UNIT 1: INTRODUCTION

1. ARTIFICIAL INTELLIGENCE :

AI is the study of intelligent behavior and how to make machines do things at which humans are doing better. According to the father of Artificial Intelligence, John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs". AI means to create computer programs that perform tasks as well as or better than humans. In simple terms, AI is human intelligence demonstrated by machines.

Definitions:

There are various definitions of AI according to different authors. These definitions are categorized into the following 4 categories.

1. Systems that think like humans(focus on reasoning and human framework).E.g. : GPS (General Problem Solver)
2. Systems that think rationally(focus on reasoning and a general concept of intelligence).Eg:Rational agents
3. Systems that act like humans(focus on behavior and human framework).Eg:Eliza
4. Systems that act rationally(focus on behavior and a general concept of intelligence) Eg : Heuristic systems

Goals:

1. To create expert system: The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.

2. To implement human intelligence in machines: Creating systems that understand, think, learn, and behave like humans.

Computational intelligence(CI)& Artificial Intelligence):

AI	CI
AI is the study of intelligent behavior and how to make machines do things at which humans are doing better.	CI is the study of adaptive mechanisms to enable or facilitate intelligent behaviour in complex and changing environments.
Hard computing Technique	Soft Computing Technique
Follows binary logic	Follows Fuzzy logic
Based on mathematical models	Nature inspired model
Not very effective	Can work in exact and incomplete data
Deterministic results	Probabilistic result
Common applications of AI are speech recognition, handwriting recognition, optical character recognition, big data solutions	The real life applications of CI include intelligent household appliances, medical diagnosis, banking and consumer electronics and so on

Example:

- Google Assistant
- Alexa
- Siri
- Social Media Feeds
- Netflix, YouTube (Recommendations)
- Chatbot

a. Problems :

AI problems are hypothesized to include computer vision, natural language understanding, and dealing with unexpected circumstances while solving any real-world problem. Currently, AI-complete problems cannot be solved with modern computer technology alone, but would also require human computation.

AI seeks to understand the computations required from intelligent behavior and to produce computer systems that exhibit intelligence. Aspects of intelligence studied by AI include perception, communicational using human languages, reasoning, planning, learning and memory.

The AI problems are:

1. **Game playing:** Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game. To solve these problems we must explore a large number of solutions quickly and choose the Best One.

2. **Theorem proving:** Logic Theorist: It proved mathematical theorems. It actually proved several theorems from Classical Math Textbooks.

3. **Commonsense Reasoning:** It includes reasoning about physical objects and their relationships to each, as well as reasoning about actions and their consequences.

   Example: GPS ,which they applied to several commonsense tasks as well as to the problem of performing symbolic manipulations of logical expressions.Again,no attempt was made to create a program with a large amount of knowledge about a particular problem domain .Only quite simple tasks were selected.

4. **Perception (vision and speech):** Perception is the process of acquiring, interpreting, selecting, and organizing sensory information. In AI, the study on perception is mostly focused on the reproduction of human perception, especially on the perception of aural and visual signals.

   Speech recognition is the front-end of a system that can perceive and understand spoken language, as used in voice command interface and speech-to-speech translation. Example: Google Assistant.

   A subset of mainstream artificial intelligence that deals with the **science** of making computers or machines visually enabled, i.e., they can analyze and understand an image.Example: self-driving cars.

5. **Natural language understanding:** NLU is Artificial Intelligence that uses computer software to interpret text and any type of unstructured data. NLU can digest a text, translate it into computer language and produce an output in a language that humans can understand. Natural language processing (NLP) describes the interaction between human language and computers. Example: Siri, Alexa, or Google Assistant.

There are three categories of tasks:

1. **Mundane Tasks:**
   - Perception
     - Vision
     - Speech
   - Natural Language
     - Understanding
     - Generation
     - Translation
   - Common-sense Reasoning
   - Robot Control
2. **Formal Tasks:**
   - Games
     - Chess
     - Backgammon
     - Checkers
     - GO
   - Mathematics
     - Geometry
     - Logic
     - Integral Calculus

- proving Properties of Programs

3. **Expert Tasks:**

- Engineering
  - Design
  - Fault finding
  - Manufacturing Planning
- Scientific Analysis
- Medical Diagnosis
- Financial Analysis

The AI Problems And Solutions Techniques Requires:

1. **Understanding Assumptions About Intelligence:** The underlying assumption is Physical Symbol system hypothesis. Computers and human minds are examples of physical symbol systems. A symbol is a meaningful pattern that can be manipulated. Examples of symbols are written words, sentences, gestures, marks on paper, or sequences of bits. A symbol system creates copies, modifies, and destroys symbols. A physical symbol system has the necessary and sufficient means for general intelligent action. Physical Symbol system hypothesis states that processing structures of symbols is sufficient, in principle, to produce artificial intelligence in a digital computer and that; moreover, human intelligence is the result of the same type of symbolic manipulations.

2. **Techniques For Solving Ai Problems. :**There are three important AI techniques:

1. **Search**: Provides a way of solving problems for which no direct approach is available. It also provides a framework into which any direct techniques that are available can be embedded.
2. **Use of knowledge:** Provides a way of solving complex problems by exploiting the structure of the objects that are involved.
3. **Abstraction:** Provides a way of separating important features and variations from many unimportant ones that would otherwise overwhelm any process.

3. Level To Model Human Intelligence . :

1. Model the program that computer could easily solve.
2. Model human performance.

4. What Is Criteria For Success.

1. What is the goal of program?

Characteristics of AI Problems:

1. AI problem have large number of combination of solution. In case of chess game, the number of possible positions has been estimated about 35^{100}.
2. AI program manipulate large number of symbolic information.
3. AI problem uses Heuristic search to cope with combination explosion of solution.
4. Ability to learn.
5. AI problems can be solved with or without using AI technique.

b. Scope and applications

Scope of AI:

The ultimate goal of artificial intelligence is to create computer programs that can solve problems and achieve goals like humans would. There is scope in developing machines in robotics, computer vision, language detection machine, game playing, expert systems, speech recognition machine and much more.

