



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L1

CSE

III/VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE
Course Faculty : G.SUMATHI
Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Constraint Satisfaction problem- Related algorithms- Measure of performance and analysis of algorithms

Introduction:

- Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem.
- Performance metrics to analyze the optimum problem solving strategy

Prerequisite knowledge for Complete understanding and learning of Topic:

- Search Strategies
- Basic Mathematics
- Time Complexity
- Space Complexity

Detailed content of the Lecture:

Artificial Intelligence (AI) is a branch of Science which deals with helping machines to find solutions to complex problems in a more human-like fashion.

Branches of Artificial intelligence:

- **Games playing:** programming computers to play games such as chess and checkers.
- **Expert systems:** programming computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms).
- **Natural Language Processing:** programming computers to understand natural human languages.
- **Neural Networks:** Systems that simulate intelligence by attempting to reproduce the types of physical connections that occur in human brains.
- **Robotics:** programming computers to see and hear and react to other sensory stimuli currently, no computers exhibit full artificial intelligence (that is, are able to simulate human behavior).

Applications of AI:

- **Game playing**
There is some AI in them, but they play well against people mainly through brute force computation--looking at hundreds of thousands of positions. To beat a world champion by brute force and known reliable heuristics requires being able to look at 200 million positions per second.
- **Speech recognition**
In the 1990s, computer speech recognition reached a practical level for limited purposes. Thus United Airlines has replaced its keyboard tree for flight information by a system using speech recognition of flight numbers and city names. It is quite convenient.
- **Understanding natural language**
Just getting a sequence of words into a computer is not enough. Parsing sentences is not enough either. 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.
- **Computer vision**
The world is composed of three-dimensional objects, but the inputs to the human eye and computers' TV cameras are two dimensional. Some useful programs can work solely in two dimensions, but full computer vision requires partial three-dimensional information that is not just a set of two-dimensional views. At present there are only limited ways of representing three-dimensional information directly, and they are not as good as what humans evidently use.
- **Expert systems**
A "knowledge engineer" interviews experts in a certain domain and tries to embody their knowledge in a computer program for carrying out some task. How well this works depends on whether the intellectual mechanisms required for the task are within the present state of AI. One of the first expert systems was MYCIN in 1974, which diagnosed bacterial infections of the blood and suggested treatments. It did better than medical students or practicing doctors, provided its limitations were observed.
- **Intelligent Robots** – Robots are able to perform the tasks given by a human. They 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. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.
- **Heuristic classification**
One of the most feasible kinds of expert system given the present knowledge of AI is to put some information in one of a fixed set of categories using several sources of information. An example is advising whether to accept a proposed credit card purchase. Information is available about the owner of the credit card, his record of payment and also about the item he is buying and about the establishment from which he is buying it (e.g., about whether there have been previous credit card frauds at this establishment).

HISTORY OF ARTIFICIAL INTELLIGENCE:

- **The gestation of artificial intelligence (1943-1955)**
There were a number of early examples of work that can be characterized as AI, but it was Alan Turing who first articulated a complete vision of AI in his 1950 article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.

- **The birth of artificial intelligence (1956)**
McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help him bring together U.S. researchers interested in automata theory, neural nets, and the study of intelligence. They organized a two-month workshop at Dartmouth in the summer of 1956. Perhaps the longest-lasting thing to come out of the workshop was an agreement to adopt McCarthy's new name for the field: artificial intelligence.
- **Early enthusiasm, great expectations (1952-1969)**
The early years of Artificial Intelligence were full of successes-in a limited way. General Problem Solver (GPS) was a computer program created in 1957 by Herbert Simon and Allen Newell to build a universal problem solver machine. The order in which the program considered sub goals and possible actions was similar to that in which humans approached the same problems. Thus, GPS was probably the first program to embody the "thinking humanly" approach.
- IBM, Nathaniel Rochester and his colleagues produced some of the first AI pro-grams.
- Herbert Gelernter (1959) constructed the Geometry Theorem Prover, which was able to prove theorems that many students of mathematics would find quite tricky.
- Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute of Technology (MIT). In 1963, McCarthy started the AI lab at Stanford.
- **Knowledge-based systems: The key to power? (1969-1979)**
Dendral was an influential pioneer project in artificial intelligence (AI) of the 1960s, and the computer software expert system that it produced. Its primary aim was to help organic chemists in identifying unknown organic molecules, by analyzing their mass spectra and using knowledge of chemistry. It was done at Stanford University by Edward Feigenbaum, Bruce Buchanan, Joshua Lederberg, and Carl Djerassi.
- **AI becomes an industry (1980-present)**
In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build intelligent computers running Prolog. Overall, the AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988.
- **AI becomes a science (1987-present)**
In recent years, approaches based on hidden Markov models (HMMs) have come to dominate the area.
- Speech technology and the related field of handwritten character recognition are already making the transition to widespread industrial and consumer applications.
- The Bayesian network formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge.
- **The emergence of intelligent agents (1995-present)**
One of the most important environments for intelligent agents is the Internet.

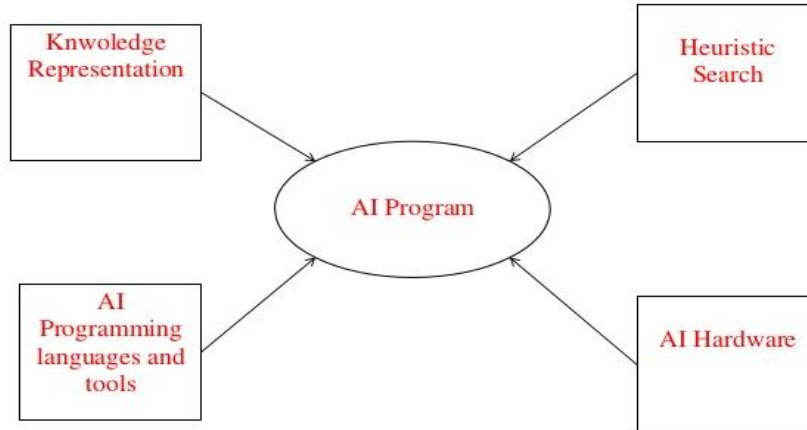
Components of Artificial Intelligence:

- **Reasoning, problem-solving:** Researchers had developed machines with algorithms that enable machines to solve puzzles or quiz similar to humans. AI can also deal with uncertain or incomplete information through advanced algorithms.
- **Knowledge Representation:** It is the representation of all the knowledge which is stored by an agent to make an expert system. Knowledge can be a set of objects, relations, concepts, or properties.
- **Planning:** Intelligent agents should be able to set goals and make plans to achieve those goals. They should be able to visualize the future and make predictions about their actions taken for

achieving the goal.

- **Learning:** It is the study of the computer algorithms which improve automatically through experiences. This concept is known as Machine Learning.
- **Natural Language Processing:** This processing enables a machine to read and understand human language by processing the human language into machine language.
- **Perception:** An ability of the machine to use input from sensors, microphones, wireless signals, etc. for understanding different aspects of the world.

Major components of an AI system



Prepared by :- Agniwesh Mishra, RCET, Bhilai

22

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 1- 29

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L2

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Agents, Environment, Problem formulation, Problem Definition and Characteristics

Introduction:

Problem Solving Approach:

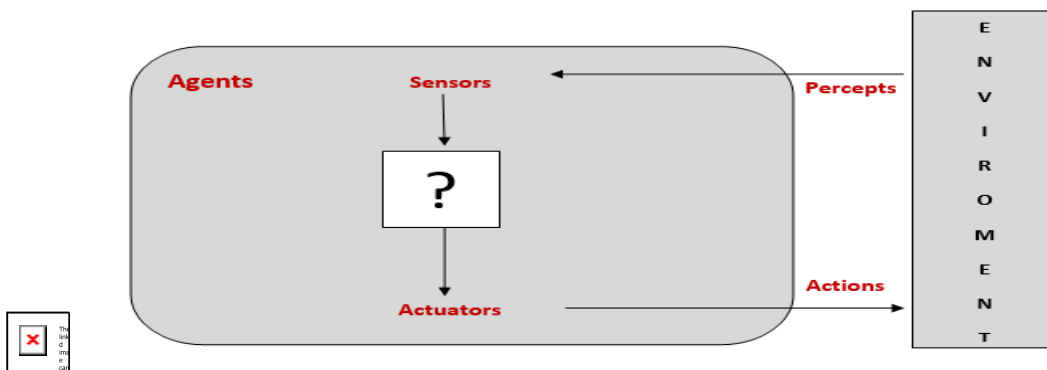
- Problem Formulation
- Defining the Problem
- Alternatives to solve the Problem
- Find a optimized solution

Prerequisite knowledge for Complete understanding and learning of Topic:

- Understand the Problem
- Define the Problem
- Alternatives to solve the problem by various method
- Find a optimized solution

Detailed content of the Lecture:

Agents and Environment:



An agent can be viewed as anything that perceives its environment through sensors and acts upon that environment through actuators.

For example, human being perceives their surroundings through their sensory organs known as sensors and take actions using their hands, legs, etc., known as actuators.

Intelligent Agent

An intelligent agent is a goal-directed agent. It perceives its environment through its sensors using the observations and built-in knowledge, acts upon the environment through its actuators.

Rational Agent

A rational agent is an agent which takes the right action for every perception. By doing so, it maximizes the performance measure, which makes an agent be the most successful.

Omniscient Agent

An omniscient agent is an agent which knows the actual outcome of its action in advance. However, such agents are impossible in the real world.

Task environment:

A task environment is a problem to which a rational agent is designed as a solution.

Rationality of an agent

It is expected from an intelligent agent to act in a way that maximizes its performance measure. Therefore, the rationality of an agent depends on four things:

- The performance measure which defines the criterion of success.
- The agent's built-in knowledge about the environment.
- The actions that the agent can perform.
- The agent's percept sequence until now.

Properties/Classification of Task Environment

Fully Observable vs. Partially Observable:

When an agent's sensors allow access to complete state of the environment at each point of time, then the task environment is fully observable, whereas, if the agent does not have complete and relevant information of the environment, then the task environment is partially observable.

Example: In the Checker Game, the agent observes the environment completely while in Poker Game, the agent partially observes the environment because it cannot see the cards of the other agent.

Note: Fully Observable task environments are convenient as there is no need to maintain the internal state to keep track of the world.

Single-agent vs. Multiagent

When a single agent works to achieve a goal, it is known as Single-agent, whereas when two or more

agents work together to achieve a goal, they are known as Multiagents.

Example: Playing a crossword puzzle – single agent
Playing chess –multiagent (requires two agents)

Deterministic vs. Stochastic

If the agent's current state and action completely determine the next state of the environment, then the environment is deterministic whereas if the next state cannot be determined from the current state and action, then the environment is Stochastic.

Example: Image analysis – Deterministic
Taxi driving – Stochastic (cannot determine the traffic behavior)

Note: If the environment is partially observable, it may appear as Stochastic

Episodic vs. Sequential

If the agent's episodes are divided into atomic episodes and the next episode does not depend on the previous state actions, then the environment is episodic, whereas, if current actions may affect the future decision, such environment is sequential.

Example: Part-picking robot – Episodic
Chess playing – Sequential

Static vs. Dynamic

If the environment changes with time, such an environment is dynamic; otherwise, the environment is static.

Example: Crosswords Puzzles have a static environment while the Physical world has a dynamic environment.

Discrete vs. Continuous

If an agent has the finite number of actions and states, then the environment is discrete otherwise continuous.

Example: In Checkers game, there is a finite number of moves – Discrete
A truck can have infinite moves while reaching its destination – Continuous.

Known vs. Unknown

In a known environment, the agents know the outcomes of its actions, but in an unknown environment, the agent needs to learn from the environment in order to make good decisions.

Example: A tennis player knows the rules and outcomes of its actions while a player needs to learn the rules of a new video game.

Note: A known environment is partially observable, but an unknown environment is fully observable.

Types of Agent Programs:

- **Simple reflex agents:** It is the simplest agent which acts according to the current percept only, pays no attention to the rest of the percept history. The agent function of this type relies

on the **condition-action rule** – “If condition, then action.” It makes correct decisions only if the environment is fully observable. These agents cannot ignore infinite loop when the environment is partially observable but can escape from infinite loops if the agents randomize its actions.

Example: iDraw, a drawing robot which converts the typed characters into writing without storing the past data.

Note: Simple reflex agents do not maintain the internal state and do not depend on the percept theory.

- **Model-based agent:** These type of agents can handle partially observable environments by maintaining some internal states. The internal state depends on the percept history, which reflects at least some of the unobserved aspects of the current state. Therefore, as time passes, the internal state needs to be updated which requires two types of knowledge or information to be encoded in an agent program i.e., the evolution of the world on its own and the effects of the agent’s actions.

Example: When a person walks in a lane, he maps the pathway in his mind.

- **Goal-based agents:** It is not sufficient to have the current state information unless the goal is not decided. Therefore, a goal-based agent selects a way among multiple possibilities that helps it to reach its goal.

Note: With the help of searching and planning (subfields of AI), it becomes easy for the Goal-based agent to reach its destination.

- **Utility-based agents:** These types of agents are concerned about the performance measure. The agent selects those actions which maximize the performance measure and devote towards the goal.

Example: The main goal of chess playing is to ‘check-and-mate’ the king, but the player completes several small goals previously.

Problem Solving:

There are four things that are to be followed in order to solve a problem:

1. *Define the problem.* A precise definition that must include precise specifications of what the initial situation(s) will be as well as what final situations constitute acceptable solutions to the problem.
2. *Problem must be analyzed.* Some of the important features land up having an immense impact on the appropriateness of various possible techniques for solving the problem.
3. *Isolate and represent the task knowledge* that is necessary to solve the problem.
4. Amongst the available ones *choose the best problem-solving technique(s)* and apply the same to the particular problem.

State Space Representation of a problem:

In this section we examine the concept of a state space and the different searches that can be used to explore the search space in order to find a solution. Before an AI problem can be solved it must be represented as a state space. The state space is then searched to find a solution to the problem. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state. Each state space takes the form of a tree or a graph. Factors that determine which search algorithm or technique will be used include the type of the problem and the how the problem can be represented.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 29-35

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L3

CSE

III/VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Production Systems

Introduction:

- Production systems provide search structures that form the core of many intelligent processes.
- it is useful to structure AI programs in a way that facilitates describing and performing the search process

Prerequisite knowledge for Complete understanding and learning of Topic:

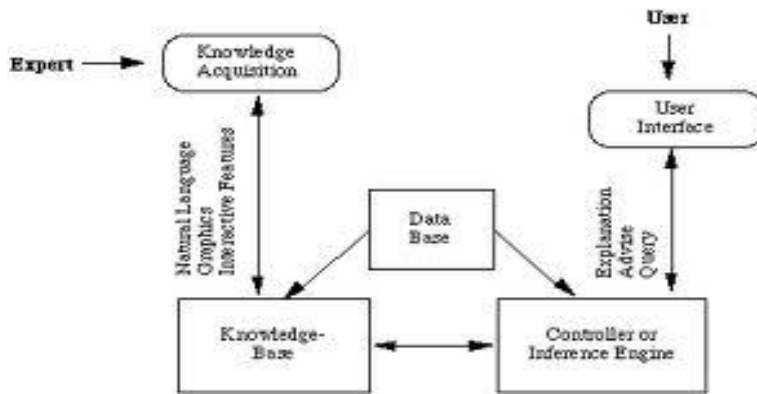
- Problem Formulation
- Problem Representation
- Problem Solving Techniques

Detailed content of the Lecture:

Production systems provide search structures that form the core of many intelligent processes. Hence it is useful to structure AI programs in a way that facilitates describing and performing the search process. Do not be confused by other uses of the word *production*, such as to describe what is done in factories. A *production system* consists of:

1. **A set of rules**, each consisting of a left side and a right hand side. Left hand side or pattern determines the applicability of the rule and a right side describes the operation to be performed if the rule is applied.
2. **One or more knowledge/databases** that contain whatever information is appropriate for the particular task. Some parts of the database may be permanent, while other parts of it may pertain only to the solution of the current problem. The information in these databases may be structured in any appropriate way.
3. **A control strategy** that specifies the order in which the rules will be compared to the database and a way of resolving the conflicts that arise when several rules match at once.

4. A rule applier.



Categories of Production System

	Monotonic	Nonmonotonic
Partially commutative	Theorem proving	Robot navigation
Not partially commutative	Chemical synthesis	Bridge

The production rules are also known as condition – action, antecedent – consequent, pattern – action, situation – response, feedback – result pairs.

For example,

If (you have an exam tomorrow)

THEN (study the whole night)

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

Features of Production System:

Some of the main features of production system are:

Expressiveness and intuitiveness: In real world, many times situation comes like “i f this happen-you will do that”, “if this is so-then this should happ en” and many more. The production rules essentially tell us what to do in a given situation.

1. **Simplicity:** The structure of each sentence in a production system is unique and uniform as they use “IF-THEN” structure. This structure provides simpli city in knowledge representation. This feature of production system improves the readability of production rules.

2. **Modularity:** This means production rule code the knowledge available in discrete pieces. Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deletetious side effects.

3. **Modifiability:** This means the facility of modifying rules. It allows the development of

production rules in a skeletal form first and then it is accurate to suit a specific application.

4. **Knowledge intensive:** The knowledge base of production system stores pure knowledge. This part does not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation.

Disadvantages of production system:

1. **Opacity:** This problem is generated by the combination of production rules. The opacity is generated because of less prioritization of rules. More priority to a rule has the less opacity.
2. **Inefficiency:** During execution of a program several rules may active. A well devised control strategy reduces this problem. As the rules of the production system are large in number and they are hardly written in hierarchical manner, it requires some forms of complex search through all the production rules for each cycle of control program.
3. **Absence of learning:** Rule based production systems do not store the result of the problem for future use. Hence, it does not exhibit any type of learning capabilities. So for each time for a particular problem, some new solutions may come.
4. **Conflict resolution:** The rules in a production system should not have any type of conflict operations. When a new rule is added to a database, it should ensure that it does not have any conflicts with the existing rules.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 36- 44

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L4

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Control and Search strategies

Introduction:

Control strategy that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once.

Prerequisite knowledge for Complete understanding and learning of Topic:

Components of AI
Production System- Basics

Detailed content of the Lecture:

a) Problem Characteristics

1. Is the problem decomposable?
2. Can solution steps be ignored or undone?
3. Is the Universal Predictable?
4. Is good solution absolute or relative ?
5. The knowledge base consistent ?
6. What is the role of Knowledge?
7. Does the task requires interaction with the person.

b) Control Strategies

One of the searches that are commonly used to search a state space is the depth first search. The depth first search searches to the lowest depth or ply of the tree.

Depth First Search Algorithm

def dfs (in Start, out State)

```

open = [Start];
closed = [];
State = failure;
while (open <> []) AND (State <> success)
begin
remove the leftmost state from
open
, call it X;
if X is the goal, then
State = success
else begin
generate children of X;
put X on closed
eliminate the children of X on
open
or
closed
put remaining children on left end of
open
end else
endwhile
return State;
enddef
Breadth First Search

```

The breadth first search visits nodes in a left to right manner at each level of the tree. Each node is not visited more than once.

Breadth First Search Algorithm

```

def bfs (in Start, out State)
open = [Start];
closed = [];
State = failure;
while (open <> []) AND (State <> success)
begin
remove the leftmost state from
open
, call it X;
if X is the goal, then
State = success
else begin
generate children of X;
put X on closed
eliminate the children of X on
open
or
closed
put remaining children on right end of
open
end else
endwhile
return State;
enddef

```

Differences Between Depth First and Breadth First

Breadth first is guaranteed to find the shortest path from the start to the goal. DFS may get “lost” in search and hence a depth-bound may have to be imposed on a depth first search.

