ The set of all possible states for a given problem is known as state space for that problem. The state space representation is basically a directed graph with all the possible states as its nodes. If the entire state space representation for a problem is given it is possible to trace the path from initial state to the goal state and identify the requirements of operators necessary for doing it.

For example :→ 8-puzzle problem.

It can be expressed with the help of state space representation. The 8-puzzle problem is made up of a frame in which 8-square tiles are placed. The tiles are numbered from 1 to 8. and the 9th square is left as blank.

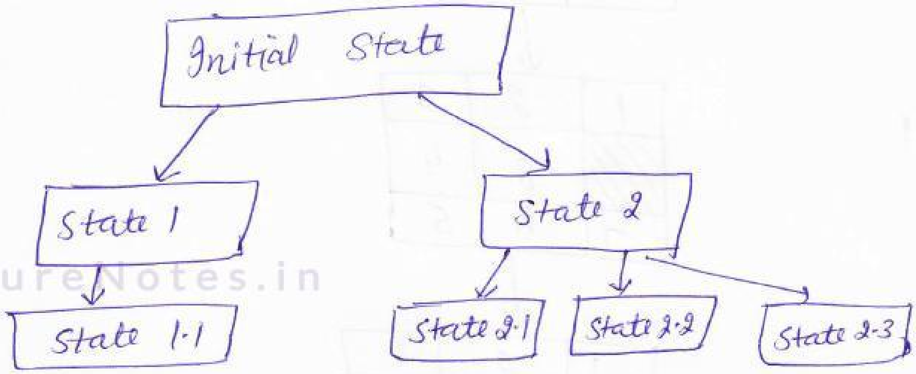
The tiles which are adjacent to blank space can be slided into that space.

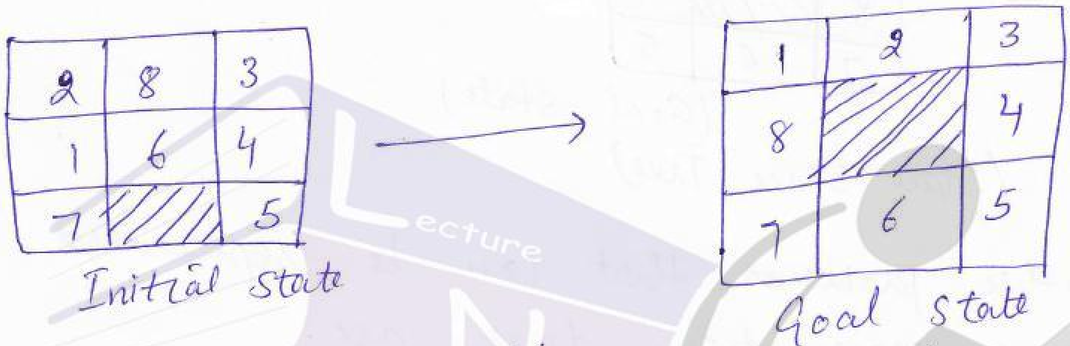
1	2	3
4	5	6
7	8	(blank)

A Game Consists of a starting position and the specified goal position there may be multiple possible steps available to move from

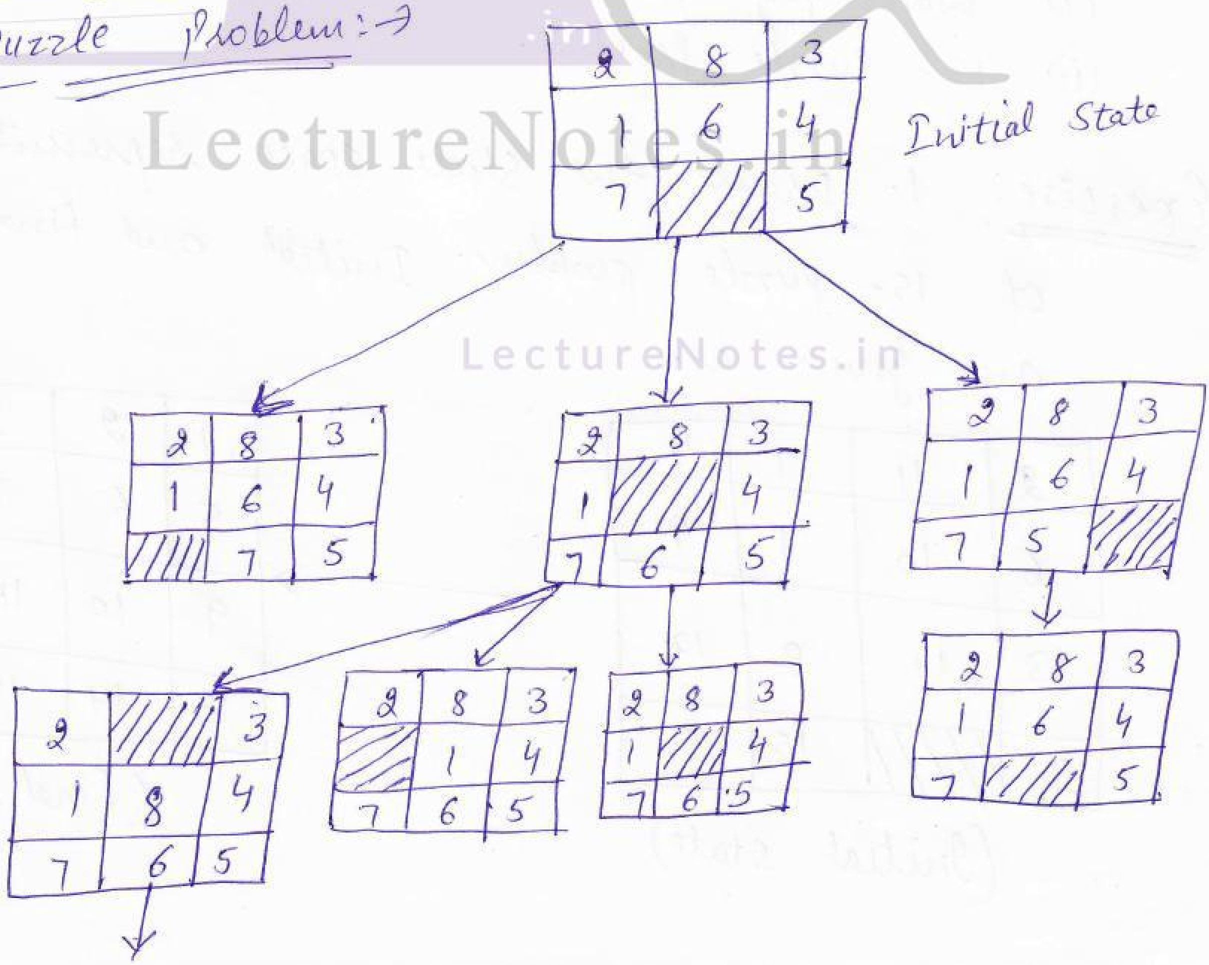
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

initial to goal state but only those steps are considered which provide better solution.





State Space Representation for 8-puzzle Problem:→



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

↓

	2	3
1	8	4
7	6	5

↓

1	2	3
	8	4
7	6	5

↓

1	2	3
8		4
7	6	5

(Goal - state)

(State Space Tree)

Some other problems that can be represented by state-space representation tree are:

(i) Water-jug Problem.

(ii) 15- puzzle Problem.

<u>Exercise</u>: 1. Draw the State Space representation of 15- puzzle problem. Initial and final states are given as:

2	11	7	4
6	10	1	9
13	14	8	12
3		15	5

(Initial state)

⟶

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(final State)

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

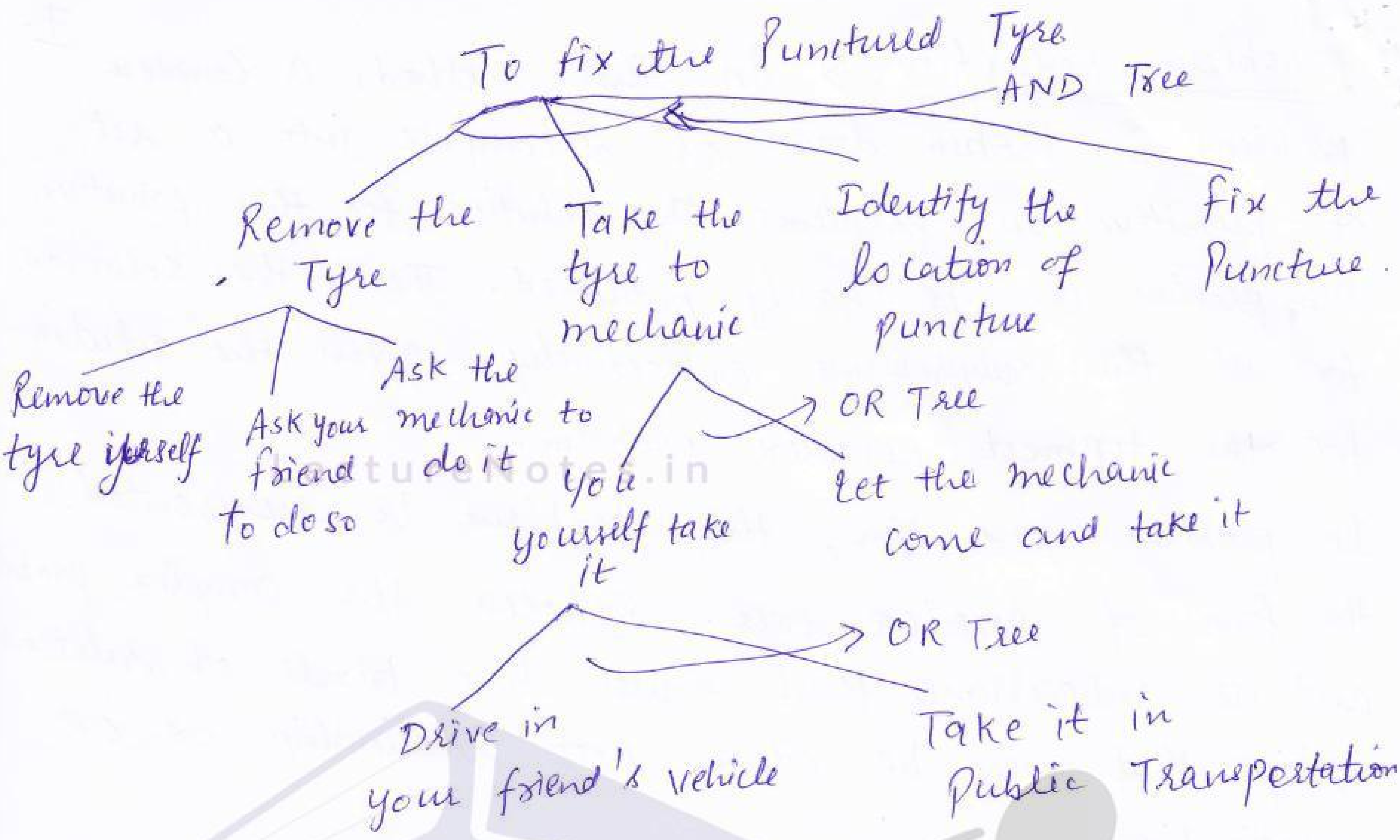
2. **Problem Reduction:→** In this method, a Complex problem is broken down or decompose into a set of primitive sub-problems. The solution for this primitive sub-problem can be easily obtained. Then the solution for all the subproblems collectively given the solution for the topmost complex problem.

In problem Reduction, the problem is represented in the form of AND/OR tree. Between the complex proble and its subproblems, there exists two kinds of relation-ships that may be either AND relationship or OR relationship.

In AND relationship, the solution for the problem obtained by solving all the subproblems whereas in OR relationship, the solution for the problem is obtai-ed by solving any of the subproblem. The branches connected by an AND relation are represented with the help of (◡) 'arc' symbol.

For example:→ AND/OR tree to solve the problem " To fix a punctured Tyre " is shown ~~below~~ as on the next page:

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



AND/OR Tree.

* **Logic:→** Logic is defined as the scientific study of process of reasoning and the system of rules and procedures that helps in reasoning process.

→ Logic takes in some information (premises) and produce output (conclusion).

→ The notion of logic ~~includes~~ begins with prime activity of human intelligence which is reasoning.

→ The activity of reasoning involves construction, organization and manipulation of statements to arrive at new conclusion.

The Logic includes:

(1) Syntax (2) Semantics (3) Inference Rules.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

1. Syntax :→ What all symbols are used and how they will be combined with each other

2. Semantics :→ It specifies what facts in the world, a sentence refers to and how to assign a truth value to that sentence.

3. Inference Rules :→ It is the mechanical method for computing new true statements from the existing statements.

* Types of Logic :→ There are two types of logic:
1.) Propositional Logic
2.) Predicate Logic

(1.) Propositional Logic :→ In this logic, all the statements are called propositions. where a proposition can take only two values: either True or False.
Propositional can be Atomic (Single) or Compound (combination of two or more atomic propositions).

Propositional Calculus include :
1.) Symbols
2.) Syntax
3.) Semantics
4.) Normal forms.

(1.) Symbols :→ Symbols of Propositional calculus are:

a.) Atomic Symbols :- P, Q, R, S, T . . . . . .

b) Truth Symbols : true and false.

c.) Connectives : ∧ (AND), ∨ (OR), ¬ (Negation)
→ (implication), ≡ (Equivalence), ↔ (Biconditional)

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Syntax :→

Propositional Calculus Sentences:→

(1.) Every propositional symbol and truth symbol is a sentenc

Eg. → True, false, P, Q and R are sentences.

(2.) The 'negation' of a sentence is a sentence.

Eg. → ¬P and ¬false are sentences.

(3.) The 'conjuction', or 'and' of two sentences is a sentence

Eg. → $P \wedge \neg P$ is a sentence.

(4.) The 'disjunction', or 'or,' of two sentences is a sentence.

Eg. → $P \vee \neg P$ is a sentence.

(5.) The 'implication' of one sentence for another is a se

Eg. → $P \rightarrow Q$ is a sentence.

(6.) The 'Equivalence' of two sentences is a sentence.

Eg. → $P \vee Q \equiv R$ is a sentence.

* WFF:→ Legal sentences are called well-formed formulas or WFFs.

* Semantics :→ It covers two aspects:

1.) To find significance of the existing sentence in real world.

2.) To assign a truth value to a sentence.

Interpretation:→

In propositional logic, this is done with the help of phenomenon called 'Interpretation' which simply means mapping of a sentence to a set. {T, F}.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(1.) The truth assignment of negation: ¬P, where P is any propositional symbol, is true if the assignment to P is false. It is false if the assignment to P is true.

P	¬P
T	F
F	T

(2.) The truth assignment of conjunction: '∧'; is true only when both conjucts are true value otherwise it is false.

P	Q	P∧Q
T	T	T
T	F	F
F	T	F
F	F	F

(3.) The truth assignment to disjunction: 'V'; is true if any one or both of the disjuncts has true value.

P	Q	PVQ
T	T	T
T	F	T
F	T	T
F	F	F

(4.) The truth assignment of implication: '→'; is false only when the symbol before the implication is true and the symbol after implication is false. otherwise it is true.

P	Q	P→Q
T	T	T
T	F	F
F	T	T
F	F	T

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

5. The truth assignment of equivalence '(≡)' is true only when both expressions have the same truth value, otherwise it is false.

P	Q	$P \equiv Q$
T	T	T
T	F	F
F	T	F
F	F	T

6. The truth assignment for biconditional (⟷) is true only when both expressions have same truth value.

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

* Propositional Logic Laws (Equivalences):→

1. Double Negation Law:→ $\neg(\neg P) \equiv P$

2.) Contropositive Law:→ $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

3.) De-Morgan's Law:→ $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$

$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$

4.) Commutative Law:→ $P \wedge Q \equiv Q \wedge P$

$P \vee Q \equiv Q \vee P$.

5.) Associative Law:→ $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$

$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

6. Distributive Law:→ $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$.

7. $P \rightarrow Q \equiv \neg P \vee Q$.

8. $P \leftrightarrow Q \equiv (\neg P \vee Q) \wedge (\neg Q \vee P)$

* Normal forms in Propositional Logic:→ There are two normal forms in propositional logic:
  1) Conjuctive Normal form (CNF).
  2) Disjunctive Normal form (DNF).

(1) CNF:→ A formula A is said to be in CNF, if it has the form

$$A = A_1 \wedge A_2 \wedge A_3 \cdots \wedge A_n,\ n \geq 1.$$

where each $A_1, A_2, A_3 \cdots A_n$ is disjunction of an atom or negation of an atom.

* Atom:→ An atom is a predicate with terms for arguments.
eg.→ $p(t\_1, t\_2, \cdots t\_n)$

(2) DNF:→ A formula is said to be in DNF if it has the form $A = A_1 \vee A_2 \vee A_3 \cdots \vee A_n,\ n \geq 1$.

where each $A_1, A_2 \cdots A_n$ are conjuction of an atom or negation of an atom.

Conversion Procedure To Normal form:→

Step 1:→ To eliminate implication and biconditional by using:

$$A \rightarrow B \equiv \neg A \vee B.$$
$$B \rightarrow A \equiv \neg B \vee A.$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$
$$\equiv (\neg A \vee B) \wedge (\neg B \vee A)$$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Step 2:→ Apply Double Negation and De-Morgan's Law.

$$\neg(\neg A) \equiv A$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B.$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B.$$

Step 3:→ To apply Distributive Law and Equivalence Law.

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C).$$

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C).$$

1. Example: Convert $((A \rightarrow B) \rightarrow C)$ into CNF.

Solution: Step 1:→ $((\neg A \vee B) \rightarrow C)$

Step 2:- $\neg(\neg A \vee B) \vee C$

Step 3:- $(A \wedge \neg B) \vee C$

$(A \vee C) \wedge (\neg B \vee C) \rightarrow$ (CNF)

Example 2:- Convert $A \rightarrow ((B \wedge C) \rightarrow D)$ into DNF.

Solution:- Step 1:- $A \rightarrow (\neg(B \wedge C) \vee D)$

$\neg A \vee (\neg(B \wedge C) \vee D)$

Step 2: $\neg A \vee (\neg B \vee \neg C \vee D)$

$(\neg A) \vee (\neg B) \vee (\neg C) \vee (D) \rightarrow$ DNF.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* **Predicate Logic :->** It is also known First Order Predicate Logic (FOPL).

In Predicate calculus, each sentence or statement is represented with the help of predicates.

Predicates :-> It is defined as that binds two atoms together.

For example : Monkey Eats Banana.

Eats (Monkey, Banana).

The above predicate can be generalized as

Eats (x, y).

where x takes anybody and y is anything.

* Every predicate can have an +ve integer associated with it called arity as argument number for that predicate.

* **Syntax :->**

-> **Predicate Calculus Symbol or Terms :->** The symbols or terms of predicate calculus are the irreducible syntactic element that cannot be broken into parts. The symbols in the predicate logic begin with a letter.

Following are the characters use to make symbols or terms of predicate logic :-

1.) Set of letters. (a-z, A-Z).
2.) Set of digits. (0-9).
3.) Underscore (_). Symbol.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For example: Jim, excel123, etc_1 etc.

The predicate calculus may represent variables, constants or functions.

(1) The variable symbols are used to designate general classes of objects or properties. variable always begin with an uppercase letter.

(2) The constant symbol represents the specific objects or properties. The constant symbols begins with a lowercase letter.

(3) The function expression is a function symbol followed by arguments. The arguments are the elements from the domain of the function.

The no. of arguments = arity.

The arguments are enclosed within parenthesis and separated by commas. For eg. → $f(x,y)$.

* Predicate Logic Connectives :→ (Predicate logic Quantifiers)

A quantifier is a symbol that permits one to declare or identify the range or scope of the variables in a logical expression. There are two types of quantifiers:

1.) Existential Quantifier. ($\exists$)

2.) Universal Quantifier ($\forall$)

1. Existential Quantifier: It means there exists a, and for some a, for at least one a.

For eg. → $\exists x$ Friends (x, Rohan).

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2.) <u>Universal Quantifier</u> :→ It means for all a, for each a, for every a.

For eg.→ $\forall x$ Likes (x, ice-cream)

* <u>Predicate Calculus Sentences</u> :→

(1.) Every atomic sentence is a sentence.
for eg.→ The Truth value i.e. true or false.
The atomic sentences are also called atomic expressions or atoms.

(2.) If S is a sentence then $\neg S$ is also a sentence.

(3.) If S_1 and S_2 are sentences then $S_1 \wedge S_2$ is also a sentence.

(4.) If S_1 and S_2 are sentences then $S_1 \vee S_2$ is also a sentence

(5.) The implication of two sentences is a sentence. $S_1 \rightarrow S_2$.

(6.) The equivalence of two sentences is a sentence.
$S_1 \equiv S_2$.

(7.) If x is a variable and S is a sentence then $\forall x S$ is also a sentence.

(8.) If x is a variable and S is a sentence then $\exists x S$ is also a sentence.

* <u>Free and Bound Variable</u> :→

A variable in a formula is FREE if and only if either its all occurences or atleast one occurence is outside the scope of quantifier having the variable. For eg.⇒ $\forall x \exists y (f(x, y, z))$
↑ Free variable.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

BOUND variable:→ A variable in a formula is BOUND iff either its all occurences or atleast one occurence is within the scope of quantifier.

for eg.→ $\forall x \exists y (f(x,y,z))$

Here x and y are Bound variables.

* Semantics for Predicate Logic:→ Semantics for it serves two purpose:

1.) To determine the meaning of predicate calculus expressions in terms of objects, properties or relation

2.) To provide the formal basis for determining the truth value of any expression.

Example: The information about a fruit named Gouvava may be expressed as:

Color (Gouvava, Red) → Assigns false value.

Color (Gouvava, Green) → Assigns True value.

* Normal forms in Predicate Logic:→ The Predicate logic has only one normal form called Prenex Normal Form.

A formula A in predicate logic is said to be in Prenex normal form if it has the form

$$\underbrace{(Q_1 x_1)(Q_2 x_2) \text{———} (Q_n x_n)}_{\text{Prefix}} \cdot \underbrace{B}_{\text{Matrix}}$$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Here $(Q_i x_i)$ is either existential $(\exists)$ or $\forall$ and B is formula without any quantifier.

For eg. → $\underbrace{\forall x \forall y \forall z}_{\text{Prefix}} \underbrace{((A(x,y,z) \vee B(y,z)) \rightarrow C(x,z))}_{\text{Matrix}}$

* Methodology for Converting Predicate Logic into Prenex Normal Form :→ (Next Page)

Step 1:- Eliminate implications and biconditionals using the laws

$$(A \rightarrow B) \equiv \neg A \vee B.$$

$$(A \leftrightarrow B) \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

$$\equiv (\neg A \vee B) \wedge (\neg B \vee A)$$

Step 2: Apply Double Negation Law and apply De-morgan's Theorm.

$$\neg (A \vee B) \equiv \neg A \wedge \neg B$$

$$\neg (A \wedge B) \equiv \neg A \vee \neg B.$$

Use the formulae:

$$\neg (\forall x (A(x))) \equiv \exists x (\neg A(x))$$

$$\neg (\exists x (A(x))) \equiv \forall x (\neg A(x))$$

Step 3: Rename Bound variable if necessary.

Step 4: Use the distribute Law and following formulae to move the quantifiers to the left of the formula.

$$\forall x\, A[x] \wedge \forall x\, B[x] \equiv \forall x (A[x] \wedge B[x])$$

$$Q_1(x)\, A[x] \wedge Q_2(x)\, B[x] \equiv Q_1(x)(Q_2 y)(A[x] \wedge B[y])$$

Example : (Page No. 14)

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Clausal Conversion Procedure:→ (Skolemization)

Step 1:) Eliminate all implication and equivalency connectives using

$$P \rightarrow Q \equiv \neg P \vee Q$$

$$P \leftrightarrow Q \equiv (\neg P \vee Q) \wedge (\neg Q \vee P).$$

Step 2:→ Move all negations into immediately precede an atom. and apply De morgan's Law.

$$\neg(\neg P) \equiv P.$$

$$\neg(\forall x) F[x] \equiv \exists x \neg F[x]$$

$$\neg(\exists x) F[x] \equiv \forall x \neg F[x].$$

Step 3:- Rename variables, if necessary, so that all quantifiers have different variable assignments; that is, rename variables so that variables bound by one quantifier are not the same as variables bound by a different quantifier.

