VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
Approved by AICTE New Delhi & Affiliated to JNTUK Kakinada
Accredited by NAAC 'A+' and NBA | ISO 9001 : 2015
Vadlamudi - 522 213, Guntur District

# Neural Networks and Soft Computing Material

# IV B,Tech CSE


Prepared by

Dr.M.Suman

Associate Professor,

EEE Department.

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
Approved by AICTE New Delhi & Affiliated to JNTUK Kakinada
Accredited by NAAC 'A+' and NBA | ISO 9001 : 2015
Vadlamudi - 522 213, Guntur District

# UNIT I: Soft Computing and Artificial Intelligence

- Introduction of Soft Computing
- Soft Computing vs. Hard Computing,
- Various Types of Soft Computing Techniques,
- Applications of Soft Computing,
- AI Search Algorithm,
- Predicate Calculus, Rules of Interference,
- Semantic Networks, Frames, Objects,
- Hybrid Models.

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
Approved by AICTE New Delhi & Affiliated to JNTUK Kakinada
Accredited by NAAC 'A+' and NBA | ISO 9001 : 2015
Vadlamudi - 522 213, Guntur District

# IV B.Tech II Sem (R20)

## Assignment-1 Question Bank
## *Neural Networks and Soft Computing(NNSC)*

## (*Computer Science Engineering*)

| Question No. | Question | Marks allotted | Course Outcome | Bloom's Taxonomy | Page No. |
|---|---|---|---|---|---|
| 1 | What is Soft Computing and discuss various types of Soft Computing Techniques. | 5 | 1 | U | Unit-1 2, 6-9. |
| 2 | Differentiate the Soft Computing vs. Hard Computing. | 5 | 1 | U | Unit-1 4 |
| 3 | Explain the various Applications of Soft Computing. | 5 | 1 | E | Unit-1 10-12 |
| 4 | Discuss the Predicate Calculus and rules of interference | 5 | 1 | E | Unit-1 20-24 |
| 5 | How knowledge is represented in terms of Semantic Networks, discuss in detail using neat diagrams | 5 | 1 | E | Unit-1 25-26 |
| 6 | Briefly discuss the different Hybrid Models of the Soft Computing. | 5 | 1 | Ap | Unit-1 27-28 |

**R** - Remember, **U** - Understand, **Ap** – Apply, **A** - Analyse, **E**- Evaluate, **C**- Create

CO1: Understand the concepts of Artificial intelligence and soft computing techniques
CO2: Analyze the concepts of Neural Networks and select the Learning Networks in modeling real world systems.
CO3: Implement the concepts of Fuzzy reasoning and concepts of Genetic algorithm and its applications to soft computing.
CO4: Classify Biologically inspired algorithm such as neural networks, genetic algorithms, ant colony optimization, and bee colony optimization.
CO5: Design hybrid system incorporating neural network, genetic algorithms, fuzzy systems

## Introduction to Soft Computing

Before understanding what is Soft computing we must know what is computing. In simple words, computing means mapping the given set of inputs to output using a formal method or an algorithm to solve a problem. In the context of computing, this input is called 'Antecedent' and the output is called 'Consequent'.

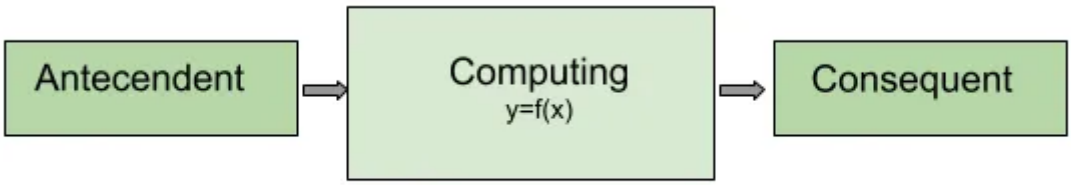



Figure 1. Basics of Computing

The method of computing must be **unambiguous**, **accurate**, and provide a **precise solution**. Computing is suitable for problems that are easy to model mathematically.Now, before moving on to soft computing let us understand what is hard computing and why there was a need to develop soft-computing algorithms. Lotfi .A. Zade was the first person to introduce the concept of hard computing in 1996. According to him, a computing concept comes under the category of **hard computing** if :

>It provides **precise results**.

>Algorithm used to solve a problem is **unambiguous**.

>The control action is **formally defined** using an algorithm or a mathematical model.

Problems like finding derivatives, integrals, searching & sorting algorithms, finding the shortest path between two points, and many more for which we can get a **precise result** by using a **mathematical model**, comes under hard computing.

**Need of Soft-Computing**

Some of the real-world tasks like handwriting recognition, image classification, music generation, etc do not have an algorithm for computation of exact solutions in polynomial time. This is where Soft computing comes into play.

The term Soft-computing was also coined by Lotfi. A. Zade [1]. He defined soft computing as follows:

Soft computing is a collection of methodologies that aim to exploit the **tolerance of imprecision** and **uncertainity** to achieve tractability, **robustness**, and **low solution cost**. Its principle components are **fuzzy logic**, **neuro-computing** and **probabilistic reasoning**. The role model for soft computing is **human mind**.

In the above definition, the following are a few key terms which one must understand:

- *Tolerance of imprecision:* the result obtained using soft-computing is not precise.
- *Uncertainty*: the soft-computing algorithm may give different results every time for the same problem.
- *robustness*: soft-computing algorithms can tackle any kind of input noise
- *low solution cost*: soft-computing makes it feasible to solve some of the problems which could be computationally very expensive if solved using hard computing.

Soft-computing algorithms are based on the biological decision-making system and follow methodologies like genetics, evolution, Ant's behaviors, partical swarming, human nervous system etc. The three computing paradigms followed by soft-computing computations are **fuzzy logics**, **neuro-computing**, and **probabilistic reasoning** (genetic algorithm)[2].

**Applications of Soft Computing**

- Image processing

- Data Compression
- Fuzzy Logic Control
- Automotive systems and Manufacturing
- Neuro-fuzzy systems
- Decision-support systems

and many more.