BFS has a bad branching factor which can lead to combinatorial explosion. The solution path found by the DFS may be long and not optimal. DFS more efficient for search spaces with many branches, i.e. the branching factor is high.

The following factors need to be taken into consideration when deciding which search to use:

- The importance of finding the shortest path to the goal.
- The branching of the state space.
- The available time and resources.
- The average length of paths to a goal node.
- All solutions or only the first solution.

b) Heuristic Search

DFS and BFS may require too much memory to generate an entire state space - in these cases heuristic search is used.

Heuristics help us to reduce the size of the search space. An evaluation function is applied to each goal to assess how promising it is in leading to the goal

Examples of heuristic searches:

- Best first search
- A* algorithm
- Hill climbing

Heuristic searches incorporate the use of domain-specific knowledge in the process of choosing which node to visit next in the search process. Search methods that include the use of domain knowledge in the form of heuristics are described as “weak” search methods. The knowledge used is “weak” in that it usually helps but does not always help to find a solution.

Calculating heuristics

Heuristics are rules of thumb that may find a solution but are not guaranteed to. Heuristic functions have also been defined as evaluation functions that estimate the cost from a node to the goal node. The incorporation of domain knowledge into this search process by means of heuristics is meant to speed up the search process. Heuristic functions are not guaranteed to be completely accurate. If they were there would be no need for the search process. Heuristic values are greater than and equal to zero for all nodes. Heuristic values are seen as an approximate cost of finding a solution. A heuristic value of zero indicates that the state is a goal state.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 55- 77



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L5

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Production system characteristics- Specialized production systems

Introduction:

- The production system is a model of computation that can be applied to implement search algorithms and model human problem solving.
- Problem solving knowledge can be packed up in the form of little quanta called productions. A production is a rule consisting of a situation recognition part and an action part.
- A production is a situation-action pair in which the left side is a list of things to watch for and the right side is a list of things to do so.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Knowledge about Production system and Rule Generation.

Production System and its Components:

A production system consists of following components.

- (a) A *set of production rules* which are of the form $A \rightarrow B$, left hand side constituent that represent the current problem state and a right hand side that represent an output state. A rule is applicable if its left hand side matches with the current problem state.
- (b) A *database*, which contains all the appropriate information for the particular task. Some part of the database may be permanent while some part of this may pertain only to the solution of the current problem.
- (c) A *control strategy* that specifies order in which the rules will be compared to the database of

rules and a way of resolving the conflicts that arise when several rules match simultaneously.

(d) A *rule applier*, which checks the capability of rule by matching the content state with the left hand

side of the rule and finds the appropriate rule from database of rules.

Production System Characteristics:

Production systems are important in building intelligent matches which can provide us a good set of production rules, for solving problems.

There are four types of production system characteristics, namely

1. Monotonic production system
 2. Non-monotonic production system
 3. Commutative law based production system, and lastly,
 4. Partially commutative law based production system.
- **Monotonic Production System (MPS):** The Monotonic production system (MPS) is a system in which the application of a rule never prevents later application of another rule that could also have been applied at the time that the first rule was selected.
 - **Non-monotonic Production (NMPS):** The non-monotonic production system is a system in which the application of a rule prevents the later application of the another rule which may not have been applied at the time that the first rule was selected, i.e. it is a system in which the above rule is not true, i.e. the monotonic production system rule not true.
 - **Commutative Law Based Production System (CBPS):** Commutative law based production systems is a system in which it satisfies both monotonic & partially commutative.
 - **Partially Commutative Production System (PCPS):** The partially commutative production system is a system with the property that if the application of those rules that is allowable & also transforms from state x to state 'y'.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., "Artificial Intelligence (SIE)", McGraw Hill- 2008.
Page No. : 65- 73

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L6

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Problem graphs with an example (Constraint Satisfaction Problem- Missionaries and Cannibals)

Introduction:

- Various strategies to solve the problem
- Most strategies may lead to solution
- Choose strategy which produces results at low cost and time.

Prerequisite knowledge for Complete understanding and learning of Topic:

Graph Theory
Discrete Mathematics
Problem formulation and representation Techniques.

Detailed content of the Lecture:

Problem solving methods- Problem graphs

Problem Representation:

- States, operators and searching
- representing state spaces using graphs
- finding a path from A to B in a graph
- Example: missionaries & cannibals: representing states & operators
- methods of search: depth-first, breadth-first

Missionaries and Cannibals

- There are three missionaries and three cannibals on the left bank of a river.
- They wish to cross over to the right bank using a boat that can only carry two at a time.
- The number of cannibals on either bank must never exceed the number of missionaries on the same bank, otherwise the missionaries will become the cannibals' dinner!

- Plan a sequence of crossings that will take everyone safely across.

This kind of problem is often solved by a graph search method. We represent the problem as a set of states

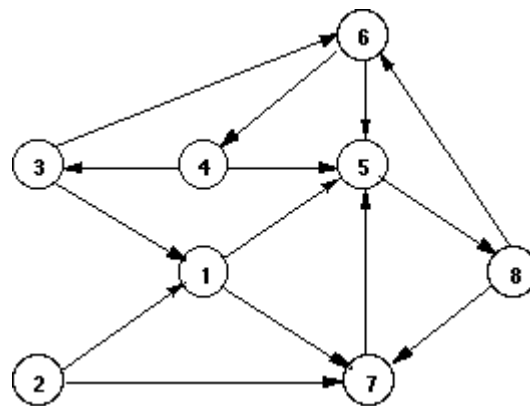
which are snapshots of the world and

operators

which transform one state into another

States can be mapped to nodes of a graph and operators are the edges of the graph. Before studying the missionaries and cannibals problem, we look at a simple graph search algorithm in Prolog. the missionaries and cannibals program will have the same basic structure.

Graph Representation



A graph may be represented by a set of edge predicates and a list of vertices.

edge(1, 5).	edge(1, 7).
edge(2, 1).	edge(2, 7).
edge(3, 1).	edge(3, 6).
edge(4, 3).	edge(4, 5).
edge(5, 8).	
edge(6, 4).	edge(6, 5).
edge(7, 5).	
edge(8, 6).	edge(8, 7).

vertices([1, 2, 3, 4, 5, 6, 7, 8]).

Finding a path

- Write a program to find path from one node to another.
- Must avoid cycles (i.e. going around in circle).
- A template for the clause is:

path(Start, Finish, Visited, Path).

Start

is the name of the starting node

Finish

is the name of the finishing node

Visited

is the list of nodes already visited.

Path

is the list of nodes on the path, including **Start** and **Finish**.

The Path Program

- The search for a path terminates when we have nowhere to go.

path(Node, Node, _, [Node]).

- A path from Start to Finish starts with a node, X, connected to Start followed by a path from X to Finish.

```
path(Start, Finish, Visited, [Start | Path]) :-
    edge(Start, X),
    not(member(X, Visited)),
    path(X, Finish, [X | Visited], Path).
```

Representing the state

Now we return to the problem of representing the missionaries and cannibals problem:

- A state is one "snapshot" in time.
- For this problem, the only information we need to fully characterise the state is:
 - the number of missionaries on the left bank,
 - the number of cannibals on the left bank,
 - the side the boat is on.

All other information can be deduced from these three items.

- In Prolog, the state can be represented by a 3-arity term,

state(Missionaries, Cannibals, Side)

Representing the Solution

- The solution consists of a list of moves, e.g.

[move(1, 1, right), move(2, 0, left)]

- We take this to mean that 1 missionary and 1 cannibal moved to the right bank, then 2 missionaries moved to the left bank.
- Like the graph search problem, we must avoid returning to a state we have visited before.
- The visited list will have the form:

[MostRecent_State | ListOfPreviousStates]

Overview of Solution

- We follow a simple graph search procedure:
 - Start from an initial state
 - Find a neighbouring state
 - Check that the new state has not been visited before
 - Find a path from the neighbour to the goal.

The search terminates when we have found the state:

state(0, 0, right).

Possible Moves

- A move is characterised by the number of missionaries and the number of cannibals taken in

the boat at one time.

- Since the boat can carry no more than two people at once, the only possible combinations are:

carry(2, 0).

carry(1, 0).

carry(1, 1).

carry(0, 1).

carry(0, 2).

- where carry(M , C) means the boat will carry M missionaries and C cannibals on one trip.

Feasible Moves

- Once we have found a possible move, we have to confirm that it is feasible.
- I.e. it is not feasible to move more missionaries or more cannibals than are present on one bank.
- When the state is state($M1$, $C1$, left) and we try carry(M , C) then

$$M \leq M1 \text{ and } C \leq C1$$

- must be true.
- When the state is state($M1$, $C1$, right) and we try carry(M , C) then

$$M + M1 \leq 3 \text{ and } C + C1 \leq 3$$

- must be true.

Legal Moves

- Once we have found a feasible move, we must check that is legal.
- I.e. no missionaries must be eaten.

legal(X , X).

legal(3, X).

legal(0, X).

- The only safe combinations are when there are equal numbers of missionaries and cannibals or all the missionaries are on one side.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., "Artificial Intelligence (SIE)", McGraw Hill- 2008.
Page No. : 82- 88

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L7

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Problem solving methods

Introduction:

- Various search strategies used to find the solution for a given problem.
- Strategies to analyze while choosing the Search method.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Knowledge about Production system and Rule Generation.
- Discrete Mathematics
- Graph Theory- Basics

Detailed Content:

- **Search Strategies**

There are two types of strategies that describe a solution for a given problem:

- **Uninformed Search (Blind Search)**

This type of search strategy does not have any additional information about the states except the information provided in the problem definition. They can only generate the successors and distinguish a goal state from a non-goal state. These type of search does not maintain any internal state, that's why it is also known as **Blind search**.

There are following types of uninformed searches:

- Breadth-first search
- Uniform cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search

- **Informed Search (Heuristic Search)**

This type of search strategy contains some additional information about the states beyond the problem definition. This search uses problem-specific knowledge to find more efficient solutions. This search maintains some sort of internal states via heuristic functions (which provides hints), so it is also called **heuristic search**.

There are following types of informed searches:

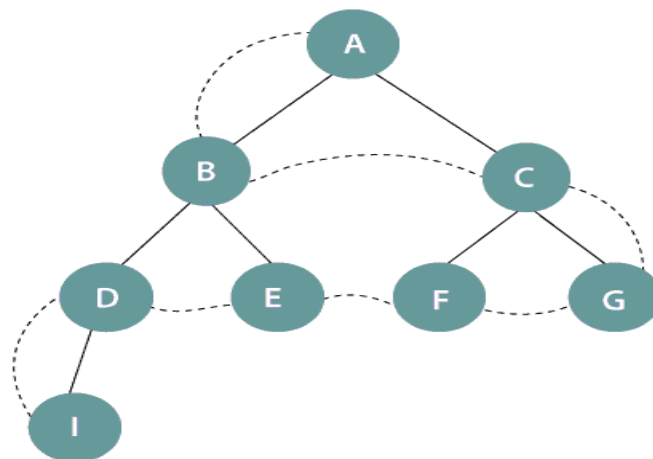
- Best first search (Greedy search)
- A* search

Breadth-first search (BFS):

It is a simple search strategy where the root node is expanded first, then covering all other successors of the root node, further move to expand the next level nodes and the search continues until the goal node is not found.

BFS expands the shallowest (i.e., not deep) node first using FIFO (First in first out) order. Thus, new nodes (i.e., children of a parent node) remain in the queue and old unexpanded node which are shallower than the new nodes, get expanded first.

In BFS, goal test (**a test to check whether the current state is a goal state or not**) is applied to each node at the time of its generation rather when it is selected for expansion.



Breadth-first search

In the above figure, it is seen that the nodes are expanded level by level starting from the root node **A** till the last node **I** in the tree. Therefore, the BFS sequence followed is: **A->B->C->D->E->F->G->I**.

BFS Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Loop each node by traversing level by level until the goal state is not found.
- While performing the looping, start removing the elements from the queue in **FIFO** order.
- If the goal state is found, **return goal state** otherwise continue the search.

The performance measure of BFS is as follows:

- **Completeness:** It is a complete strategy as it definitely finds the goal state.
- **Optimality:** It gives an optimal solution if the cost of each node is same.
- **Space Complexity:** The space complexity of BFS is **O(b^d)**, i.e., it requires a huge amount of memory. Here, **b** is the **branching factor** and **d** denotes the **depth/level** of the tree

- **Time Complexity:** BFS consumes much time to reach the goal node for large instances.

Disadvantages of BFS

- The biggest disadvantage of BFS is that it requires a lot of memory space, therefore it is a memory bounded strategy.
- BFS is time taking search strategy because it expands the nodes **breadthwise**.

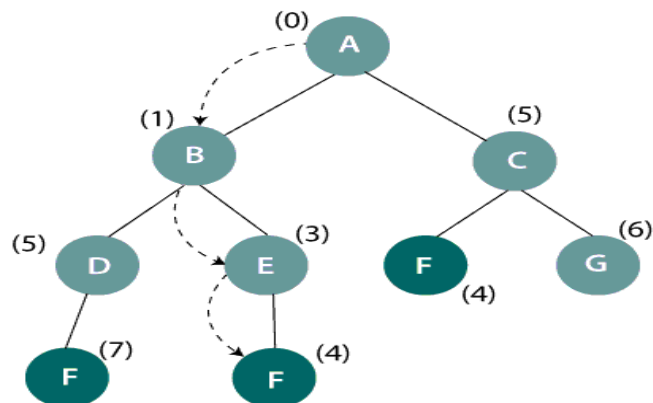
Note: BFS expands the nodes level by level, i.e., breadthwise, therefore it is also known as a **Level search technique**.

Uniform-cost search:

Unlike BFS, this uninformed search explores nodes based on their path cost from the root node. It expands a node n having the lowest path cost $g(n)$, where $g(n)$ is the total cost from a root node to node n . Uniform-cost search is significantly different from the breadth-first search because of the following two reasons:

- First, the goal test is applied to a node only when it is selected for expansion **not when it is first generated** because the first goal node which is generated may be on a suboptimal path.
- Secondly, a goal test is added to a node, only when a better/optimal path is found.

Thus, uniform-cost search expands nodes in a sequence of their **optimal path cost** because before exploring any node, it searches the optimal path. Also, the step cost is positive so, paths never get shorter when a new node is added in the search.



Uniform-cost search on a binary tree

In the above figure, it is seen that the goal-state is **F** and start/initial state is **A**. There are three paths available to reach the goal node. We need to select an optimal path which may give the lowest total cost $g(n)$. Therefore, **A->B->E->F** gives the optimal path cost i.e., **0+1+3+4=8**.

Uniform-cost search Algorithm

- Set a variable **NODE** to the initial state, i.e., *the root node and expand it*.
- After expanding the root node, select one node having the lowest path cost and expand it further. Remember, **the selection of the node should give an optimal path cost**.
- If the goal node is searched with optimal value, return **goal state**, else carry on the search.

The performance measure of Uniform-cost search

- **Completeness:** It guarantees to reach the goal state.
- **Optimality:** It gives optimal path cost solution for the search.
- **Space and time complexity:** The worst space and time complexity of the uniform-cost search is $O(b^{1+LC*/\epsilon})$.

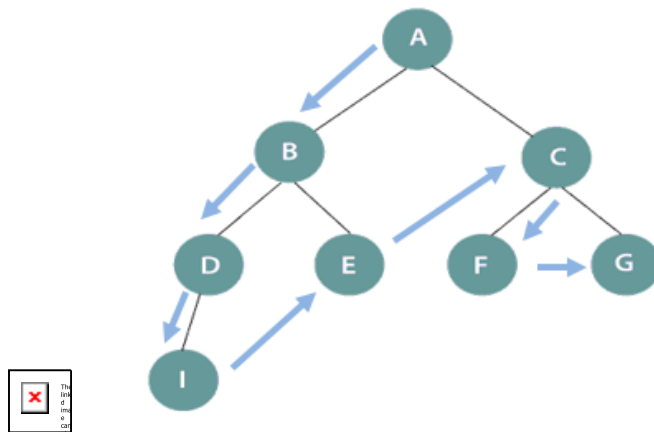
Note: When the path cost is same for all the nodes, it behaves similar to **BFS**.

Disadvantages of Uniform-cost search

- It does not care about the number of steps a path has taken to reach the goal state.
- It may stick to an infinite loop if there is a path with infinite zero cost sequence.
- It works hard as it examines each node in search of lowest cost path.

Depth-first search

This search strategy explores the deepest node first, then backtracks to explore other nodes. It uses **LIFO (Last in First Out)** order, which is based on the **stack**, in order to expand the unexpanded nodes in the search tree. The search proceeds to the deepest level of the tree where it has no successors. This search expands nodes till infinity, i.e., the depth of the tree.



DFS search tree

In the above figure, DFS works starting from the initial node **A** (root node) and traversing in one direction deeply till node **I** and then backtrack to **B** and so on. Therefore, the sequence will be **A->B->D->I->E->C->F->G**.

DFS Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Loop each node by traversing deeply in one direction/path in search of the goal node.
- While performing the looping, start removing the elements from the stack in **LIFO** order.
- If the goal state is found, **return goal state** otherwise backtrack to expand nodes in other direction.

The performance measure of DFS

- **Completeness:** DFS does not guarantee to reach the goal state.
- **Optimality:** It does not give an optimal solution as it expands nodes in one direction deeply.
- **Space complexity:** It needs to store only a single path from the root node to the leaf node. Therefore, DFS has **O(bm)** space complexity where **b** is the **branching factor**(i.e., **total no. of child nodes, a parent node have**) and **m** is the **maximum length of any path**.
- **Time complexity:** DFS has **O(b^m)** time complexity.

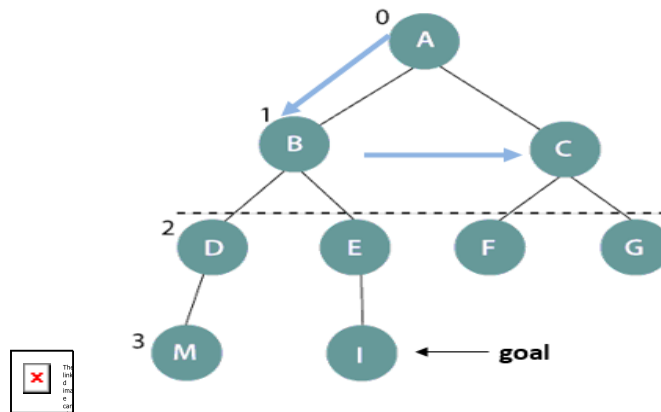
Disadvantages of DFS

- It may get trapped in an infinite loop.
- It is also possible that it may not reach the goal state.
- DFS does not give an optimal solution.

Note: DFS uses the concept of **backtracking** to explore each node in a search tree.

Depth-limited search

This search strategy is similar to DFS with a little difference. The difference is that in depth-limited search, we limit the search by imposing a **depth limit l** to the depth of the search tree. It does not need to explore till infinity. As a result, the **depth-first search is a special case of depth-limited search**, when the limit l is infinite.



Depth-limited search on a binary tree

In the above figure, the **depth-limit is 1**. So, only level 0 and 1 get expanded in **A->B->C** DFS sequence, starting from the root node **A** till node **B**. It is not giving satisfactory result because we could not reach the goal node **I**.

Depth-limited search Algorithm

- Set a variable **NODE** to the initial state, i.e., the *root node*.
- Set a variable **GOAL** which contains the value of the *goal state*.
- Set a variable **LIMIT** which carries a depth-limit value.
- Loop each node by traversing in **DFS** manner till the depth-limit value.
- While performing the looping, start removing the elements from the stack in **LIFO** order.
- If the goal state is found, **return goal state**. Else **terminate the search**.

