SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY:PUTTUR
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE
(20CS0901)

COURSE OBJECTIVES

The objective of the course is to

1. To have a basic proficiency in a traditional AI language including an ability to write simple to intermediate programs and an ability to understand code written in that language
2. To have an understanding of the basic issues of knowledge representation and blind and heuristic search, as well as an understanding of other topics such as minimax, resolution, etc. that play an important role in AI programs
3. To have a basic understanding of some of the more advanced topics of AI such as learning, natural language processing, agents and robotics, expert systems, and planning

COURSE OUTCOMES

On successful completion of the course, the students will be able to

1. Understand the foundation and current trends of AI.
2. Outline problems that are amenable to solution by AI methods, and which AI methods may be suited to solving a given problem
3. Apply the language/framework of different AI methods for a given problem
4. Understand and implement the basic and advanced knowledge representation techniques.
5. Implement basic AI algorithms- standard search algorithms or dynamic programming
6. Design and carry out an empirical evaluation of different algorithms on problem formalization, and state the conclusions that the evaluation supports

UNIT I

Introduction:

History, intelligent systems, foundations of AI, applications, tic-tac-toe game playing, development of AI languages, current trends.

Introduction to AI:

Artificial Intelligence (AI) is a branch of computer science that focuses on creating intelligent machines capable of performing tasks that typically require human intelligence.

These tasks include understanding natural language, learning, reasoning, problem-solving, and decision-making.

History of AI:

The concept of artificial intelligence has its roots in ancient times, with myths and stories about humanoid creatures coming to life. However, the formal study of AI began in the 1950s. The field has gone through several stages of development, including:

- Early Foundations (1950s-1960s): The birth of AI as a field of study can be traced back to the Dartmouth Conference in 1956, where researchers discussed the possibility of creating machines that can simulate human intelligence.
- Knowledge-Based Systems (1970s-1980s): During this period, AI focused on knowledge representation and expert systems. Researchers developed techniques to store and manipulate vast amounts of information to solve specific problems.

- Neural Networks and Machine Learning (1980s-1990s): Neural networks, inspired by the structure of the human brain, gained popularity during this period. Machine learning algorithms were developed to enable computers to learn from data and improve their performance over time.
- Knowledge Engineering and Expert Systems (1990s-2000s): Expert systems continued to be a significant area of research, with AI being applied in domains such as medicine, finance, and engineering. However, limitations in knowledge representation and reasoning led to a decline in interest in expert systems.
- Big Data and Deep Learning (2010s-present): The rise of big data and advancements in computing power enabled the training of deep neural networks, leading to breakthroughs in areas such as image recognition, natural language processing, and game playing.

Foundations of AI:

The foundations of AI include various disciplines such as mathematics, logic, cognitive science, and computer science. Key areas of study within AI include:

1. Machine Learning: This field focuses on developing algorithms that enable computers to learn patterns from data.
2. Natural Language Processing (NLP): NLP involves teaching computers to understand and generate human language.
3. Computer Vision: Computer vision deals with enabling computers to understand and interpret visual information from images or videos.
4. Robotics: Robotics combines AI with mechanical engineering to create intelligent machines capable of performing physical tasks in the real world. It encompasses areas such as autonomous navigation, manipulation, and human-robot interaction.

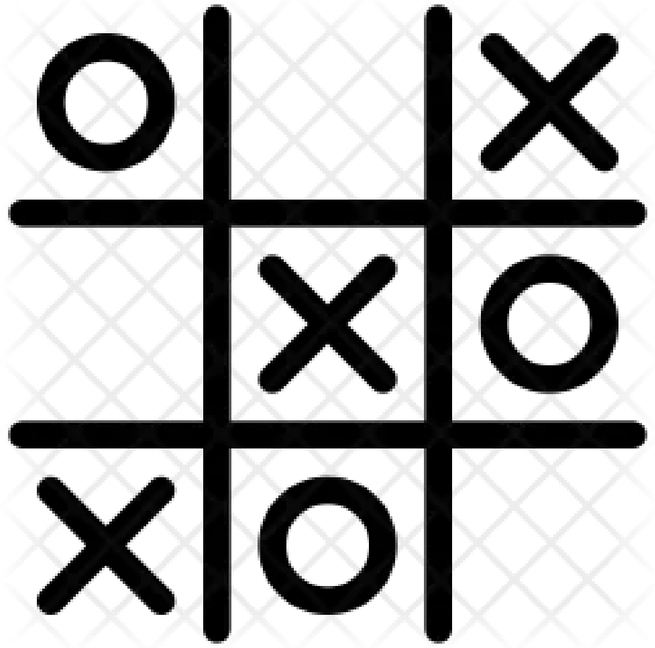
Applications of AI:

AI has found applications in various industries and domains, including:

1. Healthcare: AI is used in medical diagnosis, drug discovery, personalized medicine, and remote patient monitoring.
2. Finance: AI algorithms are employed in areas such as fraud detection, algorithmic trading, and risk assessment.
3. Transportation: Self-driving cars and intelligent traffic management systems rely on AI technologies.
4. E-commerce: AI is used for personalized recommendations, chatbots, and customer service automation.

Tic-Tac-Toe

Tic-Tac-Toe is a simple game for two players that we enjoyed playing as kids (especially in boring classrooms). The game involves 2 players placing their respective symbols in a 3x3 grid.