1. **AI in Science and Research:**

   Artificial Intelligence can handle large quantities of data and processes it quicker than human minds. This makes it perfect for research where the sources contain high data volumes.

   Eg: Eve (AI based Robot)

2. **AI in Cyber Security:**

   As organizations are transferring their data to IT networks and cloud, the threat of hackers is becoming more significant. To keep their data and resources secure, organizations are making massive investments in cyber security.

   Example1 (Cognitive AI): It detects and analyses threats, while also providing insights to the analysts for making better-informed decisions.

   Example 2(IBM Resilient): Many institutions are using AI-based solutions to automate the repetitive processes present in cyber security. IBM Resilient is an agnostic and open platform that gives infrastructure and hub for managing security responses.

Another field is fraud detection. AI can help in detecting frauds and help organizations and people in avoiding scams. For example, Recurrent Neural Networks are capable of detecting fraud in their early stages.

3. **AI in Data Analysis:**

AI can help data analysts with handling and processing large datasets. Example: Google Analytics

Another example of AI applications in this sector is predicting outcomes from data. Such systems use the analytics data to predict results.

AI systems can handle tons of data and process it much faster than humans. So, they can take customer data and make more accurate predictions of customer behavior, preferences, and other required factors. Helixa AI is a great example of such an AI application.

4. **AI in Transport:**

An autopilot system controls the trajectory of a plane. Autopilot helps the human operator and assists them in heading in the right direction.

Self-driving cars will be free from human errors, which account for 90% of traffic accidents. Many companies, including Tesla and Uber, are developing these vehicles.

5. **AI in Home:**

Amazon Echo and Google Home are popular smart home devices that let you perform various tasks with just voice commands.

Smart assistants are also present in mobile phones. Apple's Siri and Google Assistant are great examples of this sort. Microsoft also has a smart assistant, which is called Cortana.

Applications of AI:

AI has applications in all fields of human study, such as finance and economics, environmental engineering, chemistry, computer science, and so on. The various applications are:

1. **Game Playing:** AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
2. **Speech Recognition:** Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
3. **Understanding Natural Language:** The computer can now understand natural languages and hence human can now interact using natural spoken languages. The computer has to be provided with an understanding of the domain the text is about, and this is presently possible only for very limited domains.
4. **Computer Vision:** The computer vision leads the computer to understand the signals and act accordingly. There are some systems as: - Face detection system installed at airport, Medical diagnosis .
5. **Intelligent Robots :** Robots have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence.

Some of the other applications of AI are:

1. Perception

   - ❖ Machine vision

- ❖ Speech understanding
- ❖ Touch (tactile or haptic) sensation

2. Robotics
3. Natural Language Processing
   - ❖ Natural Language Understanding
   - ❖ Speech Understanding
   - ❖ Language Generation
   - ❖ Machine Translation
4. Planning
5. Expert Systems
6. Machine Learning
7. Theorem Proving
8. Symbolic Mathematics
9. Game Playing

2. PROBLEM SPACE AND SEARCH :

To build a system to solve a particular problem, we need to do 4 things:

a. Define the problem precisely. This definition must include precise specifications of what the initial situations will be as well as what final situations constitute acceptable solutions to the problem.

b. Analyze the problem.

c. Isolate and represent the task knowledge that is necessary to solve the problem.

d. Choose the best problem solving techniques and apply it to the particular problem.

Problem definition:

A problem is defined by its 'elements' and their 'relations'. A 'problem' is characterized by a set of goals, a set of objects, and a set of operations.

To provide a formal description of a problem, we need to do the following:

1. Define a state space that contains all the possible configurations of the relevant objects, including some impossible ones.

2. Specify one or more states that describe possible situations, from which the problem solving process may start. These states are called initial states.

3. Specify one or more states that would be acceptable solution to the problem. These states are called goal states.

Problem Space:

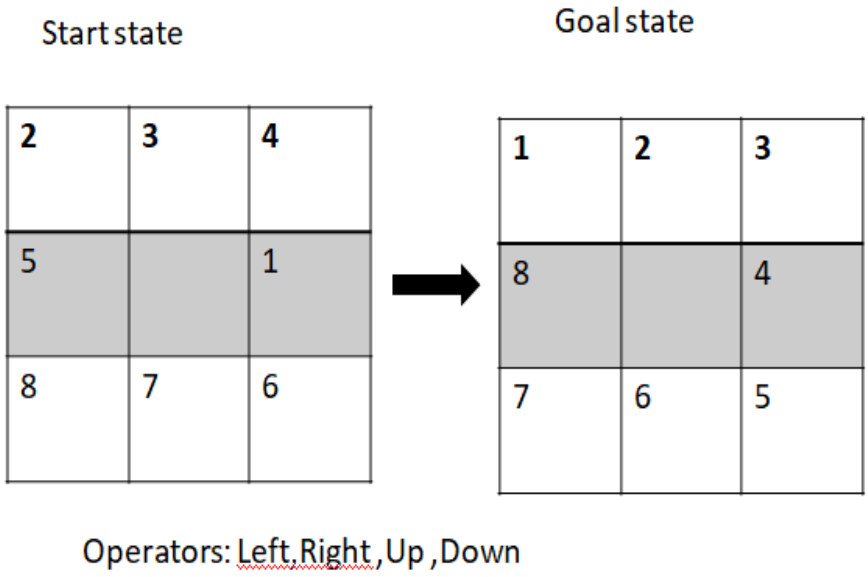
Problem Space refers to the entire range of components that exist in the process of finding a solution to a problem. This range starts with "defining the problem," then proceeds to the intermediate stage of "identifying and testing possible solutions" and ends with the final stage of "choosing and implementing a solution". Plus, it includes all of the smaller steps that exist between these identified stages.