**What is Soft Computing: Techniques and Differences**

Computation is a process of converting the input of one form to some other desired output form using certain control actions. According to the concept of computation, the input is called an antecedent and the output is called the consequent. A mapping function converts the input of one form to another form of desired output using certain control actions. The computing concept is mainly applicable to computer science engineering. There are two types of computing, hard computing, and soft computing. Hard computing is a process in which we program the computer to solve certain problems using mathematical algorithms that already exist, which provides a precise output value. One of the fundamental examples of hard computing is a numerical problem.

**What is Soft Computing?**

Soft computing is an approach where we compute solutions to the existing complex problems, where output results are imprecise or fuzzy in nature, one of the most important features of soft computing is it should be adaptive so that any change in environment does not affect the present process. The following are the characteristics of soft computing.

- It does not require any mathematical modeling for solving any given problem
- It gives different solutions when we solve a problem of one input from time to time
- Uses some biologically inspired methodologies such as genetics, evolution, particles swarming, the human nervous system, etc.

- Adaptive in nature.

There are three types of **soft computing techniques** which include the following.

- Artificial Neural Network
- Fuzzy Logic
- Genetic algorithm

**Soft Computing vs Hard Computing**

| Hard Computing | Soft Computing |
|---|---|
| • The analytical model required by hard computing must be precisely represented | • It is based on uncertainty, partial truth tolerant of imprecision and approximation. |
| • Computation time is more | • Computation time is less |
| • It depends on binary logic, numerical systems, crisp software. | • Based on approximation and dispositional. |
| • Sequential computation | • Parallel computation |
| • Gives exact output | • Gives appropriate output |
| • Examples: Traditional methods of computing using our personal computer. | • Example: Neural networks like Adaline, Madaline, ART networks, etc. |

**Advantages**

The benefits of soft computing are

- The simple mathematical calculation is performed
- Good efficiency
- Applicable in real-time
- Based on human reasoning.

**Disadvantages**

The disadvantages of soft computing are

- It gives an approximate output value
- If a small error occurs the entire system stops working, to overcome its entire system must be corrected from the beginning, which is time taking process.

**Applications**

The following are the applications of soft computing

- Controls motors like induction motor, DC servo motor automatically
- Power plants can be controlled using an intelligent control system
- In image processing, the given input can be of any form, either image or video which be manipulated using soft computing to get an exact duplicate of the original image or video.
- In biomedical applications where it is closely related to biology and medicine, soft computing techniques can be used to solve biomedical problems like diagnosis, monitoring, treatment, and therapy.
- Smart instrumentation is trendy nowadays, where intelligent devices automatically communicate with other devices using a certain set of communication protocols to perform certain tasks, but the problem here is there is no proper standard protocol to communicate. This can be overcome by using soft computing techniques, where the smart devices are communicated over multiple protocols, with high privacy and robustness.

Computing is a technique used to convert particular input using control action to the desired output. There are two types of computing techniques hard computing and soft computing. Here in our article, we are mainly focusing on soft computing, its techniques like fuzzy logic, artificial neural network, genetic algorithm, comparison between hard computing and soft computing, soft computing techniques, applications, and advantages. Here is the question “How are soft computing is applicable in the medical field?”

## Various Types of Soft Computing Techniques

Following are three types of techniques used by soft computing:

- Artificial Neural Network (ANN)
- Fuzzy Logic
- Genetic Algorithms

**NEURAL NETWORKS**

Neural networks are simplified models of the biological nervous system and therefore have drawn their motivation from the kind of computing performed by a human brain.

An NN, in general, is a highly interconnected network of a large number of processing elements called neurons in an architecture inspired by the brain. An NN can be massively parallel and therefore is said to exhibit parallel distributed processing. Neural networks exhibit characteristics such as mapping capabilities or pattern association, generalization, robustness, fault tolerance, and parallel and high speed information processing.

Neural networks learn by examples. They can therefore be trained with known examples of a problem to 'acquire' knowledge about it. Once appropriately trained, the network can be put to effective use in solving 'unknown' or 'untrained' instances of the problem. Neural networks adopt various learning mechanisms of which supervised learning and unsupervised learning methods have turned out to be very popular.

In supervised learning, a 'teacher' is assumed to be present during the learning process, i.e. the network aims to minimize the error between the target (desired) output presented by the 'teacher' and the computed output, to achieve better performance. However, in unsupervised learning, there is no teacher present to hand over the desired output and the network therefore tries to learn by itself, organizing the input instances of the problem. Though NN architectures have been broadly classified as single layer feedforward networks, multilayer feedforward networks, and recurrent networks, over the years several other NN architectures have evolved.

Some of the well-known NN systems include backpropagation network, perceptron, ADALINE ( Adaptive Linear Element), associative memory, Boltzmann machine, adaptive resonance theory, self-organizing feature map, and Hopfield network. Neural networks have been successfully applied to problems in the fields of pattern recognition, image processing, data compression, forecasting, and optimization to quote a few

**FUZZY LOGIC**

Fuzzy set theory proposed in 1965 by Lotfi A. Zadeh (1965) is a generalization of classical set theory. Fuzzy Logic representations founded on Fuzzy set theory try to capture the way humans represent and reason with real-world knowledge in the face of uncertainty. Uncertainty could arise due to generality, vagueness, ambiguity, chance, or incomplete knowledge. A fuzzy set can be defined mathematically by assigning to each possible individual in the universe of discourse, a value representing its grade of membership in the fuzzy set. This grade corresponds to the degree to which that individual is similar or compatible with the concept represented by the fuzzy set.

In other words, fuzzy sets support a flexible sense of membership of elements to a set. In classical set theory, an element either belongs to or does not belong to a set and hence, such sets are termed crisp sets. But in a fuzzy set, many degrees of membership (between 0 and 1) are allowed. Thus, a membership function $\mu A( x)$ is associated with a fuzzy set A such that the function maps every element of the universe of discourse X to the interval [0, 1]. For example, for a set of students in a class (the universe of discourse), the fuzzy set “tall” (fuzzy set A) has as its members students who are tall with a degree of membership equal to 1($\mu A( x) = 1$), students who are of medium height with a degree of membership equal to 0.75 ($\mu A( x) = 0.75$) and those who are dwarfs with a degree of membership equal to 0 ($\mu A( x) = 0$), to cite a few cases. In this way, every student of the class could be graded to hold membership values between 0 and 1 in the fuzzy set A, depending on their height.