Tic-Tac-Toe Learning System

The basic idea behind the learning system is that the system should be able to improve its performance (P) w.r.t to a set of tasks (T) by learning from training experience (E). The training experience (E) can be a direct (predefined set of data with individual labels) or indirect feedback (No labels for each training example). In our case:-

- Task (T): Playing Tic-Tac-Toe
- Performance (P): Percentage of Games won against Humans
- Experience (E): Indirect feedback via solution trace (Game History) generated from games played against itself (a clone)

Development of AI Languages:

AI has its own set of programming languages and frameworks designed to facilitate the development and implementation of AI algorithms. Some popular AI languages include:

1. Python: Python is widely used in the AI community due to its simplicity, versatility, and extensive libraries for machine learning and data analysis.
2. R: R is a language specifically designed for statistical computing and graphics. It is commonly used in data analysis and visualization tasks.
3. Java: Java is a popular general-purpose programming language that is also used in AI development, particularly for building enterprise-level AI systems.

Current Trends in AI:

Some of the current trends in AI include:

1. Deep Learning: Deep learning, a subset of machine learning, is achieving remarkable results in various domains, especially with the advent of large-scale neural networks and increased computational power.
2. Explainable AI: There is a growing demand for AI systems that can explain their decision-making processes, especially in critical domains like healthcare and finance.
3. Reinforcement Learning: Reinforcement learning, which involves training agents to learn from interactions with their environment, is gaining attention due to its potential for training autonomous systems.
4. Edge Computing: AI is being deployed on edge devices, such as smartphones and Internet of Things (IoT) devices, to perform computations locally and reduce reliance on cloud-based services.

UNIT II

Problem solving: state-space search and control strategies: Introduction, general problem solving, characteristics of problem, exhaustive searches, heuristic search techniques, iterative deepening A*, constraint satisfaction. Problem reduction and game playing: Introduction, problem reduction, game playing, alpha beta pruning, two-player perfect information games

Problem Solving:

Problem solving is a fundamental aspect of AI, and various techniques are employed to find solutions to complex problems. Let's explore some key concepts and strategies related to problem solving in AI.

State-Space Search:

State-space search is a common approach used in AI to solve problems. It involves representing a problem as a search problem, where the problem's possible states form a graph or tree-like structure. The search algorithm navigates through this structure to find a solution.

General Problem-Solving Approach:

1. Problem Understanding: Gain a clear understanding of the problem by defining the goals, constraints, and available resources.
2. Problem Representation: Represent the problem in a formal or structured manner that captures such as the current state, actions, goal state.
3. Search and Exploration: Apply search algorithms or systematic exploration strategies to traverse the problem space and find a solution.
4. Solution Evaluation: Evaluate the generated solutions based on predefined criteria, such as optimality, feasibility, or efficiency.
5. Solution Refinement: Refine and improve the initial solution through iteration, feedback, and adaptation.
6. Solution Implementation: Translate the refined solution into an executable form.
7. Solution Evaluation: Test and evaluate the implemented solution in real-world
8. Iteration and Learning: Learn from the problem-solving process and outcomes to improve future problem-solving abilities.

Control Strategies: Control strategies determine the order in which states are explored during a search. Some common control strategies include:

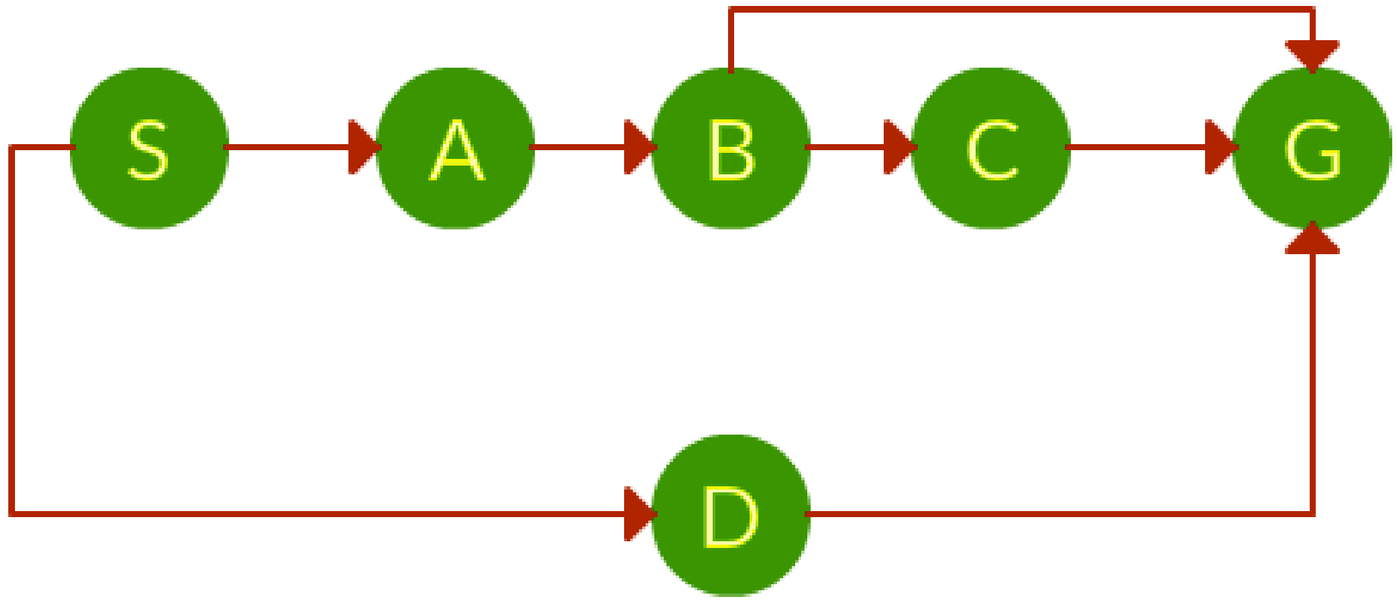
1. **Breadth-First Search (BFS):** BFS explores all the nodes at the current depth level before moving to the next depth level. It guarantees finding the shallowest goal state but may be memory-intensive.
2. **Depth-First Search (DFS):** DFS explores as far as possible along each branch before backtracking. It can quickly reach deep states but may get stuck in infinite loops or miss shallow goal states.
3. Iterative Deepening: Iterative deepening combines the benefits of BFS and DFS. It performs DFS with increasing depth limits until a solution is found.