A 'problem space' is an abstract space. The problem space may contain one or more solutions. A solution is a combination of operations and objects that achieve the goals.

It is the environment in which the search takes place. (A set of states and set of operators to change those states).

- **State** :It is the combination of all the possible states that you can take.
- **Operator:** It is the combination of all possible legal moves that you can take. It is a function that takes a state and map it to another state.

Example : 8-Puzzle Problem



Search:

The problem can then be solved by using the rules, in combination with an appropriate control strategy, to move through the problem space until a path from an initial state to a goal state is found. This process is known as 'search'.

A 'search' refers to the search for a solution in a problem space. Search proceeds with different types of 'search control strategies'. The depth-first search and breadth-first search are the two common search strategies.

Problem space Graph:

A problem space is represented by a directed graph, where nodes represent search state and paths represent the operators applied to change the state. That

is, it represents problem state. States are shown by nodes and operators are shown by edges.

3. PRODUCTION SYSTEM:

A production system in AI is a type of computer program that provides artificial intelligence based on a set of rules.

A production system (popularly known as a production rule system) is a kind of cognitive architecture that is used to implement search algorithms and replicate human problem-solving skills. This problem-solving knowledge is encoded in the system in the form of little quanta popularly known as productions.

It consists of two components: rule and action. Rules recognize the condition, and the actions part has the knowledge of how to deal with the condition.

The production system in AI contains a set of rules which are defined by the left side and right side of the system.

- The left side contains a set of things to watch for (condition), and
- The right side contains the things to do (action).

This form of operation is analogous to the popular IF-THEN structure of programming, wherein the IF and THEN correspond to the LHS and RHS respectively.

Elements of a Production System:

An AI production system has three main elements which are as follows:

1. **Global Database:** The primary database which contains all the information necessary to successfully complete a task. It is further broken down into two parts**: temporary and permanent**. The temporary part contains information relevant to the current situation only whereas the permanent part contains information about the fixed actions.

2. **A set of Production Rules:** A set of rules that operates on the global database. Each rule consists of a precondition and post condition that the global database either meets or not. For example, if a condition is met by the global database, then the production rule is applied successfully.

3. **Control System:** A control system that acts as the decision-maker, decides which production rule should be applied. The Control system stops computation or processing when a termination condition is met on the database.

Features of a Production System:

1. **Simplicity:** Due to the use of the IF-THEN structure, each sentence is unique in the production system. This uniqueness makes the knowledge representation simple to enhance the readability of the production rules.

2. **Modularity:** The knowledge available is coded in discrete pieces by the production system, which makes it easy to add, modify, or delete the information without any side effects.

3. **Modifiability:** This feature allows for the modification of the production rules. The rules are first defined in the skeletal form and then modified to suit an application.

4. **Knowledge-intensive:** As the name suggests, the system only stores knowledge. All the rules are written in the English language. This type of representation solves the semantics problem.

Classes of a Production System:

A production system is classified into four main classes which are:

1. **Monotonic Production System:** In a monotonic production system, the use of one rule never prevents the involvement of another rule

when both the rules are selected at the same time. Hence, it enables the system to apply rules simultaneously.

2. **Partially Commutative Production System:** In this production system if a set of rules is used to change state A to state B then any allowable combination of these rules will also produce the same results (convert state A to state B).

3. **Non-Monotonic Production System:** This production system increases the problem-solving efficiency of the machine by not keeping a record of the changes made in the previous search process. These types of production systems are useful from an implementation point of view as they do not backtrack to the previous state when it is found that an incorrect path was followed.

4. **Commutative Production System:** These type of production systems is used when the order of operation is not important, and the changes are reversible.

Advantages

1. Provides **excellent tools** for structuring AI programs.
2. The system is highly **modular** because individual rules can be added, removed or modified independently.
3. Separation of **knowledge** and **Control-Recognises Act Cycle.**
4. A natural **mapping** onto state-space research data or goal-driven.
5. The system uses pattern directed control which is more **flexible** than algorithmic control
6. Provides opportunities for **heuristic control** of the search

7. A good way to model the **state-driven nature** of intelligent machines
8. Quite helpful in **a real-time** environment and applications.

Disadvantages:

1. It is very **difficult** to analyze the flow of control within a production system
2. It describes the operations that can be performed in a search for a solution to the problem.
3. There is an **absence of learning** due to a rule-based production system that does not store the result of the problem for future use.
4. The rules in the production system should not have any type of **conflict resolution** as when a new rule is added to the database it should ensure that it does not have any conflict with any existing rule.

EXAMPLE :WATER JUG PROBLEM

PROBLEM:

We have to jugs of capacity 5L and 3L (liter), and a tap with endless supply of water. The objective is to obtain 4 liter exactly in the 5 liter jug with the minimum steps possible.

The state space for this problem can be described as the set of ordered pairs of integers(x,y),such that x=0,1,2,3,4,5 and y=0,1,2,3.

- x represents the quantity of water in 5L jug.
- y represents the quantity of water in 3L jug.