The capability of fuzzy sets to express gradual transitions from membership ($0 < \mu A( x) \leq 1$) to non-membership ($\mu A( x) = 0$) and vice versa has a broad utility. It not only provides for a

meaningful and powerful representation of measurement of uncertainties, but also provides for a meaningful representation of vague concepts expressed in natural language.

Operations such as union, intersection, subsethood, product, equality, difference, and disjunction are also defined on fuzzy sets. Fuzzy relations associate crisp sets to varying degree of membership and support operations such as union, intersection, subsethood, and composition of relations. Just as crisp set theory has influenced symbolic logic, fuzzy set theory has given rise to fuzzy logic.

While in symbolic logic, truth values True or False alone are accorded to propositions, in fuzzy logic multivalued truth values such as true, absolutely true, fairly true, false, absolutely false, partly false, and so forth are supported. Fuzzy inference rules (which are computational procedures used for evaluating linguistic descriptions) and fuzzy rule based systems (which are a set of fuzzy IF-THEN rules) have found wide applications in real-world problems. Fuzzy logic has found extensive patronage in consumer products especially promoted by the Japanese companies and have found wide use in control systems, pattern recognition applications, and decision making, to name a few.

## GENETIC ALGORITHMS

Genetic Algorithms initiated and developed in the early 1970s by John Holland (1973; 1975) are unorthodox search and optimization algorithms, which mimic some of the processes of natural evolution. GAs perform directed random searches through a given set of alternatives with the aim of finding the best alternative with respect to the given criteria of goodness. These criteria are required to be expressed in terms of an objective function which is usually referred to as a fitness function. Fitness is defined as a figure of merit, which is to be either maximized or minimized. It is further required that the alternatives be coded in some specific finite length which consists of symbols from some finite alphabet. These strings are called chromosomes and the symbols that form the chromosomes are known as genes.

In the case of binary alphabet (0, 1) the chromosomes are binary strings and in the case of real alphabet (0−9) the chromosomes are decimal strings. Starting with an initial population of chromosomes, one or more of the genetic inheritance operators are applied to generate offspring that competes for survival to make up the next generation of population. The genetic inheritance

operators are reproduction, cross over, mutation, inversion, dominance, deletion, duplication, translocation, segregation, speciation, migration, sharing, and mating.

However, for most common applications, reproduction, mating (cross over), and mutation are chosen as the genetic inheritance operators. Successive generations of chromosomes improve in quality provided that the criteria used for survival is appropriate. This process is referred to as Darwinian natural selection or survival of the fittest. Reproduction which is usually the first operator applied on a population selects good chromosomes in a population to form the mating pool.

A number of reproduction operators exist in the literature (Goldberg and Deb, 1991). Cross over is the next operator applied. Here too, a number of cross over operators have been defined (Spears and De Jong, 1990). But in almost all cross over operators, two strings are picked from the mating pool at random and some segments of the strings are exchanged between the strings. Single point cross over, two point cross over, matrix cross over are some of the commonly used cross over operators. It is intuitive from the construction that good substrings from either parent can be combined to form better offspring strings. Mutation operator when compared to cross over is used sparingly.

The operator changes a 1 to a 0 and vice versa with a small probability P m. The need for the operator is to keep the diversity of the population. Though most GA simulations are performed by using a binary coding of the problem parameters, real coding of the parameters has also been propounded and applied (Rawlins, 1990). GAs have been theoretically and empirically proven to provide robust search in complex space and have found wide applicability in scientific and engineering areas including function optimization, machine learning, scheduling, and others

## Applications of Soft Computing:

There are various problems in the world that require vast resources and computations to be made which cannot be solved, just by logical means. Soft computing has emerged as a way of solving problems, the way humans do. Let's take a look at some of the applications of soft computing across different industries

1. Handwritten Script Recognition
2. Image Processing and Data Compression
3. Automotive Systems and Manufacturing
4. Soft computing based Architecture
5. Decision Support System
6. Power System Analysis
7. Bioinformatics
8. Investment and Trading

**Handwritten Script Recognition using Soft Computing**

Handwritten Script Recognition is one of the demanding parts of computer science. It can translate multilingual documents and sort the various scripts accordingly. It uses the concept of "block-level technique" where the system recognizes the particular script from a number of script documents given. It uses a Discrete Cosine Transform (DCT), and discrete wavelets Transform (DWT) together, which classify the scripts according to their features.

**Image Processing and Data Compression using Soft Computing**

Image analysis is one of the most important parts of the medical field. It is a high-level processing technique which includes recognition and bifurcation of patterns. Using soft computing solves the problem of computational complexity and efficiency in the classification. Techniques of soft computing include Genetic Algorithms, Genetic Programming, Classifier Systems, Evolution Strategies, artificial life, and a few others, which are used here. These algorithms give the fastest solutions to pattern recognition. These help in analyzing the medical images obtained from microscopes as well as examine the X-rays.

**Use of Soft Computing in Automotive Systems and Manufacturing**

The use of soft computing has solved a major misconception that the automobile industry is slow to adapt. Fuzzy logic is a technique used in vehicles to build classic control methods. It takes the example of human behavior, which is described in the forms of rule – "If-Then "statements. The logic controller then converts the sensor inputs into fuzzy variables that are then defined according to these rules. Fuzzy logic techniques are used in engine control, automatic transmissions, antiskid steering, etc.

**Soft Computing based Architecture**

An intelligent building takes inputs from the sensors and controls effectors by using them. The construction industry uses the technique of DAI (Distributed Artificial Intelligence) and fuzzy genetic agents to provide the building with capabilities that match human intelligence. The fuzzy logic is used to create behavior-based architecture in intelligent buildings to deal with the unpredictable nature of the environment, and these agents embed sensory information in the buildings

**Soft Computing and Decision Support System**

Soft computing gives an advantage of reducing the cost of the decision support system. The techniques are used to design, maintain, and maximize the value of the decision process.he first application of fuzzy logic is to create a decision system that can predict any sort of risk. The second application is using fuzzy information that selects the areas which need replacement.