The performance measure of Depth-limited search

- **Completeness:** Depth-limited search does not guarantee to reach the goal node.
- **Optimality:** It does not give an optimal solution as it expands the nodes till the depth-limit.
- **Space Complexity:** The space complexity of the depth-limited search is $O(bl)$.
- **Time Complexity:** The time complexity of the depth-limited search is $O(b^l)$.

Disadvantages of Depth-limited search

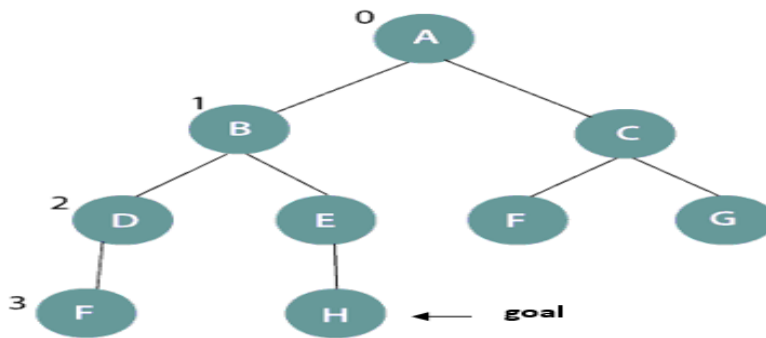
- This search strategy is not complete.
- It does not provide an optimal solution.

Note: Depth-limit search terminates with two kinds of failures: the **standard failure value** indicates “no solution,” and **cut-off value**, which indicates “no solution within the depth-limit.”

Iterative deepening depth-first search/Iterative deepening search:

This search is a combination of BFS and DFS, as BFS guarantees to reach the goal node and DFS occupies less memory space. Therefore, iterative deepening search combines these two advantages of BFS and DFS to reach the goal node. It gradually increases the depth-limit from **0,1,2** and so on and reach the goal node.

In the above figure, the goal node is **H** and initial **depth-limit** =**[0-1]**. So, it will expand level 0 and 1 and will terminate with **A->B->C** sequence. Further, change the **depth-limit** =**[0-3]**, it will again expand the nodes from level 0 till level 3 and the search terminate with **A->B->D->F->E->H** sequence where **H** is the desired goal node.



Iterative deepening search Algorithm

- Explore the nodes in DFS order.
- Set a LIMIT variable with a limit value.
- Loop each node up to the limit value and further increase the limit value accordingly.
- Terminate the search when the goal state is found.

The performance measure of Iterative deepening search

- **Completeness:** Iterative deepening search may or may not reach the goal state.
- **Optimality:** It does not give an optimal solution always.
- **Space Complexity:** It has the same space complexity as BFS, i.e., $O(b^d)$.
- **Time Complexity:** It has $O(d)$ time complexity.

Disadvantages of Iterative deepening search

- The drawback of iterative deepening search is that it seems wasteful because it generates states multiple times.

Note: Generally, iterative deepening search is required when the search space is large, and the depth of the solution is unknown.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 44- 55

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L8

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Heuristic functions- Hill climbing

Introduction:

Heuristic Functions is a **function** that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow. For example, it may approximate the exact solution.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Mathematics
- Problem solving methods

Detailed content of the Lecture:

Properties of a Heuristic search Algorithm

Use of heuristic function in a heuristic search algorithm leads to following properties of a heuristic search algorithm:

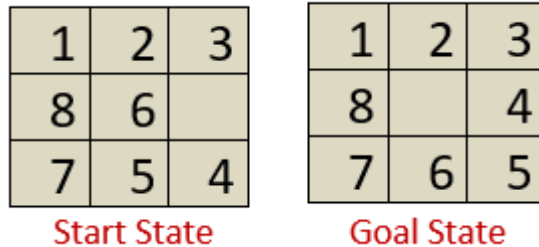
- **Admissible Condition:** An algorithm is said to be admissible, if it returns an optimal solution.
- **Completeness:** An algorithm is said to be complete, if it terminates with a solution (if the solution exists).
- **Dominance Property:** If there are two admissible heuristic algorithms **A1** and **A2** having **h1** and **h2** heuristic functions, then **A1** is said to dominate **A2** if **h1** is better than **h2** for all the values of node **n**.
- **Optimality Property:** If an algorithm is **complete**, **admissible**, and **dominating** other algorithms, it will be the best one and will definitely give an optimal solution.

Example: 8-puzzle problem

Consider the following 8-puzzle problem where we have a start state and a goal state. Our task is to slide the tiles of the current/start state and place it in an order followed in the goal state. There can be four moves either **left, right, up, or down**. There can be several ways to convert the current/start state to the goal state, but, we can use a heuristic function $h(n)$ to solve the problem more efficiently.



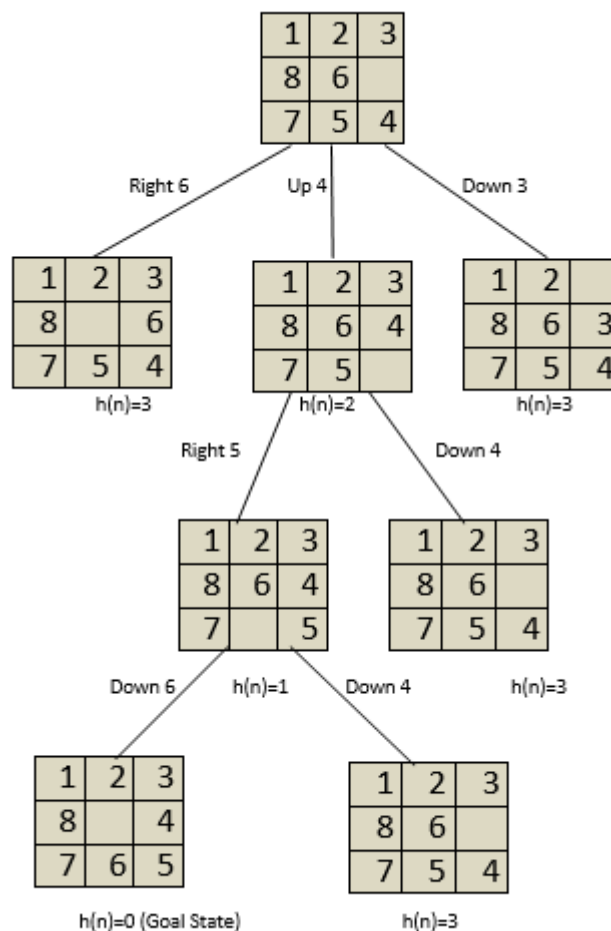
A heuristic function for the 8-puzzle problem is defined below:



$h(n)$ =Number of tiles out of position.

So, there is total of three tiles out of position i.e., 6,5 and 4. Do not count the empty tile present in the goal state). i.e. $h(n)=3$. Now, we require to minimize the value of $h(n) = 0$.

We can construct a state-space tree to minimize the $h(n)$ value to 0, as shown below:



It is seen from the above state space tree that the goal state is minimized from $h(n)=3$ to $h(n)=0$.

However, we can create and use several heuristic functions as per the requirement. It is also clear from

the above example that a heuristic function $h(n)$ can be defined as the information required to solve a given problem more efficiently. The information can be related to the **nature of the state, cost of transforming from one state to another, goal node characteristics**, etc., which is expressed as a heuristic function.

- **Hill Climbing Algorithm**

Hill climbing search is a local search problem. *The purpose of the hill climbing search is to climb a hill and reach the topmost peak/point of that hill.* It is based on the **heuristic search technique** where the person who is climbing up on the hill estimates the direction which will lead him to the highest peak.

State-space Landscape of Hill climbing algorithm

To understand the concept of hill climbing algorithm, consider the below landscape representing the **goal state/peak** and the **current state** of the climber. The topographical regions shown in the figure can be defined as:

- **Global Maximum:** It is the highest point on the hill, which is the goal state.
- **Local Maximum:** It is the peak higher than all other peaks but lower than the global maximum.
- **Flat local maximum:** It is the flat area over the hill where it has no uphill or downhill. It is a saturated point of the hill.
- **Shoulder:** It is also a flat area where the summit is possible.
- **Current state:** It is the current position of the person.



Types of Hill climbing search algorithm

There are following types of hill-climbing search:

- Simple hill climbing
- Steepest-ascent hill climbing
- Stochastic hill climbing
- Random-restart hill climbing



Simple hill climbing search

Simple hill climbing is the simplest technique to climb a hill. The task is to reach the highest peak of the mountain. Here, the movement of the climber depends on his move/steps. If he finds his next step better than the previous one, he continues to move else remain in the same state. This search focus only on his previous and next step.

- **Simple hill climbing Algorithm**

1. Create a **CURRENT** node, **NEIGHBOUR** node, and a **GOAL** node.
2. If the **CURRENT node=GOAL node**, return **GOAL** and terminate the search.
3. Else **CURRENT node<= NEIGHBOUR node**, move ahead.
4. Loop until the goal is not reached or a point is not found.

Steepest-ascent hill climbing

Steepest-ascent hill climbing is different from simple hill climbing search. Unlike simple hill climbing search, It considers all the successive nodes, compares them, and chooses the node which is closest to the solution. Steepest hill climbing search is similar to **best-first search** because it focuses on each node instead of one.

Note: Both simple, as well as steepest-ascent hill climbing search, fails when there is no closer node.

- **Steepest-ascent hill climbing algorithm**

1. Create a **CURRENT** node and a **GOAL** node.
2. If the **CURRENT node=GOAL node**, return **GOAL** and terminate the search.
3. Loop until a better node is not found to reach the solution.
4. If there is any better successor node present, expand it.
5. When the **GOAL** is attained, return **GOAL** and terminate.

- **Stochastic hill climbing**

Stochastic hill climbing does not focus on all the nodes. It selects one node at random and decides whether it should be expanded or search for a better one.

- **Random-restart hill climbing**

Random-restart algorithm is based on **try and try strategy**. It iteratively searches the node and selects the best one at each step until the goal is not found. The success depends most commonly on the shape of the hill. If there are few plateaus, local maxima, and ridges, it becomes easy to reach the destination.

- **Limitations of Hill climbing algorithm**

Hill climbing algorithm is a fast and furious approach. It finds the solution state rapidly because it is quite easy to improve a bad state. But, there are following limitations of this search:

- **Local Maxima:** It is that peak of the mountain which is highest than all its neighboring states but lower than the global maxima. It is not the goal peak because there is another peak higher than it.
- **Plateau:** It is a flat surface area where no uphill exists. It becomes difficult for the climber to decide that in which direction he should move to reach the goal point. Sometimes, the person gets lost in the flat area.
- **Ridges:** It is a challenging problem where the person finds two or more local maxima of the

same height commonly. It becomes difficult for the person to navigate the right point and stuck to that point itself.



- **Simulated Annealing**

- Simulated annealing is similar to the hill climbing algorithm.
- It works on the current situation.
- It picks a **random move** instead of picking **the best move**.
- If the move leads to the improvement of the current situation, it is always accepted as a step towards the solution state, else it accepts the move having a **probability less than 1**. T
- his search technique was first used in **1980** to solve **VLSI layout** problems.
- It is also applied for factory scheduling and other large optimization tasks.

Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 65- 73

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L9

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I - INTRODUCTION TO AI AND PRODUCTION SYSTEMS

Date of Lecture:

Topic of Lecture: Constraint Satisfaction problem- Related algorithms- Measure of performance and analysis of algorithms

Introduction:

- Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem.
- Performance metrics to analyze the optimum problem solving strategy

Prerequisite knowledge for Complete understanding and learning of Topic:

- Search Strategies
- Basic Mathematics
- Time Complexity
- Space Complexity

Detailed content of the Lecture:

Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.

Constraint satisfaction depends on three components, namely:

- **X:** It is a set of variables.
- **D:** It is a set of domains where the variables reside. There is a specific domain for each variable.
- **C:** It is a set of constraints which are followed by the set of variables.

In constraint satisfaction, domains are the spaces where the variables reside, following the problem specific constraints. These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of {**scope, rel**}. The **scope** is a tuple of variables which participate in

the constraint and **rel** is a relation which includes a list of values which the variables can take to satisfy the constraints of the problem.

Solving Constraint Satisfaction Problems

The requirements to solve a constraint satisfaction problem (CSP) is:

- A state-space
- The notion of the solution.

A state in state-space is defined by assigning values to some or all variables such as

{ $X_1=v_1$, $X_2=v_2$, and so on...}.

An assignment of values to a variable can be done in three ways:

- **Consistent or Legal Assignment:** An assignment which does not violate any constraint or rule is called Consistent or legal assignment.
- **Complete Assignment:** An assignment where every variable is assigned with a value, and the solution to the CSP remains consistent. Such assignment is known as Complete assignment.
- **Partial Assignment:** An assignment which assigns values to some of the variables only. Such type of assignments are called Partial assignments.

Types of Domains in CSP

There are following two types of domains which are used by the variables :

- **Discrete Domain:** It is an infinite domain which can have one state for multiple variables. **For example**, a start state can be allocated infinite times for each variable.
- **Finite Domain:** It is a finite domain which can have continuous states describing one domain for one specific variable. It is also called a continuous domain.

Constraint Types in CSP

With respect to the variables, basically there are following types of constraints:

- **Unary Constraints:** It is the simplest type of constraints that restricts the value of a single variable.
- **Binary Constraints:** It is the constraint type which relates two variables. A value x_2 will contain a value which lies between x_1 and x_3 .
- **Global Constraints:** It is the constraint type which involves an arbitrary number of variables.

Some special types of solution algorithms are used to solve the following types of constraints:

- **Linear Constraints:** These type of constraints are commonly used in linear programming where each variable containing an integer value exists in linear form only.
- **Non-linear Constraints:** These type of constraints are used in non-linear programming where each variable (an integer value) exists in a non-linear form.

Note: A special constraint which works in real-world is known as **Preference constraint**.

Constraint Propagation

In local state-spaces, the choice is only one, i.e., to search for a solution. But in CSP, we have two choices either:

- We can search for a solution or
- We can perform a special type of inference called **constraint propagation**.

Constraint propagation is a special type of inference which helps in reducing the legal number of values for the variables. The idea behind constraint propagation is **local consistency**.

In local consistency, variables are treated as **nodes**, and each binary constraint is treated as an **arc** in the given problem. **There are following local consistencies which are discussed below:**

- **Node Consistency:** A single variable is said to be node consistent if all the values in the variable's domain satisfy the unary constraints on the variables.
- **Arc Consistency:** A variable is arc consistent if every value in its domain satisfies the binary constraints of the variables.
- **Path Consistency:** When the evaluation of a set of two variable with respect to a third variable can be extended over another variable, satisfying all the binary constraints. It is similar to arc consistency.
- **k-consistency:** This type of consistency is used to define the notion of stronger forms of propagation. Here, we examine the k-consistency of the variables.

Example:

- **Cryptarithmic Problem:** This problem has one most important constraint that is, we cannot assign a different digit to the same character. All digits should contain a unique alphabet.
- Cryptarithmic Problem is a type of constraint satisfaction problem where the game is about digits and its unique replacement either with alphabets or other symbols. In cryptarithmic problem, the digits (0-9) get substituted by some possible alphabets or symbols. The task in cryptarithmic problem is to substitute each digit with an alphabet to get the result arithmetically correct.

We can perform all the arithmetic operations on a given cryptarithmic problem.

The rules or constraints on a cryptarithmic problem are as follows:

- There should be a unique digit to be replaced with a unique alphabet.
- The result should satisfy the predefined arithmetic rules, i.e., $2+2=4$, nothing else.
- Digits should be from **0-9** only.
- There should be only one carry forward, while performing the addition operation on a problem.
- The problem can be solved from both sides, i.e., **lefthand side (L.H.S)**, or **righthand side (R.H.S)**

Let's understand the cryptarithmic problem as well its constraints better with the help of an example:

- Given a cryptarithmic problem, i.e., **S E N D + M O R E = M O N E Y**

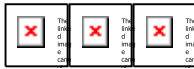


In this example, add both terms **S E N D** and **M O R E** to bring **M O N E Y** as a result.

Follow the below steps to understand the given problem by breaking it into its subparts:

- Starting from the left hand side (L.H.S) , the terms are **S** and **M**. Assign a digit which could give a satisfactory result. Let's assign **S->9** and **M->1**.

$$\begin{array}{r}
 \mathbf{S} \\
 + \mathbf{M} \\
 \hline
 \mathbf{M O}
 \end{array}
 \longrightarrow
 \begin{array}{r}
 \mathbf{9} \\
 + \mathbf{1} \\
 \hline
 \mathbf{10}
 \end{array}$$



Hence, we get a satisfactory result by adding up the terms and got an assignment for **O** as **O->0** as well.

- Now, move ahead to the next terms **E** and **O** to get **N** as its output.

$$\begin{array}{r}
 \mathbf{E} \\
 + \mathbf{O} \\
 \hline
 \mathbf{N}
 \end{array}
 \xrightarrow{\text{X}}
 \begin{array}{r}
 \mathbf{5} \\
 + \mathbf{0} \\
 \hline
 \mathbf{5}
 \end{array}$$



Adding E and O, which means 5+0=0, which is not possible because according to cryptarithmic constraints, we cannot assign the same digit to two letters. So, we need to think more and assign some other value.

$$\begin{array}{r}
 \mathbf{E} \\
 + \mathbf{O} \\
 \hline
 \mathbf{N}
 \end{array}
 \longrightarrow
 \begin{array}{r}
 \textcircled{1} \leftarrow \text{carry} \\
 \mathbf{5} \\
 + \mathbf{0} \\
 \hline
 \mathbf{6}
 \end{array}$$



Note: When we will solve further, we will get one carry, so after applying it, the answer will be satisfied.

- Further, adding the next two terms **N** and **R** we get,

$$\begin{array}{r}
 \mathbf{N} \\
 + \mathbf{R} \\
 \hline
 \mathbf{E}
 \end{array}
 \xrightarrow{\quad}
 \begin{array}{r}
 \mathbf{6} \\
 + \mathbf{8} \\
 \hline
 \mathbf{14}
 \end{array}$$


But, we have already assigned **E**->**5**. Thus, the above result does not satisfy the values because we are getting a different value for **E**. So, we need to think more.

Again, after solving the whole problem, we will get a carryover on this term, so our answer will be satisfied.

$$\begin{array}{r}
 \mathbf{N} \\
 + \mathbf{R} \\
 \hline
 \mathbf{E}
 \end{array}
 \xrightarrow{\quad}
 \begin{array}{r}
 \mathbf{6} \\
 + \mathbf{8} \\
 \hline
 \mathbf{15}
 \end{array}$$

1 ← carry

↑



where **1** will be carry forward to the above term

Let's move ahead.

- Again, on adding the last two terms, i.e., the rightmost terms **D** and **E**, we get **Y** as its result.

$$\begin{array}{r}
 \mathbf{D} \\
 + \mathbf{E} \\
 \hline
 \mathbf{Y}
 \end{array}
 \qquad
 \begin{array}{r}
 \mathbf{7} \\
 + \mathbf{5} \\
 \hline
 \mathbf{12}
 \end{array}$$

↑

where **1** will be carry forward to the above term

- Keeping all the constraints in mind, the final resultant is as follows:

$$\begin{array}{r}
 \mathbf{S E N D} \\
 + \mathbf{M O R E} \\
 \hline
 \mathbf{M O N E Y}
 \end{array}$$



- Below is the representation of the assignment of the digits to the alphabets.