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.

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

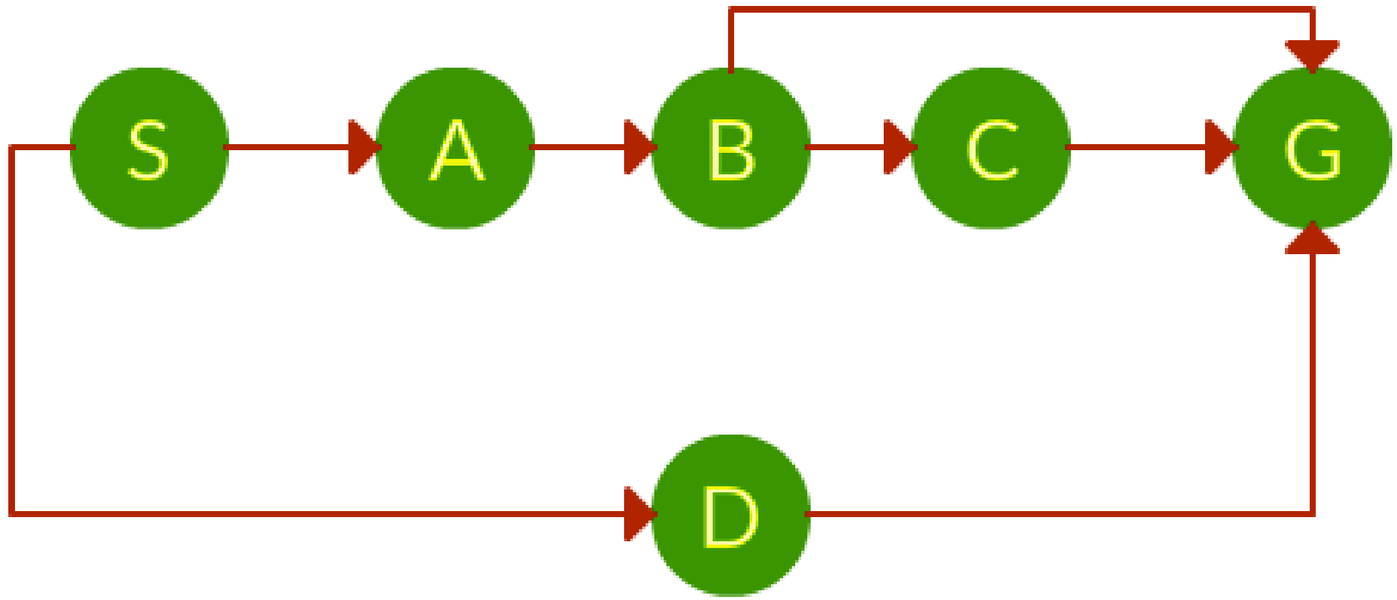
Path: S -> D -> G



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

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



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

Characteristics of Problems:

Problems can have various characteristics that affect the choice of search algorithms and strategies. Some important characteristics include:

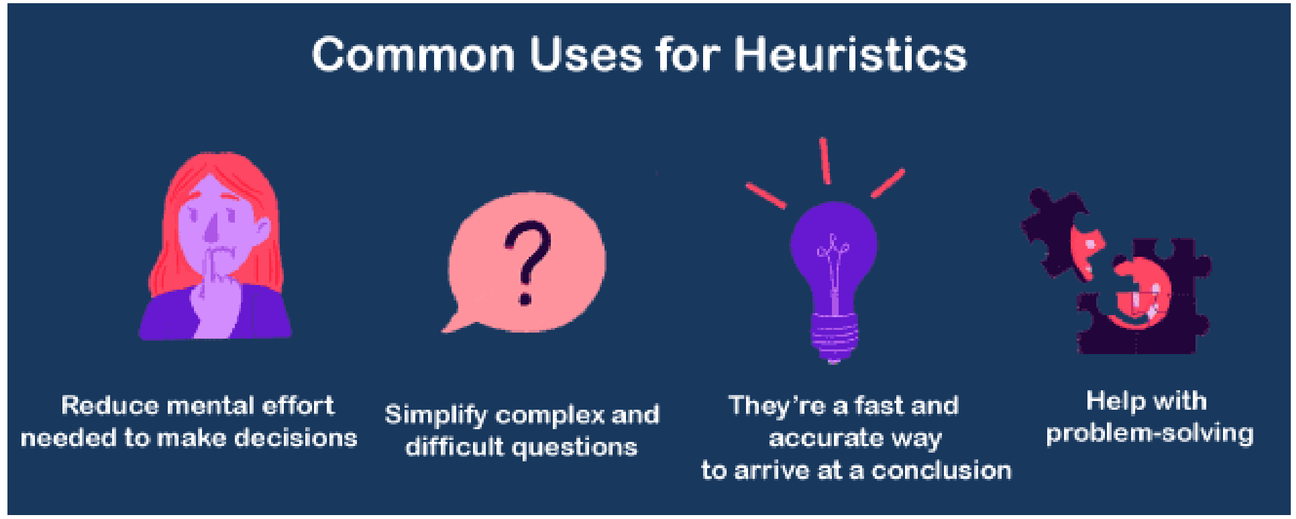
1. Single-State vs. Multi-State: Some problems require finding a single solution state, while others involve finding multiple states or optimal solutions.
2. Deterministic vs. Stochastic: In deterministic problems, the outcome of each action is known, while stochastic problems have uncertain outcomes.
3. Static vs. Dynamic: Static problems do not change over time, whereas dynamic problems have changing states and require planning over time.