**Soft Computing Techniques in Power System Analysis**

Soft computing uses the method of Artificial Neural Network (ANN) to predict any instability in the voltage of the power system. Using the ANN, the pending voltage instability can be predicted. The methods which are deployed here, are very low in cost.

**Soft Computing Techniques in Bioinformatics**

The techniques of soft computing help in modifying any uncertainty and indifference that biometrics data may have. Soft computing is a technique that provides distinct low-cost solutions with the help of algorithms, databases, Fuzzy Sets (FSs), and Artificial Neural Networks (ANNs). These techniques are best suited to give quality results in an efficient way.

**Soft Computing in Investment and Trading**

The data present in the finance field is in opulence and traditional computing is not able to handle and process that kind of data. There are various approaches done through soft computing techniques that help to handle noisy data.

Pattern recognition technique is used to analyze the pattern or behavior of the data and time series is used to predict future trading points.

**Recent developments in soft computing**

People have started using techniques of soft computing like fuzzy sets theory, neural nets, fuzzy neuro system, adaptive neuro-fuzzy inference system (ANFIS), for driving various numerical simulation analysis. Soft computing has helped in modeling the processes of machines with the help of artificial intelligence. Also, there are certain areas where soft computing is in budding stages only and is expected to see a massive evolution:

- Big Data
- Recommender system
- Behavior and decision science
- Mechanical Engineering
- Computer Engineering
- Civil Engineering

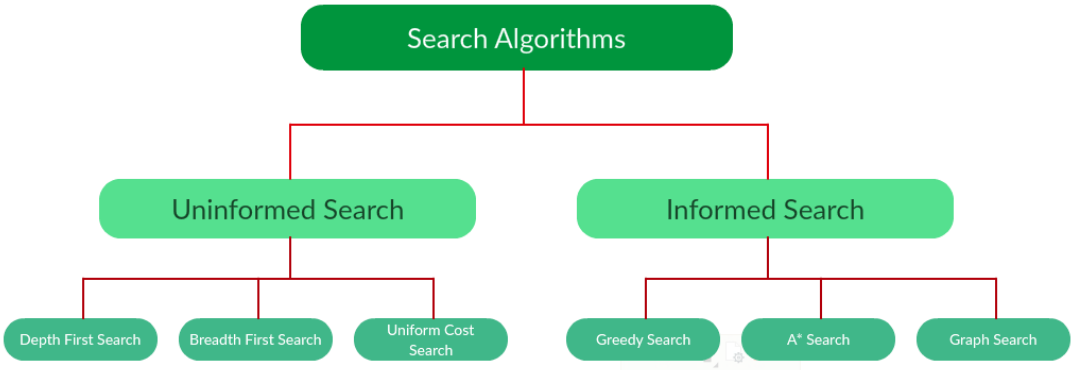
## Artificial Intelligent Searching Techniques

**Artificial Intelligence** is the study of building agents that act rationally. Most of the time, these agents perform some kind of search algorithm in the background in order to achieve their tasks.

- A search problem consists of:
    - **A State Space.** Set of all possible states where you can be.
    - **A Start State.** The state from where the search begins.
    - **A Goal State.** A function that looks at the current state returns whether or not it is the goal state.
- The **Solution** to a search problem is a sequence of actions, called the **plan** that transforms the start state to the goal state.
- This plan is achieved through search algorithms.

### Types of search algorithms:

There are far too many powerful search algorithms out there to fit in a single article. Instead, this article will discuss six of the fundamental search algorithms, divided into two categories, as shown below.




Note that there is much more to search algorithms than the chart I have provided above. However, this article will mostly stick to the above chart, exploring the algorithms given there.

### Uninformed Search Algorithms:

The search algorithms in this section have no additional information on the goal node other than the one provided in the problem definition. The plans to reach the goal state from the start state differ only by the order and/or length of actions. Uninformed search is also called **Blind search**. These algorithms can only generate the successors and differentiate between the goal state and non goal state.
The following uninformed search algorithms are discussed in this section.

1. Depth First Search
2. Breadth First Search
3. Uniform Cost Search

Each of these algorithms will have:

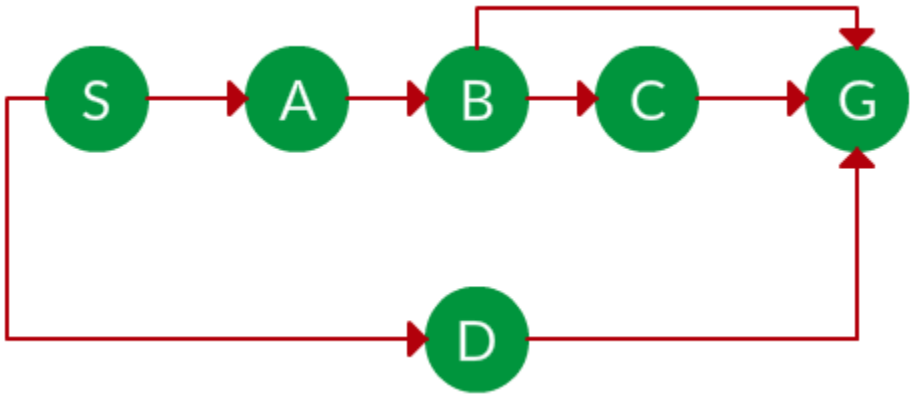
- A problem **graph,** containing the start node S and the goal node G.
- A **strategy,** describing the manner in which the graph will be traversed to get to G.
- A **fringe,** which is a data structure used to store all the possible states (nodes) that you can go from the current states.
- A **tree,** that results while traversing to the goal node.
- A solution **plan,** which the sequence of nodes from S to G.

## Depth First Search:

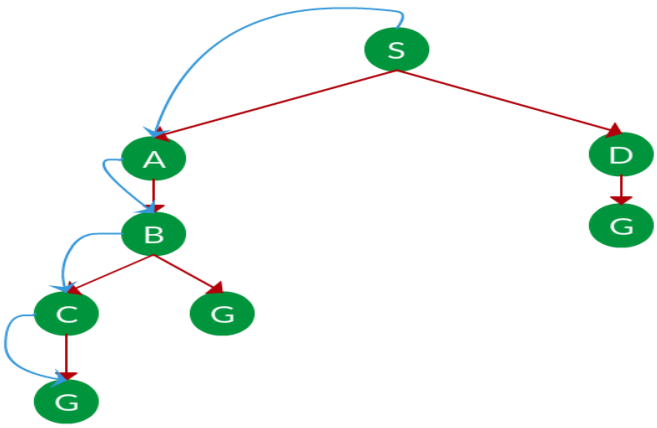
Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. It uses last in- first-out strategy and hence it is implemented using a stack.