Start state=(0,0)

Goal state=(4,n)

PRODUCTION SYSTEM(Production rules):

1. Fill the 5 liter jug from tap
2. Empty the 5 liter jug
3. Fill the 3 liter jug from tap
4. Empty the 3 liter jug
5. Empty the 3 liter jug to 5 liter
6. Empty the 5 liter jug to 3 liter
7. Pour water from 3 liter to 5 liter
8. Pour water from 5 liter to 3 liter (do not empty)

SOLUTION 1:

x (5 L)	**y (3L)**	**Rule**
0	**0**	
5	0	1.Fill the 5 liter jug from tap
2	3	8.Pour water from 5 liter to 3 liter (do not empty)
2	0	4.Empty the 3 liter jug
0	2	6.Empty the 5 liter jug to 3 liter
5	2	1.Fill the 5 liter jug from tap
4	**3**	8.Pour water from 5 liter to 3 liter (do not empty)

SOLUTION 2

x (5 L)	y (3L)	Rule
0	**0**	
0	3	3. Fill the 3 liter jug from tap
3	0	5. Empty the 3 liter jug to 5 liter
3	3	3. Fill the 3 liter jug from tap
5	1	7. Pour water from 3 liter to 5 liter
0	1	2. Empty the 5 liter jug
1	0	5. Empty the 3 liter jug to 5 liter
1	3	3. Fill the 3 liter jug from tap
4	**0**	5. Empty the 3 liter jug to 5 liter

1,8,4,6,1,8 or

3,5,3,7,2,5,3,5.

there may be other solutions but these are the shortest and the 1st sequence is chosen as it has the min no of steps.

FINAL SOLUTION :

Min no of steps =solution 1(1,8,4,6,1,8)

4. THE PREDICATE CALCULUS:

In the Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false.

Propositional logic is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power.

Consider the following sentence, which we cannot represent using Propositional logic.

1. All cats have tails.
2. Tom is a cat.

From these two sentences, one should be able to conclude that

1. Tom has a tail.

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as **Predicate calculus.**

Alternative names of Predicate calculus :

- Predicate logic
- First order logic
- Elementary logic
- Restricted predicate calculus
- Restricted functional calculus
- Relational calculus
- Theory of quantification
- Theory of quantification with equality

Predicate calculus:

Predicate calculus is another way of knowledge representation in artificial intelligence. Predicate calculus is a generalization of propositional calculus. Hence, besides **terms, predicates, and quantifiers**, predicate calculus contains propositional variables, constants and connectives as part of the language.

It is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.

The predicate calculus uses three notations. These are:

1. **Predicates:**

A predicate is defined as a relation that binds two atoms together.

Example:

Mary and Paul are siblings is represented as :

Siblings(Mary, Paul)

Generally, predicates are used to describe certain properties or relationships between individuals or objects. Here the phrase "are siblings" is a predicate that links two atoms , Mary and Paul.

Generally, predicates make statements about individuals. These statements are formed from a predicate symbol followed by a parenthesis with a sequence of terms or argument list. That is,

Predicate (term1, term2, ..., term n)

The entries of the argument list are called arguments. The arguments can be either variables or individual constants.

Example:

- In the statement "Mary and Paul are siblings", the argument list is given by Mary and Paul, in that order, whereas the predicate is described by the phrase "are siblings".
- In predicate calculus, each predicate is given a name, which is followed by the list of arguments. The list of arguments is enclosed in parentheses. Here Siblings(Mary, Paul) .

- Note that the order of arguments is important. Clearly, the statements Siblings(Mary, Paul) and Siblings(Paul, Mary) have a completely different meaning.

Arity of the predicate:

- The number of elements in the argument list of a predicate is called the arity of the predicate.
- The arity of a predicate is fixed.
- Example:

  Siblings(Mary, Paul) has arity 2.

n-place predicate :

- A predicate with arity n is often called an n-place predicate. A one-place predicate is called a property.
- **Examples:**

  1. **Tom is a cat**

  Cat(Tom) is a one-place predicate, or a property.

  2. **Mary and Paul are siblings**

  Siblings(Mary, Paul) is a two-place predicate.

  3. **The sum of 2 and 3 is 5**

  Sum(2,3,5) is a three-place predicate.

Function as an argument:

- It is also possible to have a function as an argument.
- Example:

Ravi's father is Rani's father is represented as

FATHER(father(Ravi),Rani)

Here FATHER is a predicate and father(Ravi) is a function to indicate Ravi's father.

Atomic formula:

- A predicate name, followed by an argument list in parentheses is called an atomic formula . The atomic formulas can be combined by logical connectives like propositions.

Example:

Tom is a cat and that Tom has a tail

cat(Tom) and hastail(Tom)

cat(Tom) → hastail(Tom)

2. Terms :

Terms are those arguments in a predicate . Terms play a similar role in predicate calculus as nouns and pronouns do in the English language.

Terms can be defined as follows:

a) A constant is a term.

b) A variable is a term.

c) If f is a function and $x_1,x_2,\ldots,x_n$ are terms, the $f(x_1,x_2,\ldots,x_n)$ is a term.

d) All terms are generated by applying the above mentioned rules.

Example:

Mary and Paul are siblings is represented as

Siblings(Mary, Paul)

Here mary and paul are terms.

3. 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 quantifier:

a) Universal Quantifier(for all, for each, for every)

b) Existential quantifier (there exists , for some, for least one).

Universal Quantifier:

The universal quantifier is used to indicate that a statement is always true. Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.The Universal quantifier is represented by a symbol ∀.

If x is a variable, then ∀x is read as:

- **For all x**
- **For each x**

- **For every x.**

Definition :

Let A represent a formula, and let x represent a variable. If we want to indicate that A is true for all possible values of x, we write ∀xA. Here, **∀x** is called u**niversal quantifier,** and **A** is called the **scope of the quantifier**. The variable x is said to be bound by the quantifier. The symbol ∀ is pronounced "for all".

Example1:

Express "**Everyone gets a break once in a while**" in predicate calculus.

Solution:

We define B to mean "gets a break once in a while". Hence, B(x) means that x gets a break once in a while. The word everyone indicates that this is true for all x. This leads to the following translation: **∀xB(x).**

Example2:

Express "**All cats have tails**" in predicate calculus.