S	9
E	5
N	6
D	7
M	1
O	0
R	8
y	2



Video Content / Details of website for further learning (if any):

1. www.artint.info/html/ArtInt.html
2. www.aima.cs.berkeley.edu
3. www-formal.stanford.edu/jmc/whatisai/
4. www.nptel.ac.in/courses/106106126

Important Books/Journals for further learning including the page nos.:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill- 2008.
Page No. : 74- 80

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L10

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II - REPRESENTATION OF KNOWLEDGE

Date of Lecture:

Topic of Lecture: Game playing- Introduction

Introduction :

- **Game-playing** technique where the agents are surrounded by a competitive environment.
- A conflicting goal is given to the agents (multiagent).
- These agents compete with one another and try to defeat one another in order to win the game. Such conflicting goals give rise to the adversarial search.

Prerequisite knowledge for Complete understanding and learning of Topic:

Components of Artificial Intelligence
Types of Agents and Environment

Detailed content of the Lecture:

- **Game-playing** technique where the agents are surrounded by a competitive environment. A conflicting goal is given to the agents (multiagent).
- These agents compete with one another and try to defeat one another in order to win the game. Such conflicting goals give rise to the adversarial search.
- Eg., **Tic-tac-toe, chess, checkers**, etc.

According to game theory, a game is played between two players. To complete the game, one has to win the game and the other loses automatically.'



We are opponents- I win, you loose.

Techniques required to get the best optimal solution

There is always a need to choose those algorithms which provide the best optimal solution in a limited time. So, we use the following techniques which could fulfil our requirements:

- **Pruning:** A technique which allows ignoring the unwanted portions of a search tree which make no difference in its final result.
- **Heuristic Evaluation Function:** It allows approximating the cost value at each level of the search tree, before reaching the goal node.

Elements of Game Playing search

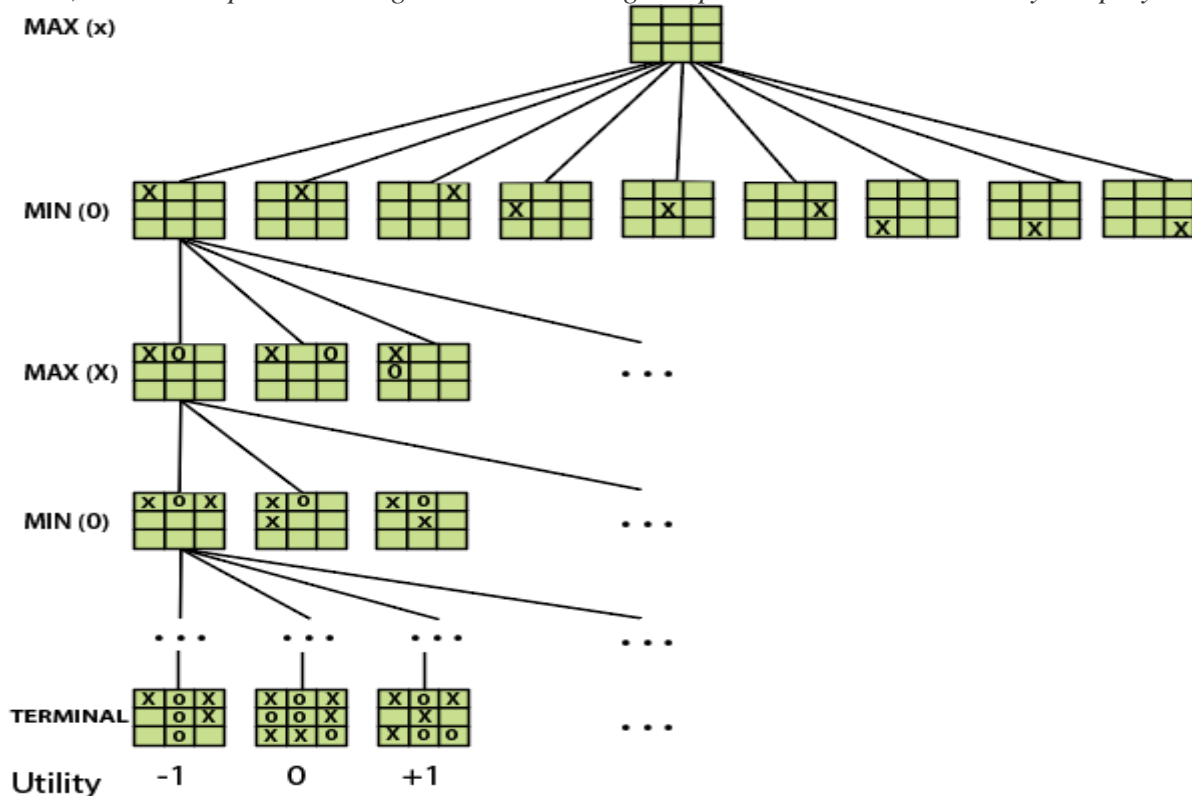
To play a game, we use a game tree to know all the possible choices and to pick the best one out. There are following elements of a game-playing:

- **S₀:** It is the initial state from where a game begins.
- **PLAYER (s):** It defines which player is having the current turn to make a move in the state.
- **ACTIONS (s):** It defines the set of legal moves to be used in a state.
- **RESULT (s, a):** It is a transition model which defines the result of a move.
- **TERMINAL-TEST (s):** It defines that the game has ended and returns true.
- **UTILITY (s,p):** It defines the final value with which the game has ended. This function is also known as **Objective function** or **Payoff function**. The price which the winner will get i.e.
 - (-1): If the PLAYER loses.
 - (+1): If the PLAYER wins.
 - (0): If there is a draw between the PLAYERS.

For example, in chess, tic-tac-toe, we have two or three possible outcomes. Either to win, to lose, or to draw the match with values +1,-1 or 0.

Let's understand the working of the elements with the help of a game tree designed for **tic-tac-toe**.

Here, the node represents the game state and edges represent the moves taken by the players.



- **INITIAL STATE (S₀):** The top node in the game-tree represents the initial state in the tree and shows all the possible choice to pick out one.
- **PLAYER (s):** There are two players, **MAX** and **MIN**. **MAX** begins the game by picking one best move and place **X** in the empty square box.
- **ACTIONS (s):** Both the players can make moves in the empty boxes chance by chance.
- **RESULT (s, a):** The moves made by **MIN** and **MAX** will decide the outcome of the game.

- **TERMINAL-TEST(s):** When all the empty boxes will be filled, it will be the terminating state of the game.
- **UTILITY:** At the end, we will get to know who wins: **MAX** or **MIN**, and accordingly, the price will be given to them.

Types of algorithms in Adversarial search

In a **normal search**, we follow a sequence of actions to reach the goal or to finish the game optimally. But in an **adversarial search**, the result depends on the players which will decide the result of the game. It is also obvious that the solution for the goal state will be an optimal solution because the player will try to win the game with the shortest path and under limited time.

There are following types of adversarial search:

- **Minmax Algorithm**
- **Alpha-beta Pruning.**

Video Content / Details of website for further learning (if any):

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L11

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II - REPRESENTATION OF KNOWLEDGE

Date of Lecture:

Topic of Lecture: Knowledge Representation

Introduction :

- **Game-playing** technique where the agents are surrounded by a competitive environment.
- Min- max is a **decision-making** strategy under **game theory**, which is used to minimize the losing chances in a game and to maximize the winning chances.
- This strategy is also known as '**Minmax,**' '**MM,**' or '**Saddle point.**' Basically, it is a two-player game strategy where *if one wins, the other lose the game.* This strategy simulates those games that we play in our day-to-day life. Like, if two persons are playing chess, the result will be in favor of one player and will unfavor the other one. The person who will make his best *try, efforts as well as cleverness, will surely win.*

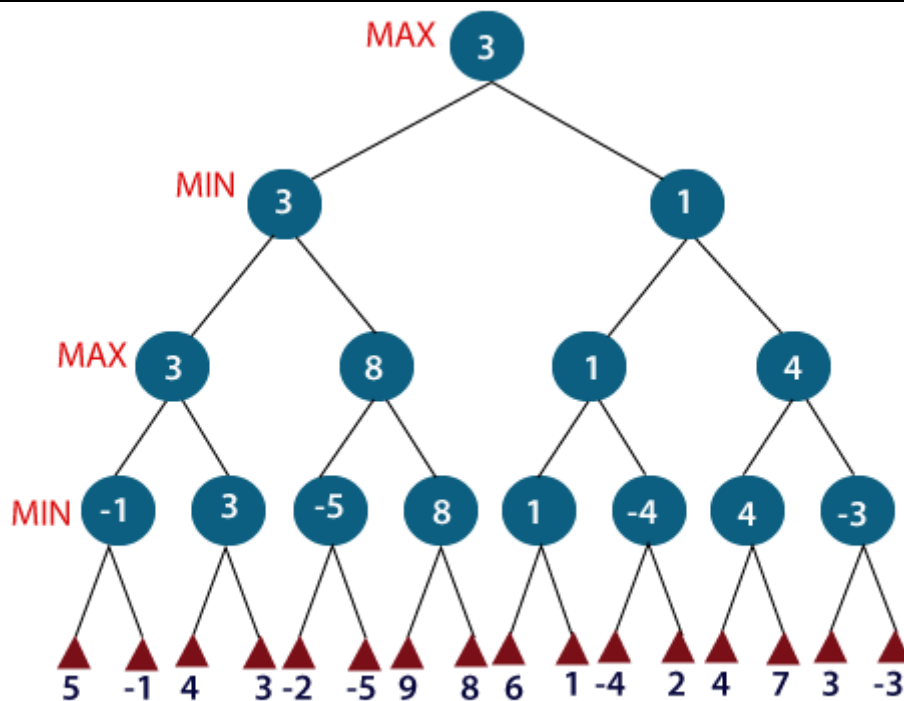
Prerequisite knowledge for Complete understanding and learning of Topic:

Concepts of Game Playing
Components of Artificial Intelligence
Types of Agents and Environment

Detailed content of the Lecture:

MINIMAX algorithm is a backtracking algorithm where it backtracks to pick the best move out of several choices. MINIMAX strategy follows the **DFS (Depth-first search)** concept. Here, we have two players **MIN and MAX**, and the game is played alternatively between them, i.e., when **MAX** made a move, then the next turn is of **MIN**. It means the move made by **MAX** is fixed and, he cannot change it. The same concept is followed in DFS strategy, i.e., we follow the same path and cannot change in the middle. That's why in MINIMAX algorithm, instead of BFS, we follow DFS.

- Keep on generating the game tree/ search tree till a limit **d**.
- Compute the move using a heuristic function.
- Propagate the values from the leaf node till the current position following the minimax strategy.
- Make the best move from the choices.



For example, in the above figure, the two players **MAX** and **MIN** are there. **MAX** starts the game by choosing one path and propagating all the nodes of that path. Now, **MAX** will backtrack to the initial node and choose the best path where his utility value will be the maximum. After this, its **MIN** chance. **MIN** will also propagate through a path and again will backtrack, but **MIN** will choose the path which could minimize **MAX** winning chances or the utility value.

So, if the level is minimizing, the node will accept the minimum value from the successor nodes. If the level is maximizing, the node will accept the maximum value from the successor.

Alpha-beta pruning is an advance version of MINIMAX algorithm. The drawback of minimax strategy is that it explores each node in the tree deeply to provide the best path among all the paths. This increases its time complexity. But as we know, the performance measure is the first consideration for any optimal algorithm. Therefore, alpha-beta pruning reduces this drawback of minimax strategy by less exploring the nodes of the search tree.

The method used in alpha-beta pruning is that it **cutoff the search** by exploring less number of nodes. It makes the same moves as a minimax algorithm does, but it prunes the unwanted branches using the pruning technique (discussed in adversarial search). Alpha-beta pruning works on two threshold values, i.e., **α (alpha)** and **β (beta)**.

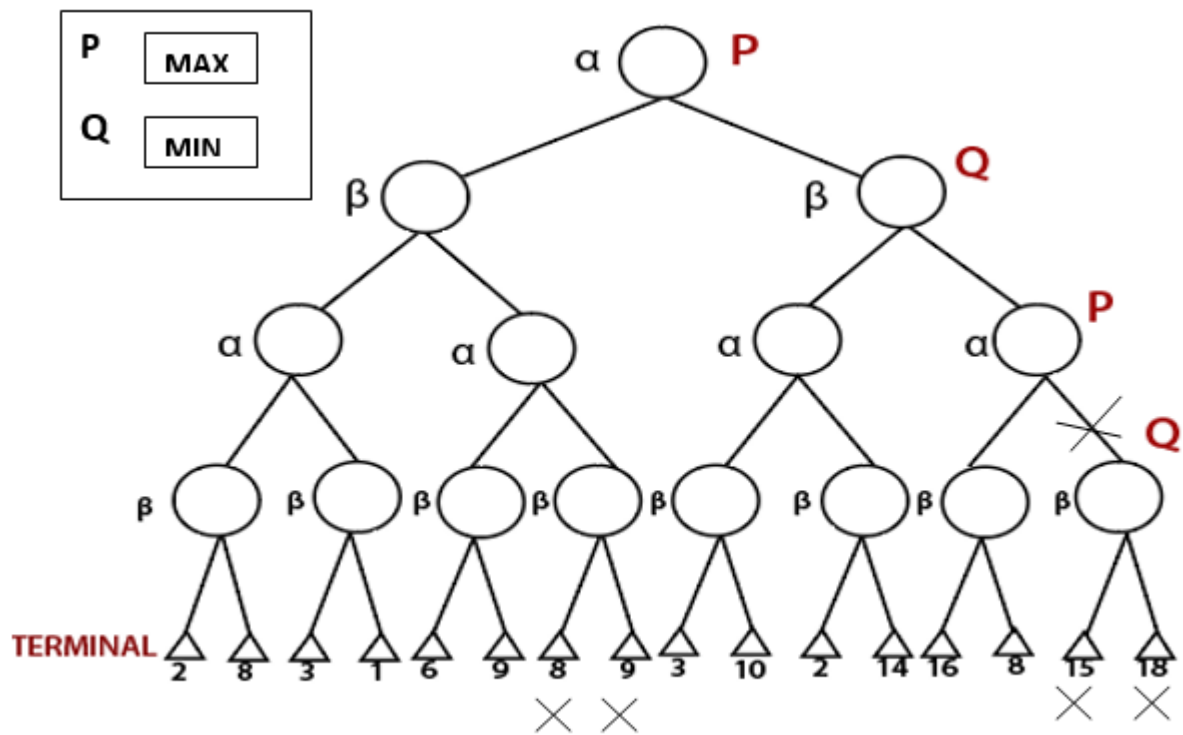
- **α :** It is the best highest value, a **MAX** player can have. It is the lower bound, which represents negative infinity value.
- **β :** It is the best lowest value, a **MIN** player can have. It is the upper bound which represents positive infinity.

So, each MAX node has α -value, which never decreases, and each MIN node has β -value, which never increases.

Note: Alpha-beta pruning technique can be applied to trees of any depth, and it is possible to prune the entire subtrees easily.

Working of Alpha-beta Pruning

Consider the below example of a game tree where **P** and **Q** are two players. The game will be played alternatively, i.e., chance by chance. Let, **P** be the player who will try to win the game by maximizing its winning chances. **Q** is the player who will try to minimize **P**'s winning chances. Here, **α** will represent the maximum value of the nodes, which will be the value for **P** as well. **β** will represent the minimum value of the nodes, which will be the value of **Q**.



Alpha-beta pruning

- Any one player will start the game. Following the DFS order, the player will choose one path and will reach to its depth, i.e., where he will find the **TERMINAL** value.
- If the game is started by player P, he will choose the maximum value in order to increase its winning chances with maximum utility value.
- If the game is started by player Q, he will choose the minimum value in order to decrease the winning chances of A with the best possible minimum utility value.
- Both will play the game alternatively.
- The game will be started from the last level of the game tree, and the value will be chosen accordingly.
- Like in the below figure, the game is started by player Q. He will pick the leftmost value of the **TERMINAL** and fix it for beta (β). Now, the next **TERMINAL** value will be compared with the β -value. If the value will be smaller than or equal to the β -value, replace it with the current β -value otherwise no need to replace the value.
- After completing one part, move the achieved β -value to its upper node and fix it for the other threshold value, i.e., α .
- Now, its P turn, he will pick the best maximum value. P will move to explore the next part only after comparing the values with the current α -value. If the value is equal or greater than the current α -value, then only it will be replaced otherwise we will prune the values.
- The steps will be repeated unless the result is not obtained.
- So, number of pruned nodes in the above example are **four** and MAX wins the game with the maximum **UTILITY** value, i.e., 3

The rule which will be followed is: **“Explore nodes if necessary otherwise prune the unnecessary nodes.”**

Video Content / Details of website for further learning (if any):

<https://www.tutorialandexample.com/alpha-beta-pruning/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L12

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II

Date of Lecture:

Topic of Lecture: Knowledge Representation

Introduction :

- **Knowledge:** The information related to the environment is stored in the machine.
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- Syntax and Semantics

Prerequisite knowledge for Complete understanding and learning of Topic:

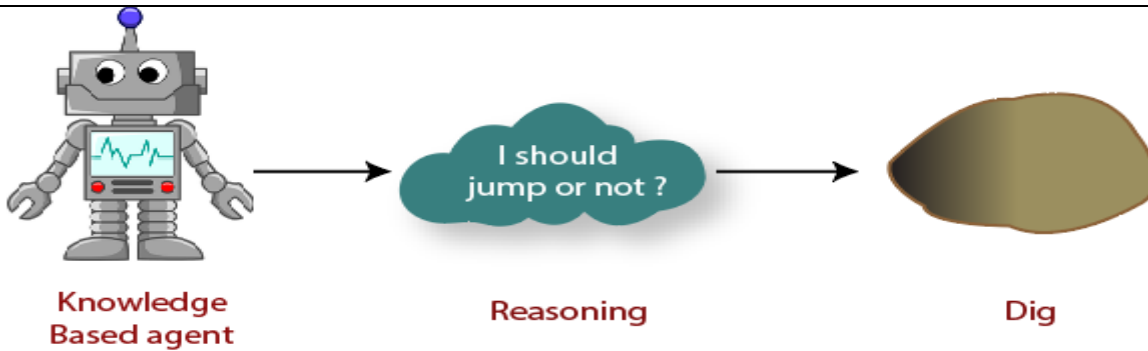
- Discrete Mathematics
- Syntax and Semantics
- Types of Learning
- Logical Reasoning

Detailed Content of the Lecture:

A machine sounds like an empty box unless it is encoded with some features or information. Therefore, to make it a valuable machine, it is required to put the necessary knowledge in it. So that it could understand it and is able to take the right decisions.

There are three factors which are put into the machine, which makes it valuable:

- **Knowledge:** The information related to the environment is stored in the machine.
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.



With the increasing demand for the knowledge representation technique, there was a need for larger and modular knowledge that could integrate and combine with one another. **Ontological engineering** is the engineering of such systems which could represent the complex domains effectively. With the help of ontological engineering, the representation of the general concepts such as **actions, time, physical objects, performance, meta-data, and beliefs** becomes possible on a large-scale.

For special type of representations, we require a special type of ontology known as **Special ontology**. But for special ontologies, there is a need to move towards a high-level generality.

There are two following major characteristics which distinguish general ontologies from the special one:

- General ontologies should be applicable in every type of special-purpose domains with some domain-specific axioms.
- When there is a sufficiently demanding domain, the areas of knowledge should be unified.

Unfortunately, none of the applications of artificial intelligence make use of the shared/general ontology, all they use is special ontology.