What is Heuristics?

A heuristic is a technique that is used to solve a problem faster than the classic methods. These techniques are used to find the approximate solution of a problem when classical methods do not. Heuristics are said to be the problem-solving techniques that result in practical and quick solutions.

History

Psychologists **Daniel Kahneman** and **Amos Tversky** have developed the study of Heuristics in human decision-making in the 1970s and 1980s. However, this concept was first introduced bythe **Nobel Laureate Herbert A. Simon,** whose primary object of research was problem-solving.



Heuristic Search Techniques:

Heuristic search techniques incorporate domain-specific knowledge to guide the search towards promising states. They estimate the desirability of states based on heuristics or rules of thumb. Some commonly used heuristic search algorithms include:

1. **A* Search:** A* search combines the cost of reaching a state from the initial state (known as the "g" value) and a heuristic estimate of the remaining cost to reach the goal (known as the "h" value). It uses this information to prioritize states with lower total costs (f = g + h).
2. **Greedy Best-First Search:** Greedy best-first search prioritizes states based solely on the heuristic estimate, without considering the cost of reaching the current state. It tends to be fast but may not always find the optimal solution.

Constraint Satisfaction: Constraint satisfaction problems (CSPs) involve finding values for variables that satisfy a set of constraints. CSPs have applications in various domains, such as scheduling, planning, and resource allocation. Constraint satisfaction techniques use algorithms to efficiently search for valid assignments to variables.

Problem Reduction and Game Playing: Problem reduction involves transforming a problem into a more manageable form or mapping it to a known problem. This approach simplifies complex problems by breaking them down into smaller, solvable sub-problems.

Game playing is a popular area of AI research, where algorithms are developed to play games against human opponents or other AI agents. Two-player perfect information games, such as chess or tic-tac-toe, are common targets for AI game-playing algorithms.

Alpha-Beta Pruning

Alpha-Beta pruning is not actually a new algorithm, but rather an optimization technique for the minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

Let’s define the parameters alpha and beta.

Alpha is the best value that the maximizer currently can guarantee at that level or above.

Beta is the best value that the minimizer currently can guarantee at that level or below.

Two-player perfect information games.

Two-player perfect information games are a specific class of games in which two players take turns making moves, and both players have complete and perfect knowledge of the game state at any given point.
In these games, the rules are well-defined, and the outcomes of moves are determined solely by the current game state and the players' actions.
Each player knows the complete history of the game and can make informed decisions based on the available information.
Examples of two-player perfect information games include classic board games like chess, checkers, tic-tac-toe, Othello, and Go. In these games, the board configuration represents the game state, and each player can observe the positions and moves made by their opponent.

UNIT III

Logic concepts: Introduction, propositional calculus, proportional logic, natural deduction system, axiomatic system, semantic tableau system in proportional logic, resolution refutation in proportional logic, predicate logic.

Introduction:

Provide an overview of the importance of logic in artificial intelligence and problem-solving.

Explain how logic serves as a foundation for reasoning and decision-making.

Propositional Calculus:

Propositional calculus, also known as propositional logic, deals with propositions and their logical relationships.

Proposals are atomic statements that can be true or false.

Logical connectives such as AND, OR, NOT, IMPLIES, and IF AND ONLY IF are used to create compound propositions.

Truth tables are used to determine the truth value of compound propositions.