For eg.→ $\forall x\,(P(x) \rightarrow (\exists x\,(Q(x)))$ to give

$\forall x\,(P(x) \rightarrow (\exists y\, Q(y)))$.

Step 4:- Skolemize by replacing all existentially quantified variables with skolem functions and deleting the corresponding existential quantifiers.

Step 5:- Move all universal quantifiers to the left of the expression the put the expression on the right into CNF.

Step 6:→ Eliminate all universal quantifiers and conjuction since they are retained implicitly.

The resulting expressions are clauses and the set of such expressions is said to be in clausal form.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For example:- Convert the expression
$\exists x \forall y (\forall z P(f(x), y, z) \rightarrow (\exists u Q(x,u) \wedge \exists v R(y,v)))$ into clausal form.

Step1:→ $\exists x \forall y (\neg(\forall z) P(f(x), y, z) \vee (\exists u Q(x,u) \wedge (\exists v) R(y,v)))$.

Step2:→ $\exists x \forall y (\exists z \neg P(f(x), y, z) \vee (\exists u Q(x,u) \wedge (\exists v) R(y,v)))$.
(3 is not required)

Step4:→ $\forall y (\neg P(f(a), y, g(y)) \vee (Q(a, h(y)) \wedge R(y, l(y)))$. | $\exists x$ is eliminated, $\exists u$ " ", u is replaced by f(y).

Step5:→ $\forall y ((\neg P(f(a), y, g(y)) \vee Q(a, h(y))) \wedge (\neg P(f(a), y, g(y)) \vee R(y, l(y)))$.

Step6:→ $\neg P(f(a), y, g(y)) \vee Q(a, h(y))$
$\neg P(f(a), y, g(y)) \vee R(y, l(y))$.
This is the clausal form.

Clausal form:→ A clause is defined as disjunction of a number of literals. for eg.→ $(P \vee Q \vee \neg R)$.
There are two types of clausal form:-

1) Ground clause:→ In which a no variable occurs in an expression.
(2) Horn clause:→ A clause with almost one positive literal is horn clause. For eg.→ $(\neg P \vee Q \vee \neg R)$.

* Example of Conversion of Expression into Prenex Normal form:→

1) Convert the formula $\forall x (A(x) \rightarrow \exists y B(x,y))$ into Prenex Normal form.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

The rules of inference are based on

→ Assertion (Given Information)

→ Implication (Given Relationship)

→ Conclusion (Drive the result using above two).

Example:- Assertion: Lion (Leo)

Implication : $\forall x$ [Lion (x) → Carnivore (x)]

Conclusion : Lio (Leo) → Carnivore (Leo)

In general, if a has property P and all objects that have property P also have Property Q, we conclude that a has property Q.

$$\frac{\begin{array}{c} P(a) \\ \forall x\, P(x) \rightarrow Q(x) \end{array}}{Q(a)}$$

LectureNotes.in

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Inference Rules:→ Inference is defined as the ability to drive new correct expressions from a set of given assertions.

In order to drive some new statements or to prove some-thing true or false, we need to use some manipulation procedures. These manipulation procedures are called Inference Rules. The most commonly used Inference Rules are:

1.) Modus Ponens.
2.) Modus Tollens.
3.) Resolution.

The rules of inference are based on three keywords:

→ Assertion (Given Information)
→ Implication (Given Relationship)
→ Conclusion (Drive the result using above two).

Example:- Assertion: Lion (Leo)

Implication : $\forall x\,[\text{Lion}(x) \rightarrow \text{Carnivore}(x)]$

Conclusion : $\text{Lion}(\text{Leo}) \rightarrow \text{Carnivore}(\text{Leo})$

In general, if a has property P and all objects that have property P also have Property Q, we conclude that a has property Q.

$$\frac{\begin{array}{c} P(a) \\ \forall x\, P(x) \rightarrow Q(x) \end{array}}{Q(a).}$$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Some impertant rules of inference:→

(1) Rules of Propositional Logic:→

(i) Modus Ponens:→ If the sentence P and $P \to Q$ are true then we can infer Q as true.

$$(P \wedge (P \to Q)) \to Q.$$

(ii) Modus Tollens:→ If $P \to Q$ is true and Q is false then we can infer $\neg P$.

$$(\neg Q \wedge (P \to Q)) \to \neg P.$$

(iii) Elimination or Simplification:→ If P and Q both are true then we can infer either P or Q.

$$P \wedge Q \to P$$

$$P \wedge Q \to Q.$$

(iv) Negation:→ If Negation of a false value is encountered then we can infer the true value itself.

$$\neg(\neg A) \to A.$$

(2) Rules of Predicate Logic:→ (Instance → Specification)

(i) Universal Instantiation:→

$$\frac{\forall x \; P(x)}{\therefore P(a)}$$

where a is from domain of x.

(ii) Universal Generalization:→

$$\frac{P(a)}{\therefore \forall x \; P(x)}$$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(iii) <u>Existential Instantiation</u> :→

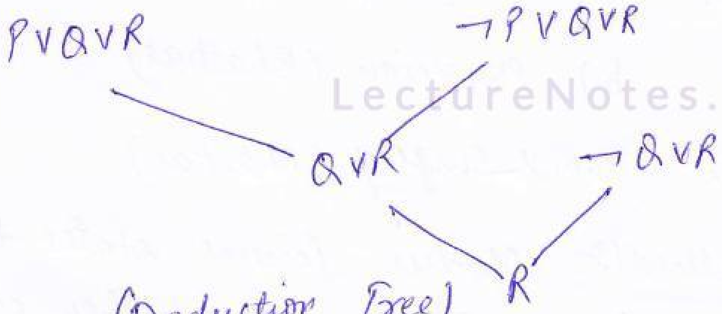
$$\frac{\exists x\ P(x)}{\therefore P(a)}$$

(iv) <u>Existential Generalization</u> :→

$$\frac{P(a)}{\therefore \exists x\ P(x)}$$

③ <u>Resolution</u> :→ Resolution is another rule of inference. Principle of resolution states that given any two clauses A and B called parent clauses, if there is a literal P_1 in A which has a complementary literal P_2 in B, delete P_1 and P_2 from A and B & construct a disjunction of the remaining clauses. The clause so constructed is called the resolvent of A and B.

For eg. →
A : $P \vee Q \vee R$
B : $\neg P \vee Q \vee R$.
C : $\neg Q \vee R$.



(Deduction Tree)

A : $P \vee Q \vee R$
B : $\neg P \vee Q \vee R$
D : $Q \vee R$
C : $\neg Q \vee R$
E : R.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

clause A has the literal P which is complementary to ¬P in B. Hence both of them are deleted and a resolvent is generated. That resolvent has again a literal Q whose negation is available in C. Hence resolving these two, one has the final resolvent.

Exercise

* Perform Resolution on the set of clauses

A : $P \vee Q \vee R$

B : $\neg P \vee R$

C : $\neg Q$

D : $\neg R$

* **Theorm Proving using Resolution:** → There are two basic methods:

1.) Start with the given axioms, use the rule of inference and prove the theorm.

2.) Prove that the negation of result cannot be true. This method is called Refutation.

Example: Given: a.) $\forall x$ [Physician (x) → knows_surgery (x)]

b.) Physician (Bhaskar)

Prove that: Knows_Surgery (Bhaskar)

Proof using Method 1: Modus Ponens states that if there is an axiom of the form P → Q and another of the form P, then Q logically follows.

Assuming Physician (Bhaskar) as P and [Physician (x) → knows_surgery (x)] as Q then

... (Bhaskar) logically follows.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Proof Using Refutation :→ Let us assume that

¬ Knows_surgery (Bhaskar) —— (1)

Given: Physician (Bhaskar) —— (2)

∀x [physician (x) → Knows_Surgery(x)] —— (3)

(3) ⇒ ¬ Physician(x) V Knows_Surgery (x) —— (4)

In Eq^n (4), substitute x = Bhaskar.

¬ Physician (Bhaskar) V Knows_Surgery (Bhaskar) —— (5)

Resolving (1) and (5)

we get ¬ Physician (Bhaskar)

But from (2), we have a contradiction.

So Knows_Surgery (Bhaskar).

* **Unification** :→ It is a process to determine most appropriate substitutions to make two predicate calculus expressions identical.

Any substitution that makes two or more expressions identical is called the unifier for those expressions and the process of identifying such unifier carried out by the process called Unification algorithm.

This algo tries to find out the Most General Unifier (MGU) between a given set of atomic formulas. Thus the basic of unification is substitution.

The Three major types of substitution allowed are:

1.) Substitution of a variable by a constant.

2.) Substitution of a variable by another variable.

3.) Substitution of a variable by a function that does

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Step 3:→ Then consider 1st argument from both the expression and make the substitution as follows:

(i) variable with a constant.

(ii) variable with another variable.

(iii) variable with a function.

Step 4:- Repeat Step 3 for all the arguments of given expression

Example:- Find the MGU for expression $A(x, f(g(x)), a)$ and $A(b, y, z)$.

Solution:-

$A(x, f(g(x)), a)$ ↑ $A(b, y, z)$ ↑

$A(x, f(g(x)), a)$ ↑ $A(b, y, z)$ ↑ {SUBST [x|b]

$A(b, f(g(b)), a)$ ↑ $A(b, y, z)$ ↑ SUBST [y|f(g(b))), (x|b

$A(b, f(g(b)), a)$ ↑ $A(b, f(g(b)), z)$ ↑ SUBST [(z|a), (y|f(g... (x|b)]

$A(b, f(g(b)), a)$ $A(b, f(g(b)), a)$

Hence the MGU for $A(x, f(g(x)), a)$ and $A(b, y, z)$ is [(z|a), (y|f(g(b))), (x|b)].

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence

Topic:

Knowledge Representation

Contributed By:

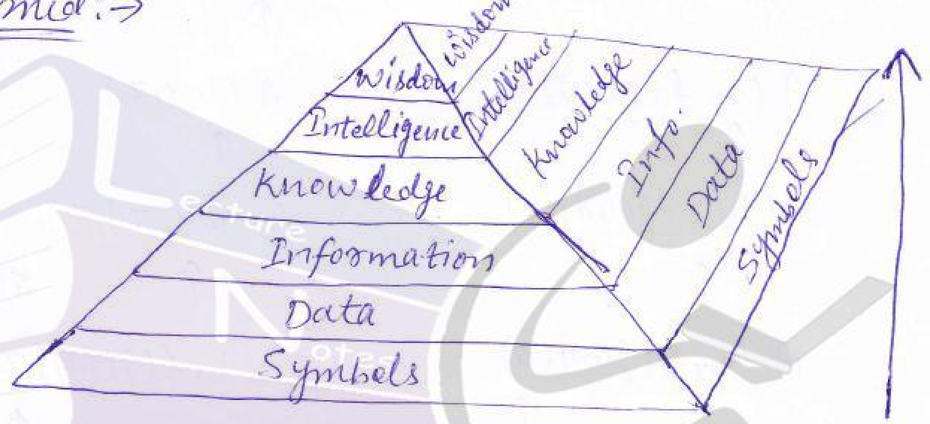
Sahil Kumar

* Knowledge Representation :→

Knowledge :→ knowledge is defined as the organised information. It is the collection of concepts, procedures, rules, ideas, facts and information that provides the to a problem.

Knowledge is the underlined force behind every intelligent System. Moreover, the quality of intelligent system depends upon how much knowledge an intelligent system has and how is that represented?

Knowledge Pyramid :→



Symbols :→ These are the base of knowledge pyramid which forms the basis of representation.

Data :→ Collection of symbols called Data.

Information :→ Information is the processed data.

Knowledge :→ Knowledge is organised information.

Intelligence :→ It is the ability to draw useful inference from the available knowledge.

Wisdom :→ It is the maturity of mind that detects its intelligence to achieve desirable goals.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Types of Knowledge :→ There are main two types of Knowledge:

1.) Domain-specific Knowledge :→ It is the knowledge that deals with all kinds of knowledge about a particular domain. There are four types of domain-specific knowledge:

a.) Knowledge about an item or object :→ This specifies the description and characteristics of the object.

b.) Knowledge about Events :→ for this knowledge is stored as set of procedures.

c.) Knowledge about task-performance :→ Here the expert possesses knowledge about the capacity of various objects and limitations of each

d.) Knowledge about Knowledge :→ Also Known as Meta-Knowledge. It helps in controlling and planning the reasoning process.

(2.) Common-Sense Knowledge :→ All other pieces of knowledge that helps in reasoning other than domain-specific knowledge are termed as common-sense knowledge.

* Knowledge Representation :→ It is the study of ways of how knowledge is actually picturised and how effectively it resembles the representation of knowledge in human brain.
A Good knowledge rep. model provides more specific and powerful problem solving mechanisms.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

The necessary characteristics of a knowledge rep. are:

(1) The representation sys. should have a set of well defined syntax and semantics.

2.) It should have good, expressive capability.

3.) It should be least complex.

4.) It should be detailed enough.

5.) It should also fascilitate effective knowledge gathering.

6.) From the computer sys. point of view, rep. must be effi

* **Types of Knowledge Representation:→**

(1.) **Declarative Representation:→** This rep. declares every piece of knowledge and permits the reasoning system to use the rules of inference to come out with new pieces of information. It tells "what about a situation"

For eg → "All carnivores have sharp_teeth."
"Cheetah is a carnivore."

Result: cheetah have sharp teeth.

Rep:→ $\forall x\ (carnivore(x) \rightarrow sharp\_teeth(x))$.
carnivore (cheetah).

Result: sharp-teeth (cheetah).

Advantages:→ 1.) Flexible.
2.) Modularity is high.
3.) Enough to rep. knowledge only once.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2) Procedural Representation :→ In this type of representation, the knowledge is rep. as procedures and the inferencing mechanisms that manipulate these procedures to arrive at the result. It tells "how of a situation"?

```
Procedure Carnivore(x):
    IF (x = cheetah) then return true
        else return false.
    END Procedure Carnivore(x).
Procedure Sharp-teeth(x):
    IF Carnivore(x) then
        return true
    else return false.
    END Procedure Sharp-teeth(x).
```

* Schemes for Knowledge Representation :→ There are various schemes used in A.I. for knowledge representation. These schemes utilize both the declarative as well as procedural knowledge. The widely known knowledge rep. schemes are:

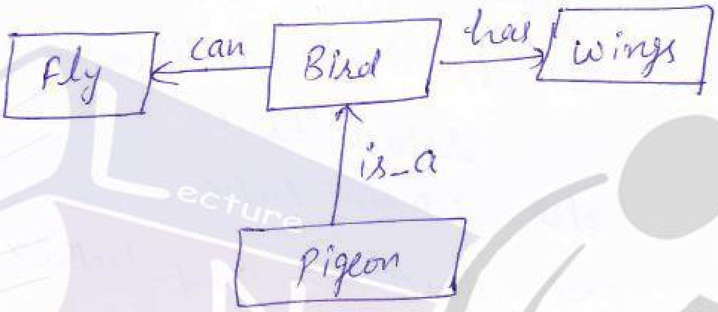
1. Semantic Networks
2. Conceptual Graphs.
3. Frames.
4. Scripts.
5. Conceptual Dependency.

(1) Semantic N/Ws :→ These are introduced by Quillian in 1968. The semantic net is a directed graph for representing knowledge as pattern of interconnected nodes & arcs.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

It is also known by name of Associated N/ws.
The nodes in the semantic net can be an object, class of objects, an event, an attribute or a state. whereas an arc rep. the relation b/w the objects. Every arc is labeled to specify the type and name of relationship that exists in between the nodes. The most common relationship arcs are is_a, has, is_in, has_part, contains, part_of, can etc.

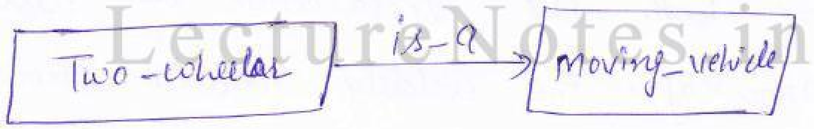
For eg.→



Classification of Nodes in a Semantic Net:→

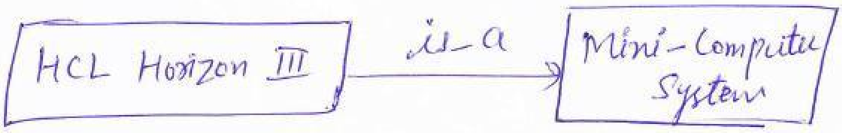
1.) Generic Node:→ A Generic node is a very General node.

Eg.→



(2.) Individual or Instance Nodes:→ Instance nodes explicitly state that they are specific instances of a generic node

For eg.→



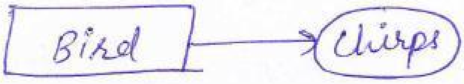
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Benefits of Semantic N/ws :→

1) Semantic net allows rep. of facts and relationships b/w facts.
2.) They support inference proper inheritance.
3.) They can be use to exhibit any hierarichal structure inherent in a group of entities.

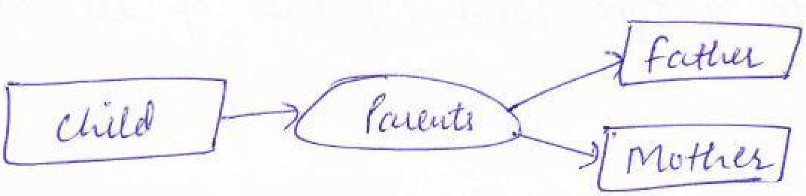
(2) Conceptual Graphs :→ The concept of conceptual graph was given by Johnsowa in 1984. A conceptual graph is defined as the finite connected Bipartite graph.
The nodes of the graph are either concepts or conceptual relations. The concept nodes may rep. an object, class of objects, entity, event or attribute. The concept nodes are rep. by rectangular box whereas the conceptual relation node rep the relationship b/w various concept nodes.
In the conceptual graph, the relationship can be of any arity (degree).

For eg.→ 1- ary relationship →



2-ary Relationship :→

3-ary Relationship :→



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Symbol Notation in Conceptual Graph :→

1. ':' (Colon) :→ This symbol is use to specialize an individual entity or class or superclass.

For eg.→

(2.) '#' :→ This is the unique token called 'marker' use to uniquely identify an individual.

For eg.→

(3.) '*' :→ This is the generic marker to indicate an unspecified individual. It is optional to use the symbol.

* **Operations Allowed in Conceptual Graph** :→ four most common opⁿs used in conceptual graphs are:

1.) Copy (2.) Restrict (3.) Join (4.) Simplify

(1.) Copy :→ This opⁿ allows to ~~use~~ form a new graph 'G' which is exact copy of already existing graph G1.

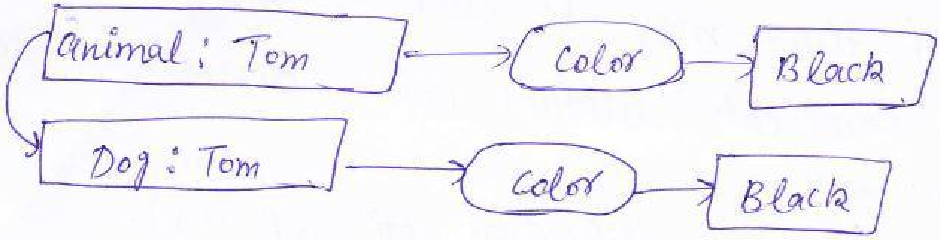
(2.) Restrict :→ This allows concept nodes in a graph to be replaced by a node which rep. its specialization. There may be two cases:

i.) Either Replace generic marker with individual marker.

ii) Replace a type label by one of its subtype.

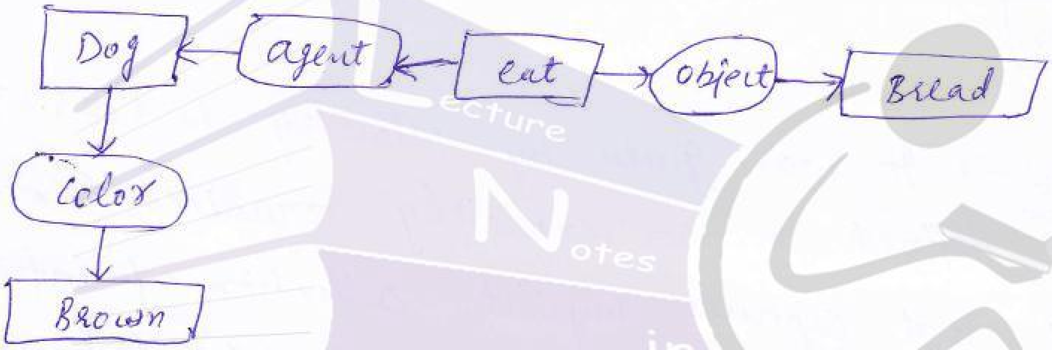
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Eg.→ Replace animal with dog.

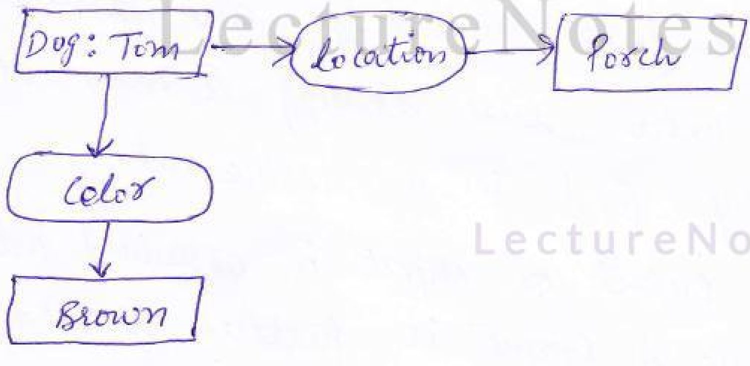


(3) Join:→ It allows to combine two graphs into a single graph, if there is a concept node C_1 in graph G_1 i.e. identical to concept node C_2 in graph G_2 then we can form a new graph G_3 by deleting C_2 and linking all of its relations incident to it to C_1.

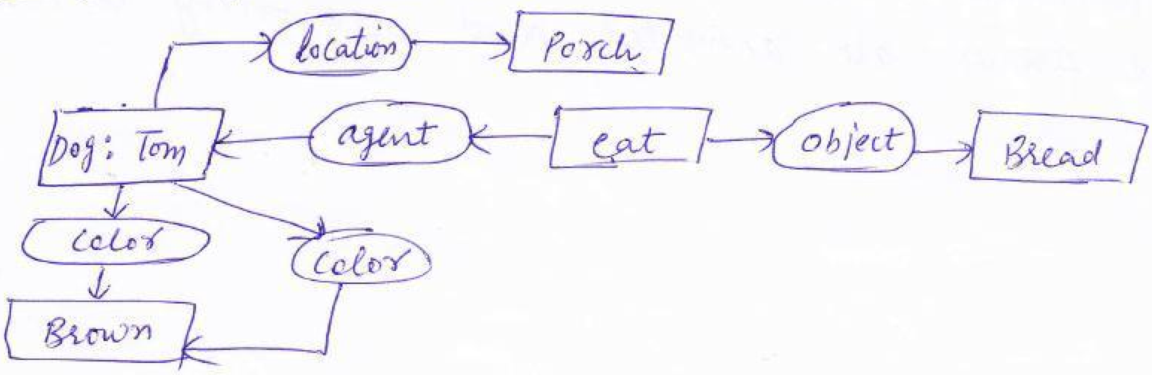
G1:



G2:



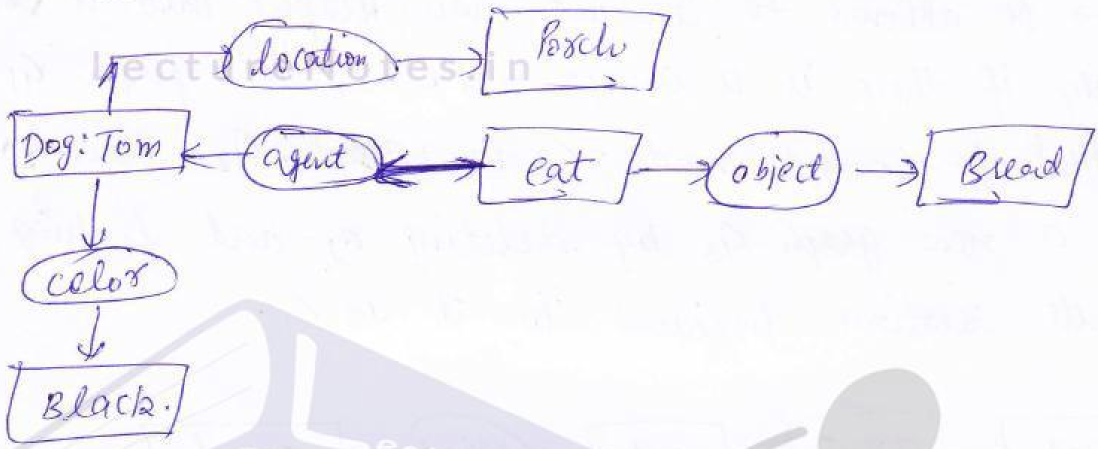
G3: (Join of G1 and G2).