Solution:

We define B to mean "**cats have tails**". Hence, B(x) means that x **cats have tails**. The word all indicates that this is true for all x. This leads to the following translation: **∀xB(x).**

Existential Quantifier:

The existential quantifier indicates that a statement is sometimes true. Existential quantifiers are the type of quantifiers, which express that the

statement within its scope is true for at least one instance of something. It is denoted by the logical operator ∃.

If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- ✓ **There exists a 'x.'**
- ✓ **For some 'x.'**
- ✓ **For at least one 'x.'**

Definition:

Let A represent a formula, and let x represent a variable. If we want to indicate that A is true for at least one value x, we write ∃xA. This statement is pronounced "There exists an x such that A." Here, **∃x** is called the **existential quantifier**, and **A** is called the **scope of the quantifier**. The variable x is said to be bound by the quantifier.

Example1:

Express "There exist people who like their meat raw" in predicate calculus.

Solution:

We define B to mean "like their meat raw". Hence, B(x) means that x like their meat raw. The word There exist indicates that this is true for all x. This leads to the following translation: ∃ **xB(x).**

Note:

- The main connective for universal quantifier **∀** is implication →.
- The main connective for existential quantifier **∃** is and **∧**.
- In universal quantifier, ∀x∀y is similar to ∀y∀x.

- In Existential quantifier, ∃x∃y is similar to ∃y∃x.
- ∃x∀y is not similar to ∀y∃x.

5. INFERENCE RULES:

In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, so generating the conclusions from evidence and facts is termed as Inference. Inference rules are the templates for generating valid arguments.

Inference rules are applied to derive proofs in artificial intelligence, and the proof is a sequence of the conclusion that leads to the desired goal.

Terminologies related to inference rules:

1. **Implication:** It is one of the logical connectives which can be represented as P → Q. It is a Boolean expression.
2. **Converse:** The converse of implication, which means the right-hand side proposition goes to the left-hand side and vice-versa. It can be written as Q → P.
3. **Contra positive:** The negation of converse is termed as contra positive, and it can be represented as ¬ Q → ¬ P.
4. **Inverse:** The negation of implication is called inverse. It can be represented as ¬ P → ¬ Q.

Truth table:

P	Q	P→Q	Q→P	~Q→~P	~P→~Q
F	F	T	T	T	T
F	T	T	F	T	F
T	F	F	T	F	T
T	T	T	T	T	T

Some of the important rules of inference are:

1. **Addition :**

The Addition rule is one the common inference rule, and it states that If P is true, then P∨Q will be true.

That is, P →(P ∨Q)

Notation:

$$\frac{P}{P \vee Q}$$

Truth Table:

P	Q	P∨Q
F	F	F
F	T	T
T	F	T
T	T	T

2. **Simplification:**

The simplification rule state that if **P∧ Q** is true, then **Q or P** will also be true.

That is, P∧ Q → Q or P∧ Q → P

Notation:

$$\frac{P \wedge Q}{Q} \quad \text{or} \quad \frac{P \wedge Q}{P}$$

Truth table:

P	Q	P∧ Q
F	F	F
F	T	F
T	F	F
T	T	T

3. Modus ponens:

The Modus Ponens rule is one of the most important rules of inference, and it states that if P and P → Q is true, then we can infer that Q will be true.

That is, (P & (P →Q) →Q)

Notation:

$$\frac{P \rightarrow Q, P}{\therefore Q}$$

Truth table:

P	Q	P →Q
F	F	T
F	T	T
T	F	F
T	T	T

4. Modus Tollens :

The Modus Tollens rule state that if P→ Q is true and ¬ Q is true, then ¬ P will also true.

That is, (~ Q & (P →Q)) → ~P

Notation

$$\frac{P \rightarrow Q, \sim Q}{\sim P}$$

Truth table:

P	Q	~ P	~ Q	P →Q
F	F	T	T	T
F	T	T	F	T
T	F	F	T	F
T	T	F	F	T

5. **Disjunctive Syllogism:**

The Disjunctive syllogism rule state that if P∨Q is true, and ¬P is true, then Q will be true.

That is , ((P ∨Q) & ~ P) → Q

Notation:

$$\frac{P \vee Q \ , \sim P}{Q}$$

Truth table:

P	Q	~ P	P ∨Q
F	F	T	F
F	T	T	T
T	F	F	F
T	T	F	T

6. **Hypothetical Syllogism:**

The Hypothetical Syllogism rule state that if P→R is true whenever P→Q is true, and Q→R is true.

That is , **[(**P→Q) & (Q→R)] → **[(**P→R)]

Notation:

P→Q, Q→R

∴P→R

Truth table:

P	Q	R	P →Q	Q →R	P →R
F	F	F	T	T	T
F	F	T	T	T	T
F	T	F	T	F	T
F	T	T	T	T	T
T	F	F	F	T	F
T	F	T	F	T	T
T	T	F	T	F	F
T	T	T	T	T	T

7. <u>Constructive Dilemma:</u>

The Constructive Dilemma rule state that, if *P* implies *Q* and *R* implies *S* and either *P* or *R* is true, then either *Q* *or* *S* has to be true.

That is , [(P→Q) & (R→S) & (P ∨ R)] →(Q ∨ S)

Notation:

$P \rightarrow Q$, $R \rightarrow S$, $P \vee R$

$\therefore Q \vee S$

Truth table:

P	Q	R	S	P→Q	R→S	P ∨R	Q∨S
F	F	F	F	T	T	F	F
F	F	F	T	T	T	F	T
F	F	T	F	T	F	T	F
F	F	T	T	T	T	T	T
F	T	F	F	T	T	F	T
F	T	F	T	T	T	F	T
F	T	T	F	T	F	T	T
F	T	T	T	T	T	T	T
T	F	F	F	F	T	T	F
T	F	F	T	F	T	T	T
T	F	T	F	F	F	T	F
T	F	T	T	F	T	T	T
T	T	F	F	T	T	T	T
T	T	F	T	T	T	T	T
T	T	T	F	T	F	T	T
T	T	T	T	T	T	T	T

8. Destructive Dilemma:

The Destructive Dilemma rule state that, if P implies Q, R implies S, and ~Q or ~S is true, then ~P or ~R is true .