**Example:**

**Question.** Which solution would DFS find to move from node S to node G if run on the graph below?




**Solution.** The equivalent search tree for the above graph is as follows. As DFS traverses the tree "deepest node first", it would always pick the deeper branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.

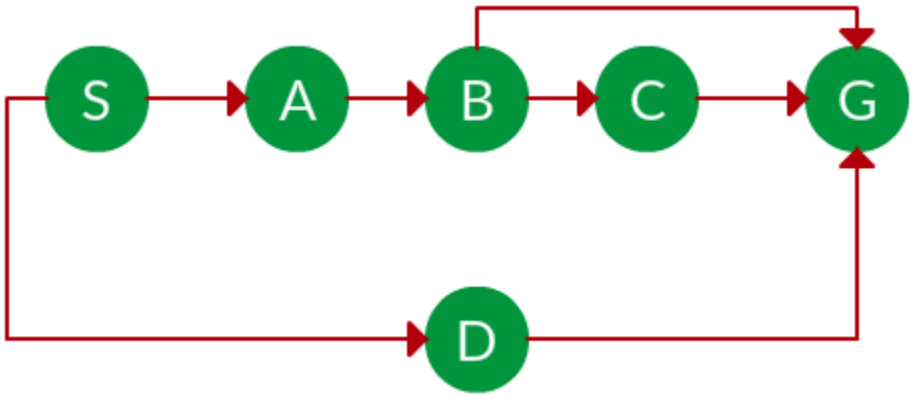



**Path:** S - > A -> B -> C -> G

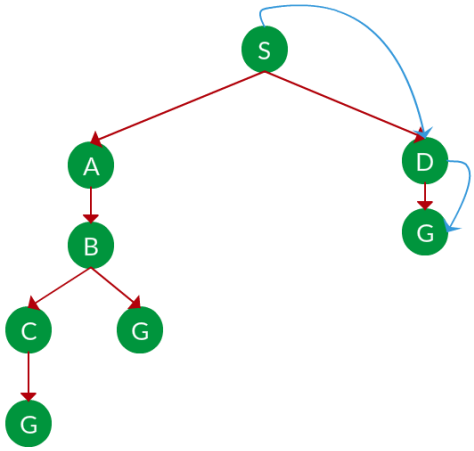
**Breadth First Search:**

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. It is implemented using a queue.

**Example:**
**Question.** Which solution would BFS find to move from node S to node G if run on the graph below?




**Solution.** The equivalent search tree for the above graph is as follows. As BFS traverses the tree "shallowest node first", it would always pick the shallower branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.

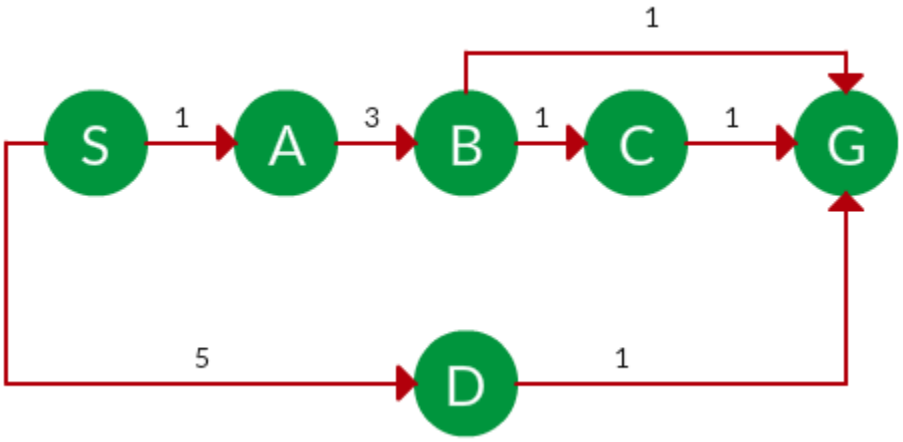



**Path:** S -> D -> G

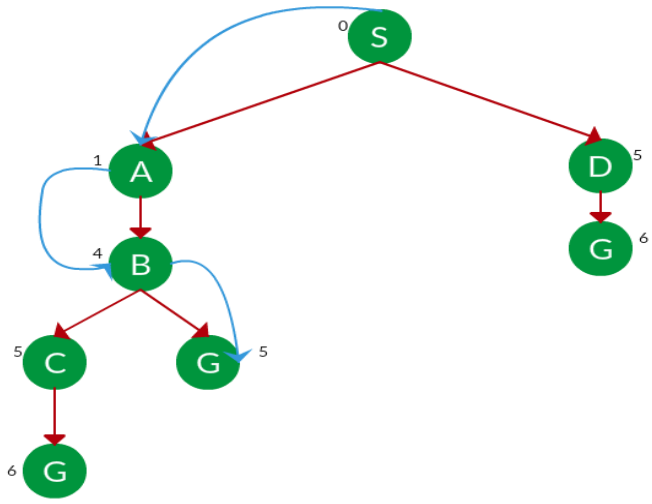
## Uniform Cost Search:

UCS is different from BFS and DFS because here the costs come into play. In other words, traversing via different edges might not have the same cost. The goal is to find a path where the cumulative sum of costs is the least.

**Example:**
**Question.** Which solution would UCS find to move from node S to node G if run on the graph below?




**Solution.** The equivalent search tree for the above graph is as follows. The cost of each node is the cumulative cost of reaching that node from the root. Based on the UCS strategy, the path with the least cumulative cost is chosen. Note that due to the many options in the fringe, the algorithm explores most of them so long as their cost is low, and discards them when a lower-cost path is found; these discarded traversals are not shown below. The actual traversal is shown in blue.



**Path:** S -> A -> B -> G
**Cost:** 5

**Advantages:**
- UCS is complete only if states are finite and there should be no loop with zero weight.
- UCS is optimal only if there is no negative cost.