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(4) Simplify:→ If a graph contains two duplicate relation then one of them may be deleted along with all of its arcs. Duplicate relations often occur as the result of join opⁿ.

Eg.→ G4: (After simplification opⁿ of G3).

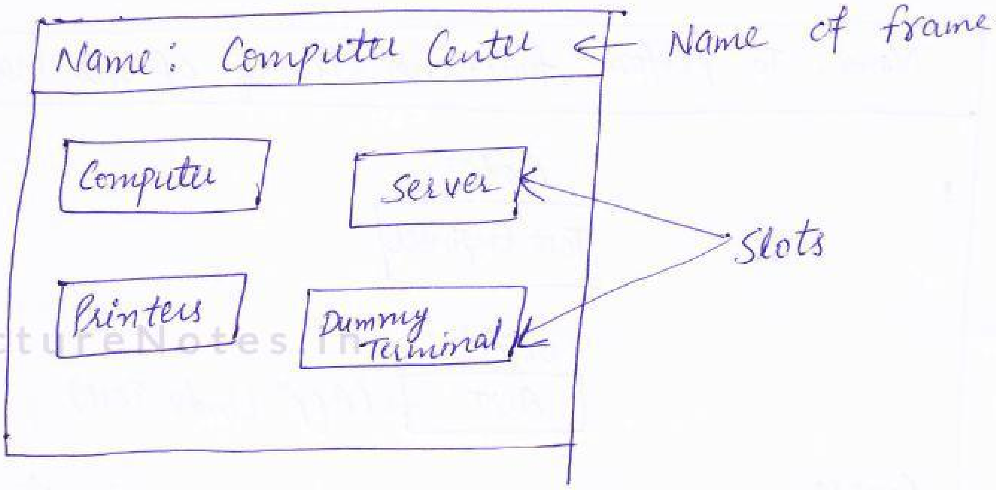


3. Frames:→ It was given by Minsky in 1975. The frame is defined as an explicitly organised data structure that captures implicit collection of information in a problem domain. Frame is a powerful tool of representing common-sense knowledge and generally represent some real life situation like driving a car, attending a meeting or eating food in a restaurant.

The knowledge about event or object is organised into small packets called frames. A frame is further divided into slots. Whenever a situation is encountered, a series of related frames are activated and reasoning is done.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Example: Sample frame for a Computer Center.



Types of frames:→ Two Types of frame:

1. Declarative (2) Procedural.

(1) Declarative frames:→ This frame merely contains description about the objects. The frame for comp. center is a declarative frame.

(2) Procedural frame:→ This frame contains procedural knowledge which explains how to perform things. Such frames which have procedure knowledge embedded in it are also called Action-Procedure frames. The action-procedure frames has got the following slots:

(i) Actor slot:→ which holds the info about who is performing the actions.

(ii) Object Slot:→ Has info about the item to be operated on.

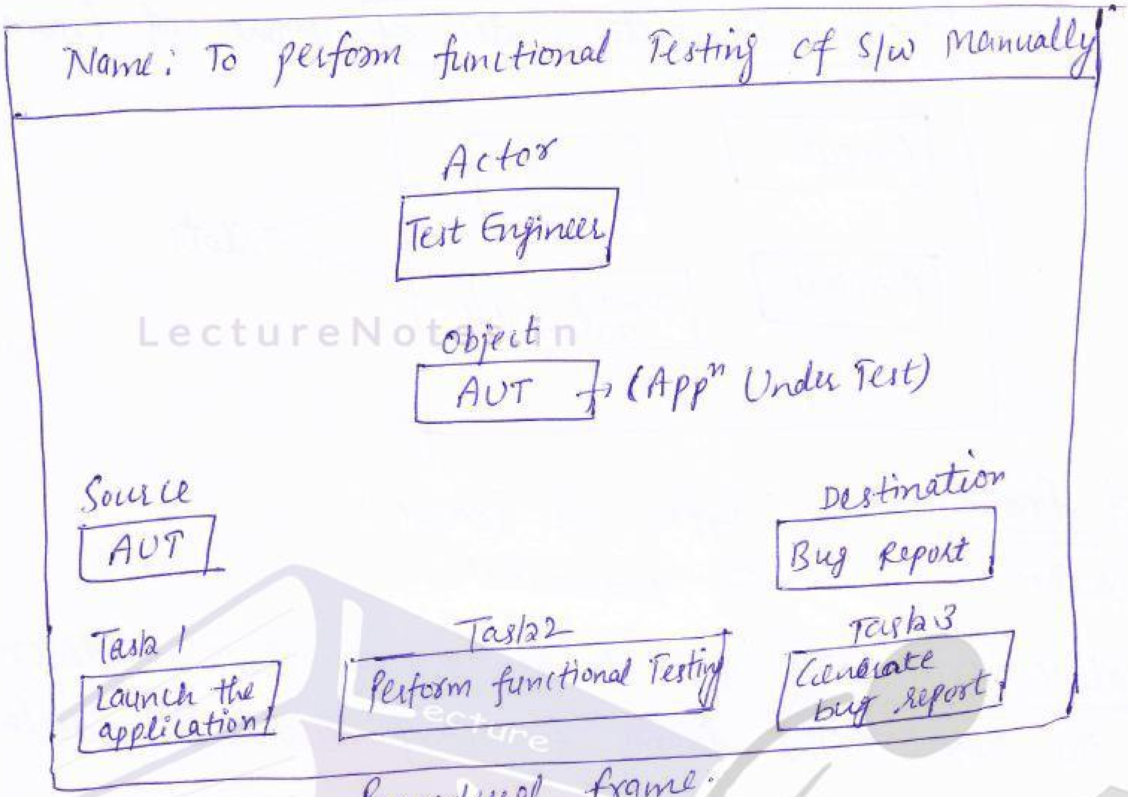
(iii) Source:→ Holds the info from where the action has to begin.

(iv) Destination:→ Holds the info where the action has to end.

(v) Task Slot:→ The necessary sub-frames required to perform the operation.

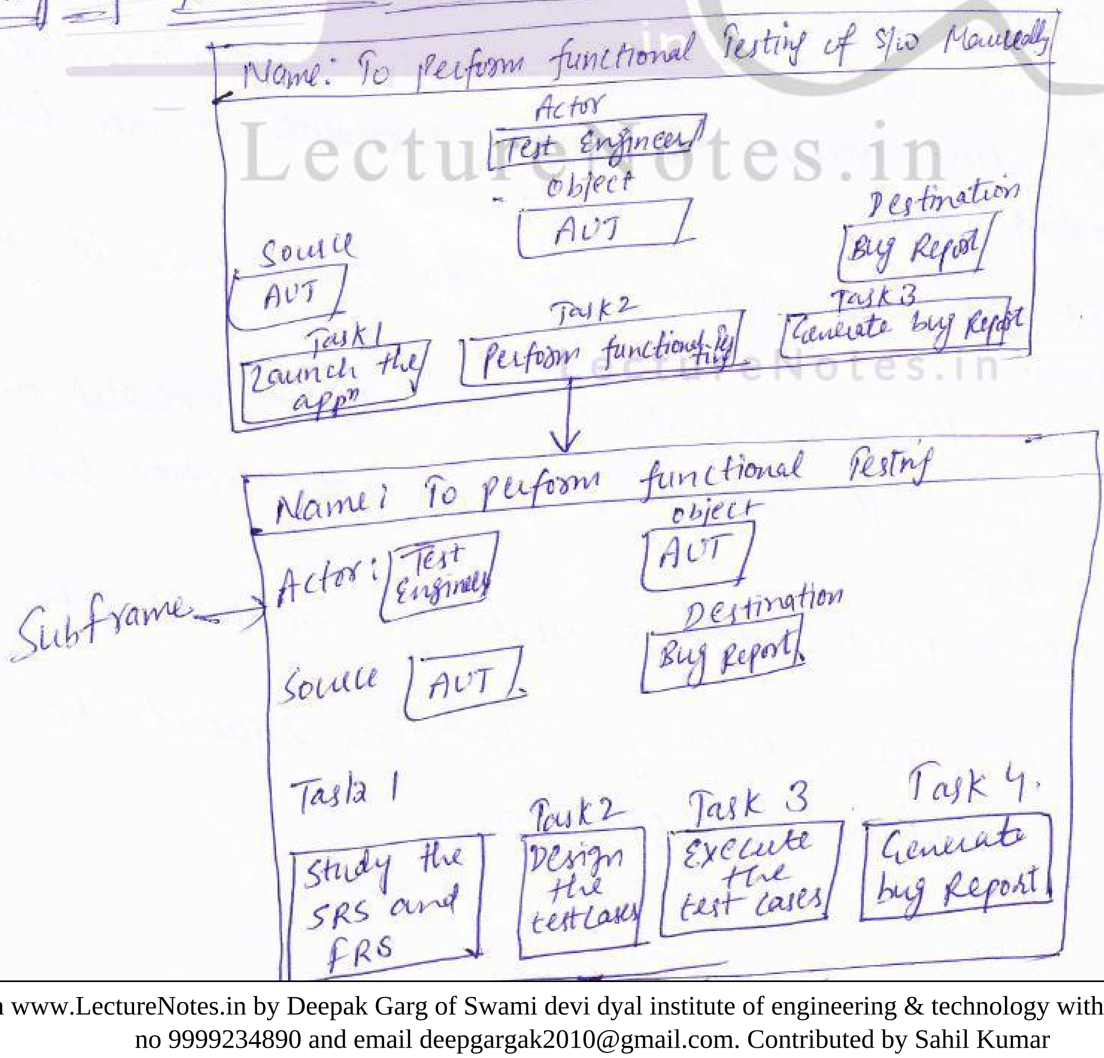
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For eg. → To perform functional Testing of S/w Manually



Procedural frame.

Linking of Procedural Subframes: →



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Reasoning Using frames:→ It is done by instantiation. Instantiation process begins when a given situation is matched with frames that are already in existance. The reasoning process tries to match the frame with the situation and later fills up slots for which values must be assigned. The values assigned to slots depict a particular situation and by this, the reasoning process tries to move towards a goal. The reasoning process can be defined as filling frame slots in frames.

Reasoning using frames also allows one to move from one frame to another to match the current situation.

(4) Scripts:→ Script is a knowledge rep. structure i.e. extensively used for describing streo type sequence of actions. It is the special case of frame structure. Similar to frames, scripts also have slots and with each slot, we associate information about the slot.

Scripts tell people what can happen in a situation, what events follow and what role every actor plays. The script structure is described in terms of actors, roles and scenes.

The important Components of scripts are:

(i) Entry Condition:→ Basic Conditions that must be fulfilled.

(ii) Result:→ Presents the situations which describe what happens after the script has occured.

(iii) Props:→ These indicate the objects that are existing in the script.

(iv) Roles:→ What various ~~actor~~ characters play is brought under the slot of roles.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(v) Track :→ Represents a specific instance of a generic pattern.

(vi) Scenes :→ Sequence of activities are described in detail.

For eg.→ A miniature restaurant script with customer going to a restaurant, ordering some eatables, eating them, paying the due amount and leaving the restaurant.

Script: Going to a restaurant Props: Food, Tables, Menu, Money Roles: Owner, Customer, Waiter, Cashier	Scene 1: Entering the restaurant Customer enters the restaurant Scans the tables chooses the best one occupies the seat
	Scene 2: Ordering the food Customer asks for menu Waiter brings it Orders the items
Entry Conditions: Customer is hungry Customer has money. Owner has food	Scene 3: Eating the food. Waiter brings the food Customer eats it.
Results: Customer is not hungry. Owner has more money. Customer has less money Owner has less food	Scene 4: Paying the bill Customer asks for the bill. Waiter brings it. Customer pays for it. Customer moves out of the restaurant

Pseudo-form of a restaurant Script

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(5) **Conceptual Dependency (CD):→** The concept of CD was given by Schank and Abelson in 1977. Conceptual Dependency is a theory of natural language processing which mainly deals with rep. of semantics of a language. CD is all about how to represent the kind of knowledge. The knowledge is rep. with the help of elements that are called conceptual structure.

In CD rep., if two sentences have identical meaning, there must be only one rep. to explicitly state them.

The main aims for development of CD:→

(1) To provide a means of representation that are language independent.

(2) To construct computer pgms that understand natural language

(3) To make inference from the statement and conditions given.

Conceptual Dependency objects:→

1) PP: (Picture Producers) - Physical objects are picture producers
2) ACT: Actions done by the actor.
3) LOC: Locations
4) T's: Time of action.
5) AA's: Action aiders - serve as modifiers of actions.
6) PA's: Picture Aiders - serve as aides of picture producers.

CD Actions:→ (1) ATRANS → Transfer of abstract relationship (e.g. → give)

(2) PTRANS → Transfer of physical location of an object. (e.g. go)

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence

Unit-II

* **State Space Search:→** Search is one of the most operational task that charaterise AI programs in best manner. Almost every AI program depends on a search procedure to perform its prescribe function. Problems are typically defined in terms of states and the solution corresponds to the goal state.

State Space Search includes:-

1.) **Predicate Calculus:→** Means of describing objects and relations in a problem domain.

2.) **Inference Rules:→** Means of infering new knowledge from given discription.

3.) **Base Rules:→** Define a space i.e searched to find a problem solution.

The **Primary Tools** for **state Space Search are:**

1) To represent the problem as a state space graph.

2) Use graph theory to analyze the structure and Complexity where state space graph is a directed graph defined as $G = (V, E)$.

$V \rightarrow$ Set of possible states (which also includes initial and goal states).

$E \rightarrow$ Set of Transitions between the states.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* **Strategies for State Space Search:** → There are two strategies known for state space search.

1.) Data Driven Search (Forward Chaining).

2.) Goal Driven Search (Backward chaining).

(1.) **Data Driven Search:** → In this approach, the problem solver begins with the given facts of the problem and a set of legal moves or rules for changing states.

Search proceeds by applying rules to facts to produce new facts which are in turn used by the rules to generate more new facts. This process continues untill it generates a path that satisfies the goal condition.

The data driven search is appropriate in following situations

(i) All or most of the data are given in the initial problem statement.

(ii) There are a large no. of potential goals but there are only a few ways to use the facts and given information

(iii) when it is difficult to form a goal state.

(2.) **Goal Driven Search:** → In this, the problem solver begins with the goal i.e. to be achieve, see what all rules or legal moves could be use to generate that goal and determine what condition must be true to see them.

These conditions become the new goals for the search.

Search contineous working backward through successive subgoals untill it works back to the facts of the problem.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

This strategy is appropriate in following situations:

(i) when the goal state can easily be formulated.

(ii) Problem data are not given but must be acquired by problem solver.

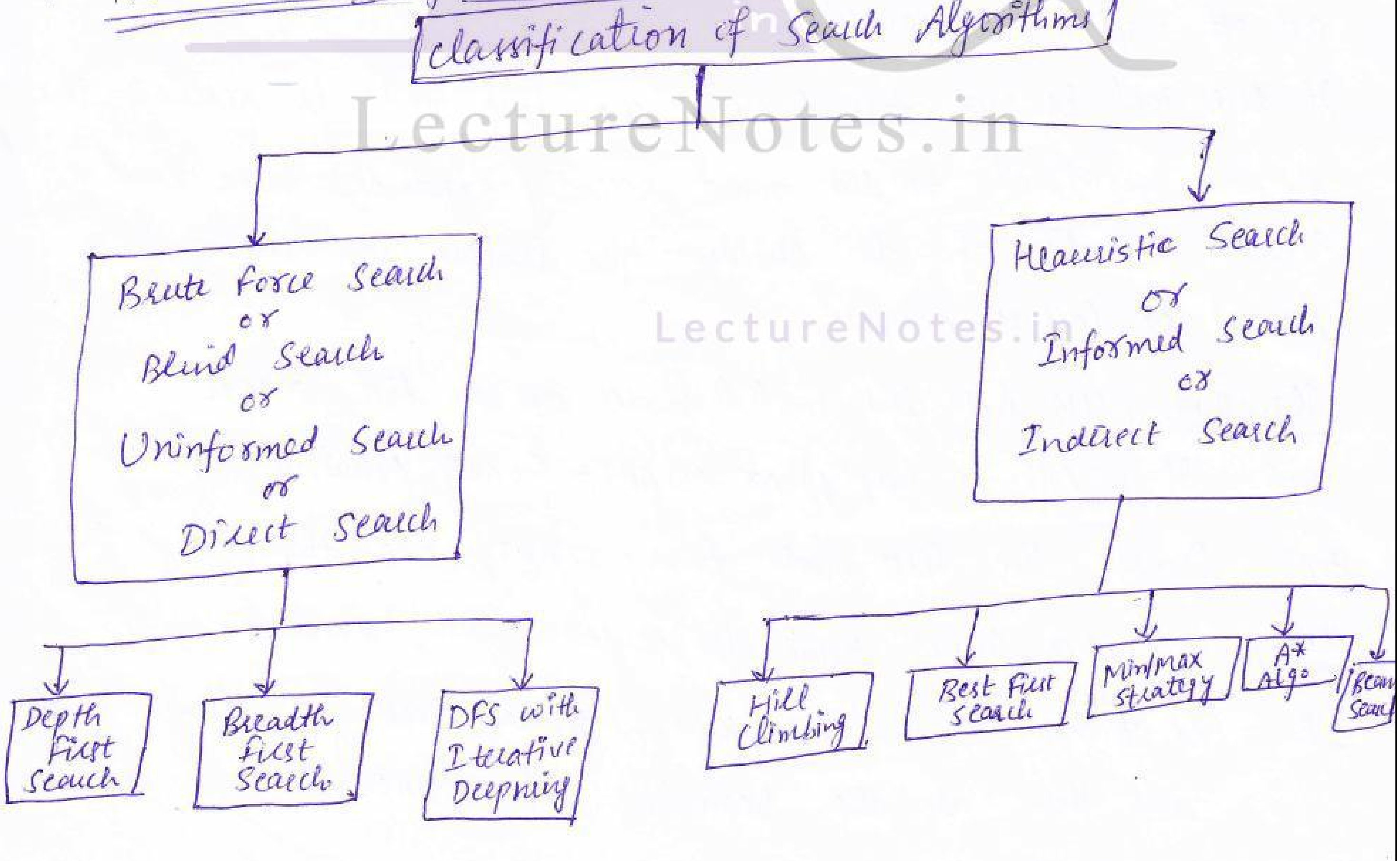
* **Research Process:→** Searching is defined as the sequence of steps that transforms the initial state into goal state. Searching is needed for solution which is not known before ~~ahead~~ ahead and have to be found. To do a search process, following things are needed:

1.) The initial state discription of the problem.

2.) A set of legal rules that changes the states.

3) The final or goal state.

* **The Search Algorithms:→**



Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(1) <u>Brute Force Search</u>:→ This is the most commonly used search procedure which explore all the alternatives during the search process. They do not have any domain specific knowledge. All that is needed is initial state, goal state and set of legal moves.

The most commonly used brute force search are:-

(i) Depth First Search (DFS). (ii) Breadth first Search (BFS).

(i) <u>DFS</u>:→ The search begins by expanding the initial node i.e. by using an operator, generate all successors of the initial node and test them.

This search is performed by dividing downwards into a tree. It does this always by generating a child node from the most recently expanded node, then generating children of that child and so on, untill the goal is found or some cut-off depth is reached.

If the goal is not found when the leaf node is reached, the program backtracks to the most recently expanded node and generate another of its children. The process continues goal is reached or failure occurs.

<u>Algorithm</u>:→ Step(1) Put the initial node on a list START.

<u>Step2</u>: If (START is empty) or (START = GOAL) terminate search.

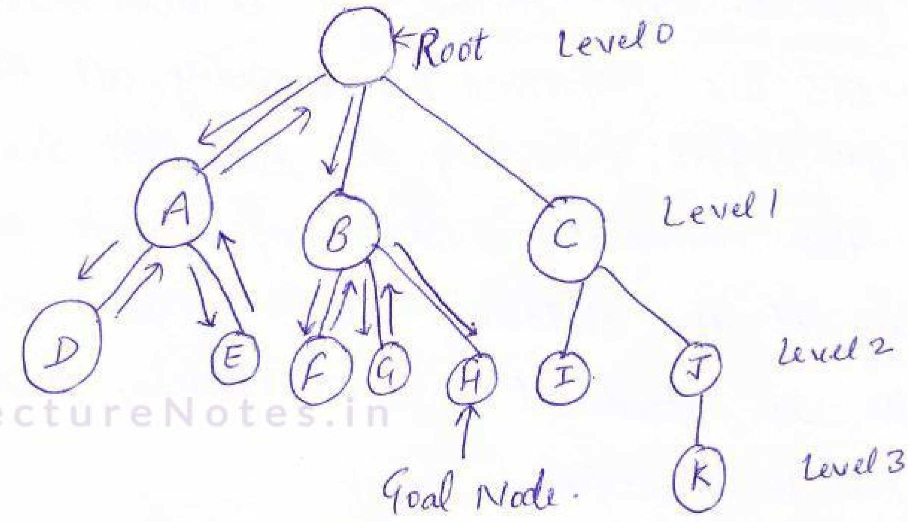
<u>Step3</u>: Remove the first node from START, call this node a.

<u>Step4</u>: If (a = GOAL), terminate search with success.

<u>Step5</u>: Else if node a has successors, generate all of them and add them at the beginning of START.

<u>Step6</u>: Go to step 2.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



Path: Root, A, D, E, B, F, G, H (Goal node)

Features of DFS:→

(1) Time Complexity:→ $1+b+b^2+ \text{——} +b^d$

$O(b^d)$

The time it takes to give the output is time complexity.

(2) Space Complexity:→ How much memory it is required.

$O(d)$.

DFS stores only current path it is pursuing. Hence the space-complexity is a linear function of the depth.

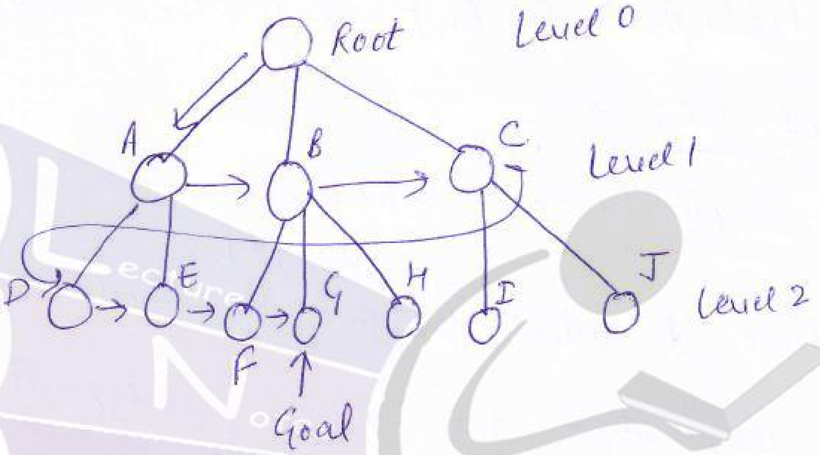
(3) Completeness:→ It is not complete because it may not be able to find the solution even if it does exist due to infinite depth of the search tree.

(4) Optimality:→ It is not optimal at all.