Propositional logic in Artificial intelligence

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

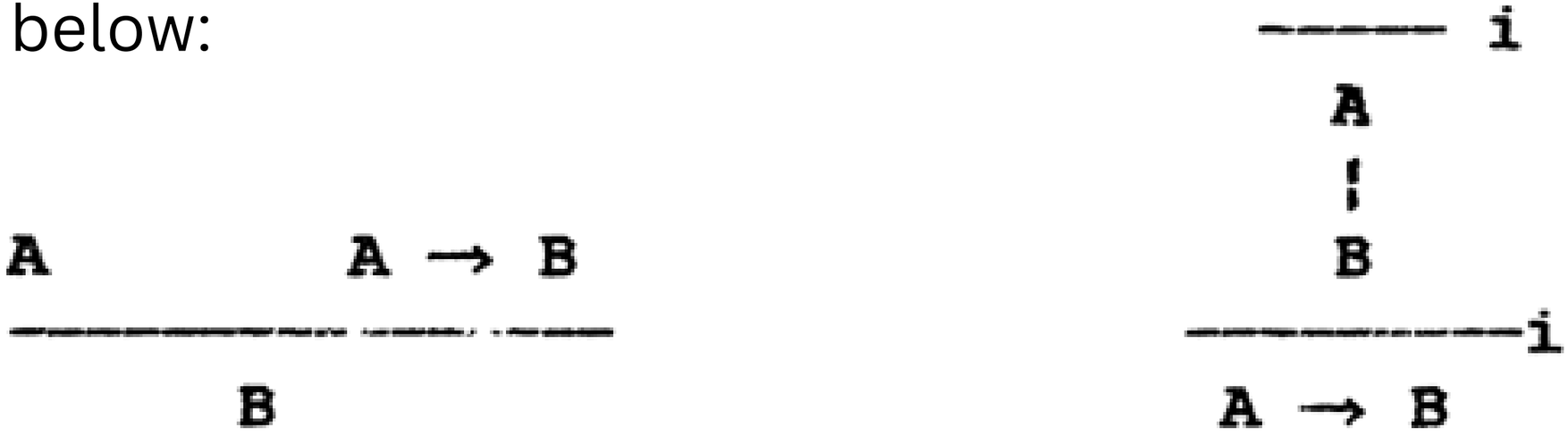
Example:

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)
4. d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic consists of an object, relations or function, and logical connectives.
- These connectives are also called logical operators.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.
- Statements which are questions, commands, or opinions are not propositions such as "**Where is Rohini**", "**How are you**", "**What is your name**", are not propositions.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.

Natural deduction is a formal inference system which is said naturally to mirror the way in which humans reason. A natural deduction system consists of rules of inference eliminating and introducing each of the connectives and quantifiers of the predicate calculus. There are twelve rules which may be used to infer conclusions. Two examples of such rules are given below:



The rule on the left is that to eliminate the → connective and that on the right to introduce it.

Axiomatic System:

An axiomatic system is a formal system based on a set of axioms and rules of inference.

Axioms are fundamental statements or principles accepted as true without proof.

The rules of inference dictate how new statements can be derived from existing.

Axiomatic systems provide a foundation for logical reasoning and proof construction.

Axiom: $A \rightarrow (B \rightarrow A)$ (Law of Implication)

Rule of Inference: Modus Ponens

If A is true and $A \rightarrow B$ is true, then B is true.

Using the axiom and the rule of inference, we can derive the following:

Given A is true

Using the axiom $A \rightarrow (B \rightarrow A)$, we can infer $(B \rightarrow A)$

Using Modus Ponens, we can conclude that B is true

Semantic Tableau System in Proportional Logic:

Semantic tableau is a proof procedure used to determine the satisfiability of a set of proportional logic formulas.

It uses a tree-like structure to systematically analyze the truth values of the formulas.

By expanding and splitting branches, the procedure determines whether a set of formulas is satisfiable or not.

$P \rightarrow (Q \vee R)$

$\neg P \vee (S \wedge T)$

$\neg Q \rightarrow (U \vee V)$

To determine if this set of formulas is satisfiable, we can use the semantic tableau method. We start by negating the conjunction of all the formulas and construct a tree-like structure. Then, we apply the rules to expand and split branches. If all branches close (result in contradictions), the set of formulas is unsatisfiable. If at least one branch remains open, it is satisfiable.

Resolution Refutation in Proportional Logic:

Resolution refutation is a proof technique used to prove the unsatisfiability of a set of proportional logic formulas.

It involves transforming the formulas into clauses and applying resolution inference to derive the empty clause.

If the empty clause is derived, it indicates that the original set of formulas is unsatisfiable.

Here's an example:

Consider the following set of formulas:

$P \vee Q$

$\neg P \vee R$

$\neg R \vee S$

$\neg S$

To prove the unsatisfiability, we convert the formulas into clauses and apply the resolution rule:

Clauses: {P, Q}, {¬P, R}, {¬R, S}, {¬S}

By applying resolution, we can derive the empty clause, indicating that the set of formulas is unsatisfiable.

Predicate Logic:

Predicate logic extends propositional logic and proportional logic by introducing predicates, variables, and quantifiers.

Predicates express properties or relationships, and variables represent objects or elements.

Universal and existential quantifiers are used to express statements about all or some objects.

Predicate logic allows for more expressive and precise representation and reasoning about the world.

1. Predicate Logic: Predicate logic extends propositional logic by introducing predicates, variables, and quantifiers.

Here's an example:

Consider the following statements:

Predicate P(x): "x is a prime number."

Predicate Q(x, y): "x is less than y."

Universal quantifier (∀): "For all x."

Existential quantifier (∃): "There exists an x."

Example 1:

Statement: ∀x P(x) → Q(x, 10)

This statement says that if every x is a prime number, then x is less than 10.

Example 2:

Statement: ∃x P(x) ∧ Q(x, 5)

This statement says that there exists an x such that x is a prime number and x is less than 5.