**Disadvantages:**
- Explores options in every "direction".
- No information on goal location.

## Informed Search Algorithms:

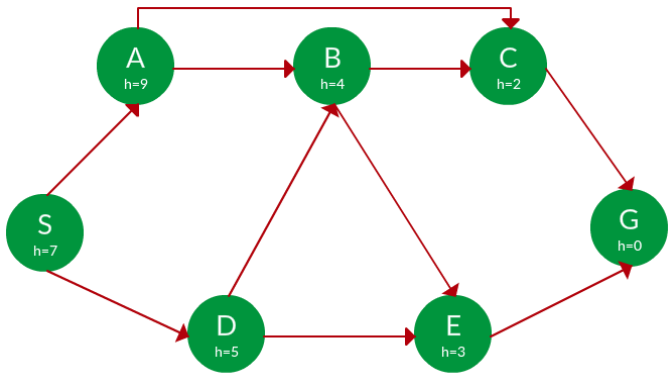
Here, the algorithms have information on the goal state, which helps in more efficient searching. This information is obtained by something called a heuristic.
In this section, we will discuss the following search algorithms.
1. Greedy Search
2. A* Tree Search

**Search Heuristics:** In an informed search, a heuristic is a function that estimates how close a state is to the goal state. For example – Manhattan distance, Euclidean distance, etc. (Lesser the distance, closer the goal.) Different heuristics are used in different informed algorithms discussed below.

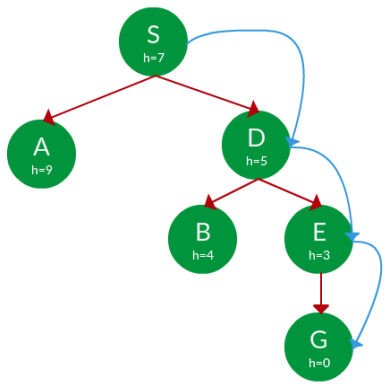
## Greedy Search:

In greedy search, we expand the node closest to the goal node. The "closeness" is estimated by a heuristic h(x).

**Heuristic:** A heuristic h is defined as-
h(x) = Estimate of distance of node x from the goal node.
Lower the value of h(x), closer is the node from the goal.
**Strategy:** Expand the node closest to the goal state, i.e. expand the node with a lower h value.
**Example:**
**Question.** Find the path from S to G using greedy search. The heuristic values h of each node below the name of the node.



**Solution.** Starting from S, we can traverse to A(h=9) or D(h=5). We choose D, as it has the lower heuristic cost. Now from D, we can move to B(h=4) or E(h=3). We choose E with a lower heuristic cost. Finally, from E, we go to G(h=0). This entire traversal is shown in the search tree below, in blue.




**Path:** S - > D -> E -> G
**Advantage:** Works well with informed search problems, with fewer steps to reach a goal.
**Disadvantage:** Can turn into unguided DFS in the worst case.

## A* Tree Search:

A* Tree Search, or simply known as A* Search, combines the strengths of uniform-cost search and greedy search. In this search, the heuristic is the summation of the cost in UCS, denoted by g(x), and the cost in the greedy search, denoted by h(x). The summed cost is denoted by f(x).

**Heuristic:** The following points should be noted wrt heuristics in A*

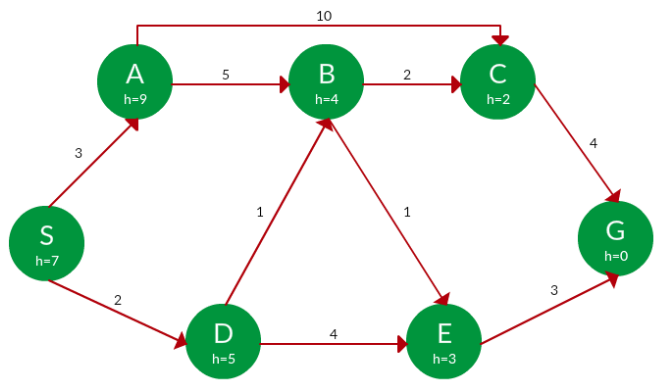
search.

- Here, h(x) is called the **forward cost** and is an estimate of the distance of the current node from the goal node.
- And, g(x) is called the **backward cost** and is the cumulative cost of a node from the root node.
- A* search is optimal only when for all nodes, the forward cost for a node h(x) underestimates the actual cost h*(x) to reach the goal. This property of A* heuristic is called **admissibility**.

  - **Strategy:** Choose the node with the lowest f(x) value.

$$\text{Admissibility:} \quad 0 \leqslant h(x) \leqslant h^*(x)$$

**Strategy:** Choose the node with the lowest f(x) value.
**Example:**
**Question.** Find the path to reach from S to G using A* search.




**Solution.** Starting from S, the algorithm computes g(x) + h(x) for all nodes in the fringe at each step, choosing the node with the lowest sum. The entire work is shown in the table below.
Note that in the fourth set of iterations, we get two paths with equal summed cost f(x), so we expand them both in the next set. The path with a lower cost on further expansion is the chosen path.

| Path | h(x) | g(x) | f(x) |
|---|---|---|---|
| S | 7 | 0 | 7 |
| | | | |
| S -> A | 9 | 3 | 12 |
| S -> D | 5 | 2 | 7 |
| | | | |
| S -> D -> B | 4 | 2 + 1 = 3 | 7 |
| S -> D -> E | 3 | 2 + 4 = 6 | 9 |
| | | | |
| S -> D -> B -> C | 2 | 3 + 2 = 5 | 7 |
| S -> D -> B -> E | 3 | 3 + 1 = 4 | 7 |
| | | | |
| S -> D -> B -> C -> G | 0 | 5 + 4 = 9 | 9 |
| S -> D -> B -> E -> G | 0 | 4 + 3 = 7 | 7 |

**Path:** S -> D -> B -> E -> G
**Cost:** 7

# First-Order Logic in Artificial intelligence or Predicate Calculus- Inferencing

In the topic of 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. PL 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 PL logic.

- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order logic:

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.
- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
    - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, .....
    - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
    - **Function:** Father of, best friend, third inning of, end of, .....