Major Drawback of DFS:→ Drawback of DFS is the determination of the depth untill which the search has to proceed. This depth is called cut-off depth. This cut-off depth is a critical factor because if this depth is too small, the solution may not be found and if it is too large then search procedure will go hanged.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(ii) BFS (Breadth first Search):→ It is also brute-search procedure. BFS are performed by exploring all the nodes at a given depth before proceeding to the next level. This means that all the immediate children of a node are explored before any of the children's children are considered.
If pointer is introduced in the algorithm, then entire path slaimed can be identified.



Algorithm:→ Step1: Put the initial node on a list START.
Step2: If (START is empty) or (START = GOAL) terminate search.
Step3: Remove the first node from START, call this node a.
Step4: If (a = GOAL) terminate search with success.
Step5: Else if node a has successors, generate all of them and add them at the tail of START.
Step6: Go to step 2.

Features of BFS:→ (1) Space Complexity:→ $1+b+b^2+ \text{---} +b^d = O(b^d)$
Since the procedure has to keep track all the children it has generated, the space-complexity is also a function of the depth d and branching factor

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2) <u>Time-Complexity</u>:→ The amount of time taken for generating these nodes is proportional to the depth d and branching factor b and is given by

$$1+b+b^2+b^3+ \text{———} +b^d = O(b^d).$$

(3) <u>Completeness</u>:→ It is Complete.

(4) <u>Optimality</u>:→ BFS will be optimal.

* BFS is better than DFS *.
* BFS is going to give the shortest path solution. *.

<u>Drawbacks of BFS</u>:→

1. Amount of time needed to generate all the nodes is considerable because of the time-Complexity.
2. Memory constraint is also a major hurdle because of the space-Complexity.
3. The searching process remembers all unwanted nodes which is of no practical use for the search.

*(iii) <u>DFS with Iterative Deepning</u>:→ This search is performed as a form of repetitive DFS moving to a successively deeper depth with each iteration. It begins by performing a DFS to a depth of level 1. It then discards all nodes generated and starts over doing a search to a depth of level 2. If no Goal node has been found, it discards all nodes generated and does a DFS to a depth of level 3. This process continues untill a Goal node is found or some maximum depth is reached. Since this algorithm expand

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

all nodes at a given depth before expanding nodes at a greater depth, it is guaranteed to find a shortest path solution like the BFS.

Properties:→ (1) Time Complexity:→ $O(b^d)$.

(2) Space Complexity:→ $O(d)$.

(3) Completeness:→ It is complete.

(4) Optimality:→ It is optimal.

Disadvantages:→ It performs wasted computations before reaching to the goal state.

(2) Heuristic Search:→ Heuristic is some prior knowledge. Heuristics are approximations used to minimize the search process.

Generally two categories of problems used heuristics:

(i) Problems for which no exact algos are known, & one needs to find an approximate and satisfying solⁿ.

(ii) Problems for which exact solⁿs are known, but computationally infeasible.

The following algos make use of heuristic functions:

(i) Hill climbing:→ This algo is the variant of DFS. It is also known by the name of Discrete optimization Algorithm. This algo uses a very single heuristic function i.e. the amount of distance, the node is from the goal.

Algo:→ Step 1: Put the initial node on a list START.

Step 2: If (START is empty) or (START = GOAL) terminate search.

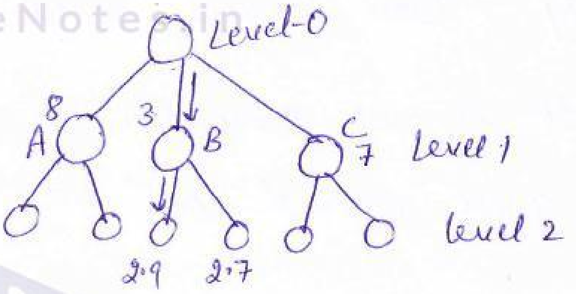
Step 3: Remove the first node from START, call this node a.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Step 4: If (a = GOAL) terminate search with success.

Step 5: Else If node a has successors, generate all of them. Find out how far they are from the goal node. Sort them by the remaining distance from the goal and add them to the beginning of START.

Step 6: Goto step 2.



Problems of Hill-Climbing Technique:→

(1) Local maximum:→ A state that is better than all its neighbours but not so when compared to the states that are farther away.

(2) Plateau:→ A flat area of the search space, in which all neighbours have the same value.

(3) Ridge:→ This is an area in the path which must be traversed very carefully because movement in any direction might maintain one at the same level or result in fast descent.

In order to overcome these problems, adopt one of the following or a combination of the following methods:

(1) Backtracking for local maximum. Backtracking helps in undoing what has been done so far and permits to try a totally different path to attain the global peak.

(2) A big jump is the soln to escape from the plateau. A huge jump is recommended because in a plateau all neighbours

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

points have the same value.

(3) Trying different paths at the same time is the solution for circumventing ridges.

Properties of Hill climbing:→

(1) Time Complexity:→ Depends on heuristic function, H(n).

2.) Space Complexity:→ low storage space is required.

3) Completeness:→ It is not complete.

(4) Optimality:→ It gives optimal solⁿ but in specific cases only.

(ii) Best First Search:→ It is the variation of BFS. The heuristic function used here called an evaluation function is an indicator of how far the node is from the goal node. Goal nodes have an evaluation function value of zero.

Algorithm:- Step 1: Put the initial node on a list START.

Step 2: If (START is empty) or (START = GOAL) terminate search.

Step 3: Remove the first node from START, call this node a.

Step 4: If (a = GOAL) terminate search with success.

Step 5: Else if node a has successors, generate all of them. find out how far they are from the goal node. Sort all the children generated so far by the remaining distance from the goal.

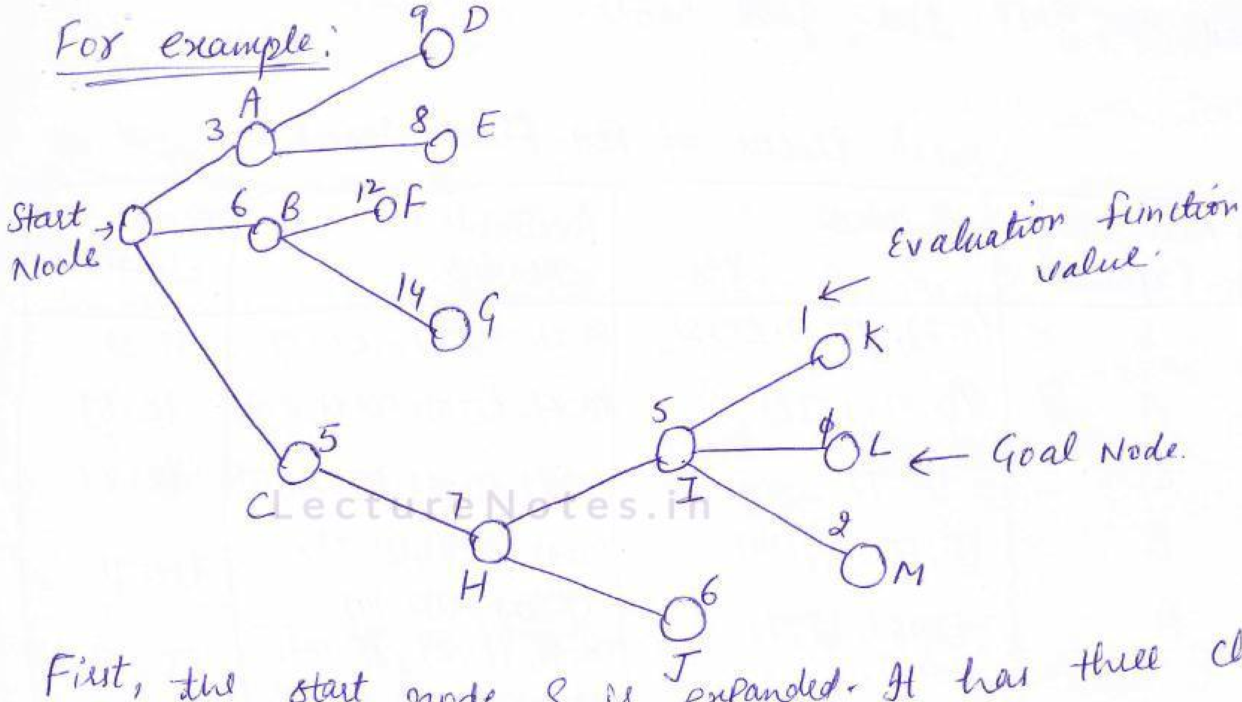
Step 6: Name this list as START1.

Step 7: Replace START with START1.

Step 8: Goto Step 2.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For example:



First, the start node S is expanded. It has three children A, B and C with values 3, 6 and 5 resp. These values approximately indicate how far they are from the goal node. The child with min value A is chosen. The children of A are generated. They are D and E with values 9 and 8.

The search process has now four nodes to search for i.e. node D with value 9, node E with value 8, node B with value 6 and node C with value 5. Of them, node C has got the minimal value which is expanded to give node H with value 7.

At this pt, the nodes available for search are (D:9), (E:8), (B:6) and (H:7) where (α:β) indicates that (α) is the node and β as its evaluation value. Of these, B is minimal and hence B is expanded to give (F:12), (G:14).

At this pt, the nodes available for search are (D:9), (E:8), (H:7), (F:12) and (G:14) out of which (H:7) is minimal and is expanded to give (I:5), (J:6).

→ Nodes now available for expansion are (D:9), (E:8), (F:12), (G:14), (I:5), (J:6).

Of these, the node with minimal value is (I:5) which is

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

expanded to give the goal node.

Search Process of Best-First Search

Step No.	Node being Expanded	Children	Available Nodes	Node chosen
1.	S	(A:3), (B:6), (C:5)	(A:3), (B:6), (C:5)	(A:3)
2.	A	(D:9), (E:8)	(B:6), (C:5), (D:9), (E:8)	(C:5)
3.	C	(H:7)	(B:6), (D:9), (E:8), (H:7)	(B:6)
4.	B	(F:12), (G:14)	(D:9), (E:8), (H:7), (F:12), (G:14)	(H:7)
5.	H	(I:5), (J:6)	(D:9), (E:8), (F:12), (G:14), (I:5), (J:6)	(I:5)
6.	I	(K:1), (L:0), (M:2)	(D:9), (E:8), (F:12), (G:14), (J:6), (K:1), (L:0), (M:2)	Search stops as goal is reached

Properties :→ (1.) Time Complexity :→ Depends on heuristic function.

(2.) Space Complexity :→ High storage space required.

(3.) Completeness :→ It is complete only if tree is of finite depth

(4.) Optimality :→ It is optimal.

(iii) A* Algorithm :→ In A* algo, we consider fitness no. f(n) as the heuristic function which is the combination of evaluation function H(n) and cost function C(n).

Fitness no. f(n) = Heuristic function H(n) + cost function C(n)

Algorithm :→ Step 1: Put the initial node on a list START.

Step 2: If (START is empty) or (START = GOAL) terminate search.

Step 3: Remove the first node from START. Call this node a.

Step 4: If (a = GOAL) terminate search with success.

Step 5: Else if node a has successors, generate all of them. Estimate

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

the fitness no. of the successors by totaling the evaluation function value and the cost-function value. 7.

Sort the list by fitness number.

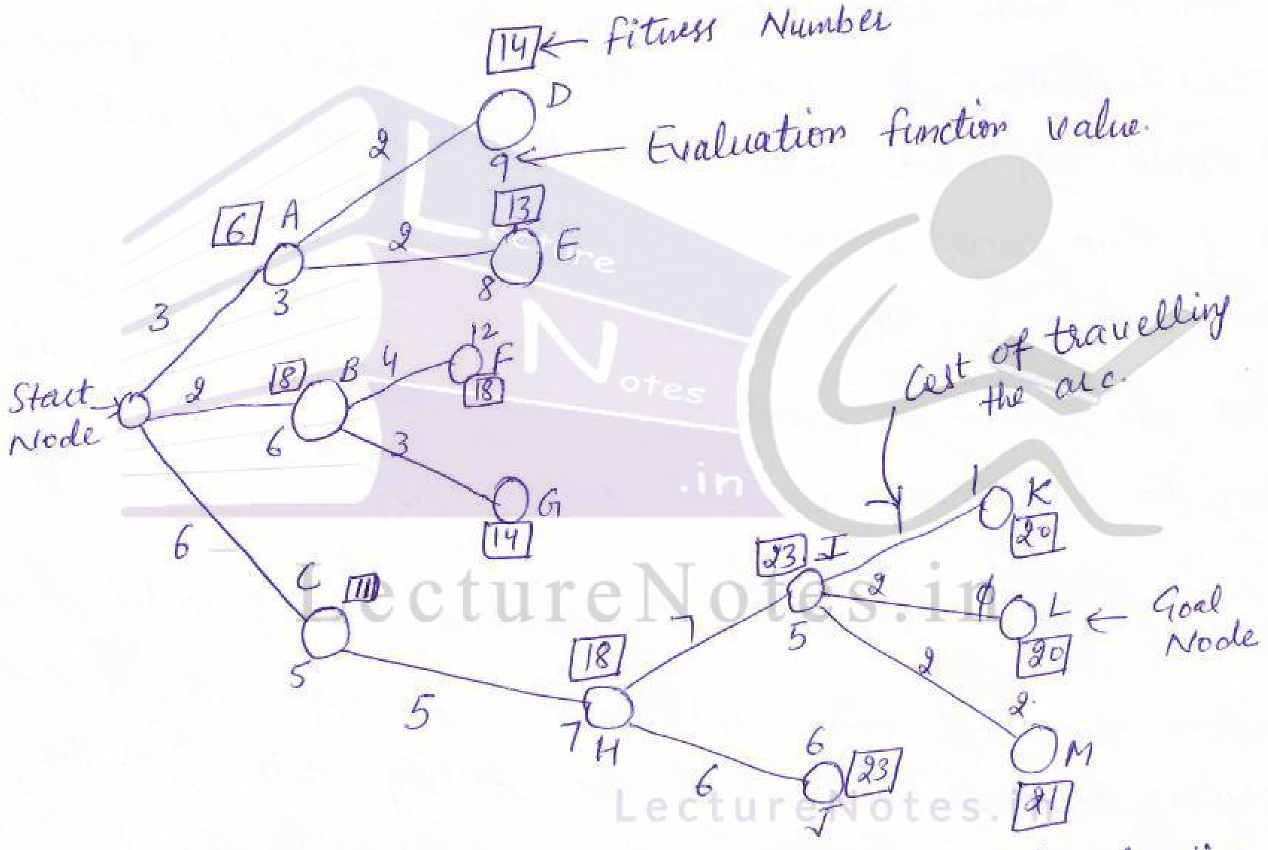
step6: Name the new list as START1.

Step7: Replace START with START1.

Step8: Goto step 2.

LectureNotes.in

For example:



The fitness no is the total of the evaluation function value and the cost-function value.

For eg → Consider node K, the fitness no. is 20, which is obtained as follows:

(Evaluation function of K) + (Cost function involved from start node S to node K)

= 1 + (Cost function from S to C + Cost function from C to H + Cost function from H to I + Cost function from I to K)

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

A* uses the fitness no. for its Computation while the best-first search uses the evaluation function value Only for expanding the best node.

(iv) AO* Algorithm :→ (AND-OR Graph) :→ The AO* algorithm carries out the search process with the help of AND-OR tree. It considers the AND as well as OR cases that could happen in real life problem thats why this searching is also known as AND-OR Graph Searching.

In this algorithm, the problem could be solved is decomposed into small subproblems which are then represented with the help of AND arcs.

This algo requires that nodes traverse in the tree be labeled as solved or unsolved in the solution process to account for and Node Solutions which ~~req~~

The major steps of the AO* algorithm are presented below:

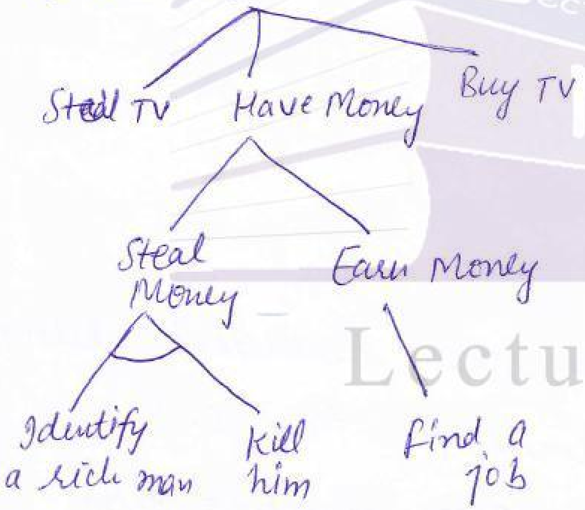
1. Given the Goal node, called the starting node, find the possible offsprings (children) of the starting state, s.t. the Goal can be derived from them by AND/OR clauses.
2. Estimate the h' values at the leaves and find the leaf (leaves) with minimum h'. The cost of the parent of the leaf (leaves) is the minimum of the cost of the OR clauses plus one or the cost of the AND clauses plus the number of AND clauses. After the children with minimum h's are estimated, a pointer is attached a point from the parent

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

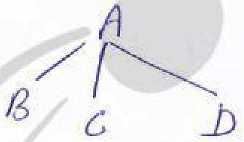
(3) One of the unexpanded OR clauses / the set of unexpanded AND clauses, where the pointer points from its parent, is now expanded and the h' of the newly generated children are estimated. The effect of this h' has to be propagated up to the root by recalculating the f' of the parent or the parent of the parents of the newly created child / children clauses through a least cost path. Thus the pointers may be modified depending on the revised cost of the existing clauses.

Eg.→

Goal : Acquire a TV set



Symbols



IF (B OR C OR D)
THEN A.

IF ((B AND C) OR D)
THEN A.

Procedure AO*

Begin

1. Given the goal node INIT in the graph G; evaluate h' at INIT.

2. Repeat

a.) Trace the marked arcs from node INIT, if any such exists, and select one of the unexpanded nodes,

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

named NODE, that occurs on this path, for expansion.

(b.) If NODE cannot be expanded, Then assign FUTILITY as the h' value of NODE, indicating that NODE is not solvable. Else for each successor, called SUCCESSOR, which is not an ancestor of NODE,

do

Begin

(i) Append SUCCESSOR to the Graph G.

(ii) If SUCCESSOR is a terminal node Then label it SOLVED and set its h' value 0.

(iii) If SUCCESSOR is not a terminal node Then estimate its h' value.

End;

(c.) Initialize S to NODE.

(d.) Repeat

(i) Select from S a node, none of whose descendants belong to S. Call it CURRENT and remove it from S.

(ii) Estimate the cost of each of its arcs, emerging from CURRENT. The cost of each arc is equal to the sum of h' value of each of the nodes at the end of arc plus the cost of the arc itself. The new h' value of CURRENT is the minimum of the cost just computed for the arcs emerging from it.

(iii) Label the best path out of CURRENT by marking the arc that had the least cost as computed in the last step.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(iv) If all of the nodes connected to CURRENT through the new marked arcs have been labeled SOLVED, Then mark the CURRENT SOLVED.

(v) If CURRENT is marked SOLVED or the cost of CURRENT was changed, Then propagate its new status back up the tree, add all the ancestors of CURRENT to S. Until S is empty.

Until INIT is labeled solved or its h' value becomes greater than a maximum level called FUTILITY.

End.

* Beam Search:→ The searching process in beam search is similar to breadth-first search wherein searching proceeds level by level. At each level, heuristic functions are applied to reduce the no. of paths to be explored.

→ It is done to keep the width of the beam to be minimal. The width of the beam is fixed & whatever be the depth of the tree, the no. of alternatives to be scanned is the product of the width and the depth.

Algo:→ Step1: Let width_of_beam = w

Step2: Put the initial node on a list START

Step3: If (START is empty) or (START = GOAL) terminate search.

Step4: Remove the first node from START. Call this node a.

Step5: If (a = GOAL) terminate search with success.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Step 6: Else If node a has successors, generate all of them and add them at the tail of START.

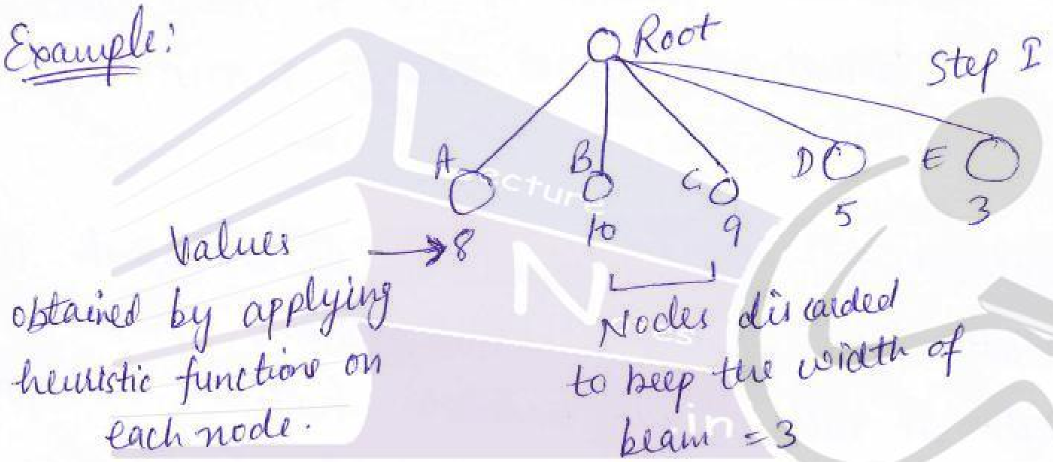
Step 7: Use a heuristic function to rank and sort all elements of START.

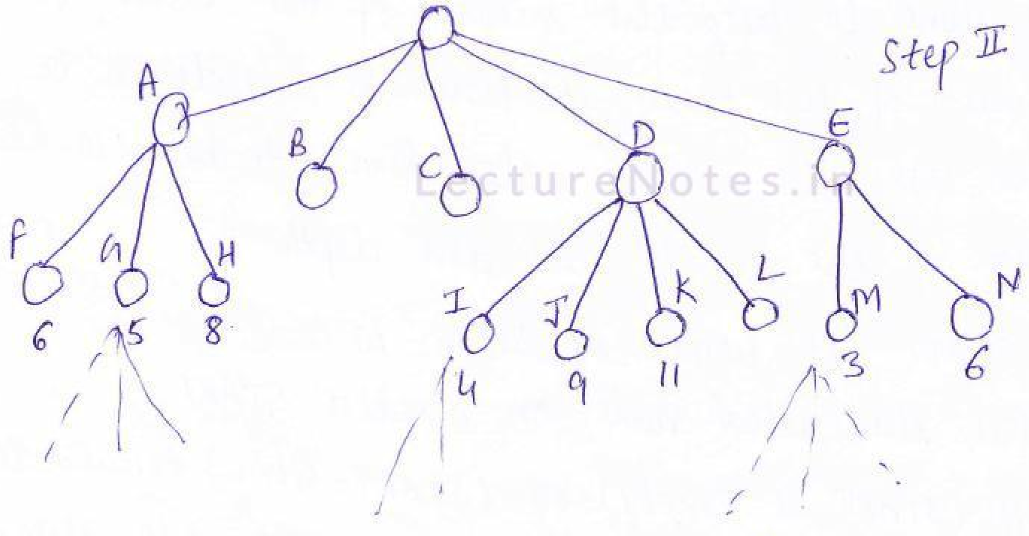
Step 8: Determine the nodes to be expanded. The no. of nodes should not be greater than 10. Name these as START1.

Step 9: Replace START with START1.

Step 10: Goto step 2.

Example:





How Beam Search Proceeds

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Game Playing :→ The reasons why game-playing occupies a pivotal role in AI are:

1.) The rules of the game are limited. Hence extensive amounts of domain-specific knowledge are seldom needed.

2.) Many human experts exist to assist in the developing of the programs.

3.) Games provide a structured task wherein success or failure can be measured with least effort.

In Game-playing literature, the term play is used for a move. With this background information, let's look at what are the major components of game-playing program.

Major Components of Game-playing Program :→

There are two major components of a game-playing program, a plausible move generator and a static evaluation function generator.

Plausible Move Generator :→ For every move a player makes in the game of chess, the branching factor is 35, i.e. the opponent can make 35 different moves.