UNIT IV

Knowledge representation:

Introduction Approaches to knowledge representation, knowledge representation using semantic network, extended semantic networks for KR, knowledge representation using frames. Advanced knowledge representation techniques: Introduction, conceptual dependency theory, script structure, CYC theory, case grammars, semantic web

Approaches to Knowledge Representation

A good system for the representation of knowledge in a particular domain should possess the following four properties:

Representational Adequacy: the ability to represent all of the kinds of knowledge that are needed in that domain.

Inferential Adequacy: the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.

Inferential efficiency: the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.

Acquisitional Efficiency: the ability to acquire new information easily.

Simple Relational Knowledge

The simplest way to represent declarative facts is as a set of relations of the same sort used in database systems.

The reason that this representation is simple is that standing alone it provides very weak inferential capabilities, but knowledge represented in this form may serve as the input to more powerful inference engines.

The relational knowledge corresponds to a set or attributes and associated values that together describe the objects of the knowledge base.

Providing support for relational knowledge is what database systems are designed to do.

RepresentationsandMappings

• In order to solve the complex problems encountered in artificial intelligence; we need both a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to
new problems.

• Avarietyofwaysofrepresentingknowledge(facts)havebeenexploitedinAlprograms.

We deal with two different kinds of entities:

• Facts:truthsinsomerelevantworld,thesearethethingswewanttorepresent.

• Representationsoffactsinsomechosenformalism.Thesearethethingswewillactuallybeableto manipulate.

These entities are at two levels:

•

Theknowledgelevel,atwhichfacts(includingeachagent'sbehaviour.andcurrentgoals)aredescri

bed.

• Thesymbollevel,atwhichrepresentationsofobjectsattheknowledgelevelaredefinedintermsof symbols that can be manipulated by programs.

Inheritable Knowledge

Knowledge about objects, their attributes, and their values need not be as simple as that shown in relational knowledge.

In particular, it is possible to augment the basic representation with inference mechanisms

that
operate on the structure of the representation.

For this to be effective, the structure must be designed to correspond to the inference mechanisms that are desired.

One of the most useful forms of inference is property inheritance, in which elements of specific classes inherit attributes and values from more general classes in which they are included.

In order to support property inheritance, objects must be organized into classes and classes

must
be arranged in a generalization hierarchy.

Lines represent attributes. Boxed nodes represent objects and values of attributes of objects.

Inferential Knowledge

The power of traditional logic is necessary to describe the inferences that are n eeded.

This knowledge is useless unless there is also an inference procedure that can exploit it The required inference procedure is one that implements the standard logical rules of inference.

The procedures, may reason forward from given facts to conclusions,
or may r eason backwar d fr om desir ed conclusions to given facts.
One of the most commonly used of these procedures is Resolution, which exploits a pr oof by contr adiction str ategy.

Procedural Knowledge

Thiskind of knowledge is operational.Procedural knowledge specifies what to do and when to do.

Procedural knowledge can be repr esented in programs in many ways. The most common way is simply as code (in some
programming language such as LISP) for doing something.
perform a task.

But,thiswayofrepresentingproceduralknowledgegetslowscoresforthefollowing properties:

Inferentialadequacy,becauseitisverydifficulttowriteaprogramthatcanreasonaboutanotherprogramsbehaviour.

Acquisitionalefficiency,becausetheprocessofupdatinganddebugginglargepiecesofcodebecomesunwieldy.

The most commonly used technique for representing proceduralknowledge in
Al Programs is the use of production rules.

Semantic Nets

In a semantic net, information is represented as a set of nodes connected to each other by a set of labeled arcs, which represent relationships among the
nodes.

The main idea behind semantic nets is that the meaning of aconceptcomes from the ways in which it is connected to other concepts

This network contains examples of both the isa and instance relations, and domain-specific relations like team and uniform-color.

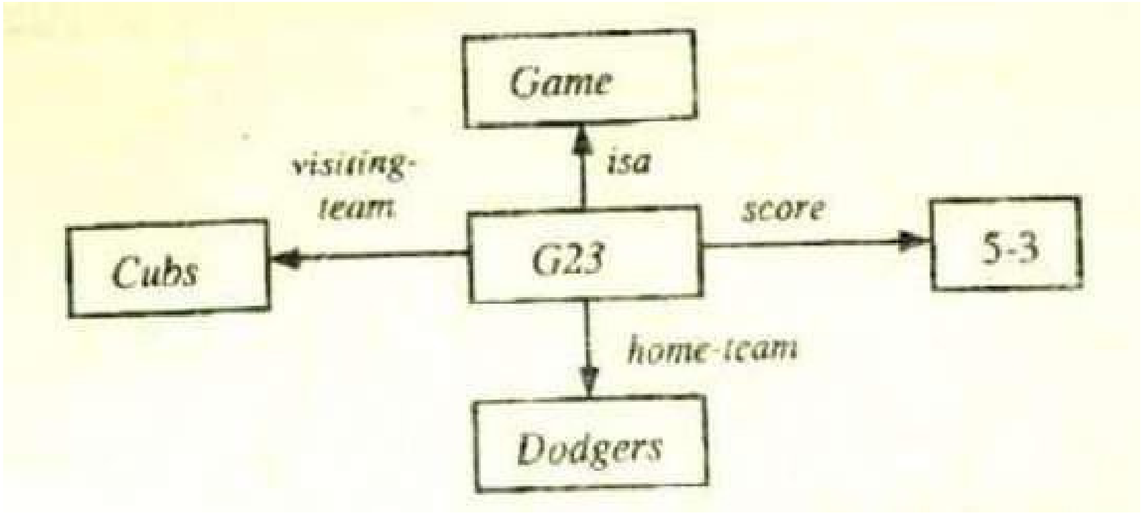
In thisnetwork, we could use inheritance to derive the additional relation:

has-part (Pee- Wee-Reese, Nose)