- As a natural language, first-order logic also has two main parts:
    a. **Syntax**
    b. **Semantics**

Syntax of First-Order logic:

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

| | |
|---|---|
| **Constant** | 1, 2, A, John, Mumbai, cat,.... |
| **Variables** | x, y, z, a, b,.... |
| **Predicates** | Brother, Father, >,.... |
| **Function** | sqrt, LeftLegOf, .... |
| **Connectives** | ∧, ∨, ¬, ⇒, ⇔ |
| **Equality** | == |
| **Quantifier** | ∀, ∃ |

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ....., term n)**.

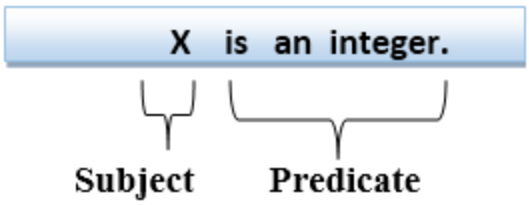
**Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay). Chinky is a cat: => cat (Chinky)**.

Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.

**First-order logic statements can be divided into two parts:**

- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
    a. **Universal Quantifier, (for all, everyone, everything)**
    b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

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 ∀, which resembles an inverted A.

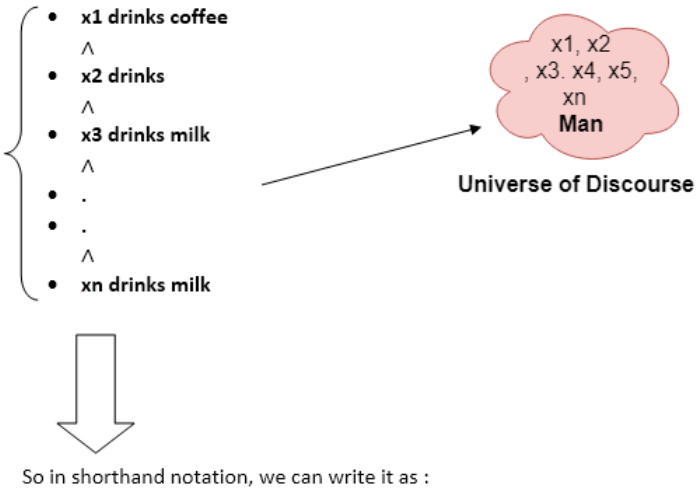
If x is a variable, then ∀x is read as:

- **For all x**
- **For each x**
- **For every x.**

Example:

**All man drink coffee.**

Let a variable x which refers to a cat so all x can be represented in UOD as below:




**∀x man(x) → drink (x, coffee).**

It will be read as: There are all x where x is a man who drink coffee.

Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

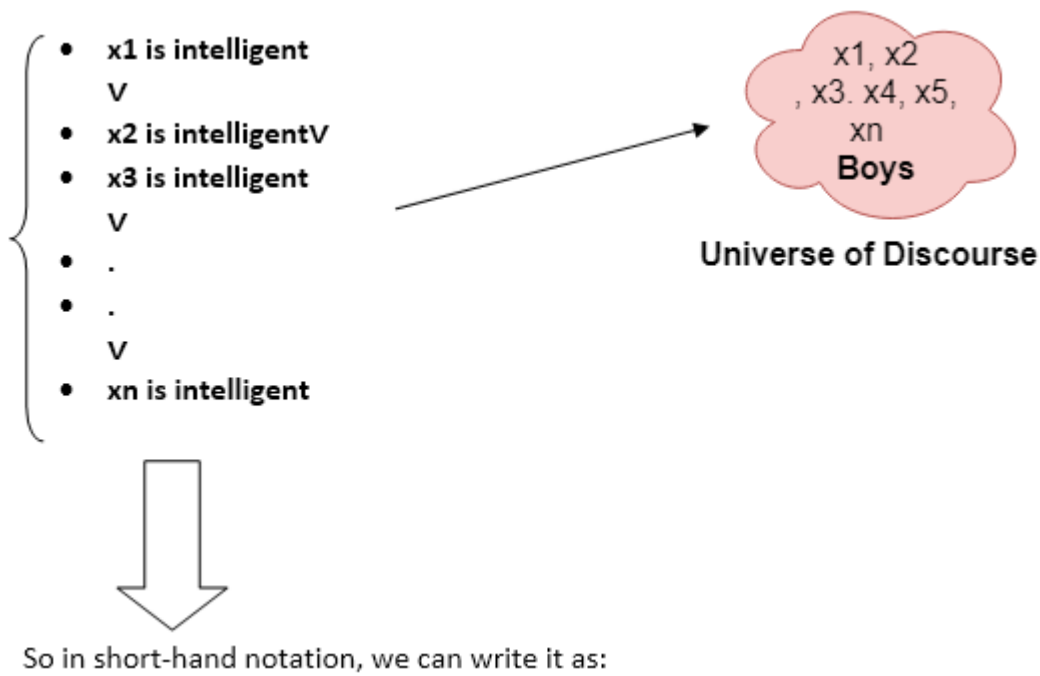
It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

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.'**

Example:

**Some boys are intelligent.**




**∃x: boys(x) ∧ intelligent(x)**

It will be read as: There are some x where x is a boy who is intelligent.

Points to remember:

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

Properties of Quantifiers:

- 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.

Some Examples of FOL using quantifier:

**1. All birds fly.**
In this question the predicate is "**fly(bird)**."
And since there are all birds who fly so it will be represented as follows.