That is , $[(P \rightarrow Q) \& (R \rightarrow S) \ \& \ (\sim Q \vee \sim S)] \rightarrow (\sim P \ \vee \sim R)$

Notation:

$P \rightarrow Q , R \rightarrow S , \sim Q \vee \sim S$

$\therefore \sim P \vee \sim R$

Truth table:

P	Q	R	S	~P	~Q	~R	~S	P→Q	R→S	~Q ∨ ~S	~P ∨ ~R
F	F	F	F	T	T	T	T	T	T	T	T
F	F	F	T	T	T	T	F	T	T	T	T
F	F	T	F	T	T	F	T	T	F	T	T
F	F	T	T	T	T	F	F	T	T	T	T
F	T	F	F	T	F	T	T	T	T	T	T
F	T	F	T	T	F	T	F	T	T	F	T
F	T	T	F	T	F	F	T	T	F	T	T
F	T	T	T	T	F	F	F	T	T	F	T
T	F	F	F	F	T	T	T	F	T	T	T
T	F	F	T	F	T	T	F	F	T	T	T
T	F	T	F	F	T	F	T	F	F	T	F
T	F	T	T	F	T	F	F	F	T	T	F
T	T	F	F	F	F	T	T	T	T	T	T
T	T	F	T	F	F	T	F	T	T	F	T
T	T	T	F	F	F	F	T	T	F	T	F
T	T	T	T	F	F	F	F	T	T	F	F

9. Negation:

The Negation rule state that, If a statement is true, then it is not the case that the statement is not true.

That is , (~(~P))=P

Notation:

(~(~P))

P

Truth Table :

P	~ P	~ ~ P
F	T	F
T	F	T

10. Resolution:

The Resolution rule state that if P∨Q and ~ P∧R is true, then Q∨R will also be true.

That is , (P∨Q and ~ P∧R) → (Q∨R)

Notation:

P∨Q , ~ P∧R

Q∨R

P	Q	R	~P	PVQ	~ PAR	QVR
F	F	F	T	F	F	F
F	F	T	T	F	T	T
F	T	F	T	T	F	T
F	T	T	T	T	T	T
T	F	F	F	T	F	F
T	F	T	F	T	F	T
T	T	F	F	T	F	T
T	T	T	F	T	F	T

11. Universal instantiation:

Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

As per UI, we can infer any sentence obtained by substituting a ground term for the variable.

The UI rule state that we can infer any sentence P(a) by substituting a ground term a (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse.

Notation:

$\forall x P(x)$

$\therefore P(a)$

Example:

IF "**Every person like ice-cream**" => **∀x P(x)**

so we can infer that "**John likes ice-cream**" => **P(a)**

12. Universal generalization:

Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).

This rule can be used if we want to show that every element has a similar property.

Notation:

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

Example:

Let's represent,

P(a): "**A byte contains 8 bits**", so for

∀ x P(x) :"**All bytes contain 8 bits**.", it will also be true.

13. Existential instantiation:

Existential instantiation is also called as Existential Elimination.It can be applied only once to replace the existential sentence.

This rule states that one can infer P(a) from the formula given in the form of ∃x P(x) for a new constant symbol a.

The restriction with this rule is that a used in the rule must be a new term for which P(a) is true.

Notation:

$\exists x\, P(x)$

$\therefore P(a)$

Example:

Let's represent,

$\exists x\, P(x)$: " someone got good marks in English."

P(a) : " Privanka got good marks in English".

14. Existential generalization :

An existential generalization is also known as an existential introduction. This rule states that if there is some element a in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.

Notation:

$P(a)$

$\therefore \exists x\, P(x)$

Example:

Let's represent,

P(a) : " Privanka got good marks in English".

Therefore **$\exists x$ P(x) :** " someone got good marks in English."

6. STRUCTURES AND STRATEGIES FOR STATE SPACE SEARCH:

Predicate calculus expressions provide a means of describing objects and relations in a problem domain, and inference rules such as modus phonens allow us to infer new knowledge from these descriptions. These inferences define a space that is searched to find a problem solution.

To successfully design and implement search algorithms, a programmer must be able to analyze and predict their behaviour.By representing a problem as a state space graph, we can use graph theory to analyze the structure and complexity of both the problem and the search procedures that we employ to solve it.

A graph consists of a set of nodes and a set of arcs or links connecting pairs of nodes.

In the state space model of problem solving, the nodes of a graph are taken to represent discrete states in a problem solving process, such as the results of logical inferences or the different configurations of a game board. The arcs of the graph represent transitions between states. These transitions correspond to logical inferences or legal moves of a game.

State space search:

It characterizes problem solving as the process of finding a solution path from the start state to a goal.

Arcs of the state space correspond to steps in a solution process and paths through the space represent solutions in various stages of completion.

Paths are searched, beginning at the start state and continuing through the graph, until either the goal description is satisfied or they are abandoned.

The actual generation of new states along the path is done by applying operators, such as “legal moves” in a game or inference rules in a logic problem., to existing states on a path.

The task of a search algorithm is to find a solution path through such a problem space.

Search algorithms must keep track of the paths from a start state to a goal node, because these paths contain the series of operations that lead to the problem solution.