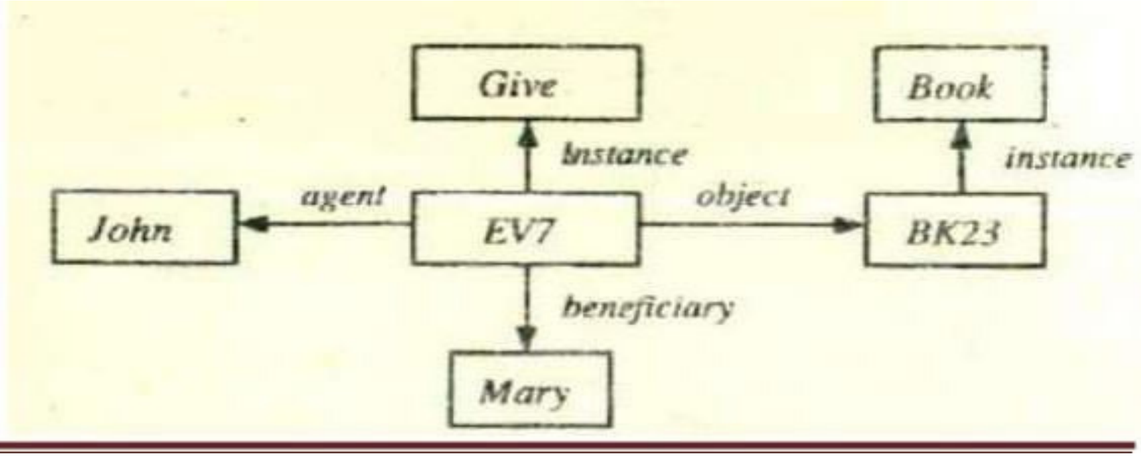
Example for Representing Non binary Predicates:

Can be represented in a semantic net by creating a node to represent the specific game and then relating each of the three pieces of information to it.

- The sentence: John gave the book to Mary could be represented by the network as follows:



John gave the book to Mary



Basic mechanism of inheritance:

Algorithm: Property Inheritance

To retrieve a value V, for attribute A of an instance object O:

1. Find O in the knowledge base.
2. IfthereisavaluethereforttheattributeA,reportthatvalue.
3. Otherwise,seeifthereisavalueforttheattributeinstance.Ifnot,thenfail.
4. Otherwise,movetothenodecorrespondingtothatvalueandlookforavalueforthe attribute A. If one is found, report it.
5. Otherwise,dountilthereisnovalueforttheisaattributeoruntilananswerisfound:

(a) Get the value of the isa attribute and move to that node.

(b) See if there is value for the attribute A. If there is, repori it.

Extended semantic networks for KR

Partitioned Semantic Networks allow for:

Propositions to be made without commitment to truth.Expressions to be quantified.

The basic idea is to br eak networ k into spaces which consist of gr oups of nodes

and ar cs and r egar d

each space as a node.

By partitioning the semantic net into a hierarchical set of spaces, eachpartition corresponds t
to the scope of one or more variables.

Consider the statement: The dog bit the mail car r ier .

The nodes Dogs, Bite, and Mail-Carrier represent the classes of dogs, bitings and mail carriers.

Respectively, while the nodes d, b and m represent a particular dog,

Fr ames

A frame is a collection of attributes (usually called slots) andassociated values (and possibly constraints on values) that describes some entity in the world.

A single frame taken alone is rarely useful. Instead, we build frame systems out of collection of frames that are connected to each other. The value of an attribute of one frame may be another frame.

ConceptualDependency

Semantic networ ks and fr ame systems may have specialized links and infer ence pr ocedur es,
but there are no hard and fast rules about what kinds of objects and links arc good in general for knowledge representation.

Conceptual Dependency specifies what types of objects and relations arepermitted.

Conceptual Dependency (CD) is a theory of how to represent the kind
of knowledge about events that is usually contained in natural language sentences.
The goal is to represent the knowledge in a way that:

Faci l i t at es dr awi n g i nfer en ces fr om t he sen t en ces.

Is independent of the language in which the sentences were originally stated.

The CD representation of a sentence is built not out of primitives

corresponding to the words used in the sentence, but rather out of conceptual primitives

that can be
combined to form the meanings of words in any particular language.

The Dependencies of CD

Rule 1 describes the relationship between an actor and the event heor she causes. This is a two-way dependency since neither actor nor event can be considered primary. The letter p above the dependency link indicates past tense.

Rule 2 describes the relationship between a PP and a PA that isbeing asserted to describe it.

Rule 3 describes the relationship between two PPs, one of whichbelongs to the set defined by the other.

Rule 4 describes the relationship a PP and an attribute that has already been predicated of it.

The di r ecti on of the ar r ow i s tow ar dt he PP bei ng descr i bed.

Rule 5 describes the relationship between two PPs, one of which provides a particular kind of
information about the other.