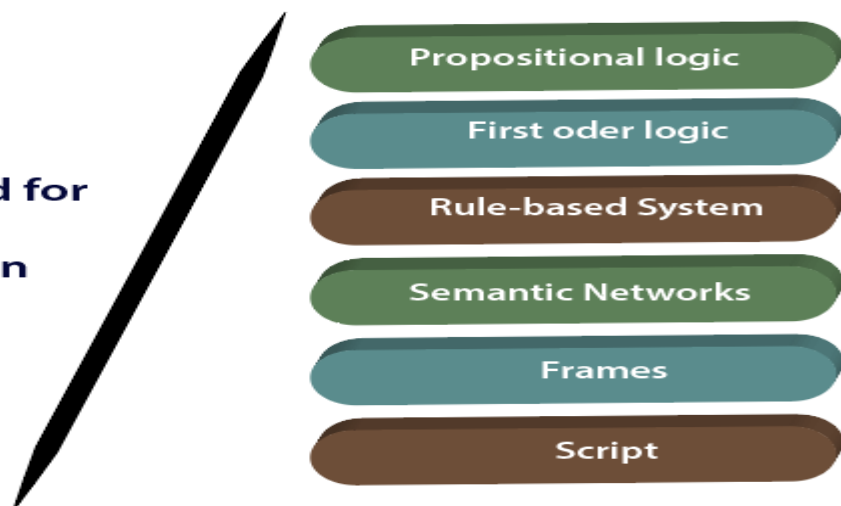
Properties of a Knowledge Representation system:

There are following properties of a Knowledge Representation system:

- **Representational Adequacy:** It is the ability of the system to represent all kinds of knowledge needed in a specific domain.
- **Inferential Adequacy:** It is the ability of a knowledge representation system to manipulate the current stored knowledge so that newly gained knowledge could be added.
- **Inferential Efficiency:** It is the ability of the system to directly add new knowledge in the system with efficiency
- **Acquisitional Efficiency:** It is the ability of the system to automatically acquire new knowledge from the environment. This leads the system to give more productive result as more knowledge adds up with the current knowledge.

Techniques used for Knowledge Representation

Techniques used for Knowledge Representation



- **Logic:** It is the basic method used to represent the knowledge of a machine. The term logic means to apply intelligence over the stored knowledge.

Logic can be further divided as:

- **Propositional Logic:** This technique is also known as **propositional calculus, statement logic, or sentential logic**. It is used for representing the knowledge about **what is true and what is false**.
- **First-order Logic:** It is also known as **Predicate logic or First-order predicate calculus (FOPL)**. This technique is used to represent the objects in the form of **predicates or quantifiers**. It is different from Propositional logic as it removes the complexity of the sentence represented by it. In short, FOPL is an advance version of propositional logic.
- **Rule-based System:** This is the most commonly used technique in artificial intelligence. In the rule-based system, we impose rules over the propositional logic and first-order logic techniques. If-then clause is used for this technique. **For example**, if there are two variables A and B. Value of both A and B is True. Consequently, the result of both should also be True and vice-versa.
- If the value of A and B is True, then the result will be True. So, such a technique makes the propositional as well as FOPL logics bounded in the rules.
- **Semantic Networks:** The technique is based on storing the knowledge into the system in the form of a graph. Nodes of a graph represent the objects which exist in the real world, and the arrow represents the relationship between these objects. Such techniques show the connectivity of one object with another object. **For example**, Consider the given knowledge stored in a machine:

Ram has a cycle.

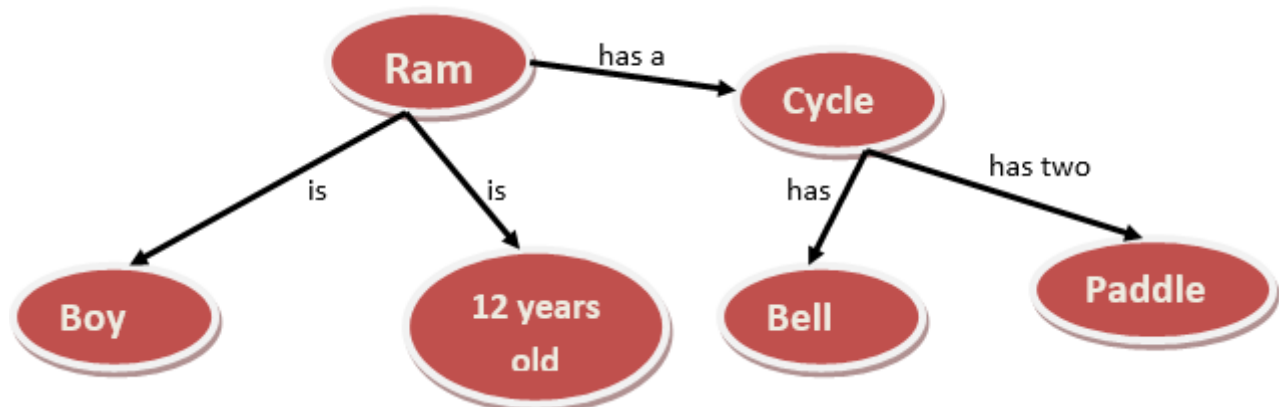
Ram is a boy.

Cycle has a bell.

Ram is 12 years old.

Cycle has two paddles.

Representing the above information in the form of a graph looks like:



- **Frames:** In this technique, the knowledge is stored via **slots and fillers**. As we have seen in DBMS, we store the information of an employee in the database, where:

Similarly, the Slots are the entities and Fillers are its attributes. They are together stored in a frame. So, whenever there is a requirement, the machine infers the necessary information to take the decision. **For example**, Tomy is a dog having one tail.

It can be framed as:



- **Script:** It is an advanced technique over the Frames. Here, the information is stored in the form of a script. The script is stored in the system containing all the required information. The system infers the information from that script and solves the problem

Challenges/Issues in Knowledge Representation

- **Important Attributes:** Some basic attributes were occurring in almost every problem domain.
- **Relationship among attributes:** Any important relationship which exists among object attributes.
- **Choosing Granularity:** How much detailed knowledge is needed to be represented?
- **Set of Objects:** How to represent the set of objects?
- **Finding the right structure:** The information is stored in a large amount. The question is how to access the relevant information out of whole?

Video Content / Details of website for further learning (if any):

https://www.tutorialandexample.com/knowledge_representation

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L13

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II

Date of Lecture:

Topic of Lecture: Knowledge Representation using Predicate Logic

Introduction :

- **Knowledge:** The information related to the environment is stored in the machine.
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

Propositional Logic

It is a branch of logic which is also known as **statement logic, sentential logic, zeroth-order logic**, and many more. It works with the propositions and its logical connectivities. It deals with the propositions or statements whose values are **true, false**, or maybe unknown.

Syntax and Semantics of Propositional Logic

Syntax and semantics define a way to determine the truth value of the sentence.

Syntax: The statements given in a problem are represented via propositional symbols. Each sentence consists of a single propositional symbol. The propositional symbol begins with an uppercase letter and may be followed by some other subscripts or letters. We have two fixed propositional symbols, i.e., True and False.

To convert simple sentences into complex one, following connectives (a connective is used to combine two or more sentences) are used:

- **not(\neg):** It is known as the **negation** of a sentence. A literal can be a positive literal or a negative literal.
- **and(\wedge):** When a sentence is having (\wedge) as the main connective. It is known

as **Conjunction**, and its parts are known as **Conjuncts**. For example, $(Y_1 \vee Y_2) \wedge (Y_3 \vee Y_4) \wedge \dots (Y_n \vee Y_m)$, such type of sentences are known as **Conjunctive sentences**.

- **or(\vee)**: When a sentence is having (\vee) as the main connective. It is known as **Disjunction**, and its parts are known as **Disjuncts**. For example, $(Y_1 \wedge Y_2) \vee (Y_3 \wedge Y_4) \vee \dots (Y_n \wedge Y_m)$, such type of sentences are known as **Disjunctive sentences**.
- **implies(\Rightarrow)**: When $(Y_1 \vee Y_2) \Rightarrow Y_3$ is given, it is known as the **Implication of a sentence**. It is like **if->then** clause, where if this implies then it will happen. Implication is sometimes referred to as **Rules or if-then** statement. It can also be denoted as () or ().
- **if and only if (\Leftrightarrow)**: It represents implication at both sides where the expression is $(a_1 \vee a_2) \Leftrightarrow a_3$. Such type of connective is called **biconditional implication**. It returns true if both sides satisfy one another, else returns false. This can also be denoted as (\equiv).

Precedence Order of the Connectives

Below table shows the precedence order of the connectives in their decreasing order:

Name	Symbol
Parenthesis/ Brackets	()
Negation/not	\neg or \sim
Conjunction/and	\wedge
Disjunction/or	\vee
Implication	\rightarrow
Biconditional/ if and only if	\Leftrightarrow

Semantics: It defines the rules to determine the truth of a sentence with respect to a specific model. A semantic should be able to compute the truth value of any given sentence.

There are following five rules regarding the semantics of the complex sentences P and Q in a given model m :

$\neg P$: Its value will be false, iff it is true in the model m.

$(P \wedge Q)$: Its value is true, iff both P and Q are true in m.

$(P \vee Q)$: Its value is true, iff either P is true, or Q is true in m.

$(P \Rightarrow Q)$: Its value is true, iff the value of P is false, and that of Q is true in m.

$(P \Leftrightarrow Q)$: The value will be true, iff P and Q value is either true or false in the given model m.

Note: Here, iff means if and only if.

These five connectives can also be understood with the help of the below described truth table:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Truth tables for five logical Connectives

Examples of Propositional Logic

Example 1: Consider the given statement:

If it is humid, then it is raining.

Solution: Let, P and Q be two propositions.

P=It is humid.

Q=It is raining.

It is represented as $(P \rightarrow Q)$.

Example 2: It is noon and Ram is sleeping.

Solution: A= It is noon.

B= Ram is sleeping.

It is represented as $(A \vee B)$.

Example 3: If it is raining, then it is not sunny.

Solution: P= It is raining.

Q= It is sunny.

It is represented as $P \rightarrow (\sim Q)$

Example 4: Ram is a man or a boy.

Solution: X= Ram is a man.

Y= Ram is a boy.

It is represented as $(X \wedge Y)$.

Example 5: I will go to Delhi if and only if it is not humid.

Solution: A= I will go to Delhi.

B= It is humid.

It is represented as $(A \Leftrightarrow B)$.

There can be many examples of Propositional logic.

Propositional Theorem Proving

Theorem proving means to apply rules of inference directly to the sentences.

There are following concepts which are used for theorem proving:

- **Logical Equivalence:** If the value of P and Q is true in the same set of models, then they are said to be logically equivalence.

Rule Name	Rule
Idempotency Law	$(A \wedge A) = A(A \vee A) = A$
Commutative Law	$(A \wedge B) = (B \wedge A)(A \vee B) = (B \vee A)$
De morgan's Law	$\sim(A \wedge B) = (\sim A \vee \sim B)\sim(A \vee B) = (\sim A \wedge \sim B)$
Associative Law	$A \vee (B \vee C) = (A \vee B) \vee CA \wedge (B \wedge C) = (A \wedge B) \wedge C$
Distributive Law	$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
Contrapositive Law	$A \rightarrow B = \sim A \rightarrow \sim B$ $\sim A \rightarrow \sim B$ (Converse of Inverse)
Implication Removal	$A \rightarrow B = \sim A \vee B$
Biconditional Removal	$A \Leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$
Absorption Law	$A \wedge (A \vee B) \equiv AA \vee (A \wedge B) \equiv A$

Double-negation elimination

$$\sim(\sim A)=A$$

Table defining the rules used in Propositional logic where A, B, and C represents some arbitrary sentences.

- **Validity:** If a sentence is valid in all set of models, then it is a valid sentence. Validity is also known as **tautology**, where it is necessary to have true value for each set of model.
- **Satisfiability:** If a sentence is true atleast for some set of values, it is a satisfiable sentence.

Let's understand validity and satisfiability with the help of examples:

Example 1:

$$(P \vee Q) \rightarrow (P \wedge Q)$$

P	Q	$P \vee Q$	$P \wedge Q$	$(P \vee Q) \rightarrow (P \wedge Q)$
False	False	False	False	True
False	True	True	False	False
True	False	True	False	False
True	True	True	True	True

So, from the above truth table, it is clear that the given expression is satisfiable but not valid.

Example 2:

A	B	$A \rightarrow B$	$(A \rightarrow B) \wedge A$	$((A \rightarrow B) \wedge A) \rightarrow B$
False	False	True	False	True
False	True	True	False	True
True	False	False	False	True
True	True	True	True	True

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

So, it is clear from the truth table that the given expression is valid as well as satisfiable.

Note: If the given expression is valid, it is by default satisfiable.

Video Content / Details of website for further learning (if any):

<https://www.tutorialandexample.com/theory-of-first-order-logic/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L14



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II

Date of Lecture:

Topic of Lecture: Knowledge Representation using Predicate Logic

Introduction :

- **Knowledge:** The information related to the environment is stored in the machine.
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**
- **Inference Rules in Propositional Logic**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

Inference rules are those rules which are used to describe certain conclusions. The inferred conclusions lead to the desired goal state.

In propositional logic, there are various inference rules which can be applied to prove the given statements and conclude them.

There are following laws/rules used in propositional logic:

- **Modus Tollen:** Let, P and Q be two propositional symbols:

Rule: Given, the negation of Q as ($\sim Q$).

If $P \rightarrow Q$, then it will be ($\sim P$), i.e., the negation of P.

Example: If Aakash goes to the temple, then Aakash is a religious person. Aakash is not a religious person. Prove that Aakash doesn't go to temple.

Solution: Let, P= Aakash goes to temple.

Q= Aakash is religious. Therefore, (\sim Q)= Aakash is not a religious person.

To prove: \sim P \rightarrow \sim Q

By using **Modus Tollens** rule, P \rightarrow Q, i.e., \sim P \rightarrow \sim Q (because the value of Q is (\sim Q)).

Therefore, Aakash doesn't go to the temple.

- **Modus Ponens:** Let, P and Q be two propositional symbols:

Rule: If P \rightarrow Q is given, where P is positive, then Q value will also be positive.

Example: If Sheero is intelligent, then Sheero is smart. Sheero is intelligent. Prove that Sheero is smart.

Solution: Let, A= Sheero is intelligent.

B= Sheero is smart.

To prove: A \rightarrow B.

By using **Modus Ponens** rule, A \rightarrow B where A is positive. Hence, the value of B will be true. **Therefore, Sheero is smart.**

- **Syllogism:** It is a type of logical inference rule which concludes a result by using deducting reasoning approach.

Rule: If there are three variables say P, Q, and R where

P \rightarrow Q and Q \rightarrow R then P \rightarrow R.

Example: Given a problem statement:

If Ram is the friend of Shyam and Shyam is the friend of Rahul, then Ram is the friend of Rahul.

Solution: Let, P= Ram is the friend of Shyam.

Q= Shyam is the friend of Rahul.

R= Ram is the friend of Rahul.

It can be represented as: **If (P \rightarrow Q) \wedge (Q \rightarrow R)= (P \rightarrow R).**

- **Disjunctive Syllogism**

Rule: If (\sim P) is given and (P \vee Q), then the output is Q.

Example: Sita is not beautiful or she is obedient.

Solution: Let, (\sim P)= Sita is beautiful.

Q= She is obedient.

P= Sita is not beautiful.

It can be represented as (P \vee Q) which results Sita is obedient.

Note: Logical equivalence rules can also be used as Inference rules in Proposition logic.

We can also apply the inference rules to the logical equivalence rules as well.

- **Biconditional Elimination:** If A \leftrightarrow B then (A \rightarrow B) \wedge (B \rightarrow A) or
If (A \rightarrow B) \wedge (B \rightarrow A) then A \leftrightarrow B. (using one side implication rule).

- **Contrapositive:** If \neg (A \leftrightarrow B) then \neg ((A \rightarrow B) \wedge (B \rightarrow A))

We can re-obtain it using **De Morgan's** and **Modus Ponens** rule.

Using inference rules, we can also define a proof problem as follows:

- **Initial State:** It is the initial knowledge base.
- **Actions:** It is the set of actions which contains all the inference rules applied over all the sentences that match the top half of the inference rule.
- **Result:** When we add the sentence at the bottom of the inference rule, it gives the result of the applied action.
- **Goal:** It is the state which contains that sentence we are trying to prove.

Video Content / Details of website for further learning (if any):

<https://www.tutorialandexample.com/theory-of-first-order-logic/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L15

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II

Date of Lecture:

Topic of Lecture: Predicate calculus- Inference Rules- Resolution

Introduction :

- **Knowledge:** The information related to the environment is stored in the machine.
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**
- **Predicate Calculus**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

First-order logic is also called **Predicate logic** and **First-order predicate calculus (FOPL)**. It is a formal representation of logic in the form of quantifiers. In predicate logic, the input is taken as an entity, and the output it gives is either true or false.

Syntax and Semantics of FOPL

Syntax: It defines the way of representing the given predicates. As these predicates are represented via quantifiers, there are different types of quantifiers used:

- **Universal Quantifier(For all/every):** When the predicate is indicating about all/everything, we use **for all** quantifier. It is denoted as “ \forall ”
- **Existential Quantifier(For some):** When the predicate is indicating about some quantity, we use **for some** quantifier. It is denoted as “ \exists ”
- **Nested Quantifiers:** It is the nesting of the same type of quantifier. One predicate is nested under the other predicate.
- **Atomic Sentences:** These sentences are formed via predicate symbols may or may not be

followed by a list of terms.

Example: Parents(Ram, Sita) where Ram and Sita are the parents.

- **Complex Sentences:** These sentences make use of logical connectives to construct more complex sentences.
- **Equality:** We use the equality symbol to express that two terms refer to the same object. **For example,** Eleventh_President(India)= Dr. APJ Abdul Kalam. Here, both LHS is equal to RHS. It means that both terms refer to the same entity/ person.

Elements and their symbols in Predicate Logic

The elements for which different symbols are defined are:

- **Objects:** It refers to an entity that exists in the real world. **For example,** Ram, John, etc. are referred to as Objects.
- **Functions:** Any function performed by the object/on the object. **For example,** LeftLeg, writes, eats, etc. are some of the functions.
- **Relations:** The relation of an object with the other object defines its relation. **For example,** brother, mother, king, etc. are some types of relations which exist in the real world.

Now, let's discuss the symbols used to represent these elements. They are as follows:

- **Constant Symbols:** These symbols are used to represent the objects.
- **Predicate Symbols:** These symbols are used to represent relations.
- **Function Symbols:** These symbols are used to represent the functions performed by the object.

Semantics: It defines the sense of the given predicate. It allows to make more logical expression by devising its semantics. Semantics allow us to understand the sentence meaning.

Let's understand Predicate logic with the help of below examples:

Example 1: Lipton is a tea.

Solution: Here, the object is Lipton.

It will be represented as **Tea(Lipton)**.

Note: In this example, there is no requirement of quantifiers because the quantity is not specified in the given predicate. **Let's see more.**

Example 2: Every man is mortal.

Solution: Here, the quantifier is the universal identifier, and the object is man.

Let x be the man.

Thus, it will be represented as **x: man(x) → mortal(x)**.

Example 3: All girls are beautiful.

Solution: Here, we are talking about all girls. It means universal quantifier will be used. The object is girls. Let, y be the girls.

Therefore, it will be represented as **girls(y) → beautiful(y)**.

Example 4: All that glitters is not gold.

Solution: Here, we will represent gold as x.

Therefore, it will be represented as **glitters(x) → ¬gold(x)**.