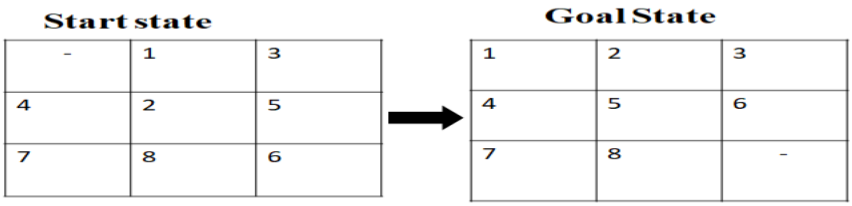
Definition of state space search:

A state space is represented by a 4-tuple **[N,A,S,GD]** ,where:

- N is the set of nodes or states of the graph. These correspond to the states in a problem solving process.
- A is the set of arcs (or links) between nodes. These correspond to the steps in a problem solving process.
- S,a nonempty subset of N,contains the start states of the problem.
- GD,a nonempty subset of N,contains the goal states of the problem. The states in GD are described using either:
  - A measurable property of the states encountered in the search.
  - A property of the path developed in the search, for example the transition costs for the arcs of the path.

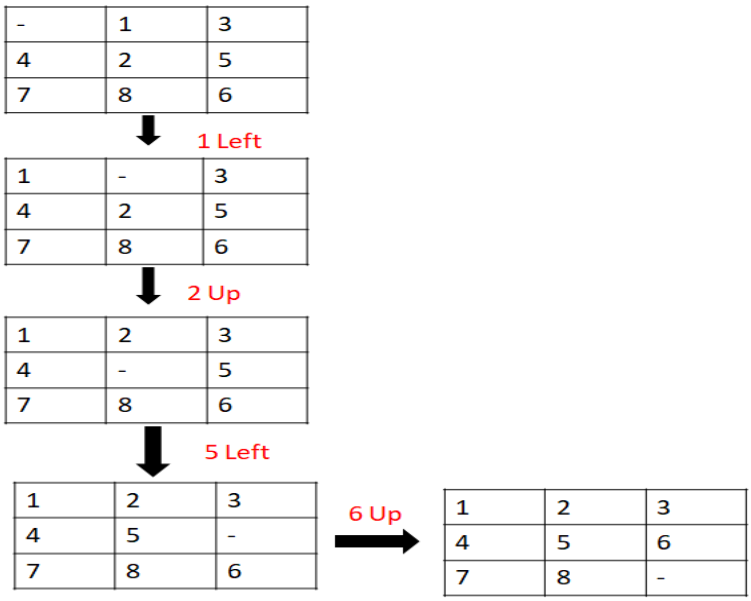
A solution path is a path through this graph from a node in S to a node in GD.

Example : 8-Puzzle Problem



Operators:

Left , Right , Up ,Down



Structures for state space search:

A graph is a set of nodes or states and a set of arcs that connect the nodes.

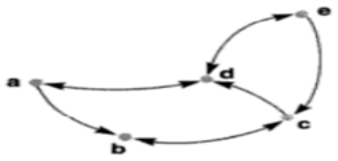
A labelled graph has one or more descriptors (labels) attached to each node that distinguish that node from any other node in the graph.

In a state space graph, these descriptors identify states in a problem solving process. If there are no descriptive differences between two nodes, they are considered the same. The arc between two nodes is indicated by the labels of the connected nodes.

The arcs of a graph may also be labelled.Arc labels are used to indicate that an arc represents a named relationship or to attach weights to arcs .If there are different arcs between the same two nodes,these can also be distinguished through labelling.

A graph is directed if arcs have an associated directionality. The arcs in a directed graph are usually drawn as arrows or have an associated directionality.

The arcs in a directed graph are usually drawn as arrows or have an arrow attached to indicate direction. Arcs that can be crossed in either direction may have two arrows attached but more often have no direction indicators at all.

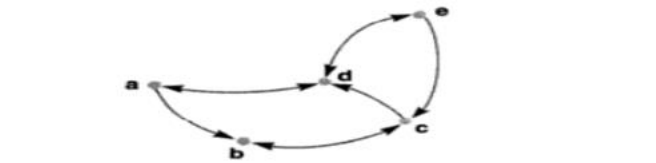


Nodes = {a,b,c,d,e}
Arcs = {(a,b),(a,d),(b,c),(c,b),(c,d),(d,a),(d,e),(e,c),(e,d)}

A labeled directed graph.

In this figure, arc(a,b) may only be crossed from node a to node b,but arc (b,c) is crossable in either direction.

A path through a graph connects a sequence of nodes through successive arcs. The path is represented by an ordered list that records the nodes in the order they occur in the path.

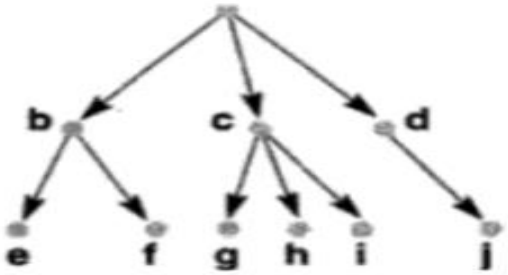


Nodes = {a,b,c,d,e}
Arcs = {(a,b),(a,d),(b,c),(c,b),(c,d),(d,a),(d,e),(e,c),(e,d)}

A labeled directed graph.

Here [a,b,c,d] represents the path through nodes a,b,c, and d,in that order.

A rooted graph has a unique node, called the root, such that there is a path from the root to all nodes within the graph. A tree is a graph in which two nodes have at most one path between them. For rooted trees or graph, relationships between nodes include parent, child and sibling. The children of a node are called siblings.



A rooted tree, exemplifying family relationships.

Strategies for space search:

1. **Data-driven search (forward chaining) :**

Data Driven approach collects the facts about the problem , apply rules on the data and produce new facts from it to move towards the goal. It is also termed as forward chaining.

Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties:

- It is a bottom-up approach, as it moves from bottom to top.
- It operates in the forward direction.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- It is suitable for the planning, monitoring, control, and interpretation application. It is commonly used in the expert system, such as CLIPS, business, and production rule systems.
- Forward chaining reasoning applies a breadth-first search strategy.
- It tests for all the available rules.
- It can generate an infinite number of possible conclusions.It is aimed for any conclusion.