If we are to employ a simple move-generator, then it might not be possible to examine all the states. Hence it is essential that only very selected moves or paths be examined. For this one has a plausible move generator, that expands or generates only selected moves. It is not possible for all moves to be examined because

(1) The amount of time given for a move is limited.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2) The amount of computational power available at the disp-al for examining various state is also limited.

(3) Static Evaluation Function Generator:→ This is the most im component of the game-playing program.

Based on heuristics, this generates the static evaluation func value for each and every move that is being made. More the static evaluation function value, more is the probability for a victory.

Static evaluation function generator occupies a crucial role in game-playing programs because of the following factors:

(i) It utilizes the heuristic knowledge for evaluating the static evaluation function value.

(ii) The static evaluation function generator acts like a pointe to point the way the plausible move generator has to generate future paths.

The basic characteristic of the strategy must be look ahea in nature i.e. explore the tree for two or more level downwards and choose the optimal one.

The basic methods available for game playing are:

(1) Minimax Strategy

(2) Minimax strategy with alpha-beta cutoffs.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

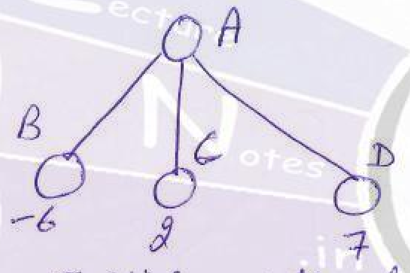
(1) <u>Minimax Strategy</u> :→ Minimax strategy is a simple look ahead strategy for two-person game playing.

Here one player is called a maximizer and the other is called a minimizer.

→ Both the maximizer and minimizer fight it out to see to see that the opponent gets the minimum benefit while they get the maximum benefit.

→ The plausible move generator generates the necessary states for further evaluation and the static evaluation function "ranks" each of the positions.

Eg.→



Initial state of the game

Let A be the initial state of the game. The plausible move generator generates three children for that move and the static evaluation function generator assigns the values given along with each of the states.

It is assumed that the static evaluation function generator returns a value from -20 to +20, wherein a value of +20 indicates a win for the maximizer and a value of -20 a win for the minimizer.

A value of 0 indicates a tie or draw.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

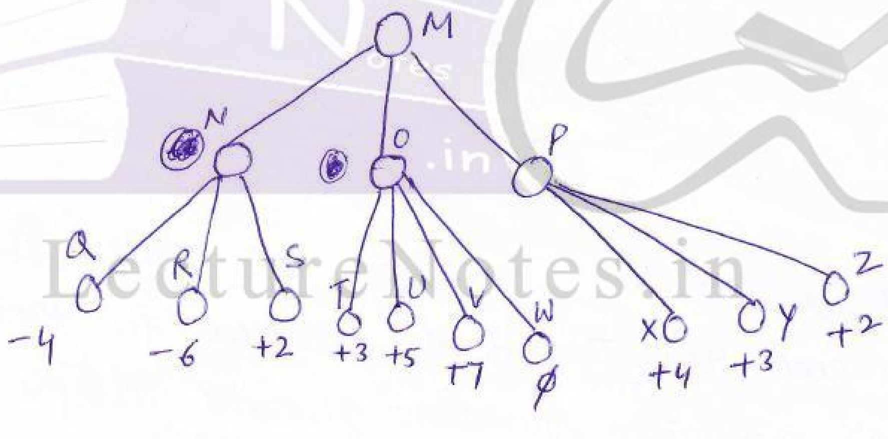
Case-I:

→ It is also assumed that the maximizer makes the first move. The maximizer, always tries to go to a position where the static evaluation value is the maximum positive value.

→ The maximizer, being the player to make the first move, will move to node D because the static evaluation function value for that node is maximum.

→ If the minimizer has to make the first move, he will go to node B because the static evaluation function value at that node is advantageous to him.

→ But ~~the~~ a game-playing strategy never stops with one level but looks ahead.



Let's assume that it is the maximizer again who will have to play first followed by the minimizer. The search strategy here tries for only two moves, the root being M and the leaf nodes being Q, R, S, T, U, V, W, X, Y and Z.

→ Before the maximizer moves to N, O or P, he will have to think which move would be highly beneficial to him. In order to evaluate the children of intermediate nodes N, O and P are generated and the static evaluation

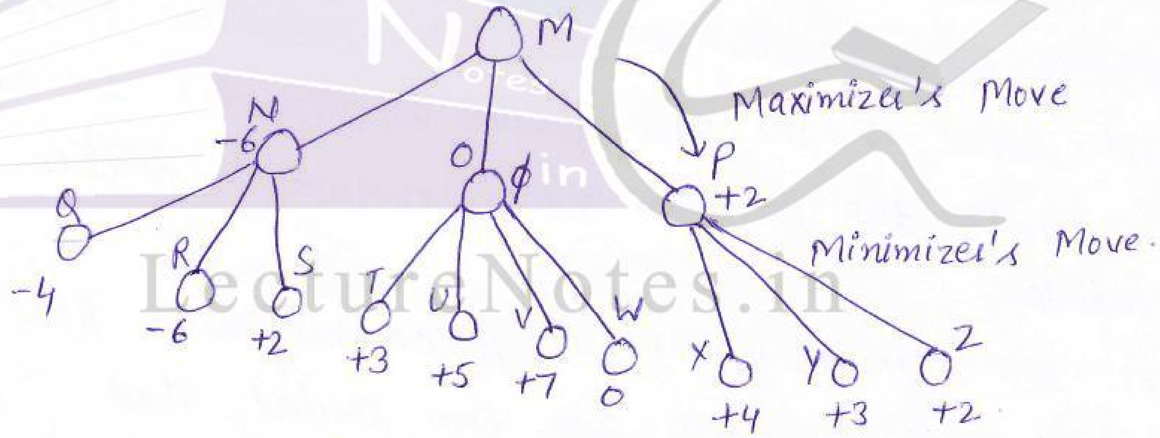
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

function value generator has assigned values for all the leaf nodes.

→ If M moves to N, it is the minimizer who will have to play next. The minimizer always tries to give the minimum benefit to the other and hence he will move to R. This value is backed up at N.

→ If M moves to O, then the minimizer will move to W, which is minimum of +3, +5, +7 and 0. So the value 0 is backed up at O.

→ Similarly, the value ~~the value~~ that is backed up at P is 2. The tree now with the backed up values is as follows:



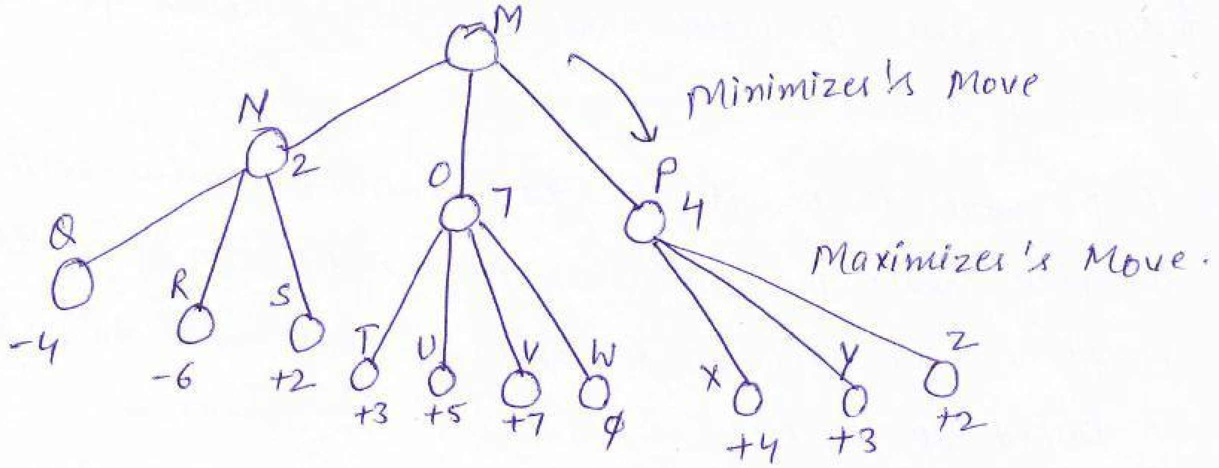
Maximizer's Move.

→ The maximizer will now have to choose between N, O or P with the values -6, 0 and 2.

→ Being a Maximizer, he will choose node P because of getting a maximum value of 2.

Case II: If minimizer will have to make the first move: ~~[illegible]~~ following Tree show this:

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



This search has just stopped with two levels only. However, it is possible to consider more levels for accurate results.

<u>Algorithm for Minimax:</u> The algorithm determines who is making the first move, the maximizer or the minimizer. If the maximizer is the player, the algo is recursively used to estimate the maximum of the static evaluation function value and report it. The same process is repeated for minimizer also with the minimum static evaluation function value being reported.

Step1: Set FINAL_VALUE to be as minimum as possible.

Step2: If limit of search has been reached, then
FINAL_VALUE = GOOD_VALUE of the current position.

Step3: Else do
step 3.1: Generate the successors of the position.
Step 3.2: Recursively call MINIMAX again with the present position with depth incremented by unity.

Step4: Evaluate the GOOD_VALUE.

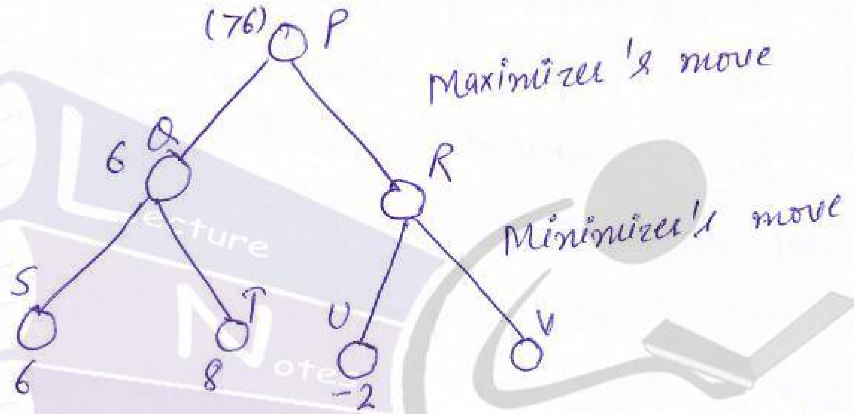
Step5: If GOOD_VALUE > FINAL_VALUE then
FINAL_VALUE = GOOD_VALUE.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* **Modified Minimax with Alpha-Beta Cutoffs** :→ This is a modified version of minimax algo wherein two threshold values are maintained for future expansion.

→ One threshold value is called alpha, which is the lower bound on the value, the maximizer can be assigned and the other is beta, which represents the upper bound on the value the minimizer can be assigned.

Example: Consider the tree structure:

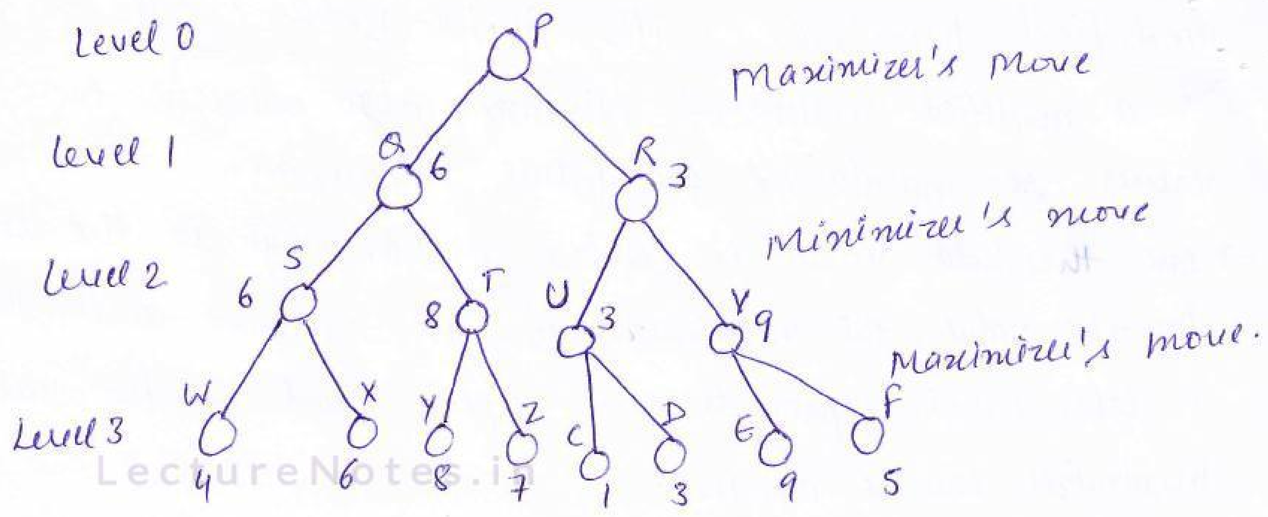


→ The maximizer has to play first followed by the minimizer. As in minimax, look ahead search is done.

→ The maximizer assigns a value of 6 at Q (minimum of S & T). This value is passed back to P. So, the maximizer is assured of a value of 6 when he moves to Q.

→ Now, the value at U is -2 and V is unknown. Since the move is a minimizing one, by moving to R, P can get only a value of -2 or less than that. It is unacceptable for P because by moving to Q, he is assured of a value of 6. Hence P will not examine V or other children of R.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



Role of Alpha.

Here P is the maximizing player. Before P can branch to Q & R, a look ahead search is done upto level 3. The static evaluation function generator has assigned values which are given for the leaf nodes. Since S, T, U and V are also maximizers, the maximum of the leaf nodes are assigned to them. Thus S, T, U and V have the values 6, 8, 3 and 9 resp.

→ The Predecessor level i.e. the nodes Q and R are minimizers. Thus Q takes the minimum of 6 and 8, R takes the minimum of 3 and 9. Since P is a maximizer, P will opt for Q.

Let's examine the roles of alpha and beta.

Role of Alpha:→

For P, the maximizer, a value of 6 is assured by moving to node Q.

P, being a maximizer, would follow any path whose value is greater than 6. Hence, this value of 6, being the least that a maximizing node can obtain is set as the value of alpha

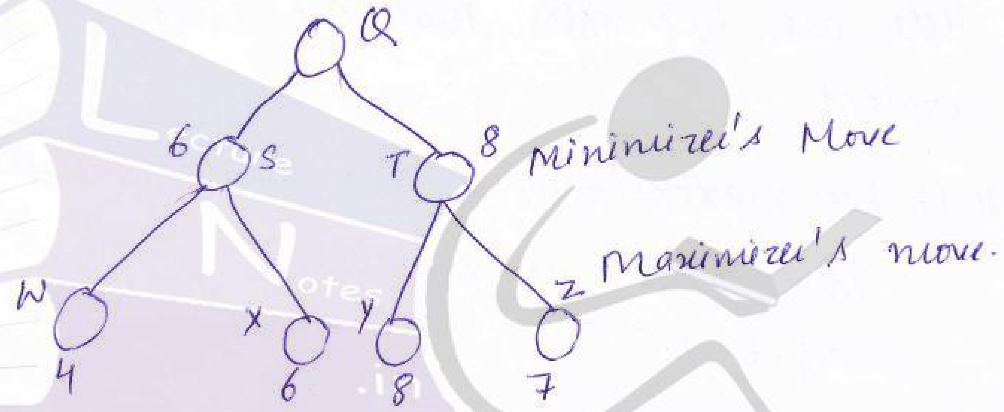
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

14.

→ This value of alpha is now used as a reference point. Any node whose value is greater than alpha is acceptable and all nodes whose values are less than the alpha value are rejected.

→ Here the value at R is 3, which is much lower than the alpha cut off. Hence the entire tree under R is totally rejected.

Role of Beta:→ Consider the tree, which is the portion of tree considered before:



Role of Beta.

Here Q is the root. Q is a minimizer and the paths for the expansion are chosen from the values at the leaf nodes.

Since S and T are maximizers, the maximum values of their children are backed up as their static evaluation function values.

→ Node Q, being a minimizer, will always move to S rather than T.

The value at S (6) is now used by Q as a reference. This value is called Beta, the maximum value, a minimizer can

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

be assigned. Any ~~statue~~ node whose value is less than this beta value (6) is acceptable and values more than beta are prefered.

→ The value of beta is passed to node T. Comparing it with the static evaluation function value, the minimizer will be benefited by only moving S rather than T. Hence the entire tree under node T is pruned.

Here are two rules that are used in modified minimax strategy:

Rule 1: For maximizers, if the static evaluation function value found at any node is less than alpha value, reject it.

Rule 2: For minimizers, if the static evaluation function value found at any node is more than the beta value, reject it.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Main Characteristics of Search Algorithms:→

Desirable Properties of Heuristic Search Algorithm:→

(1) **Admissibility Property:→** An algorithm is admissible if it is guaranteed to return an optimal solution when one exists. Example: BFS and A^* algorithms are admissible search algorithms.

(2) **Completeness Property:→** An algorithm is Complete if it always terminates with a solution when one exists.

(3) **Dominance Property:→** Let A_1 and A_2 be admissible algos with heuristic estimation function h_1^* and h_2^* respectively. Then A_1 is said to be more informed than A_2 whenever $h_1^*(n) > h_2^*(n)$. $\forall n$.

Here A_1 is said to dominate A_2.

(4) **Optimality Property:→** Algorithm A is optimal over a class of algorithms if A dominates all members of the class. i.e. $h_A^*(n) > h_i^*(n)$ $\forall i$ {Class of algorithms}

(5) **Monotonicity:→** It states that, " If a state is discovered by using heuristic search and it is guaranteed that same state would not be found later in the search at the cheaper cost then that algorithm is called to have monotoncity.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(6) Informedness :→ For two A^* heuristics h_1 and h_2, if $h_1(n) <= h_2(n)$, for all states n in the search space, heuristic h_2 is said to be more informed than h_1.

This property is a means to distinguished a heuristic as better than another heuristic. So if for two heuristic A_1 and A_2, cost functions are found to be h_1 and h_2 respectively and if $h_1(n) < h_2(n)$ $\forall n$ then A_1 is said to be more informed than A_2.

LectureNotes.in

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(3) PROPEL → Application of physical force of an object.
(e.g. throw)

4) MOVE → To make a move. (e.g. kick)

5) GRESP → Grasping of an object. (e.g. hold)

6.) INGEST → Taking an object by an animal (e.g. eat, drink)

— etc.

* Advantages of CD :→

(1) CD helps in representing wide knowledge in simple w

(2) CD is to make explicit of which is implicit.

(3) CD brings forward the notion of language independe

Disadvantages :→

1.) Representation may be Complex even for simple actions.

2) Complex representation require a lot of storage.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence
MCA-405.

Topics Covered :⇒

1. Introduction to Knowledge Acquistion
2. Types of Learning
3. General Learning Model
4. Factors Affecting the Performance of a Learning System
5. Performance Measures
6. Learning Automata
7. Genetic Algorithm
8. Intelligent Editors
9. Learning By Induction
10. Generalization & Specialization
11. Introduction to PROLOG
12. Parts of PROLOG Program.
13. Basis of PROLOG Language
14. Controlling Execution in PROLOG.
15. Recussion
16. Rules and facts.

Nidhi Kalra
Assistant Professor

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

LECTURENOTES.IN

Artificial Intelligence

Topic:

Production System

Contributed By:

Sahil Kumar

Unit-III

Artificial Intelligence

MCA-405

Production System:→ It is the powerful tool in AI which is meant for two purposes:

(i) For Problem Solving.

(ii) For implementing search algorithm.

A production system provides pattern-directed control of a problem solving process.

* **Components of Production System:→**

1) Set of Production Rules.

2) Working Memory

3) Recognize-act cycle.

(1) **Set of Production Rules:→** These are simply called productions. A production is called a condition-action pair. ~~and define~~ The Condition part of the rule is a pattern that determines when that rule may be applied to a problem instance. The action part defines the associated problem-solving step.

(2) **Working Memory:→** Working memory contains a description of the current state of the world in a reasoning process. This description is a pattern that is matched

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

against the condition part of a production to select appropriate problem-solving actions.

When the condition element of a rule is matched by the contents of working memory, the action associated with that condition may then be ~~preferred~~ performed.

The actions of production rules are specificially designed to alter the contents of working memory.

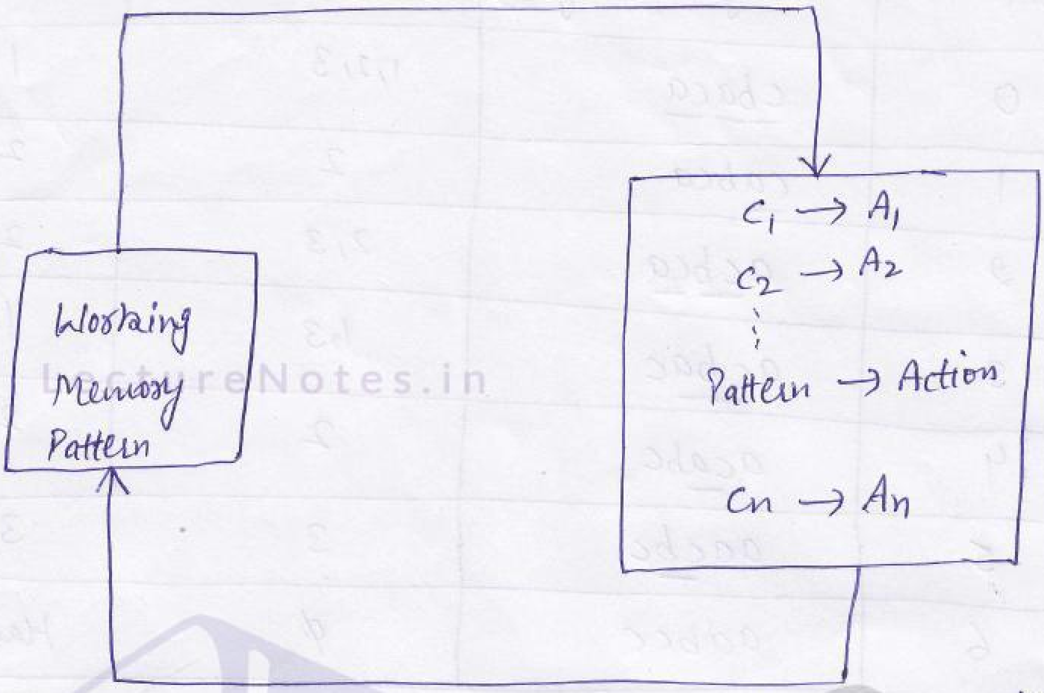
(3) The recognize-act cycle :→ The control structure for a production system is simple:

Working memory is initialized with the beginning problem description. The current state of the problem-solving is maintained as a set of patterns in working memory. These patterns are matched against the conditions of the production rules: this produces a subset of the production rules, called the conflict set, whose conditions match the patterns in working memory.

The productions in the conflict set are said to be enabled. One of the productions in the conflicts set is then selected (conflict resolution) and the production is fired. To fire a rule, its action is performed, changing the contents of working memory.

After the selected production rule is fired, the control cycle repeats with the modified working memory. The process terminates when the contents of working memory

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



A production system. Control loops until working memory pattern no longer matches the conditions of any productions.

Example: Here is an example of production system execution.

Production Set:
1. ba → ab
2. ca → ac
3. cb → bc.

This is a production system program for sorting a string composed of the letters a, b and c. In this example, a production is enabled if its condition matches a portion of the string in working memory. When a rule is fired, the substring that matched the rule condition is replaced by the string on the right-hand side of the rule.

→ Production systems are a general model of computation that can be programmed to do anything that can be done on a computer.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Iteration #	Working Memory	Conflict Set	Rule fired
0	cbaca	1,2,3	1
1	cabca	2	2
2	acbca	2,3	2
3	acbac	1,3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	ϕ	Halted

Trace of a simple production system.

* **Control of Search in Production Systems:→** Control of search in production system is done in three ways

(1.) With the help of search strategy.
↳ Data-Driven strategy
↳ Goal-Driven strategy.

(2.) Through Rule structure.

(3.) Through Conflict Resolution.

(1) **Control of Search through Data Driven Strategy:→** Data Driven search begins with a problem description and infers new knowledge from the data. This is done by applying rules of inference, legal moves in a game or other state-generating op's to the current description of the world and adding the results to that problem description. This process continues

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

until a goal state is reached.