Example 5: Some boys are obedient.

Solution: Here, boys are objects. The quantifier used will be existential quantifier. Let x be the boys.

Thus, it will be represented as

∃x: boys(x) → obedient(x).

Example 6: Some cows are black and some cows are white.

Solution: Let, x be the cows. Therefore, it will be represented as:

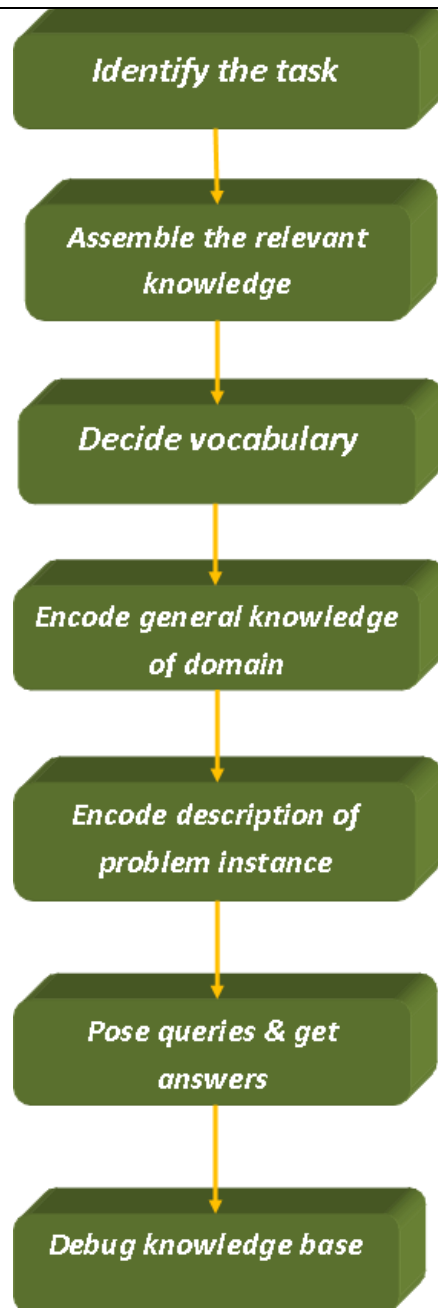
∃x: cows(x) → black(x) ∧ white(x).

Knowledge Engineering in FOPL

Knowledge engineering is the process where a knowledge engineer investigates a specific domain, learn the important concepts regarding that domain, and creates the formal representation of the objects and relations in that domain.

Knowledge Engineering Process

An engineering term is used when we are talking about any project. **Therefore, knowledge engineering over a project involves the below described steps:**



- **Identify the task:** A knowledge engineer should be able to identify the task by asking a few questions like:
- Do the knowledge base will support?
- What kinds of facts will be available for each specific problem?

The task will identify the knowledge requirement needed to connect the problem instance with the answers.

- **Assemble the relevant knowledge:** A knowledge engineer should be an expert in the domain. If not, he should work with the real experts to extract their knowledge. This concept is known as **Knowledge Acquisition**.

Note: Here, we do not represent the knowledge formally. But to understand the scope of the knowledge base and also to understand the working of the domain.

- **Decide on a vocabulary of constants, predicates, and functions:** Translating important domain-level concepts into logical level concepts.

Here, the knowledge engineer asks questions like:

- What are the elements which should be represented as objects?
- What functions should be chosen?

After satisfying all the choices, the vocabulary is decided. It is known as the **Ontology of the domain**. Ontology determines the type of things that exists but does not determine their specific properties and

interrelationships.

- **Encode general knowledge about the domain:** In this step, the knowledge engineer pen down the axioms for all the chosen vocabulary terms.

Note: Here, misconceptions occur between the vocabulary terms.

- **Encode description of the specific problem instance:** We write the simple atomic sentences for the selected vocabulary terms. We encode the chosen problem instances.
- **Raise queries to the inference procedure and get answers:** It is the testing step. We apply the inference procedure on those axioms and problem-specific facts which we want to know.
- **Debug the knowledge base:** It is the last step of the knowledge engineering process where the knowledge engineer debugs all the errors.

Video Content / Details of website for further learning (if any):

<https://www.tutorialandexample.com/theory-of-first-order-logic/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L16



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : II

Date of Lecture:

Topic of Lecture: Knowledge Representation using Predicate Calculus

Introduction :

- **FOPL – First Order Predicate Logic**
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**
- **Inference Rules**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

FOPL offers the following inference rules:

- **Inference rules for quantifiers**
- **Universal Instantiation (UI):** In this, we can infer any sentence by substituting a ground term (a term without variables) for the variables. In short, when we create the FOPL of the given statement, we can easily infer any other statement using that FOPL

Notation: Let, $SUBST(\theta, \alpha)$ be the result of the given sentence α , and its substitute is θ .

The rule can be written as: $\frac{\alpha}{\forall v \alpha}$

$SUBST(\{v/g\}, \alpha)$

where v is the variable and g is the ground term.

For example: Every man is mortal.

It is represented as $\forall x: man(x) \rightarrow mortal(x)$.

In UI, we can infer different sentences as:

man(John) → mortal(John)
man(Aakash) → mortal(Aakash), etc.

- **Existential Instantiation(EI):** In EI, the variable is substituted by a single new constant symbol.

Notation: Let, the variable be **v** which is replaced by a constant symbol **k** for any sentence α . The value of **k** is unique as it does not appear for any other sentence in the knowledge base. Such type of constant symbols are known as **Skolem constant**. As a result, EI is a special case of **Skolemization process**.

Note: UI can be applied several times to produce many sentences, whereas EI can be applied once, and then the existentially quantified sentences can be discarded.

For example: $\exists x:steal(x, Money)$.

We can infer from this: **steal(Thief, Money)**

- **Generalized Modus Ponens:** It is a lifted version of Modus Ponens as it uplifts the Modus Ponens from ground propositions to FOPL. Generalized Modus Ponens is more generalized than Modus Ponens. It is because, in generalized, the known facts and the premise of the implication are matched only upto a substitution, instead of its exact match.

Notation: For atomic sentences like $p_i.p_i'$ and q where we have a substitute θ that **SUBST**(θ, p_i) for each i .

Thus, $p_1', p_2', \dots, p_n', (p_1 p_2 \dots p_n \Rightarrow q)$
SUBST(θ, q)

For example:

p_1' is Thief(Charlie)	p_1 is Thief(x)
p_2' is Silent(y)	p_2 is Silent(y)
θ will be {x/Charlie, y/Charlie}	q is evil(x)
SUBST (θ, q) is evil(Charlie).	

- **Unification:** It is the key component of First-order inference algorithms. Unification is the process used by the lifted inference rules to find substituents that could give identical but different logical expressions. It means the meaning of the sentence should not be changed, but it should be expressed in multiple ways. The **UNIFY** algorithm in unification takes two sentences as input and then returns a unifier if one exists:

UNIFY(p,q) = θ where **SUBST(θ, p) = **SUBST**(θ, q).**

Let see how **UNIFY** works with the help of the below example:

Given: Knows(Ram,x). The question is- **Whom does Ram knows?**

The **UNIFY** algorithm will search all the related sentences in the knowledge base, which could unify with Knows(Ram,x).

UNIFY (Knows(Ram, x), Knows(Ram, Shyam)) = {x/Shyam}

UNIFY (Knows{Ram,x}, Knows{y, Aakash}) = {x/Aakash, y/Ram}

UNIFY (Knows{Ram,x}, Knows{x, Raman}) = fails.

The last one failed because we have used the same variable for two persons at the same time.

Unification Algorithm

Earlier, we have studied **TELL** and **ASK** functions which are used to inform and interrogate a knowledge base. These are the primitive functions of **STORE** and **FETCH** functions. **STORE** function is used to store a sentence **s** into the knowledge base and **FETCH** function is used to return all the unifiers with some

function **UNIFY**(a, b, θ) returns a substitution to make a, b identical
inputs: a, a variable, constant, list, or compound expression

```

a, a variable, constant, list, or compound expression
θ, the substitution built up earlier (optional, defaults to empty)
if θ = failure then return failure
else if a = b then return θ
else if VARIABLE?(a) then return UNIFY-VAR(a, b, θ)
else if VARIABLE?(b) then return UNIFY-VAR(a, b, θ)
else if COMPOUND?(a) and COMPOUND?(b) then
return UNIFY(a.ARGS, b.ARGS, UNIFY(a.OP, b.OP, θ))
else if LIST?(a) and LIST?(b) then
return UNIFY(a.REST, b.REST, UNIFY(a.FIRST, b.FIRST, θ))
else return failure
function UNIFY-VAR(var, a, θ) returns a substitution
if {var/val} ∈ θ then return UNIFY(val, a, θ)
else if {a/val} ∈ θ then return UNIFY(var, val, θ)
else if OCCUR-CHECK?(var, a) then return failure
else return add {var/a} to θ

```

Note: In the above example: Knows(Ram, x) → It is an instance of FETCH function.

- **Forward Chaining:** In forward chaining, we start with the atomic sentences in the knowledge base and apply Modus Ponens in forward direction. Also adding new sentences until any inference is not made.

Note: We will understand Forward Chaining in detail in Forward Chaining section.

- **Backward Chaining:** This algorithm is opposite of forward chaining algorithm. It works in backward direction. It starts from the goal, chain through the rules to search the known facts and infer sentences from the given facts in backward direction.

Note: We will understand Backward Chaining in detail in Backward Chaining section.

- **Resolution Method:** Unification, Backward Chaining as well as Forward Chaining, all are based on the resolution method. The resolution method is a decision-making rule where a machine decides what to do and what not to do. It can be understood as:

Firstly, we convert the given statement into FOPL.

Secondly, we infer some related sentences from it.

Then, realizing our goal, we need to prove it.

Here, we make use of the resolution method to decide the best possible FOPL from various.

Video Content / Details of website for further learning (if any):

<https://www.tutorialandexample.com/theory-of-first-order-logic/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L17



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit II

Date of Lecture:

Topic of Lecture: Other Knowledge Representation- Structured Knowledge Representation

Introduction :

- **FOPL – First Order Predicate Logic**
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**
- **Inference Rules**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

Approaches to knowledge representation:

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

1. Simple relational knowledge:

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

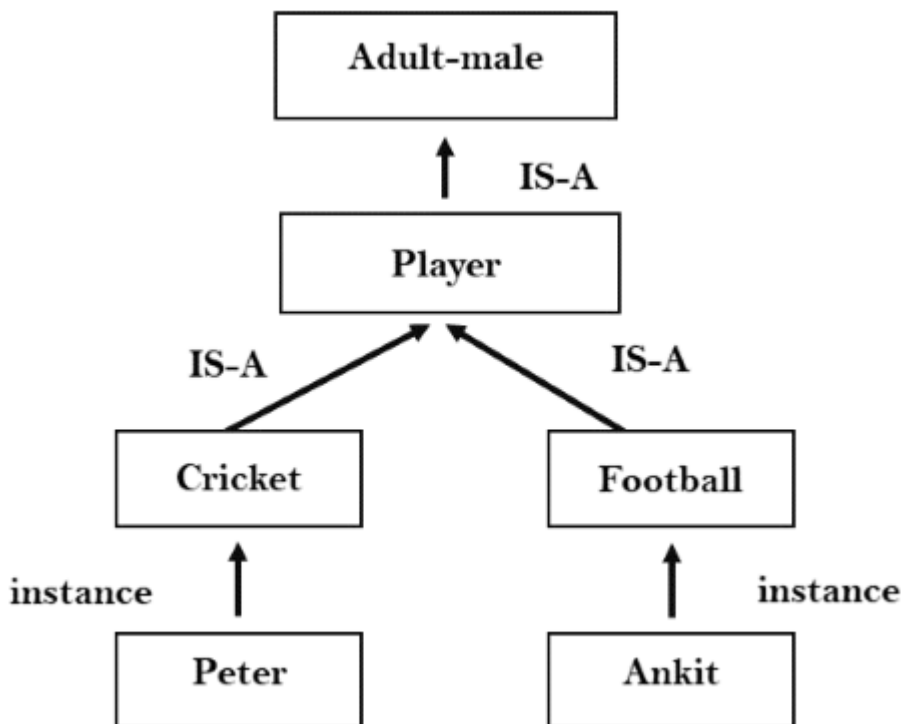
- This approach has little opportunity for inference.

Example: The following is the simple relational knowledge representation.

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

2. Inheritable knowledge:

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



3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.

- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
 - a. Marcus is a man
 - b. All men are mortal
 Then it can represent as;

man(Marcus)

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

4. Procedural knowledge:

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

Requirements for knowledge Representation system:

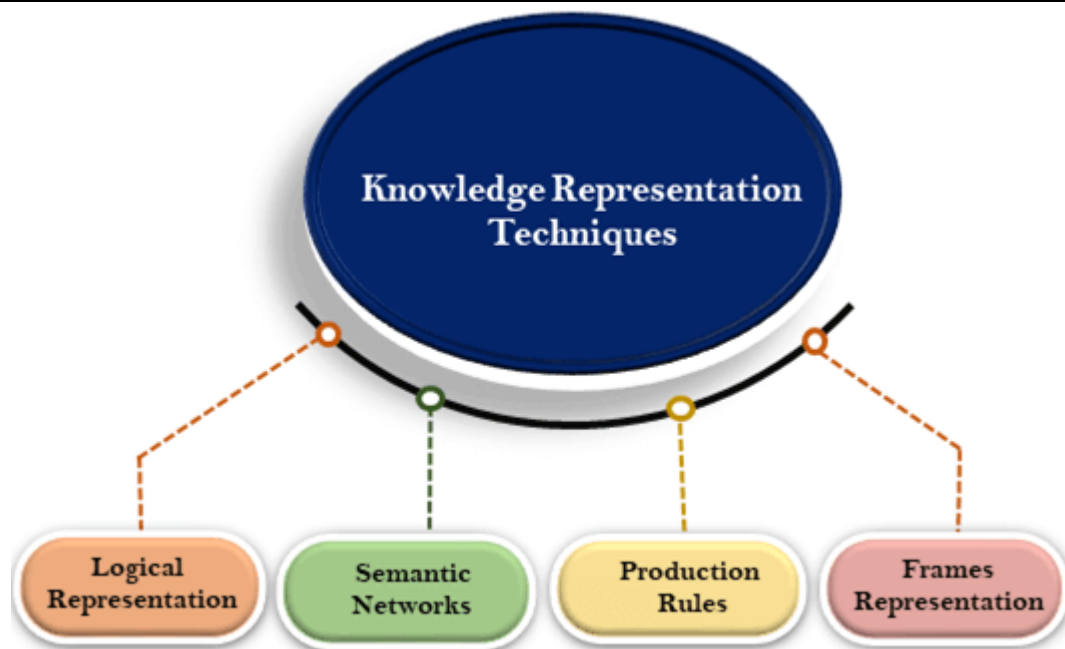
A good knowledge representation system must possess the following properties.

1. **1.Representational Accuracy:**
KR system should have the ability to represent all kind of required knowledge.
2. **2. Inferential Adequacy:**
KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.
3. **3. Inferential Efficiency:**
The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
4. **4. Acquisitional efficiency-** The ability to acquire the new knowledge easily using automatic methods.

Techniques of knowledge representation:

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules



1. Logical Representation

Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

Syntax:

- Syntaxes are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

Semantics:

- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.

Logical representation can be categorised into mainly two logics:

- Propositional Logics
- Predicate logics

Note: We will discuss Prepositional Logics and Predicate logics in later chapters.

Advantages of logical representation:

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

Disadvantages of logical Representation:

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.

Note: Do not be confused with logical representation and logical reasoning as logical representation is a representation language and reasoning is a process of thinking logically.

2. 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.

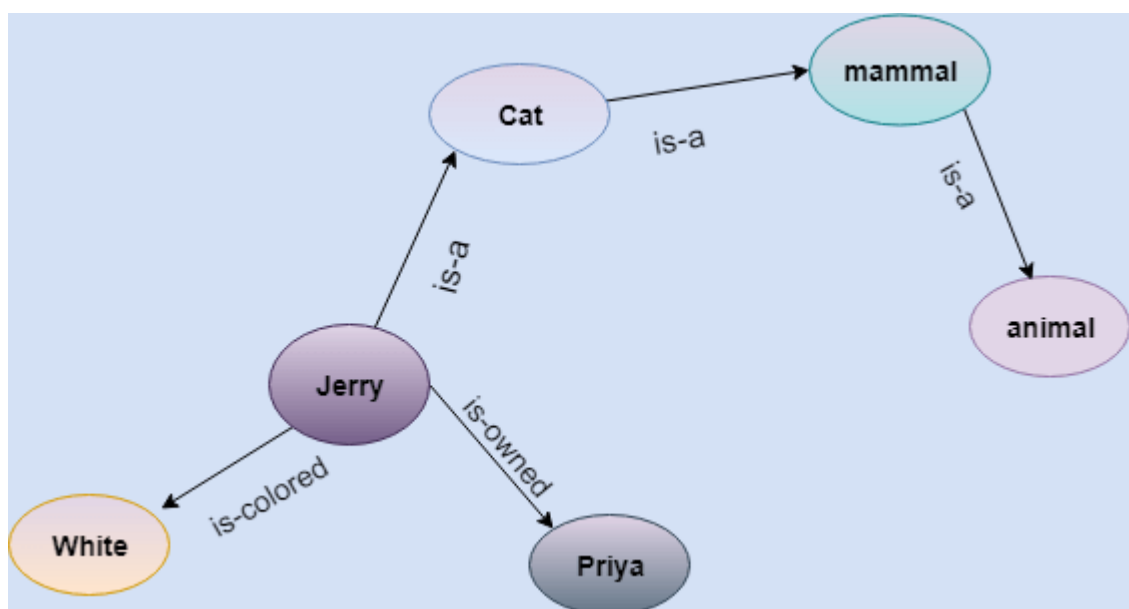
This representation consist of mainly two types of relations:

- IS-A relation (Inheritance)
- Kind-of-relation

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

Statements:

- Jerry is a cat.
- Jerry is a mammal
- Jerry is owned by Priya.
- Jerry is brown colored.
- 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.

Video Content / Details of website for further learning (if any):

<https://www.javatpoint.com/ai-techniques-of-knowledge-representation>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L18



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit II

Date of Lecture:

Topic of Lecture: Other Knowledge Representation- Structured Knowledge Representation

Introduction :

- **FOPL – First Order Predicate Logic**
- **Reasoning:** The ability of the machine to understand the stored knowledge.
- **Intelligence:** The ability of the machine to make decisions on the basis of the stored information.
- **Syntax and Semantics**
- **Inference Rules**

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Syntax and Semantics
- Logical Reasoning

Detailed Content of the Lecture:

Approaches to knowledge representation:

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

3. Frame Representation

- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.
- **Facets:** The various aspects of a slot is known as **Facets**. Facets are features of frames which

enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

- Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example: 1

Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Example 2:

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.

3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly preceded by frame representation.
3. Frame representation has a much generalized approach.

4. Production Rules

Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.

If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

Example:

- **IF (at bus stop AND bus arrives) THEN action (get into the bus)**
- **IF (on the bus AND paid AND empty seat) THEN action (sit down).**
- **IF (on bus AND unpaid) THEN action (pay charges).**
- **IF (bus arrives at destination) THEN action (get down from the bus).**

Advantages of Production rule:

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

Disadvantages of Production rule:

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. During the execution of the program, many rules may be active hence rule-based production

systems are inefficient.