**∀x bird(x) →fly(x)**.

**2. Every man respects his parent.**
In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.
Since there is every man so will use ∀, and it will be represented as follows:

**∀x man(x) → respects (x, parent)**.

**3. Some boys play cricket.**
In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use **∃, and it will be represented as**:

**∃x boys(x) → play(x, cricket)**.

**4. Not all students like both Mathematics and Science.**
In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.
Since there are not all students, so we will use **∀ with negation, so** following representation for this:

**¬∀ (x) [ student(x) → like(x, Mathematics) ∧ like(x, Science)].**

**5. Only one student failed in Mathematics.**
In this question, the predicate is "**failed(x, y)," where x= student, and y= subject**.
Since there is only one student who failed in Mathematics, so we will use following representation for this:

**∃(x) [ student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)]**.

Free and Bound Variables:

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

**Free Variable:** A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

**Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.**

**Bound Variable:** A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

**Example: ∀x [A (x) B( y)], here x and y are the bound variables.**

## Semantic Network Representation

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

This representation consist of mainly two types of relations:

a. IS-A relation (Inheritance)

b. Kind-of-relation

**Example:** Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

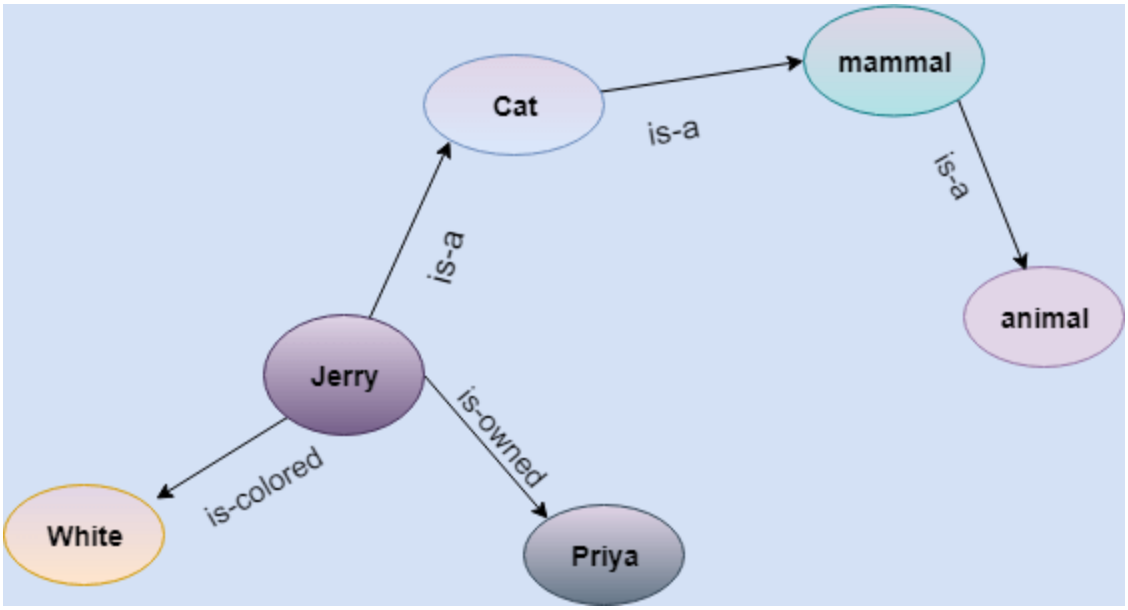
a. Jerry is a cat.

b. Jerry is a mammal

c. Jerry is owned by Priya.

d. Jerry is brown colored.

e. All Mammals are animal.




In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network:

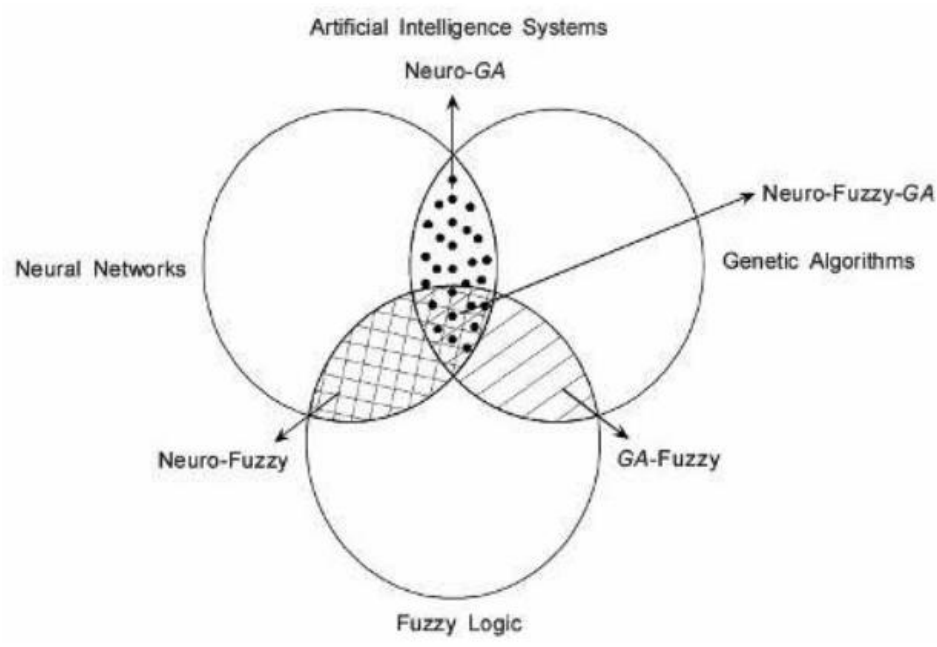
1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

## Hybrid Models of Soft Computing

According to Zadeh, soft computing differs from hard computing ( conventional computing) in its tolerance to imprecision, uncertainty and partial truth. In effect, the role model is the human mind. Hard computing methods are predominantly based on mathematical approaches and therefore demand a high degree of precision and accuracy in their requirements. But in most engineering problems, the input parameters cannot be determined with a high degree of precision and therefore, the best estimates of the parameters are used for obtaining solution to problems. This has restricted the use of mathematical approaches for the solution of inverse problems when compared to forward problems.

On the other hand, soft computing techniques, which have drawn their inherent characteristics from biological systems, present effective methods for the solution of even difficult inverse problems. The guiding principle of soft computing is exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low cost solution… .

Hybrid intelligence systems deal with the synergistic integration of two or more of the technologies. The combined use of technologies has resulted in effective problem solving in comparison with each technology used individually and exclusively.



Hybrid intelligence systems deal with the synergistic integration of two or more of the technologies. The combined use of technologies has resulted in effective problem solving in comparison with each technology used individually and exclusively. The focus on three technologies, namely Neural Networks (NN), Fuzzy Logic (FL) and Genetic Algorithms (GA) and their hybrid combinations. As illustrated in the above Figure, each of these technologies individually and in combination can be employed to solve problems. The combinations include neuro-fuzzy, GA-fuzzy, neuro-GA, and neuro-fuzzy-GA technologies.