This description of data-driven reasoning emphasizes its close fit with the production system model of computation.

→ The current state of the world is placed in working memory. The recognize-act cycle then matches the current state against the set of productions.

→ All productions have the form CONDITION → ACTION. When the CONDITION matches some elements of working memory, its ACTION is performed.

Example: production set:

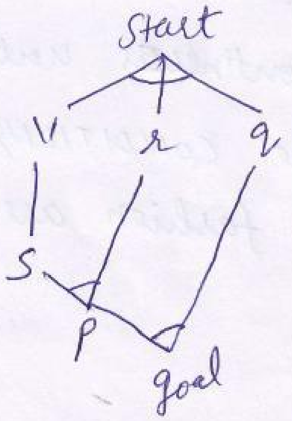
1. $p \wedge q \rightarrow$ Goal
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow q$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. start $\rightarrow v \wedge r \wedge q$

Trace of Execution:

iteration	Working Memory	Conflict Set	Rule fired
0	Start	6	6
1	Start, v, r, q	6, 5	5
2	Start, v, r, q, s	6, 5, 2	2
3	Start, v, r, q, s, p	6, 5, 2, 1	1
4	Start, v, r, q, s, p, goal	6, 5, 2, 1	Halt

⇓ Direction of search

Space searched by execution:



Data-Driven Search in a production System.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

This figure presents a simple data-driven search on a set of productions expressed as propositional calculus implications. The conflict resolution strategy is a simple one of choosing the enabled rule that has fired least recently, in the event of ties, the first rule is chosen.

→ Execution halts when a goal is reached.

Control of Search Through Goal-Driven Strategy:⇒ Goal-Driven search begins with a goal and works backward to the facts of the problem to satisfy that goal.
To implement this in a production system the goal is placed in working memory and matched against the ACTIONS of the production rules. These ACTIONS are matched just as the CONDITIONS of the productions were matched in the data-driven reasoning.

→ When the ACTION of a rule is matched, the CONDITIONS are added to working memory and become the new subgoals (state) of the search.
The new states are then matched to the ACTIONS of the other production rules. The process continues until a fact is found. The search stops when the CONDITIONS of all productions fired in this backward fashion are found to be true.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

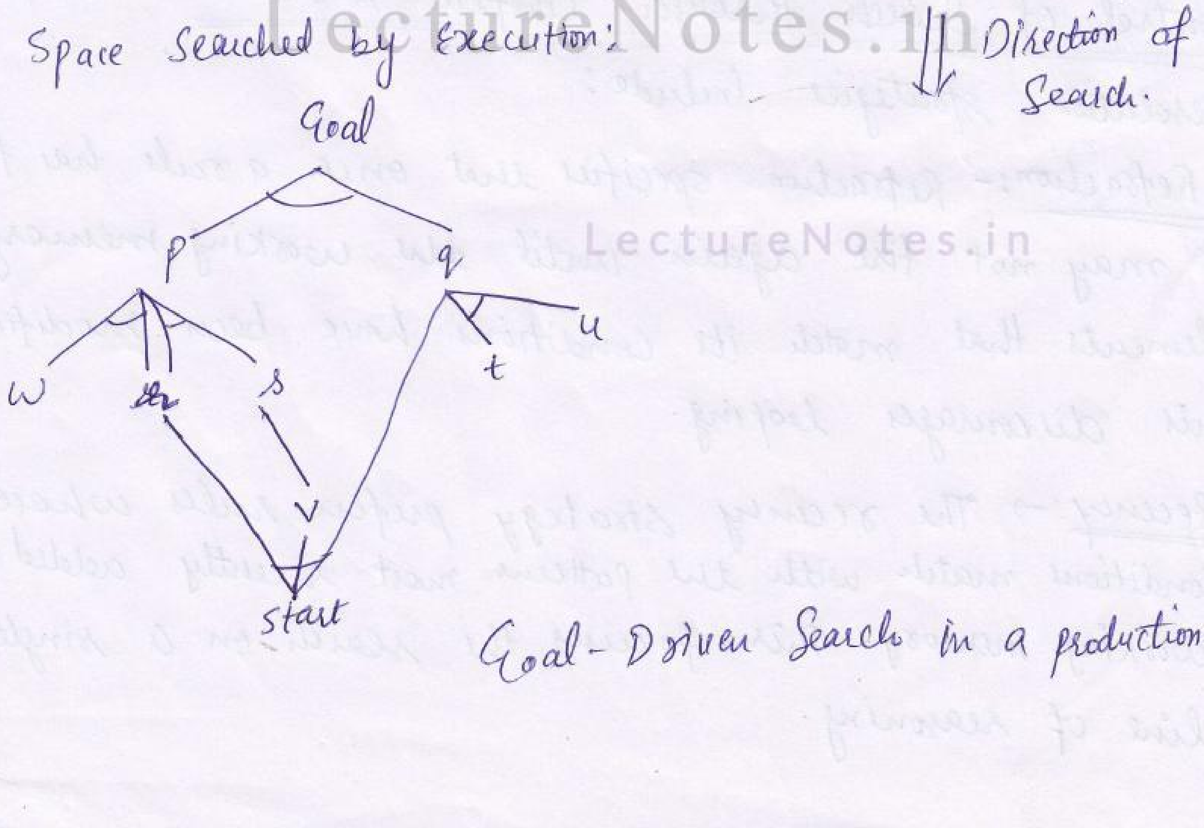
Example: Production Set:

1. $p \wedge q \rightarrow$ Goal
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow p$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. start $\rightarrow v \wedge r \wedge q$

Trace of Execution:

Iteration	Working Memory	Conflict Set	Rule fired
0	Goal	1	1
1	Goal, p, q	1, 2, 3, 4	2
2	Goal, p, q, r, s	1, 2, 3, 4, 5	3
3	Goal, p, q, r, s, w	1, 2, 3, 4, 5	4
4	Goal, p, q, r, s, w, t, u	1, 2, 3, 4, 5	5
5	Goal, p, q, r, s, w, t, u, v	1, 2, 3, 4, 5, 6	6
6	goal, p, q, r, s, w, t, u, v, start	1, 2, 3, 4, 5, 6	Halt

Space Searched by Execution:



Goal-Driven Search in a production System.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2) Control of Search through Rule structure :→ The structure of rules in a production system determines the fashion in which the space is searched.

For eg. → A rule $\forall x\ (f(x) \wedge g(x) \rightarrow m(x))$ and alternative of the same rule is $\forall x\ (f(x) \rightarrow m(x) \vee \neg g(x))$.

Although these formulations are logically equivalent, they do not lead to the same results when interpreted as productions because the production system implementation imposes an order on the matching and firing of rules.

→ Because the production system tries each of its rules in a specific order, the programmer may control search through the structure and order of rules in the production set.

(3) Control of Search through Conflict Resolution :→ Conflict Resolution strategies include:

(i) Refraction :- Refraction specifies that once a rule has fired, it may not fire again until the working memory elements that match its conditions have been modified. This discourages looping.

(ii) Recency :→ The recency strategy prefers rules whose conditions match with the patterns most recently added to working memory. This focuses the search on a single line of reasoning.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(iii) Specificity:→ This strategy assumes that it is appro-priate to use a more specific problem-solving rule rather than to use a more general one. One rule is more specific than another if it has more conditions, which implies that it will match fewer working memory patterns.

* Types of Production System:→ four types:

1.) Commutative

2.) Decomposable

3.) Monotonic

4.) Non-monotonic

(1) Commutative:→ A production system is called commutative if for a given set of rules "R" and a working memory "W", it provides the freedom in order of rules firing i.e. for a given set of rules, the random order of firing. The applicable rule will not make a difference in content of working memory as well as it not affect the goal state.

→ The most significant advantage of this system is that, rules can be fired in any order without having the risk of loosing the goal in case if it is attainable.

(2) Decomposable Production System:→ It is the production system in which the contents of working memory and goal state to be acheived can be decomposed or partitioned into disjoint sets, that can be processed independently and made identical to the set of production rules given.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For example:- Consider a production system with initial contents C, B, Z and the set of production rules as follows:

Production Set:

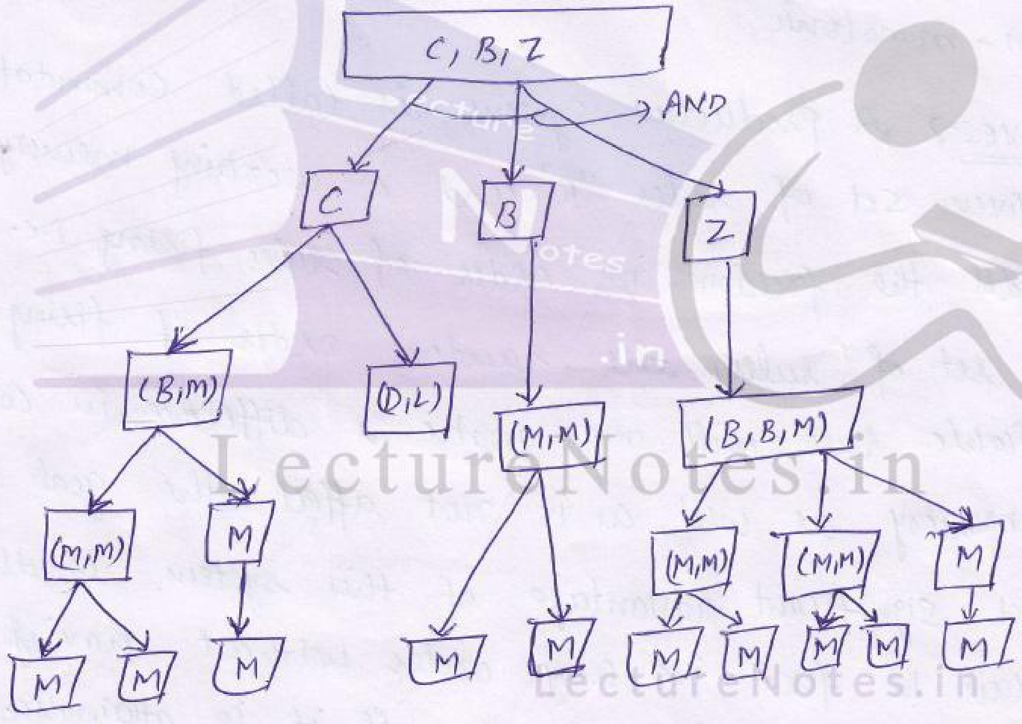
R1: C → (D, L)

R2: C → (B, M)

R3: B → (M, M)

R4: Z → (B, B, M).

The target is to produce working memory contains M only.



The search strategy used here is BFS in which component at each level is processed independently.

③ Monotonic :→ A logic system is monotonic if the truth of proposition does not change when some new information is added to the system.

For eg.→ If a proposition is given: 'Birds can Fly' and

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

later one more proposition is added to the system. 'Oastrich cannot fly' will never affect the earlier proposition.

(4) Non-Monotonic:→ It is the reverse of monotonic production system i.e. if the new information is added to the system, the truth of already existing information may change.

* Advantages of Production Systems for AI:→

(1) Separation of Knowledge and Control:→ The Production System is an elegant model of separation of knowledge and control in a computer program. The advantages of this separation include ease of modifying the knowledge base without requiring a change in the code for program control and conversly, the ability to alter the code for program control without changing the set of production rules.

(2) A natural Mapping onto State Space Search:→ The Components of a production system map naturally into the constructs of state space search. The production rules are set of possible transitions b/w states, with conflict resolution implementing the selection of a branch in the state space.

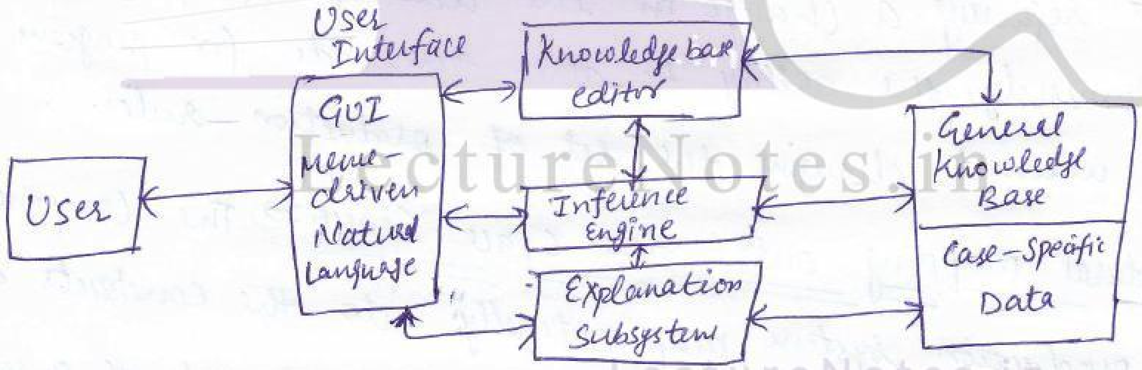
(3) Modularity of Production Rules:→ Rules may only effect the firing of other rules by changing the pattern in working memory, they may not call another rule directly as if it were a subroutine, nor may they set the value of variables in other production rules.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(4) Pattern-Directed Control :→ The problems addressed by AI programs require particular flexibility in program execution. This goal is served by the fact that the rules in a production system may fire in any sequence.

(5) Language Independence :→ The production system control model is independent of the representation chosen for rules and working memory. Although there are many advantages to using logic as both the basis for representation of knowledge and the source of sound inference rules, the production system model may be used with other representations.

* Overview of Expert System Technology :→ (Luger)
↳ Design of Rule-Based Expert Systems :→ (Components / Architecture)



Architecture of Expert System for a problem Domain.

(1.) User Interface :→ The user interacts with the system through a user interface that simplifies communication and hides some details such as internal structure of the rule base. The final decision on the interface type is a compromise b/w user needs and requirements of the knowledge base and inferencing system.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Knowledge Base:→ The heart of the expert system is the knowledge base, which contains the knowledge of a particular application domain. In a rule-based expert system this knowledge is represented in the form of if-then-rules. The KB contains both general knowledge as well as case-specific info.

(2) Inference Engine:→ It applies the knowledge to the solution of actual problems. In production sys., the inference engine performs the recognize-act control cycle. The procedures that implement the control cycle are separate from the production rules themselves. It is imp. to maintain this separation of the KB and inference engine for several reasons:

(i) This separation makes it possible to represent knowledge in a more natural fashion. for eg.→ If-then-rules are closer to the way in which humans describe their problem-solving skills than a lower-level computer code.

(ii) Because the KB is separated from the program's lower-level control structures, expert system builders can focus on capturing and organizing problem-solving knowledge rather than on the details of its computer implementation.

(iii) The separation of knowledge & control allows changes to be made in one part of KB without creating side-effects in others.

(iv) The separation of knowledge & control elements of the program allows the same control and interface s/w to be used in a variety of systems.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

③ Knowledge-Base Editor:→ It help the programmer locate and correct bugs in the program's performance, often accessing the information provided by the explanation subsystem. They also assist in the addition of new knowledge, help maintain correct rule syntax and perform consistency checks on the updated KB.

④ Explanation Subsystem:→ It allows the program to explain its reasoning to the user. These explanation include justific-ations for the system's conclusions, in response to 'how queries', explanations of why the system needs a particular piece of data, 'why queries' and where useful, tutorial explanations or deeper justifications of the program's actions.

⑤ Case-Specific Data:→ The expert system must keep track of case-specific data. These are facts, conclusions and other information relevant to the case under consideration. This includes the data given in a problem instance, partial conclusions, confidence measures of conclusions and dead ends in the search process. This info is separate from the general KB.

* Characterstics of an Expert System:→ (Janki Raman)

① They should solve difficult programs in a domain as good as or better than human experts.

② They should possess vast quantities of domain-specific knowledge to the minute details. For successful operations, vast quantities of domain specific knowledge is needed.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(3) These systems permit the use of heuristic search process. As brute force search techniques are impractical and to manage the problem, heuristic search procedures are used. Expert systems provide fascilities for incorporating these heuristic search procedures.

(4) They explain why they ask a question and justify their conclusions.

(5) They accept advice, modify, update & expand.

(6) They deal with uncertain & irrelevant data.

(7) They communicate with the users in their own natural language.

(8) They provide extensive fascilities for symbolic processing rather than numeric processing. Symbolic processing is the core of any AI program and hence an ES should provide fas-cilities for doing so.

→* Expert System Life Cycle:→ (Development of Expert System)

There are five major stages in the development of an ES. Each stage has its own unique features:

Stage 1: Identification of the Problem:→ In this stage, the expert and the knowledge engineer interact to identify the problem. The amount of resources needed, eg. men, computing resources, finance etc. are identified. Areas in the problem which can give much trouble are identified and a conceptual solution for that problem and the overall specification is made.

Stage 2: Decision about the mode of development:→ Once the problem is identified, the immediate step would be to decide on the vehicle for development. The knowledge engineer can develop

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

the system from scratch using a programming language like LISP or PROLOG or any conventional language. In this, various shells and tools are identified and analyzed for the suitability.

Stage 3: Development of a prototype:→ Before developing a prototype, the following are prerequisite activities:-

(a) Decide on what concepts are needed to produce the sol? one imp. factor to be decided here is the level of knowledge. Starting with coarse granularity, the sys. development proceeds towards fine granularity.

(b.) After this, the task of knowledge acquisition begins. The knowledge engineer and the domain expert interact frequently and domain-specific knowledge is extracted.

(c.) Once the knowledge is acquired, the knowledge engineer decides on the method of representation.

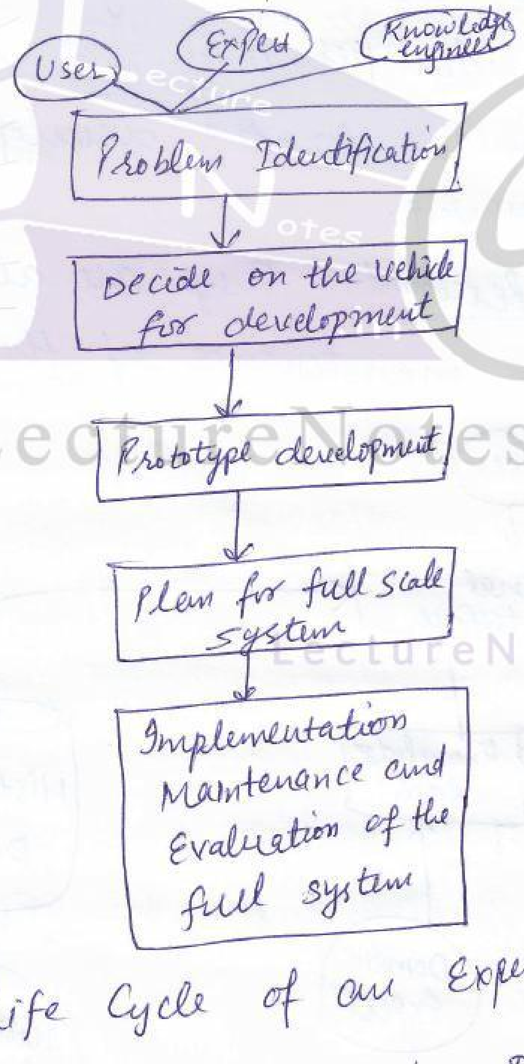
(d.) When the knowledge rep. scheme and the knowledge is available, a prototype is constructed. This prototype undergoes the process of testing for various problems and revision of the prototype takes place. By this process, knowledge of fine granularity emerges and this is effectively coded in the KB.

Stage 4: Planning for a full-Scale System:→ In prototype construction, the area in the problem which can be implemented with relative ease is first chosen. In the full-scale implementation, subsystem development is assigned a group leader and schedules are drawn.

Stage 5: Final Implementation, Maintenance & Evaluation:→ This is the final life cycle stage of an ES. The full scale system developed is implemented at the site. The basic resource

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

requirements at the site are fulfilled and parallel conv-ersion and testing techniques are adopted. The final system undergoes regorous testing & later handed over to the user.
Maintenance of the system implies tuning of the KB because knowledge, the environment and types of problems that arrive are never static. The historical DB has to be maintained and the minor modifications made on inference engine has to be kept track off.
Evaluation is a difficult task for any AI pgms. However, utmost what one can do is to supply a set of problems to the system and a human expert and compare the results.



Life Cycle of an Expert System.

* Personnel Involved in Expert System Development:→ four classes of personnel are involved:

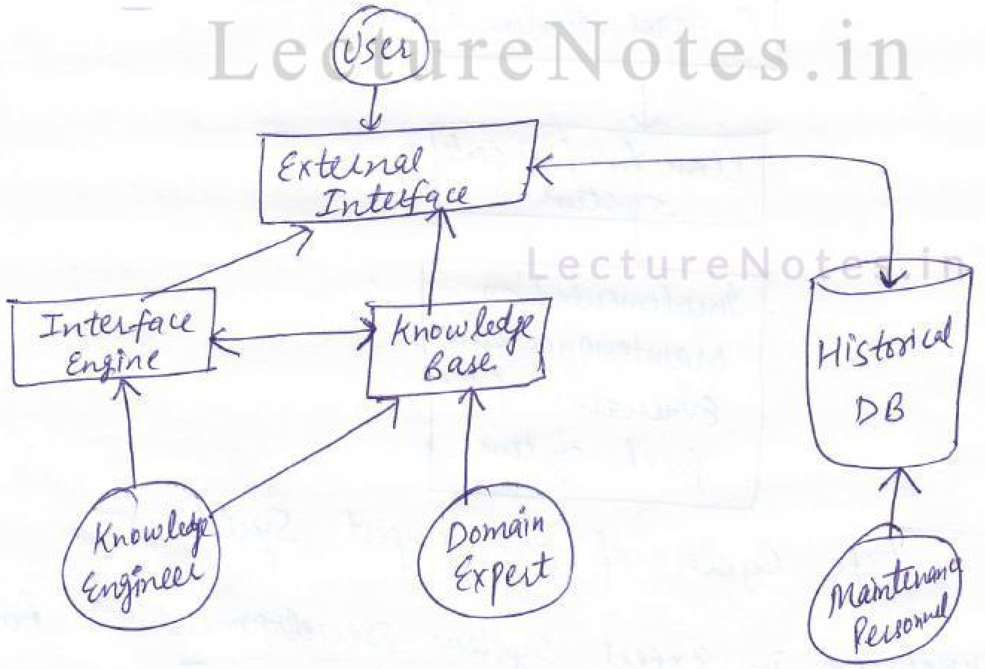
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(1) User :→ A very passive element in the development process.
→ Specifies the overall problem for which ES is needed.
→ Plays a small role in problem identification & comes into picture when the system is fully developed and implemented.

(2) Knowledge Engineer :→ An active player in the development team. Associated from the other right from the identification phase untill the implementation of the full system. Involved in the development of the inference engine, structure of KB, and user interface if a pgmming language is chosen.

(3) Domain Expert :→ Another active player in the development process. Transfers the entire knowledge about the domain of the system. Also identifies and plugs loop-holes in the system. Totally committed for the development of the system right from the beginning.

(4) System Maintenance Personnel :→ They are also passive in nature and are involved in maintenance of the sys. & historical DB



Personnel Involved in ES development

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* Advantages of Using ES :→ Expert systems are being developed for a wide variety of domains. Some of the advantages are:

1.) Expert systems have in their KB vast quantities of domain specific knowledge. ES not only serves as an archive of this knowledge but also plays the role of knowledge disseminator.

(2) In an ES, one can view the entire reasoning process. The reasoning process is not only transparent but also provides answer for questions like "why and How". This is the fascility one calls as the "human window" which enables one to look into its internal working.

(3) In ES, the knowledge engineer after eliciting the knowledge from ~~the~~ expert codes it into the machine understandable form. This fascility has minimized human intervention and has become intimate the domain experts who might not be a very good computer literate.

(4.) Human experts, under stress or in bad moods or when time is critical, either makes default assumptions or forget relevant factors. Since ES do not have these characteristics of stress or moods, they do not make default assumptions or forget relevant factors. Hence one can have greater reliability.

(5.) ESs have the major advantage of increased accessibility which cannot be imagined in case of human experts.

(6.) The time for duplication of an ES is very very short.