Video Content / Details of website for further learning (if any):

<https://www.javatpoint.com/ai-techniques-of-knowledge-representation>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Machine Learning-Supervised learning

Introduction :

- Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves
- Regularization comes to picture when the model is either overfit or underfit. It is basically used to minimize the error in the dataset. A new piece of information is fit in the data set to avoid fitting issues

Prerequisite knowledge for Complete understanding and learning of Topic:

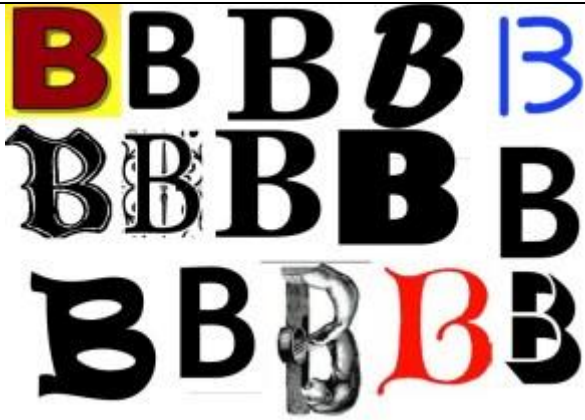
- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed content of the Lecture:

Machine Learning is a set of rules that a computer develops on its own to correctly solve problems. The basic idea is that a Machine Learning computer will find patterns in data and then predict the outcome of something it has never seen before. Machine Learning is a critical component to any Artificial Intelligence (AI) development

Until recently Machine Learning was not possible because we lacked the very large data sets computers require to find patterns in, the storage capacity to keep all of that data and the computing power to find those patterns in a reasonable amount of time

Supervised Learning is a type of machine learning that feeds a computer system many of examples of a given item and having the computer calculate the similarities between those items so that it can recognize other examples of that item which it has not seen yet. For example, if you fed the computer the following set of graphics (and thousands more!) and told it they were all examples of the capital letter B:



it should be able to calculate distances between various parts of each of those letters to develop ratios that let it identify the following letter B graphic even though it has never 'seen' before like this one:



Video Content / Details of website for further learning (if any):

<https://www.sciencedirect.com/topics/computer-science/supervised-learning>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 401

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Unsupervised learning-Reinforcement Learning

Introduction :

- **Unsupervised learning** is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human-labeled data, unsupervised learning, also known as self-organization allows for modeling of probability densities over inputs.
- It forms one of the three main categories of machine learning, along with supervised and reinforcement learning. Semi-supervised learning, a related variant, makes use of supervised and unsupervised techniques.
- **Reinforcement Learning** is a type of Machine Learning, and thereby also a branch of Artificial Intelligence. It allows machines and software agents to automatically determine the ideal behaviour within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed content of the Lecture:

Unsupervised learning

Two of the main methods used in unsupervised learning are principal component and cluster analysis. Cluster analysis is used in unsupervised learning to group, or segment, datasets with shared attributes in order to extrapolate algorithmic relationships. Cluster analysis is a branch of machine learning that groups the data that has not been labelled, classified or categorized. This approach helps detect anomalous data points that do not fit into either group.

- The process of unsupervised learning is required in many recognition problems, where the target pattern is unknown. The unsupervised Learning process attempts to generate a unique set of weights for one particular class of patterns

- The objective of unsupervised learning process is to adjust the weights autonomously, until an equilibrium condition is reached when the weights do not change further.
- The process of unsupervised learning, thus, maps a class of objects to a class of weights. Generally, the weight adaptation process is described by a recursive functional relationship. Depending on the topology of neural nets and their applications, these recursive relations are constructed intuitively.

Reinforcement Learning

- This type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment.
- The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and accordingly adjusts its parameters
- Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters.
- The self organizing neural learning may be categorized under this type of learning.

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Unsupervised_learning

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 419

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Learning by Inductive Logic Programming- Computational Learning Theory

Introduction :

Inductive Logic programming

The theory of ILP is based on proof theory and model theory for the first order predicate calculus. Inductive hypothesis formation is characterised by techniques including inverse resolution , relative least general generalisations , inverse implication , and inverse entailment. The removal of redundancy, and use of search procedures also play an important role in the theory

computational learning theory (or just learning theory) is a subfield of Artificial Intelligence devoted to studying the design and analysis of machine learning algorithms.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed Content of the Lecture

Inductive Logic Programming systems develop predicate descriptions from examples and background knowledge. The examples, background knowledge and final descriptions are all described as logic programs. A unifying theory of Inductive Logic Programming is being built up around lattice-based concepts such as refinement, least general generalisation, inverse resolution and most specific corrections.

- Inductive Logic Programming (ILP) is a sub territory of AI which deals with the induction of hypothesized predicate definitions from examples and background knowledge. Logic programs are treated as a single representation, for example, background knowledge, and hypotheses.
- ILP is differentiated from the other forms of machine learning both by its use of an expressive representation language and its ability to make use of logically encoded background knowledge.
- Molecular biology and natural language are two significant areas of successful applications of ILP. Both the fields have rich sources of background knowledge and benefit from the use of an expressive concept representation languages.

- In the natural language area, ILP has not demonstrated higher accuracies than various other machine learning approaches in learning the past tense of English but also better capabilities of learning accurate grammar which translate sentences into deductive database queries.
- Language applications of ILP, however, would require revision and extension of a hierarchically defined set of predicates in which the examples are typically only provided for predicates at the top of the hierarchy. New predicates often need to be invented, and complex recursion is usually involved.
- Theoretical results in machine learning mainly deal with a type of inductive learning called supervised learning. In supervised learning, an algorithm is given samples that are labeled in some useful way.
- For example, the samples might be descriptions of mushrooms, and the labels could be whether or not the mushrooms are edible. The algorithm takes these previously labeled samples and uses them to induce a classifier.
- This classifier is a function that assigns labels to samples including the samples that have never been previously seen by the algorithm. The goal of the supervised learning algorithm is to optimize some measure of performance such as minimizing the number of mistakes made on new samples.
- In addition to performance bounds, computational learning theory studies the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results:
 - Positive results – Showing that a certain class of functions is learnable in polynomial time.
 - Negative results – Showing that certain classes cannot be learned in polynomial time.

Negative results often rely on commonly believed, but yet unproven assumptions, such as:

- Computational complexity – $P \neq NP$ (the P versus NP problem);
- Cryptographic – One-way functions exist.

There are several different approaches to computational learning theory. These differences are based on making assumptions about the inference principles used to generalize from limited data. This includes different definitions of probability (see frequency probability, Bayesian probability) and different assumptions on the generation of samples.

Video Content / Details of website for further learning (if any):

<https://cs.uwaterloo.ca/~klarson/teaching/W15-486/lectures/22Colt.pdf>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 427

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Neural Nets- Artificial Neural Nets

Introduction :

- **Artificial neural networks (ANN) or connectionist systems** are computing systems vaguely inspired by the biological neural networks that constitute animal brains.^[1] Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules.
- For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

Prerequisite knowledge for Complete understanding and learning of Topic:

- **Discrete Mathematics**
- **Statistics and Probability**
- **Logical Reasoning**

Detailed Content of the Lecture:

- An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.
- In ANN implementations, the "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a *weight* that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection.
- Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.
- The original goal of the ANN approach was to solve problems in the same way that a human brain would. But over time, attention moved to performing specific tasks, leading to deviations from biology. ANNs have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, medical diagnosis, and even in activities that have traditionally been considered as reserved to humans, like painting.

Video Content / Details of website for further learning (if any):

<https://www.google.com/search?q=NURAL+NETS&oq=NURAL+NETS&aqs=chrome.0.69i59j0l7.6161j0j8&sourceid=chrome&ie=UTF-8>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 433

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Topology of AI- Learning using Neural Nets

Introduction :

- Artificial neural nets have been successfully used for recognizing objects from their feature patterns. For classification of patterns, the neural networks should be trained prior to the phase of recognition process

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed Content of the Lecture:

The process of training a neural net can be broadly classified into three typical categories, namely,

- Supervised learning
 - Unsupervised learning
 - Reinforcement learning
- The supervised learning process requires a trainer that submits both the input and the target patterns for the objects to get recognized.
 - The process of unsupervised learning is required in many recognition problems, where the target pattern is unknown. The unsupervised Learning process attempts to generate a unique set of weights for one particular class of patterns
 - Reinforcement learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment

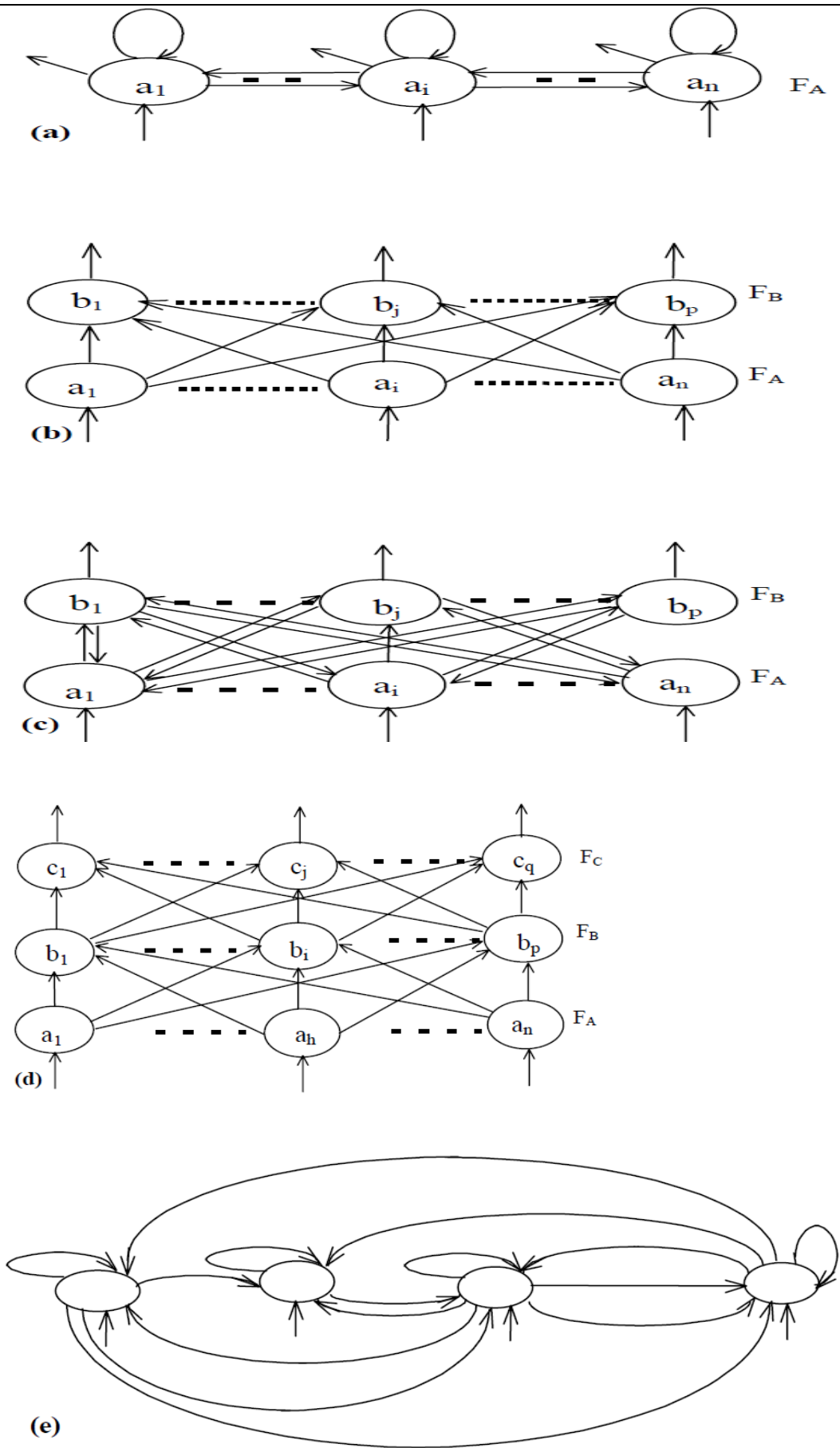


Fig.14.4: Common topologies of artificial neural net: (a) single layered recurrent net with lateral feedback, (b) two layered feed-forward structure, (c) two layered structure with feedback, (d) three layered feed-forward structure, (e) a recurrent structure with self-loops.

Video Content / Details of website for further learning (if any):

<https://www.google.com/search?q=NURAL+NETS&oq=NURAL+NETS&aqs=chrome.69i59j0l7.61j0j8&sourceid=chrome&ie=UTF-8>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 430

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Back Propagation Training Algorithm

Introduction :

- Backpropagation, an abbreviation for "backward propagation of errors" is a common method of training artificial neural networks.
- The method calculates the gradient of a loss function with respects to all the weights in the network.
- The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

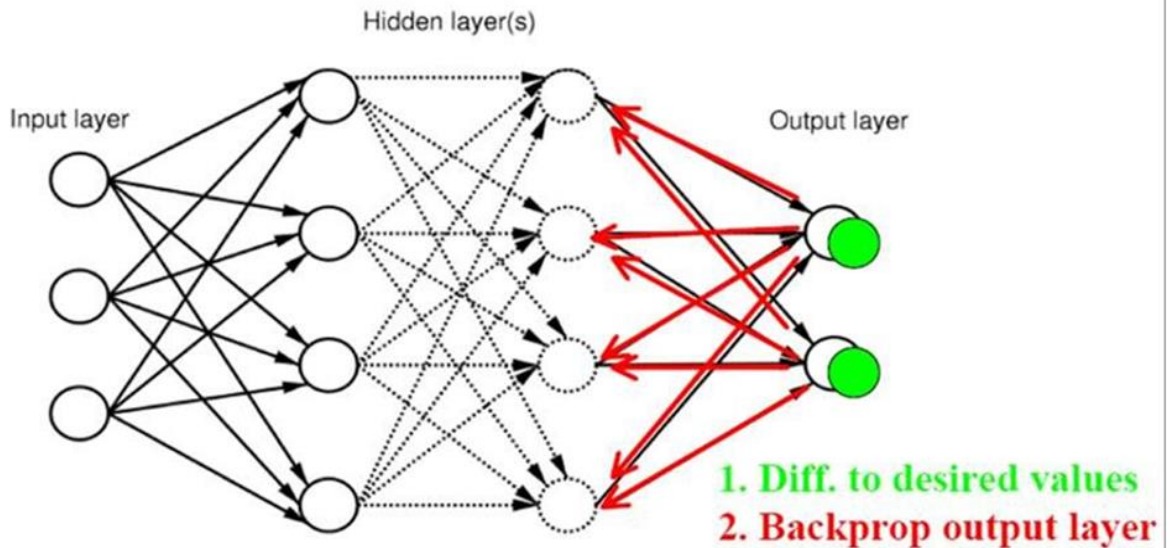
Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed Content of the Lecture:

- *Learning* is the process of modifying the weights in order to produce a network that performs some function.

Backpropagation Learning



Back Propagation Learning Algorithm

It is the most commonly used generalization of the delta rule. This procedure involves two phases

- (i) **Forward phase:** when the input is presented, it propagates forward through the network to compute output values for each processing element. For each PE all the current outputs are compared with the desired outputs and the error is computed.
- (ii) **Backward phase:** The calculated error is now fed backward and weights are adjusted.

After completing both the phases, a new input is presented for the further training.

This technique is slow and can cause instability and has tendency to stuck in a local minima, but it is still very popular.

Algorithm

The following steps is the recursive definition of algorithm:

Step:-

1. Randomly choose the initial weights.
2. For each training pattern apply the inputs to the network.
3. Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer.
4. Calculate the error at the outputs:
 - Use the output error to compute error signals for pre-output layers.
$$\text{Error}_B = \text{Output}_B(1 - \text{Output}_B) (\text{Target}_B - \text{Output}_B)$$

Applications

- Mapping character strings into phonemes so they can be pronounced by a computer
- Neural network trained how to pronounce each letter in a word in a sentence, given the three letters before and three letters after it in a window
- In the field of Speech Recognition
- In the field of Character Recognition
- In the field of Face Recognition

Video Content / Details of website for further learning (if any):

<https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 440

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : III

Date of Lecture:

Topic of Lecture: Multi-Layered ADALINE Models

Introduction :

- **ADALINE (Adaptive Linear Neuron** or later **Adaptive Linear Element**) is an early single-layer artificial neural network and the name of the physical device that implemented this network. The network uses memistors
- It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch–Pitts neuron. It consists of a weight, a bias and a summation function
- The difference between Adaline and the standard (McCulloch–Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights
- A multilayer network of ADALINE units is known as a **MADALINE**

Prerequisite knowledge for Complete understanding and learning of Topic:

- **Discrete Mathematics**
- **Statistics and Probability**
- **Logical Reasoning**

Detailed Content of the Lecture:

- ADALINE units in its hidden and output layers, i.e. its activation function is the sign function.^[9] The three-layer network uses memistors. Three different training algorithms for MADALINE networks, which cannot be learned using back propagation because the sign function is not differentiable, have been suggested, called Rule I, Rule II and Rule III
- The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.^[10] The second training algorithm improved on Rule I and was described in 1988.^[8] The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation

The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction,
- tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced,
- stops when the error is zero.

Additionally, when flipping single units' signs does not drive the error to zero for a particular example, the training algorithm starts flipping pairs of units' signs, then triples of units, etc

Video Content / Details of website for further learning (if any):

<https://en.wikipedia.org/wiki/ADALINE>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 447

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit III

Date of Lecture:

Topic of Lecture: Hopfield Neural Net-Associative Memory

- **Introduction :** The Hopfield network (model) consists of a set of neurons and a corresponding set of unit delays, forming a multiple-loop feedback system
- An associative memory is a content-addressable structure that maps a set of input patterns to a set of output patterns

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Logical Reasoning

Detailed Content of the Lecture:

4. The number of feedback loops is equal to the number of neurons.
5. The output of each of each neuron is fed neuron is fed back via a unit via a unit delay element, to each of the other neurons in the network. ◻ i.e., there is no self-feedback in the network
6. Two types of associative memory: auto associative and hetero associative.
 - Auto-association ◻ retrieves a previously stored pattern that most closely resembles the current pattern.
 - Hetero-association ◻ the retrieved pattern is, in general, different from the input pattern not only in content but possibly also in type and format.

Video Content / Details of website for further learning (if any):

<http://ce.sharif.edu/courses/92-93/1/ce957-1/resources/root/Lectures/Lecture23.pdf>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 453

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to
Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit III

Date of Lecture:

Topic of Lecture: Fuzzy Neural Nets- Self Organizing Neural Net-Adaptive Resonance Theory.

Introduction :

- A **fuzzy neural network** or **neuro-fuzzy system** is a learning machine that finds the parameters of a fuzzy system (i.e., fuzzy sets, fuzzy rules) by exploiting approximation techniques from neural networks.
- **Adaptive resonance theory (ART)** is a theory developed by Stephen Gross berg and Gail Carpenter on aspects of how the brain processes information. It describes a number of neural network models which use supervised and unsupervised learning methods, and address problems such as pattern recognition and prediction.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete Mathematics
- Statistics and Probability
- Fuzzy Logics

Detailed Content of the Lecture:

Combining fuzzy systems with neural networks

- Both neural networks and fuzzy systems have some things in common. They can be used for solving a problem (e.g. pattern recognition, regression or density estimation) if there does not exist any mathematical model of the given problem. They solely do have certain disadvantages and advantages which almost completely disappear by combining both concepts.
- Neural networks can only come into play if the problem is expressed by a sufficient amount of observed examples. These observations are used to train the black box. On the one hand no prior knowledge about the problem needs to be given. On the other hand, however, it is not straightforward to extract comprehensible rules from the neural network's structure.
- On the contrary, a fuzzy system demands linguistic rules instead of learning examples as prior knowledge. Furthermore the input and output variables have to be described linguistically. If the

knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. Since there is not any formal approach for it, the tuning is performed in a heuristic way. This is usually very time consuming and error-prone.