2. Goal-driven search (backward chaining):

Goal driven approach focus on the goal, search the facts and rules that could be able to produce that goal. It works by processing in backward direction via rules and sub goals to reach to the facts of the problem. It is also called backward chaining.

A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties:

- It is a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- It operates in the backward direction.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward chaining reasoning applies a depth-first search strategy.
- Backward chaining only tests for few required rules.
- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- Backward chaining is suitable for diagnostic, prescription, and debugging application.
- Backward chaining generates a finite number of possible conclusions.
- Backward chaining is only aimed for the required data.

3. Bidirectional search :

Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small sub graphs in which one starts the search

from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.

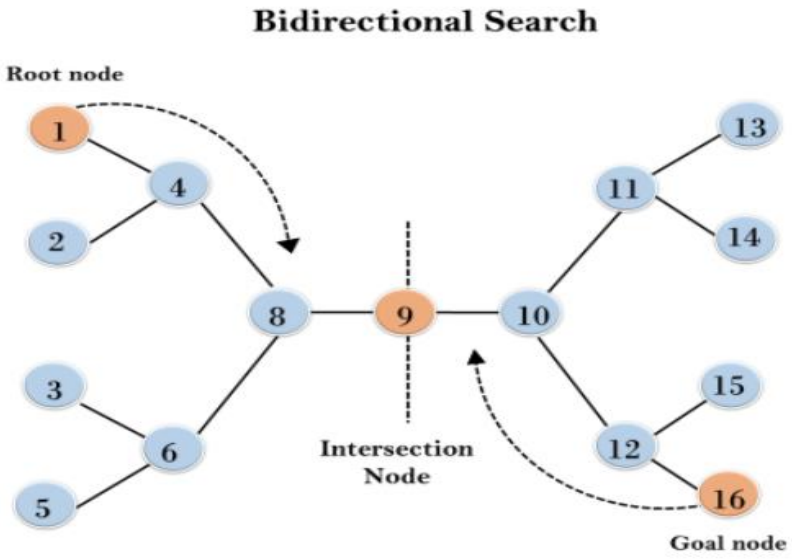
Bidirectional search can use search techniques such as BFS, DFS, DLS(Depth Limited Search), etc.

Advantages:

1. Bidirectional search is fast.
2. Bidirectional search requires less memory

Disadvantages:

3. Implementation of the bidirectional search tree is difficult.
4. In bidirectional search, one should know the goal state in advance.



This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.

4. Blind search methods:

Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and

test for the goal, so it is also called blind search. It examines each node of the tree until it achieves the goal node.

It can be divided into five main types:

- Breadth-first search
- Uniform cost search
- Depth-first search
- Iterative deepening depth-first search
- Bidirectional Search

Using state space to represent reasoning with the predicate calculus:

State Space Description of a Logical System:

Predicate calculus can be used as the formal specification language for making nodes distinguishable as well as for mapping the nodes of a graph onto the state space.Inference rules can be used to create and describe the arcs between states.

Problems in the predicate calculus, such as determining whether a particular expression is a logical consequence of a given set of assertions, may be solved using graph search.

Example:

A set of assertions:

q →p

r →p

v →q

s → r

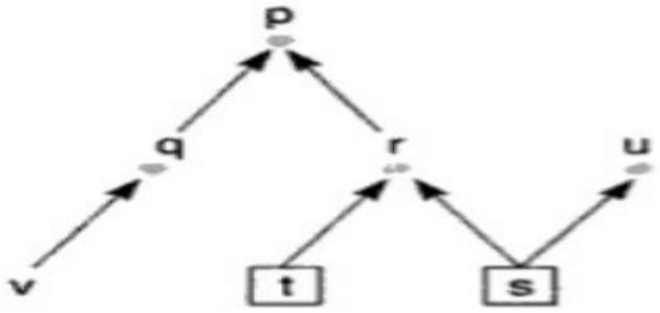
t → r

s → u

s

t

The relationship between the initial assertions and these inferences is expressed in the directed graph. The arcs correspond to logical implications.



The path [s,r,p] corresponds to the sequence of inferences:

s and s → r yields r.

r and r → p yields p.

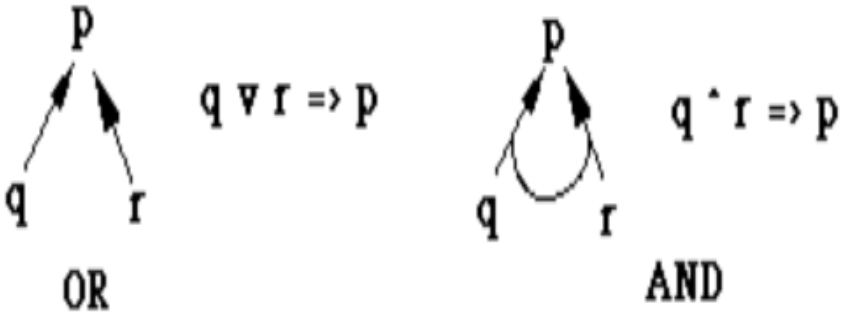
Data driven proof of p:Find a path from a boxed node (start node) to the goal node(p).

And/Or graphs:

Expressing the logical relationships defined by AND and OR operators requires an extension to the basic graph model known as an And/Or graph.

And/Or graphs are an important tool for describing the search spaces generated by many AI problems, including those solved by logical theorem provers and expert systems.

If the premises of an implication are connected by an ∧ operator, they are called AND nodes, and the arcs from this node are joined by a curved link.



Assertions:	And/Or graph
a b c a ∧ b → d a ∧ c → e b ∧ d → f f → g a ∧ e → h