(7) An ES plays three major roles: Role of a problem solver, a tutor and an archive. A human expert who is a good problem solver need not be a good tutor.

(8.) ESs are highly advantageous in interdisciplinary domains where

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

multiple experts are needed.

(9.) Another advantage of high reliability is that factors which need more importance are always given high priority. Hence one can expect a consistent reasoning methodology in an ES.

~~[Limitations of ES]~~

Major Application Areas of ES:→ (Janki Raman)

1) Analysis (2) Control (3) Designing (4) Diagonosis (5) Instruction (6) Monitoring (7) Planning (8) Prediction (9) Repair.

* Logic - Basic (Luger) Abductive Inference:→ With logic, pieces of knowledge are explicitly used in reasoning and can be part of the explanations of derived conclusions.
We present several extensions to traditional logic that allow it to support abductive inference.

Montonicity:→ Traditional mathematical logic is monotonic means it begins with a set of axioms, assumed to be true and infers their consequences. If we add new info to this system, it may cause the set of true statements to increase. Adding new knowledge will never make the set of true statements decrease

Non-monotonicity:→ Non-monotonic reasoning addresses the problem of changing beliefs. A non-monotonic reasoning system handles uncertaintly by making the most reasonable assumptions in light of uncertain information.

→ Logics for Nonmonotonic Reasoning:→ Non-montonicity is an important feature of human problem solving and commonsense reasoning.
Conventional reasoning using predicate logic is based on three

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

assumptions:

1.) The predicate descriptions must be sufficient w.r.t. our application domain. That is, all the info necessary to solve the problem must be represented.

2.) The Information base must be consistent, i.e. pieces of knowledge cannot contradict each other.

3) Through the use of inference rules, the known information grows monotonically.

If any of these three assumptions is not satisfied, the Conventional logic-based approach will not work.

Nonmonotonic systems address each of these three issues:

(1) Reasoning systems are often faced with a lack of knowledge about a domain.

(2) Second assumptions required of traditional logic-based systems is that the knowledge supporting reasoning must be consistent.

(3.) If we wish to use logic we must address the problem of how a knowledge base is updated. There are two issues are: First, how can we possibly add knowledge that is based on assumption only.

Second, what can we do when one of our assumptions is later shown to be incorrect.

To address the first issue, we can allow the addition of new KB based on assumptions. This new knowledge is assumed to be correct and so it may be used to infer more new knowledge.

→ In <u>implementing</u> nonmonotonic reasoning, we may extend our logic with the operator 'unless'. 'unless' supports inference based on the belief that its argument is not true.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Example: Suppose we have the following set of Predicate logic sentences:

(matter of belief)

$p(X)$ unless $q(X) \to r(X)$

$p(Z)$

$r(W) \to s(W)$.

First rule means that we may infer $r(X)$ if $p(X)$ is true and we do not believe $q(X)$ to be true. When these conditions are met, we infer $r(X)$ and using $r(X)$, can then infer $S(X)$.

'unless' deals with matters of belief rather than truth.

→ Another non-monotonic logic system is default logic. Default logic employs a new set of inference rules of the form:

$A(Z) \wedge : B(Z) \to C(Z)$.

which is read: If $A(Z)$ is provable and it is consistent with what we know to assume $B(Z)$ then we can conclude $C(Z)$.

* Truth Maintenance Systems: → A TMS may be employed to protect the logical integrity of the conclusions of an inferencing system.

As it is necessary to recompute support for items in a KB whenever beliefs expressed by the clauses of the KB are revised.

One way of viewing this problem is to review the backtracking algorithm. Backtracking is a systematic method for exploring all the alternatives for decision points in search-based problem solving.

The shortcoming of backtracking algo is the way it blindly backs out of dead end states of the space and looks

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

for alternatives from its most recent choices. This approach is called chronological backtracking.

The soln of chronological backtracking pbm is dependency-directed backtracking which has the ability to back-track directly to the point in the space where the problem occurs to make adjustments to the soln at that state.

In order to use dependency-directed backtracking in a reasoning system, we must:

(1.) Associate with the production of each conclusion is its justification. This justification indicates the derivation process for that conclusion. The justification must contain all the facts, rules and assumptions used to produce the conclusion.

(2.) Provide a mechanism that, when given a contradiction along with its justification, finds the set of false assumptions within that justification that led to the contradiction.

(3.) Retract the false assumption.

(4.) Create a mechanism that follows up the retracted assumption and retracts any conclusion that uses within its justification a retracted false assumption.

TMT

- ATMS (Assumption Based Truth Maintenance System)
- JTMS (Justification based Truth Maintenance System)

JTMS explicitly separate the truth maintenance system, the a n/w of propositions and their justifications, from the reasoning system operating in some domain. The result of this split is that the JTMS communicates with the problem solving

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

receiving info about new propositions and justifications & in turn supplying the pbm solver with info about which propositions should be believed based on the current existing justifications.

There are three main operations that are performed by JTMS.

(1) JTMS inspects the n/w of justifications. This inspection can be triggered by queries from the pbm solver such as: Should I believe in proposition p? Why should I believe proposition p?

(2) JTMS modify the dependency n/w, where modifications are driven by info^n supplied by the pbm solver. Modifications include adding new propositions, adding or removing premises, adding contradictions, & justifying the belief in a proposition.

(3) final op^n of JTMS is to update the work. The update op^n recomputes the labels of all propositions in a manner that is consistent with existing justifications

* <u>Abduction: Alternatives to Logic</u> (Luger):→

(1) <u>The Stanford Certainity Factor Algebra</u>:→ This theory is based on a no. of observations. (i) The first is that in traditional prob. theory, the sum of confidence for a relationship and confidence against the same relationship must add to one.

(ii) Another assumption that underpins certainty theory is that the knowledge content of the rules is much more imp. than the algebra for computing the confidences.

⇒ Confidence measures correspond to the informal evaluations that human experts attach to their conclusions, such as " it is

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Probably true", "it is almost certainly true" or "it is highly unlikely". (13)

The Stanford certainty theory makes some simple assumptions for creating confidence measures and has some equally simple rules for combining these confidences as the pgm moves toward its conclusion.

The first assumption is to split "confidence for" from "confidence against" a relationship.

Call MB(H|E) the measure of belief of a hypothesis H given edivence E.

Call MD(H|E) the measure of disbelief of a hypothesis H given evidence E.

Now either:

$$1 > MB(H|E) > 0 \text{ while } MD(H|E) = 0 \text{ or}$$

$$1 > MD(H|E) > 0 \text{ while } MB(H|E) = 0.$$

Once the link b/w measures of belief and disbelief has been established, they may be tied together again, by:

$$CF(H|E) = MB(H|E) - MD(H|E)$$

Now if CF = -1, → uncertaintty

CF = 0, → neutral

CF = 1, → certainty.

→ If there are two premises P_1 & P_2 then

$$CF(P1 \text{ and } P2) = MIN(CF(P1), CF(P2))$$

and

$$CF(P1 \text{ or } P2) = MAX(CF(P1), CF(P2)).$$

The combined CF of the premises, using the above rules, is then multiplied by the CF of the rule itself to get the CF for the conclusions of the rule.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

For eg. → Consider the rule in a KB:

(P1 and P2) or P3 → R1 (0.7) and R2 (0.3).

where P1, P2 & P3 are premises and R1 and R2 are the conclusions of the rule having CFs 0.7 & 0.3 resp.
If the running pgm has produced P1, P2 & P3 with CFs of 0.6, 0.4 and 0.2 resp. then R1 & R2 may be added to the collected case-specific results with CFs 0.28 & 0.12 resp.

Sol: CF (P1 (0.6) and P2 (0.4)) = MIN (0.6, 0.4) = 0.4

CF (P2 (0.4) or P3 (0.2)) = MAX (0.4, 0.2) = 0.4.

The CF for R1 is 0.7, so R1 is added to the set of case-specific knowledge with associated CF of (0.7) x (0.4) = 0.28

The CF for R2 is 0.3 in the rule, so R2 is added to the set of case-specific knowledge with associated CF of (0.3) x (0.4) = 0.12.

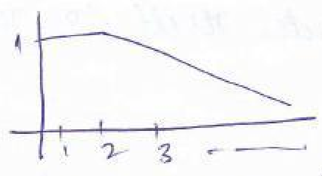
*

(2) Reasoning With fuzzy Sets:

→ The notion of a fuzzy set can be described as follows: Let S be a set and s a member of that set. A fuzzy subset F of S is defined by a membership function mF(s) that measures the "degree" to which s belongs to F.

Example of Fuzzy Set:- Let S be the set of +ve integers and F be the fuzzy subset of S called small integers. Now various integer values can have a possibility distribution defining their "fuzzy membership" in the set of small integers:

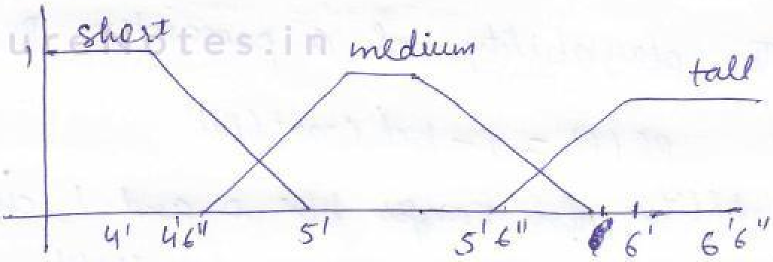
mF(1) = 1.0, mF(2) = 1.0, mF(3) = 0.9, mF(4) = 0.8 --- mF(50) = 0.001 etc



The fuzzy set representation for "small integers".

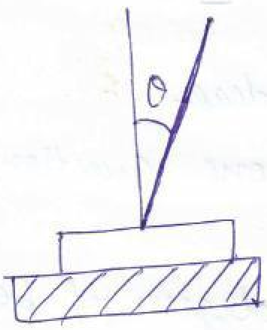
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

→ fuzzy set theory is not concerned with how these possibility distributions are created but rather with the rules for computing the combined possibilities over expressions that contain fuzzy variables.
The laws for the 'or', 'and' and 'not' of these expressions are similar to these just presented for the stanford certainty factor algebra.



A fuzzy set representation for the sets short, medium and tall males.

This fig. offers a set membership function for the concept of short, medium and tall male humans. Any one person can belong to more than one set. for eg.→ a 5'10" male belongs to both the set of medium as well as to the set of tall males.



This fig. presents a pendulum, inverted which we desire to keep in balance and pointing upward. We keep the pendulum in balance by moving the base of the sys. to offset the force of gravity acting on the pendulum.

The advantage of the fuzzy approach to controlling this pendulum system is that an algo. may be established to control the system efficiently & in real time.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(5) <u>The Dempster-Shafer Theory of Evidence</u>:→ It considers set of propositions and assigns to each of them an interval [belief, plausibility] within which the degree of belief for each proposition must lie.

The belief measure, denoted bel, ranges from zero, indicating no evidence of support for a set of propositions, to one, denoting certainty. The plausibility of a proposition p, pl(p) is defined:

$$pl(p) = 1 - bel(not(p))$$

Thus, plausibility also ranges b/w 0 and 1 and reflects how evidence of not(p) relates to the possibility for belief in p. If we have certain evidence of not(p) then bel(not(p)) will be 1 and pl(p) will be 0. The only possible value for bel(p) is also 0.

~~[Suppose we have two competing hypothesis h_1 and h_2]~~

Dempster and Shafer address the problem of measuring certainty by making a fundamental distinction b/w lack of certainty and ignorance.

The Dempster-Shafer Theory is based on two ideas—

(1) The idea of obtaining degrees of belief for one question from subjective probabilities for related questions.

(2) The use of a rule for combining these degrees of belief when they are based on independent items of evidence.

Eg.→ I have subjective probabilities for the reliability of my friend X; The prob. that she is reliable is 0.9 & that she is unreliable is, 0.1.

Suppose X tells me that my computer was broken into. This statement is true if X is reliable, but it is not necessarily false if she is unreliable. So X's testimony

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

alone justifies a degree of belief of 0.9 that my computer was broken into and a 0.0 belief that it was not. Belief of 0.0 does not mean that I am sure that my computer was not broken into, as a probability of 0.0 would. It merely means that X's testimony gives me no reason to believe that my computer was not broken into. The plausibility measure, pl, is:

pl (Computer_broken_into) = 1 - bel (not (Computer_broken_into))

= 1 - 0.0

or 1.0 & my belief measure for X is [0.9, 1.0]. There is still no evidence that my computer was not broken into.

→ Now consider Dempster's rule for combining evidence. Suppose my friend Y also tells me that my computer was broken into. Suppose the prob. that Y is reliable is 0.8 and that he is unreliable is 0.2. I also must suppose that X's and Y's testimonies about my computer are independent of each other. i.e. they have separate reasons for telling me that they think my computer was broken into. The reliability of Y must also be independent of X's reliability.

The prob. that both X & Y are reliable is the product of their reliabilities or 0.72, the prob. that they both are unreliable is the product 0.02.

The prob. that at least one of the two is reliable is 1-0.02 is 0.98. Since they both said that my computer was broken into and there is a prob. of 0.98 that at least one of them is reliable.

Suppose that Y and X disagree on whether my computer was

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

broken into: X says that it was & Y says that it was not. In this case, they cannot both be correct and they cannot both be reliable. Either both are unreliable or only one is reliable.

The prior prob. that only X is reliable is $0.9 \times (1-0.8) = 0.18$, that only Y is reliable is $0.8 \times (1-0.9) = 0.08$ & that neither is reliable is $0.2 \times 0.1 = 0.02$.

Given that atleast one is not reliable, $(0.18 + 0.08 + 0.02) = 0.28$, we can also compute the posterior prob. that only X is reliable as $0.18/0.28 = 0.643$ and my computer was broken into or the posterior prob. that only Y was right, $0.08/0.28 = 0.286$ and my computer was not broken into.

(4) Bayesian Reasoning :→ Bayesian theory supports the calculation of more complex probabilities from previously known results. In simple prob. calculations, we are able to conclude, for eg, how cards might be distributed to a no. of players. If I do not have queen of spades, I can conclude that each of the other players has it with prob. 1/3.

Given that A and B are independent, using the rule:

probability (A & B) = probability (A) * Probability (B)

Definition: Prior Probability :- The prior probability, often called the unconditioned probability, of an event is the probability assigned to an event in the absence of knowledge supporting its occurrence or absence i.e the prob. of the event prior to any evidence. The prior prob. of an event is defined as P(event)

Posterior Probability :→ It is also called the conditional prob. of an event is the prob. of an event given some evidence. It is defined as P(event|evidence).

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

The prior prob. of getting a two or a three on the roll of a fair die is the sum of these options divided by the total no. of possible options or 2/6. (16)

The posterior prob. of a person having disease d with symptom s, $P(d|s)$ is:

$$P(d|s) = |d \cap s| / |s|.$$

where "|"'s surrounding a set indicate the no. of elements in that set. The R.H.S of this equation indicates that the no. of people having both (intersection) the disease d and the symptom s divided by the total no. of people having the sympton s. We extend this eqn taking:

$$P(d|s) = P(d \cap s) / P(s).$$

and have an equivalent relationship for $P(s|d)$ and from that, of $P(d \cap s)$:

$$P(s|d) = P(d \cap s) / P(d).$$

$$P(d \cap s) = P(s|d) * P(d).$$

Substituting, this result in $P(d|s)$ is Bayes Theorm (for one disease and one symptom):

$$P(d|s) = (P(s|d) * P(d)) / P(s).$$

Here is a form of Bayes with multiple Symptoms:

$$P(d|s_1 \& s_2 \& — \& s_n) = (P(d) * P(s_1 \& s_2 — s_n|d)) / P(s_1 \& s_2 \& — s_n)$$

Bayes provides a way of computing the prob. of a hypothesis Hi, following from a particular piece of evidence, given only the probabilities with which the evidence follows from actual causes.

$$P(H_i|E) = \frac{P(E|H_i) \times P(H_i)}{\sum_{k=1}^{n} P(E|H_k) \times P(H_k)}$$

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

where $P(Hi|E)$ is the prob. that Hi is true given evidence E.

$P(Hi)$ is the prob. that Hi is true overall.

$P(E|Hi)$ is the prob. of observing evidence E when Hi is true.

n is the no. of possible hypothesis.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence
MCA- 405

Topics Covered:

1) State Space Search
2) Strategies for State Space Search.
    ↳ Data Driven Search
    ↳ Goal Driven Search.
3.) Search Algorithms ⇒
4.) Brute Force Search
    ↳ Depth first Search Algorithm
    ↳ Breadth first Search.
    ↳ DFS with iterative Deepening.
5.) Heuristic Search
    ↳ Hill climbing
    ↳ Best first Search
    ↳ A* Algorithm
    ↳ AO* Algorithm
6.) Beam Search
7.) Game Playing
    ↳ MINIMAX Algorithm.
    ↳ Modified Minimax with α-β cutoff.
8) Properties of Heuristic Search Algorithm.

Nidhi Kalra
Assistant Professor
MCA Department.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial - Intelligence
MCA - 405.
Unit - IV

(Patterson)

* **Knowledge Acquistion:→** Knowledge acquistion is the process of adding new knowledge to a KB and refining or other-wise improving knowledge that was previously acquired. Therefore we can think of knowledge acquistion as the goal oriented creation and refinement of knowledge.

Acquired knowledge may consist of facts, rules, concepts, procedures, heuristic or other useful information. Sources of this knowledge may include one or more of the following:-

1.) Experts in the domain
2.) Books
3) Technical Papers.
4.) Data Bases
5.) Reports
6.) The environment.

In context of AI, machine learning is considered as the specialized form of knowledge acquistion.

The more important points to be considered under knowledge acquistion are:-

1.) The newly acquired knowledge should be integrated with existing knowledge in some meaningful way.
2.) The knowledge should be accurate, non-redundant, consistent & fairly complete.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Artificial Intelligence

Topic:
Knowledge Acquistion

Contributed By:
Sahil Kumar

Imp/ Types of Learning:→ There are five different learning methods:

1.) Learning By Memorization (Rote Learning)
2.) Learning By Direct Instruction (By being told)
3.) Learning by Analogy.
4.) Learning by Induction
5.) Learning by Deduction.

(1) Learning By Memorization:→ It is the simplest form of learning. It acquires the least amount of knowledge and is accomplished by simply copying the knowledge in the same form that it will be use directly in the KB. We use this memorization learning when we memorize multiplication table & alphabets.

(2) Learning By Direct Instruction:→ This is slightly complex form of learning which requires more inference than rote learning because the knowledge must be transformed into an operational form before being integrated into the KB.

(3) Learning By Analogy (Analogical Learning):→ It is the process of learning a new concept or solution through the use of similar known concepts or solutions.

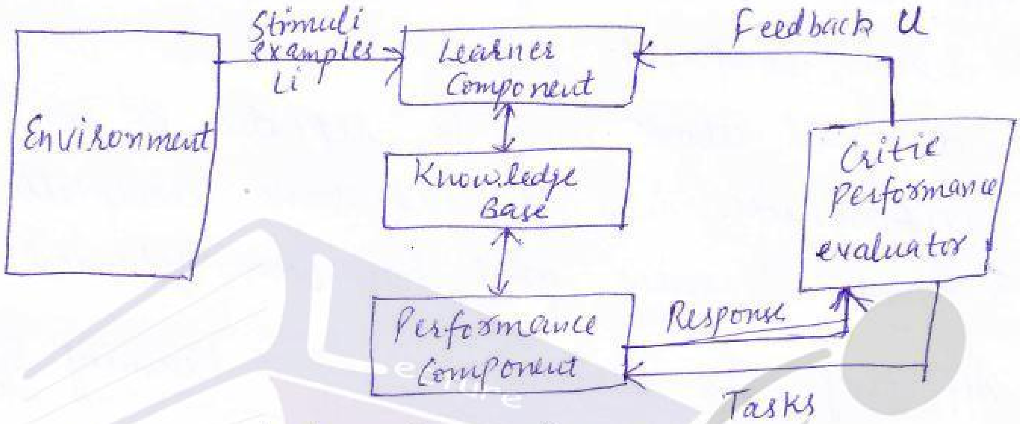
(4) Learning By Induction:→ It is the powerful form of learning. This form requires the use of inductive inference in which we formulate a general concept after examining a no. of instances or examples of a concept. For eg.→ We learn the concepts of appearance, color and taste after experiencing the several examples of objects.

(5) Learning By Deduction:→ This type of learning is

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

using known facts. From the known facts, new facts or relationships are logically derived. For eg → If we say A is Brother of B then we can very well deduce that both A and B are children of same parent.

* General Learning Model :→



(General Learning Model)

Learning requires the new knowledge structure be created from some form of examples. This new knowledge must be assimilated into a KB and can be tested in someway for its utility. The major components of general learning model are:-

(1) Environment :→ It may be regarded as a form of nature which produces random examples or a perfect trainer such as a teacher which provides carefully selected training examples from the learner component.

(2) Learner Component :→ Input to the learner component may be physical stimuli of some type or discriptive, symbolic training examples. (3) KB:- The info conveyed to the learner component is used to create and modify knowledge structures in the KB which is further used by performance comp.

→ The learner comp. acquire knowledge by any of the learning

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(4) Performance Component :→ When given a task, the performance component produces the response describing its actions in performing a task.

(5) Critic Performance Evaluator :→ This component is made for assigning some task to the performance component, getting the response, evaluating it and sending the feedback to the learner component.

The cycle described above may be repeated a no. of times until the performance has reached some acceptable level or until a known learning goal has been achieved.

* Factors Affecting the performance of a Learning System :→

1. Background knowledge
2. Feedback
3. Training Scenario
4. Representation Scheme

→ Learning Algorithms ⇒ Resolution Performance

(1) Background Knowledge :→ It is used to constraint the search space or exercise the control operation which limits the search process. To make the learning more efficient, it is necessary to constraint the background knowledge.

(2) Feedback :→ It is essential to the learner component because without it, the learner component would never know that if the knowledge structure in KB were improving or if they were adequate for the performance of given task. The feedback may be simple yes or no,

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

describing why a particular action was good or bad.

(3) Training Scenario:→ The type of training used in system can have strong effect on performance of learning system. The training material should consist of carefully selected and ordered examples.

(4) Representation Scheme:→ The kind of representation scheme used in learning model placed a vital role. The representation scheme should be simple, easy to understand & illustrative.

(5) The Learning Algorithm:→ These are themselves determined to a large extent, how successful a learning system will be. The algorithms control the search to find and build the knowledge structures. The algorithm should be able to extract much of the useful information from training examples and background knowledge.

* Performance Measures:→ (Janki Raman)

1.) Generality:→ 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.

(2) Effeciency:→ The effeciency of a method is a measure of the average time required to construct the target knowledge structures from some specified initial structures. For eg.→ The relative effeciency of a method can be defined as the ratio of the time required for the given method to ... target structure

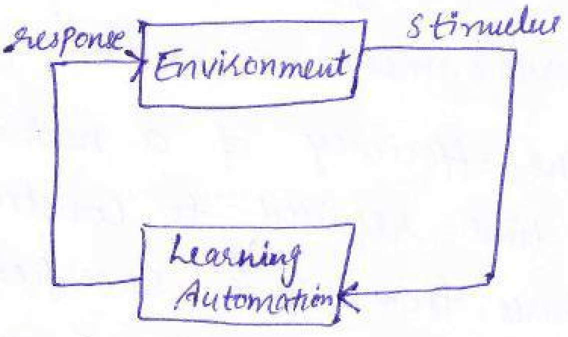
Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(3) 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.

(4) 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.

(5) 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.

* Learning Automata (Patterson):→ Learning automata systems are finite state adaptive systems which interact iteratively with a general environment.



Learning Automation model.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

learning automata has two components: An automaton (learner) and an environment.

The learning cycle begins with an input to the learning automata system from the environment. This input elicits one of the ~~learning automata system~~ finite no. of possible responses from the automaton. The env. receives and evaluates the response and then provides some form of feedback to the automaton in return.

This feedback is used by the automation to alter its stimulus-response mapping structure to improve its behavior.

Learning automata have been generalized. One such generalization is Collective learning Automata (CLA). CLAs are standard learning automata systems except that feedback is not provided to the automaton after each response. In this case, several collective stimulus-response actions occur before feedback is passed to the automaton. This type of learning resembles that of human beings in that we usually perform a gp. of primitive actions before receiving feedback on the performance of such actions, such as solving a complete problem on a test.