The set of conceptual tenses in a CD are:

• The set of conceptual tenses in a CD are:
•p Past
•f Future
•t Transition
• txStart transition
• tyFinished transition
• kContinuing
•?Intnil Present
• delta Timeless
• cConditional

Scripts

CD is a mechanism for representing and reasoning about events. A script is a structure that describes a stereotyped sequence of events in a particular con t ext .

Script is a mechanism for representing knowledge about commonsequences of events.

A script consists of a set of slots. Associated with each slot may be some information about what kinds of values it may contain as well asa default value to be used if no other information is available.

The important components of a script are:

Entr y conditions: Conditions that must, be satisfied before theevents described in the script can occur.

Result: Conditions that will, be true after the events described inthe script have occurred. Props: Slots representing objects that are involved in the events described in the script. The
presence of these objects can be inferred even if they are not mentioned explicitly.

Role: Slots representing people who are involved in the events described in the script.

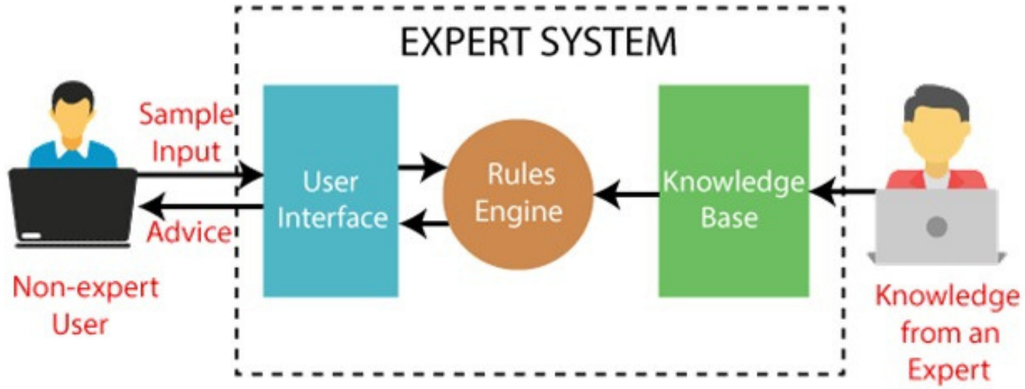
Track: The specific variation on a more general pattern that isrepresented by this particular
script. Different tracks of the same script will share many but not all components.

Scenes: The actual sequences of events that occur.

UNIT V

Expert system and applications: Introduction phases in building expert systems, expert system versus traditional systems Uncertainty measure: probability theory: Introduction, probability theory, Bayesian belief networks, certainty factor theory, dempster-shafertheory

Expert system and applications:



Introduction phases in building expert systems

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using**both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as**medicine, science,**etc.

Expert system versus traditional system

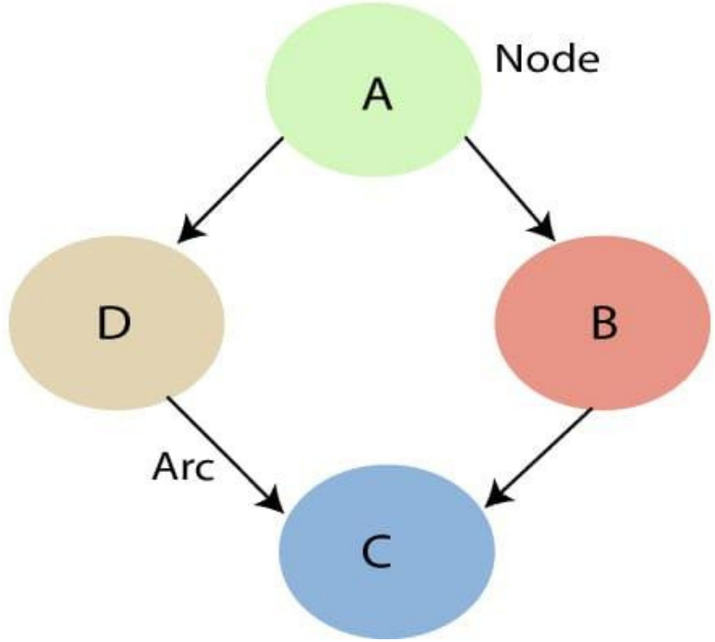
A key distinction between the traditional system as opposed to the expert system is the way in which the problem related expertise is coded. Essentially, in conventional applications, the problem expertise is encoded in both program as well as data structures. On the other hand, in expert systems, the approach of the problem related expertise is encoded in data structures only. Moreover, the use of knowledge in expert systems is vital. However, traditional systems use data more efficiently than the expert system.

One of the biggest limitations of conventional systems is that they are not capable of providing explanations for the conclusion of a problem. That is because these systems try to solve problems in a straightforward manner. However, expert systems are capable of not only providing explanations but also simplifying the understanding of a particular conclusion.

Bayesianbelief networks

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.



certainty factor theory

The Certainty Factor (CF) is a numeric value which tells us about how likely an event or a statement is supposed to be true. It is somewhat similar to what we define in probability, but the difference in it is that an agent after finding the probability of any event to occur cannot decide what to do. Based on the probability and other knowledge that the agent has, this certainty factor is decided through which the agent can decide whether to declare the statement true or false.

Dempster-shafertheory

DST is an evidence theory, it combines all possible outcomes of the problem. Hence it is used to solve problems where there may be a chance that a piece of different evidence will lead to some different result.