* <u>Genetic Algorithms</u> :→ Genetic algorithm learning methods are based on models of natural adaptation and evolution.

Genetic algorithm systems start with a fixed size population of data structures which are used to perform some given

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

some no. of times, the structures are rated on their performance, and a new generation of data structures is then created. The new generation is created by mating the higher performing structures to produce offspring. These offspring and their parents are then retained for the next generation while the poorer performing structures are discarded.

Genetic Algorithm

After multiple attempts at executing the tasks, each of the participating structures would be rated and tagged with a utility value i.e. commensurate with its performance. The next population would then be generated using the higher performing structures as parents and the process would be repeated with the newly produced generation.

Genetic opns:- (1) Mating b/w two strings is accomplished with ... randomly selects a bit position

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

in the 8-bit string and concatnates the head of one Parent to the tail of the second parent to produce the offspring. Suppose the two parents are designated as xxxxxxxx and yyyyyyyy resp and suppose the third bit position has been selected at the crossover point (xxx:xxxxx). (5)

After the crossover opⁿ is applied, two offsprings are then generated, namely xxxyyyyy and yyyxxxxx. Such offspring and their parents are then used to make up the next generation of structures.

(2) Inversion:→ Inversion is a transformation applied to a single string. A bit position is selected at random, and when applied to a structure, the inversion operation concatenates the tail of the string to the head of the same string. So, if sixth position were selected $(x_1 x_2 x_3 x_4 x_5 x_6 : x_7 x_8)$, the inverted string would be $x_7 x_8 x_1 x_2 x_3 x_4 x_5 x_6$.

(3) Mutation:→ Mutation is used to insure that all locations of the rule space are reachable, that every potential rule in the rule space is available for evaluation. This ensures that the selection process does not caught in a local minimum.

The new generation is created from the old population by first selecting a fraction of the members having the utility values. from these, offspring are obtained by appⁿ of appropriate genetic operators.

The three operators, crossover, inversion and mutation, ... to give new offspring

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

move sequences. Each offspring inherits a utility value from one of the parents. Population members having low utility values are discarded to keep the population size fixed.

The whole process is repeated until the genetic system has learned all the optimal moves.

* **Intelligent Editors:** → An intelligent editor acts as an interface between a domain expert and an expert system. They permit a domain expert to interact directly with the system without the need for an intermediary to code the knowledge.

Acquisition using an Intelligent Editor.

The expert carries on a dialog with the editor in a restricted subset of English which includes a domain-specific vocabulary. The editor has direct access to the knowledge in the expert system and knows the structure of that knowledge.
Through the editor, an expert can create, modify, and delete rules without a knowledge of the internal structure of the rules.

The editor assists the expert in building and

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

specific topic, and reviewing and modifying the rules if necessary.

Through the editor, the expert system can query the expert system for conclusions when given certain facts. If the expert is unhappy with the results, a trace can be obtained of the steps followed in the inference process. When faulty or deficit knowledge is found, the problem can then be corrected.

* Learning By Induction:- (Patterson).

Some Definitions:

(1) object:- Any entity, physical or abstract, which can be described by a set of attribute values and attribute relations is an object. We will refer to an object either by its name O_i or by an appropriate representation such as the vector of attribute values $x = (x_1, x_2 — x_n)$.

(2) class:- Given some universe of objects U, a class is a subset of U. for eg → Given the universe of four-legged animals, one class is the subset horses.

(3) Concept:→ This is a description of some class or rule which partitions the universe of objects U into two sets, the set of objects that satisfy the rule and those that do not. Thus, the concept of horse is a description is a description or rule which asserts

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

the set of all horses and excludes all nonhorses.

(4.) Hypothesis:-> A Hypothesis H is an assertion about some objects in the universe. It is a candidate or tentative concept which partitions the universe of objects. One such hypothesis related to the concept of horse is the class of four-legged animals with a tail. This is a candidate for the concept horse.

(5.) Target Concept:-> The target is one concept which correctly classifies all objects in the universe.

(6) Positive Instances:-> These are example objects which belong to the target concept.

(7) Negative Instances:-> These are examples opposite to the target concept.

(8.) Consistent classification Rule:-> This is a rule that is true for all positive instances and false for all negative instances.

(9.) Induction:-> Induction is the process of class formation. Here we are more interested in the formation of classes which are goal-oriented, therefore we will define induction as purposeful class formation.

(10.) Selective Induction:-> In this form of induction class descriptions are formed using only the attributes ... positive instances.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

11. Constructive induction:→ This form of induction creates new descriptors not found in any of the instances.

12.) Expedient Induction:→ This is the application of efficient, efficacious inductive learning methods which have some scope, methods which span more than a single domain. The combined performance in efficiency, efficacy and scope has been termed the inductive power of a system.

Generalization and Specialization :→ Concept learning requires that a guess or estimate of a larger class, the target concept, be made after having observed only some fraction of the objects belonging to that class. This is essentially a process of generalization, of formulating a description or a rule for a larger class but one which is still consistent with the observed positive examples.

For eg → Given, three positive instances of objects:

(blue cube rigid large)

(small flexible blue cube)

(rigid small cube blue).

is a proper generalization which implies the three instances is blue cube. Each of the instances satisfies the general description.

Specialization is the opposite of generalization. To specialize

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

blue cubes is required such as small cubes or flexible blue cube or any of the original instances given above.

Specialization may be required if the learning algorithm over-generalizes in its search for the target concept. An over-generalized hypothesis is inconsistent since it will include some negative instances in addition to the positive ones.

* Generalization Rules :→ Since specialization rules are essentially the opposite of rules for generalization, to specialize a description, one could change variables to constants, add a conjuct or remove a disjunct from a description & so forth.

These methods are useful tools for constructing knowledge structures. They give us methods with which to formulate and express inductive hypothesis. Unfortunately, they do not give us much guidance on how to select hypothesis effeciently. For this, we need methods which more directly limit the number of hypothesis which must be considered.

Types of Generalization on basis of rules :→

① Selective Generalization :→ Selective generalization rules ... (attributes

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

and relations) that appear in the instances.

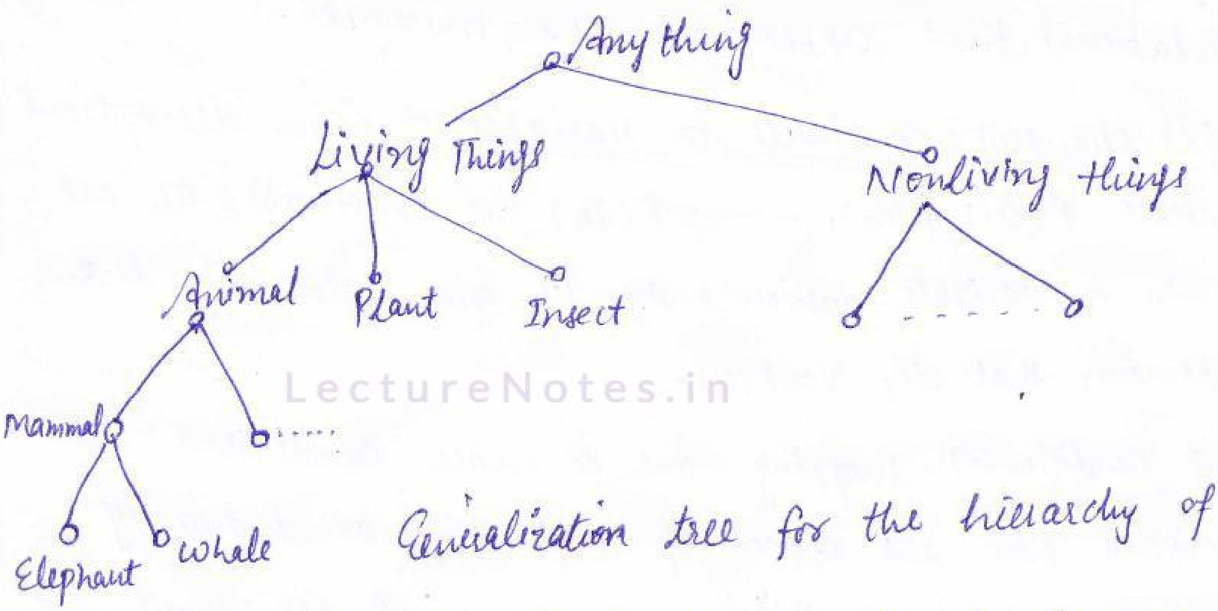
Rules:- (i) Changing constants to variables:→ Given descriptions or predicate $P(a_1), P(a_2), \ldots P(a_n)$ the constants a_i are changed to a variable which may be any value in the given domain, that is, $\forall x P(x)$.

(ii) Dropping Conditions:→ Dropping one or more conditions in a description has the effect of expanding or increasing the size of the set. for eg.→ The set of all small red spheres is less general than the set of small spheres. Another way of stating this rule, when the conjuctive description is given is that a generalization results when one or more of the conjucts is dropped.

(iii) Adding an Alternative:→ This is similar to the dropping condition rule. Adding a disjunctive term generalizes the resulting description by adding an alternative to the possible objects. For eg.→ Transforming red sphere to (red sphere) V (green pyramid) expands the class of red spheres to the class of red spheres or green pyramids. Note that the internal disjunction could also be used to generalize An internal disjunction is one which appears inside the parentheses such as (red V green sphere).

(iv) Climbing a Generalization Tree:→ When the classes of ... can be represented as a tree hierarchy.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar



Generalization tree for the hierarchy of All Things.

Generalization is accomplished by simply climbing the tree to a node which is higher and therefore gives a more general description. For eg, moving up the tree one level from elephant we obtain the more general class description of mammal. A greater generalization would be the class of all animals.

(v) <u>closing an interval</u>:→ When the domain of a descriptor is ordered ($d_1 < d_2 < \cdots < d_k$) and a few values lie in a small interval, a more restricted form of generalization than changing constants to variables can be performed by generalizing the values to a closed interval. Thus, if two or more instances with values $D = d_i$ and $D = d_j$ where $d_i < d_j$ have been observed, the generalization $D = [d_i \cdots d_j]$ can be made; that is, D can be any value in the interval d_i to d_j.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(2) Constructive Generalization :→ As Selective generalization (9) rules build descriptions using only the descriptors that appear in the instances, whereas constructive generalization rules do not.

(i) Generating Chain Properties :→ If an order exists among a set of objects, they may be described by their ordinal position such as Ist, 2nd — nth. For eg.→ suppose the relations for a four story building are given as:

above (f_2, f_1) & above (f_3, f_2) & above (f_4, f_3)

then a Constructive generalization is

most_above (f_4) & least_above (f_1).

The most_above, least_above relations are created. They did not occur in the original descriptors.

* Inductive Bias :→ There are two general types of bias:

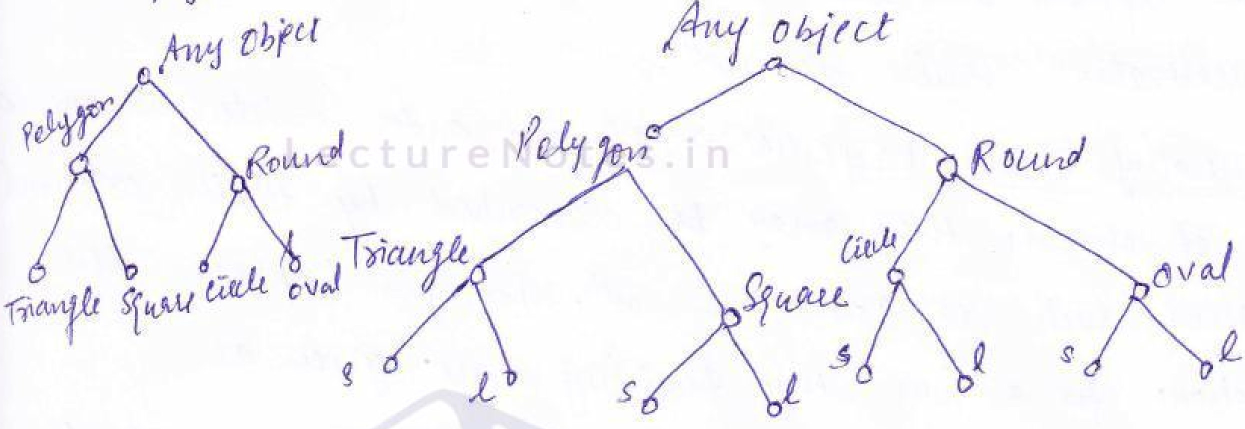
1) Restricting hypothesis from the hypothesis space.
2) The use of a preferential ordering among the hypothesis or use of an informed selection scheme.

Each of these methods can be implemented in different ways. For eg., the size of the hypothesis space can be limited through the use of syntactic constraints in a rep. language which permits attribute descriptions only.

Representations based on more abstract descriptions

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

will often limit the size of the space as well.
Eg- of which limits the no of hypotheses is illustrated in fig.



Tree Representation for object descriptions (s=small, l=large)

The tree representation on the right contains more information and therefore, will permit a large no. of object descriptions to be created than with the tree on the left.
On the other hand, if one is only interested in learning general descriptions of geometrical objects without regard to details of size, the tree on the left will be a superior choice since the smaller tree will result in less search.

Methods based on the second general type of bias limit the search through preferential hypothesis selection. One way this can be achieved is through the use of heuristic evaluation functions. If it is known that a target concept should not contain some object or class of objects, all hypothesis which contain these objects can be eliminated from consideration.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

Bias can be strong or weak, correct or incorrect. A strong bias is one which focuses on a relatively small no. of hypothesis.

A weak bias does not. A correct bias is one which ~~focuses on a relatively small no of hypothesis~~ allows the learner to consider the target concept, whereas an incorrect bias does not.

A learner's task is simplified when the bias is both strong and correct. Bias can also can implemented in a program as either static or dynamic. When dynamic bias is employed, it is shifted automatically by the program to improve the learner's performance.

* Example of an Inductive Learner:→ There are many concepts which simply cannot be described well in conjuctive terms only. ~~one~~ for eg.→ The concept of uncle. Since an uncle can be either the brother of the father or the brother of the mother of a child. To state it any other way is cumbersome.

For eg.→ Here is a system that learns descrip-tions which are essentially in disjunctive normal form. The system can learn either concept descriptions from attribute values or structural descriptions of objects. The training set we use here consists of a sequence

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

target concept. Each instance is presented to the learner as an unordered list of attributes together with a label which specifies whether or not the instance is positive or negative.

For this example, we require our learner to learn the disjunctive concept "something that is either a tall flower or a yellow object". One such instance of this concept is represented as (short skinny yellow flower +), whereas a negative instance is (brown fat tall weed -).

Given a no. of positive and negative training instances such as these, the learner builds frame-like structures with groups of slots we will call clusters as:

Concept-name: (tall flower or yellow object)
    positive part:
        cluster: description: ———
                 examples: ———
        cluster: description: ———
                 examples: ———
        ⋮
    negative part:
        examples: ———

The target concept is given in the concept name. The actual description

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* **PROLOG:→** The name PROLOG is taken from the phrase "programming and logic." This language was developed in 1973 by Alan Collmeraue and P. Roussel in university of Marseilles in France.

PROLOG is an AI programming language that supports formal symbolic reasoning making theortically possible intelligent computers that can understand human language and perform routine tasks without the need for a procedure defined by a human.

* **Features of PROLOG that it makes it suitable for AI Programming:→**

(1.) The syntax and semantics of PROLOG are very close to formal logic.

(2.) This language has high productivity and ease of program maintenance.

(3) PROLOG's free data structure is amenable (willing to accept) to complex data structures.

~~(4.)~~ ~~PROLOG~~ is an object oriented language. It uses no procedures and essentially no ~~program~~.

(5.) PROLOG uses only data about objects and their relationships.

(6.) With PROLOG, the user defines a goal and the computer must find both the procedure & solution.

... programmers.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(4.) The clauses of PROLOG have a procedural and declarative meaning. Because of this, understanding of language is easier.

(7.) In PROLOG, each clause can be executed separately as though it is a separate program.

* **Parts of PROLOG Program:** → PROLOG program is divided into three sections:-

(1.) **Domain Section:** → This section defines the type of each object being used in a PROLOG program. The six basic object types allowed in PROLOG are:-

(i) char	Single character enclosed between single quotation mark.
(ii) Integer	Integer from -32768 to 32767.
(iii) Real	Floating point no. ($1e^{-307}$ to $1e^{308}$).
(iv) String	Character sequence enclosed b/w double quotation mark.
(v) Symbol	Character sequence of letters, numbers and underscore with the 1st character of lowercase letter.
(vi) File	Symbolic Filename.

(2.) **Clause Section:** → It contains the clauses and consist of facts and rules. A fact is used to indicate a simple data relationship b/w elements (objects). Each clause ends with a period (.) symbol.

For ex. → is(Ball, Round).

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(3) Predicate Section:→ The relations used in the clauses of clause section are defined in the predicate section. Each relation in each clause must have a predicate definition in predicate section. The only exceptions are the built-in predicates.

The predicate definition in the predicate section never ends with a (·) period symbol. A predicate can have any no. of arguments.

is (object, shape)

* Basis of PROLOG Language:→

(i) Horn clause:→ In a horn clause, one condition is followed by zero or more conditions. It is represented as:

Conclusion:-

Condition-1,
Condition-2,
Condition-3,
⋮
Condition-n.

i.e. the conclusion is true iff Condition-1 is true and Condition-2 is true & Condition-3 is true and so on until Condition-n is true.

In other words, a horn clause consists of a set of statements joined by logical ANDs.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(ii) Robinson's Resolution Rule:→ The principle of resolution states that two clauses can be resolved with one another if one of them contains a positive literal and other contains a corresponding negative literal with same predicate symbols and the same no. of arguments.

Eg.→ Consider the clauses:

$$\neg X(a) \vee Y(p,q)$$

$$\neg Y(p,q) \vee T(r,s)$$

These two clauses can be resolved to give

$$T(r,s) \vee \neg X(a).$$

* PROLOG Variables:→ (1) Free and (2) Bound Variables
(3) Anonymous Variables.

. Bound variables :→ These are those variables which have some value at any instance of time.

Free Variables:→ Free variables are those that does not have any value at any instance of time.

Anonymous Variables:→ It is a special variable that instructs the system to ignore the value of an argument. It unifies with anything but does not print. Its symbol is underscore (_). It is also known by the name of "for all" variable. So the query

likes(ram, _)

will return the value true because the system can match from the DB the predicate name and the arguments

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

* **The Compound Goals or Queries:→** Sometimes we need to combine two query conditions. In that case, concept of compound query is used which combines two atomic queries using AND (,) symbol.

For eg → Symptom (Disease, headche),
Symptom (Disease, Sneezing).

Conclusion:- Disease = cold.

* **Adding Comments:→** /* ------ */.

* **Operators:→**

(1.) **Arithmetic Operators:→**

Symbol	Meaning
+	Addition
−	Subtraction
*	Multiplication
/	Division
mod	modulo
div	

(2.) **Comparison Operators:→** >, <, >=, <=, =, <> (Not equal)

(3.) **Logical Connectives:→** Not () → Negation
, → Conjuction
; → Disjunction
:- → Implication

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(13)

* **Equal operator:** → The equal sign '=' is an operator in PROLOG, whose values to the left and right are considered as the operands.

The equal (=) operator functions in two ways:-

1. If operand is a variable and it is not bound then equal operator functions like an assignment operator for binding the variable.
2. otherwise, the equal operator functions as the comparison operator and a test is performed to give the result as true or false.

For eg. →

```
go:-
    X = 4+3
    write (X)
output: 7
```

```
go:-
    X = 4+3
    write (X)
    X= 4-3   (Comparison)
    write (X)
output: 7
```

* **Queries To a Database:** → Once a DB has been created, one can make queries to it. A simple query consists of a predicate name and its arguments.

For eg.→, for the 'likes' database created, the query

likes (ram, cars)

would return the value true.

If the query has a variable, then the system will try to evaluate those predicates for which the variable is True. As, the variables will start with an uppercase letter.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

What = aircrafts
What = car

How does Prolog solve a Query :→

PROLOG tries to match the arguments of the query with the facts in the DB. If the unification succeeds, the variable is said to be instantiated.

It is also possible that one can have variables for all the arguments. The query

likes (Who, What)

Who = Kumar, What = toffees
Who = ram, What = aircrafts.
Who = mani, What = toffees.

The sequence adopted for this is the same as the sequence in the DB.

* **Controlling Execution in PROLOG :→** There are three predicates which are use to control the execution.

These predicates are:

1.) fail Predicate
2.) cut Predicate
3.) Not Predicate

(1) **fail Predicate :→** The fail predicate will make a clause to fail during execution. In order to force backtracking, this predicate is useful.

Eg.→ clause1 :-
person (Name, Designation),
write (Name),
... ignation),

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

person (raman, researcher).
person (kumar, manager).
person (ravi, accountant).
person (selvan, partner).

When this program is executed, the system will bind raman to Name, researcher to Designation and print them. This clause is deliberately failed using fail predicate. This forces backtracking and the system instantiates another value to the variable. Thus, the system will print all the Names and Designation and will fail because of the fail predicate.

In order to make the clause succeed, all that has to be done is to make the clause true. This is done by adding the clause without any conditions to it. for eg. →

clause 1:-

```
person (Name, Designation),
write (Name),
write (Designation),
fail.
```

clause1. /* This clause will make clause 1 to succeed */

```
person (raman, researcher).
person (kumar, manager).
person (ravi, accountant).
person (selvan, partner).
```

Here the variables in the clause lose their bindings every

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

(Q) **Cut Predicate:** → The cut is a built-in predicate that instructs the intepreter not to backtrack beyond the point at which it occurs. It is used to prune the search space. It's symbol is "!". (15)

For eg. →

```
state(tamilnadu).
state(kerala).
state(andhra_Pradesh).

state(S):-
    write("Do you belong to"),
    write(S),
    write("?"),
    readln(Reply),
    Reply = "yes",
    !
    write("so you belong to"),
    write(S).
```

Output:- when a person belonging to kerala will answer.

Do you belong to tamilnadu?
no
Do you belong to kerala?
yes
So you belong to kerala.

The system reads the user's variable in Reply. If it is "no", then Reply subgoal fails & system backtracks to get ... variable for S. If Reply is "yes", then the system

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

allows the program to proceed beyond the cut. The cut will see it to that the query will end after the first "yes" answer & will not permit to backtrack. This is the reason ~~why~~ system will not ask about another pardesh.

⑧ Not Predicate :→ This predicate allows the programmer to encode negative information. The NOT predicate succeeds if unification its arguments fails. The operation of the NOT predicate will be represented by NOT (True predicate).

For eg.→ The Predicate Have-a-gun (Ram) will be return by NOT predicate as NOT (Have-a-gun (Ram)). Given the above statement as Goal, the PROLOG unification mechanism proceeds first to prove the truth of the statement Have-a-gun (Ram). If a fact Have-a-gun (Ram) does not exist in DB, the unifier fails and consequently NOT (Have-a-gun (Ram)) succeeds.

* Recursion in PROLOG :→ If a function during execution calls itself again, then such a function is said to be recursive in nature.

Eg.→ Consider the program that finds the "ancestor".

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar

```
ancestor(A,B) :-                    /* clause 1 */
        parent(A,B).

ancestor(A,B) :-
        parent(C,B),
        ancestor(A,C)               /* clause 2 */
```

Clause 1 states A is an ancestor of B, when A is parent of B.

Clause 2 states A is an ancestor of B, when C is a parent of B and A is an ancestor of C.

To verify this, consider the DB

```
parent(person_1, person_2).
parent(person_1, person_3).
parent(person_3, person_4).
```

The query ancestor(Person_1, Whom) will have the answer

```
Whom = person_2.
Whom = person_3.
Whom = person_4.
```

Any recursive procedure has to have:

(1) A non-recursive clause to indicate when the recursion has to stop.

(2) A recursive rule.

Downloaded from www.LectureNotes.in by Deepak Garg of Swami devi dyal institute of engineering & technology with registered phone no 9999234890 and email deepgargak2010@gmail.com. Contributed by Sahil Kumar