Neural Networks	Fuzzy Systems
no mathematical model necessary	no mathematical model necessary
learning from scratch	apriori knowledge essential
several learning algorithms	not capable to learn
black-box behavior	simple interpretation and implementation

Table 1: Comparison of neural control and fuzzy control

It is desirable for fuzzy systems to have an automatic adaption procedure which is comparable to neural networks. As it can be seen in Table 1, combining both approaches should unite advantages and exclude disadvantages.

Characteristics

Compared to a common neural network, connection weights and propagation and activation functions of fuzzy neural networks differ a lot. Although there are many different approaches to model a fuzzy neural network (Buckley and Hayashi, 1994, 1995; Nauck and Kruse, 1996), most of them agree on certain characteristics such as the following:

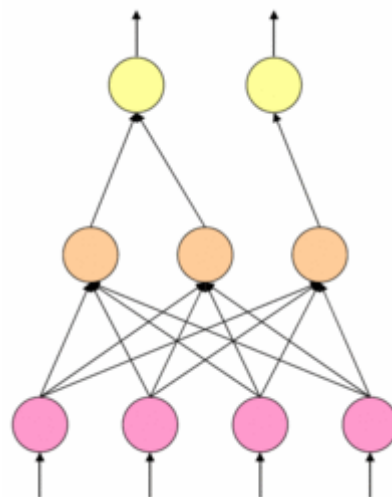


Figure 1: The architecture of a neuro-fuzzy system

- A neuro-fuzzy system based on an underlying fuzzy system is trained by means of a data-driven learning method derived from neural network theory. This heuristic only takes into account local information to cause local changes in the fundamental fuzzy system.
- It can be represented as a set of fuzzy rules at any time of the learning process, i.e., before, during and after. Thus the system might be initialized with or without prior knowledge in terms of fuzzy rules.
- The learning procedure is constrained to ensure the semantic properties of the underlying fuzzy

system.

- A neuro-fuzzy system approximates a n-dimensional unknown function which is partly represented by training examples. Fuzzy rules can thus be interpreted as vague prototypes of the training data.
 - The first layer corresponds to the input variables.
 - The second layer symbolizes the fuzzy rules.
 - The third layer represents the output variables.
 - The fuzzy sets are converted as (fuzzy) connection weights.
 - Some approaches also use five layers where the fuzzy sets are encoded in the units of the second and fourth layer, respectively. However, these models can be transformed into a three-layer architecture.

Cooperative Fuzzy Neural Network

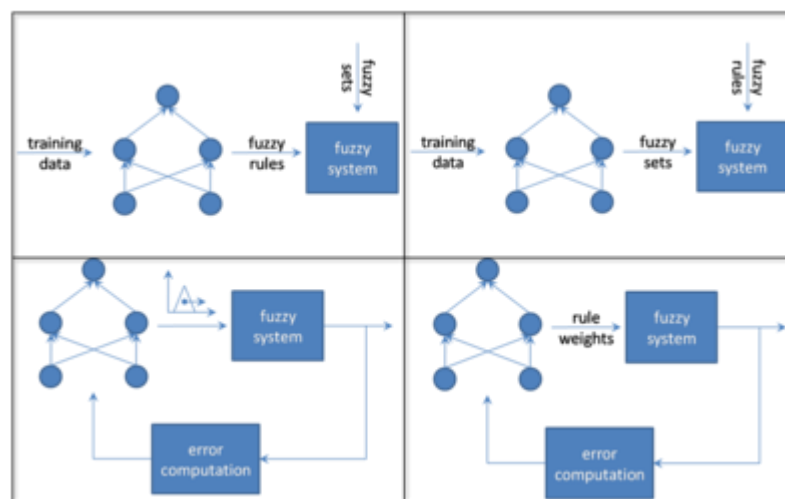


Figure 2: Different cooperative fuzzy neural networks

- In the case of cooperative neural fuzzy systems, both artificial neural network and fuzzy system work independently from each other. The ANN tries to learn the parameters from the fuzzy system. This can be either performed offline or online while the fuzzy system is applied. Figure 2 depicts four different kinds of cooperative fuzzy neural networks.
- The upper left fuzzy neural network learns fuzzy set from given training data. This is usually performed by fitting membership functions with a neural network. The fuzzy sets are then determined offline. They are then utilized to form the fuzzy system by fuzzy rules that are given (not learned) as well.
- The upper right neuro-fuzzy system determines fuzzy rules from training data by a neural network. Here as well, the neural networks learns offline before the fuzzy system is initialized. The rule learning usually done by clustering on self-organizing feature maps (Bezdek et al., 1992; Vuorimaa, 1994). It is also possible to apply fuzzy clustering methods to obtain rules.
- In the lower left neuro-fuzzy model, the system learns all membership function parameters online, i.e., while the fuzzy system is applied. Thus initially fuzzy rules and membership functions must be defined beforehand. Moreover, the error has to be measured in order to improve and guide the learning step.

- The lower right one determines rule weights for all fuzzy rules by a neural network. This can be done online and offline. A rule weight is interpreted as the influence of a rule (Kosko, 1992). They are multiplied with the rule output.
- In (Nauck et al., 1997) the authors argue that the semantics of rule weights are not clearly defined. They could be replaced by modified membership functions. However, this could destroy the interpretation of fuzzy sets. Moreover, identical linguistic values might be represented differently in dissimilar rules.
- **self-organizing fuzzy neural network model** is presented which is able to learn and reproduce different sequences accurately. Sequence learning is important in performing skillful tasks, such as writing and playing piano.
- The structure of the proposed network is composed of two parts: 1-sequence identifier which computes a novel sequence identity value based on initial samples of a sequence, and detects the sequence identity based on proper fuzzy rules, and 2-sequence locator, which locates the input sample in the sequence.
- Therefore, by integrating outputs of these two parts in fuzzy rules, the network is able to produce the proper output based on current state of the sequence

Adaptive Resonance Theory (ART)

- On-line clustering algorithm
- Recurrent ANN
- Competitive output layer
- Data clustering applications
- Stability-plasticity dilemma
- Stability: system behaviour doesn't change after irrelevant events
- Plasticity: System adapts its behaviour according to significant events
- Dilemma: how to achieve stability without rigidity and plasticity without chaos?
 - Ongoing learning capability
 - Preservation of learned knowledge

Video Content / Details of website for further learning (if any):

http://www.scholarpedia.org/article/Fuzzy_neural_network

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 457

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Genetic Algorithms

Introduction :

- Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics
-

Prerequisite knowledge for Complete understanding and learning of Topic:

- Optimization Techniques
- Mining Algorithm

Detailed content of the Lecture:

- Generate initial population of Candidate solutions
- Apply fitness function to population members
- Choose the fittest member to form the new population
- Apply genetic operators and generate new population

GENETIC ALGORITHM FLOW CHART

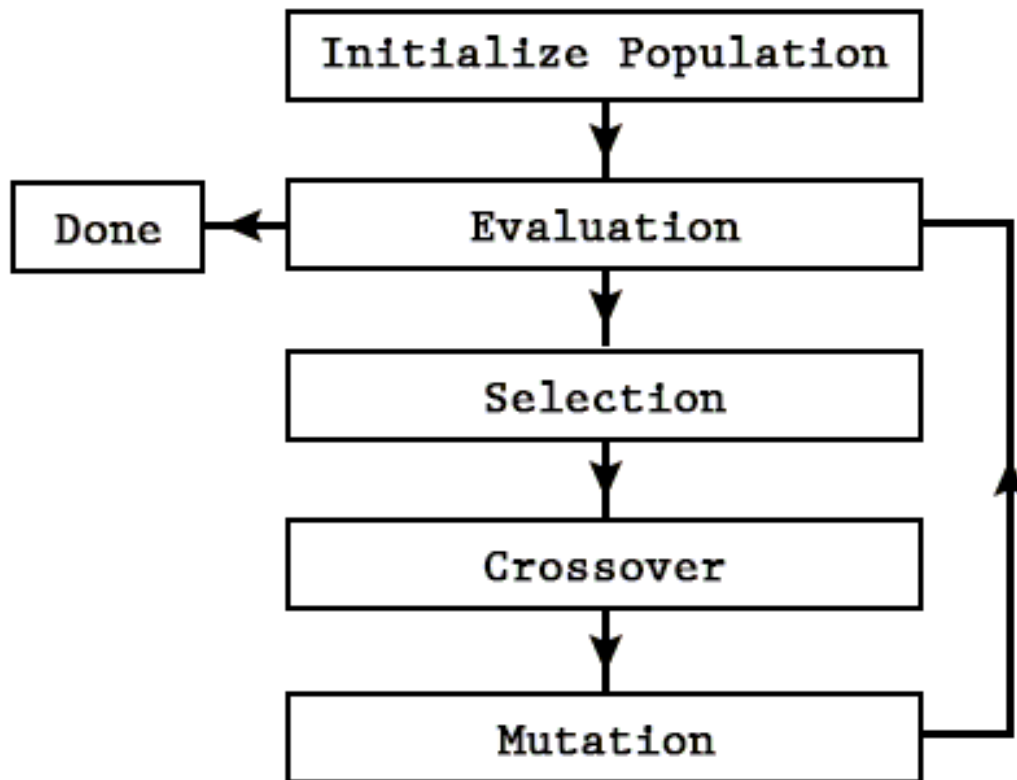


FIGURE 2

Steps of genetic algorithm

A genetic algorithm works by maintaining a pool of candidate solutions (named generation). Iteratively, the generation evolved to produce the next generation which has candidate solutions with higher fitness values than the previous generation. This process is repeated for a pre-specified number of generations or until a solution with goal fitness value is found.

The next generation is created from current generation in a biologically inspired manner that consists of 3 steps:

Selection: we evaluate the fitness of members of the current generation, then we select the subset with the best fitness values in order to act as parents for the next generation. In short, survival for the fittest.

Crossover: we merge pairs from the selected parents to generate new offspring child solution. Crossover can happen in different forms, simplest form is the one-point crossover which splits the string representation of each solutions into two parts at the same position, then concatenate the first part of one solution with the second part of the second one to form the offspring(children) solution representation.

Mutation: Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. Hence GA can come to a better solution by using mutation

Video Content / Details of website for further learning (if any):

<https://www.hindawi.com/journals/mpe/2018/9270802/>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 467

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Hollands Observation

Introduction

- Holland's schema theorem, also called the fundamental theorem of genetic algorithms is an inequality that results from coarse-graining an equation for evolutionary dynamics

Prerequisite knowledge for Complete understanding and learning of Topic:

Optimization Techniques
Mining Algorithm

Detailed content of the Lecture:

To explain Holland's observation in a deterministic manner let us presume the following assumptions .

- There are no recombination or alternations to genes.
- Initially, a fraction f of the population possesses the schema S and those individuals reproduce at a fixed rate r .
- All other individuals lacking schema S reproduce at a rate $s < r$.

Thus with an initial population size of N , after t generations, we find $Nf r^t$ individuals possessing schema S and the population of the rest of the individuals is $N(1 - f) s^t$.

Therefore, the fraction of the individuals with schema S is given by

$$(N f r^t) / [N (1 - f) s^t + N f r^t] = f (r / s)^t / [1 + f \{(r / s)^t - 1\}]$$

- Holland's schema theorem, also called the fundamental theorem of genetic algorithms, is an inequality that results from coarse-graining an equation for evolutionary dynamics. The Schema Theorem says that short, low-order schemata with above-average fitness increase exponentially in frequency in successive generations.
- The theorem was proposed by John Holland in the 1970s. It was initially widely taken to be the foundation for explanations of the power of genetic algorithms. However, this interpretation of its implications has been criticized in several publications reviewed in [2], where the Schema
- Theorem is shown to be a special case of the Price equation with the schema indicator function as the macroscopic measurement.
- A schema is a template that identifies a subset of strings with similarities at certain string positions. Schemata are a special case of cylinder sets, and hence form a topological space

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Holland%27s_schema_theorem

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 470

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Fundamental Theorem of Genetic Algorithms

Introduction :

Fundamental theorem of genetic algorithms, is an inequality that results from coarse-graining an equation for evolutionary dynamics. The Schema Theorem says that short, low-order schemata with above-average fitness increase exponentially in frequency in successive generations.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete mathematics
- Optimization Techniques
- Mining Algorithm

Detailed Content of the Lecture:

Theorem: The number of representatives of any schema, S , increases in proportion to the observed relative performance of S .

The Schema Theorem is called as “The Fundamental Theorem of Genetic Algorithm”, which gives insight in the way a GA works. It offers a different perspective on the GA by considering how it processes schemata (instead of chromosomes)

$$E [m(H, t + 1)] \geq m(H, t) \frac{f(H)}{\bar{f}} \left[1 - P_c \frac{\delta(H)}{l-1} - O(H)P_m \right]$$

For a given schema H , let:

$m(H, t)$ be the relative frequency of the schema H in the population of the t^{th} generation.

$f(H)$ be the mean fitness of the elements of H .

$O(H)$ be the number of fixed bits in the schema H called the order of the schema.

$d(H)$ be distance between the first and the last fixed bit of the schema, called the definition length of the schema.

f is the mean fitness of the current population.

P_c is the crossover probability.

P_m is the mutation probability.

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Holland%27s_schema_theorem

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 473

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Markov Model for Convergence Analysis

Introduction :

- In order to perfect the convergence theory of firefly algorithm (FA), the Markov chain model of the firefly algorithm is established. It was shown that the firefly group state sequence containing both the firefly states, the current local and the global optimal firefly states constructs a limited and homogeneous Markov chain by analyzing the property of the Markov chain.
- The transition process of the firefly group state sequence was analyzed, and the conclusion that sequence will eventually converges to the optimal state set was drawn; and it was proved that the firefly algorithm ensures global convergence as it meets the global convergence criterions of random search algorithms

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete mathematics
- Optimization Techniques
- Mining Algorithm

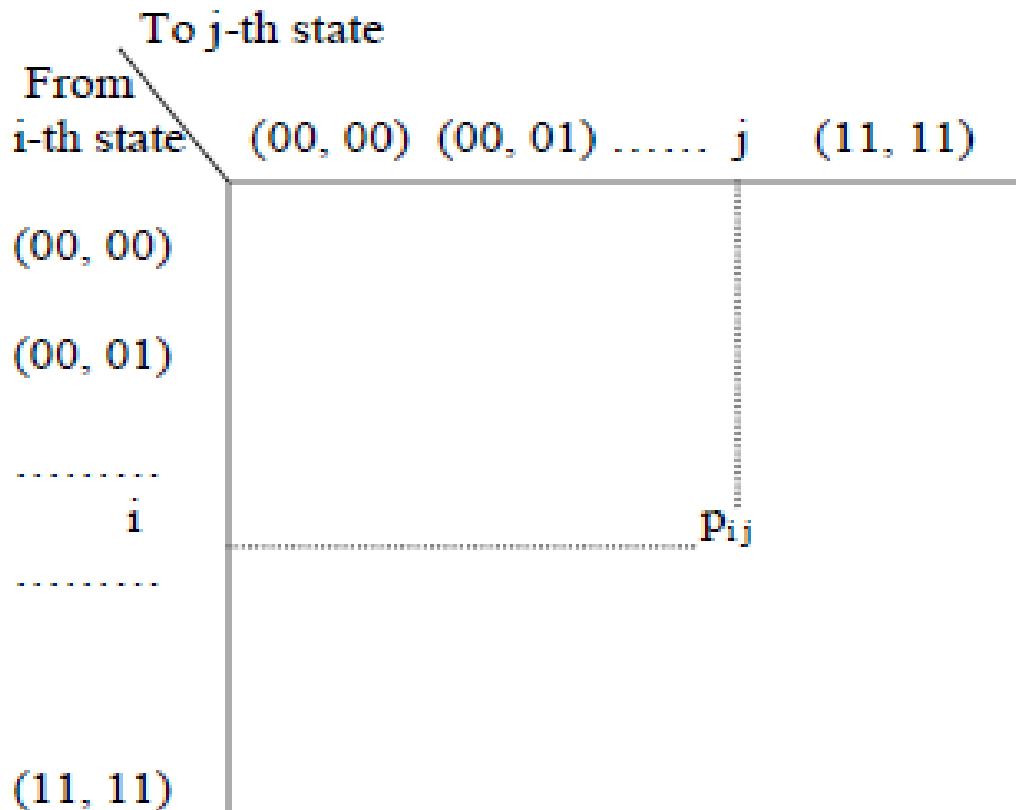
Detailed Content of the Lecture:

Markov Model for Convergence Analysis

To study the convergence of the GA, let us consider an exhaustive set of population states, where 'state' means possible members (chromosomes) that evolve at any GA cycle.

As an illustration, let us consider 2-bit chromosomes and population size = 2, which means at any GA cycle we select only two chromosomes. Under this circumstance, the possible states that can evolve at any duplication are the members of the set S, where

$S = \{ (00, 00), (00,01), (00,10), (00,11), (01, 00), (01, 01),$
 $(01, 10), (01, 11), (10, 00), (10, 01), (10, 10), (10, 11),$
 $(11, 00), (11, 01), (11, 10), (11, 11) \}$



Now, let us assume a row vector π_t , whose k-th element denotes the probability of occurrence of the k-th state at a given genetic iteration (cycle) t; then π_{t+1} can be evaluated by

$$\pi_{t+1} = \pi_t \cdot P$$

Thus starting with a given initial row vector π_0 , one can evaluate the state probability vector after n-th iteration π_n by

$$\pi_n = \pi_0 \cdot P^n$$

where P^n is evaluated by multiplying P matrix with itself (n-1) times.

Identification of a P matrix for a GA that allows selection, crossover and mutation, undoubtedly, is a complex problem

$$P = \begin{pmatrix} I & 0 \\ R & Q \end{pmatrix}$$

where I is an identity matrix of dimension (a x a) that corresponds to the absorbing states;

R is a (t x a) transition sub-matrix describing transition(passing) to an absorbing state;

Q is a (t x t) transition

Video Content / Details of website for further learning (if any):

https://www.researchgate.net/publication/298078057_Markov_model_and_convergence_analysis_based_on_firefly_algorithm

Important Books/Journals for further learnig including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 478

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Applications of Optimization problem, Intelligent Systems

Introduction :

- Intelligent Systems started out in the video game industry when programmer Toru Narihiro was hired by Nintendo to port Famicom Disk System software to the standard ROM-cartridge format that was being used outside Japan on the NES.
- Similarly to the origins of HAL Laboratory, the team soon became an auxiliary program unit for Nintendo that provided system tools and hired people to program, fix, or port Nintendo-developed software. Much of the team's original work consists of minor contributions to larger games developed by Nintendo R&D1 and Nintendo EAD

Prerequisite knowledge for Complete understanding and learning of Topic:

- Discrete mathematics
- Optimization Techniques
- Mining Algorithm

Detailed Content of the Lecture:

Solving optimization problems in practice In practice, the following issues need to be considered

1. Modelling of the problem
2. Modelling of the optimization problem
3. Choosing an appropriate optimization method
4. Coupling of optimization software and a modelling tool
5. Optimization and analysis of the solution obtained

Narihiro programmed his first video games, Famicom Wars and Fire Emblem: Shadow Dragon and the Blade of Light, towards the end of the Famicom's life cycle, although the game design, graphic design, and music was provided by the Nintendo R&D1 team. Because of Narihiro's success, Intelligent Systems began to hire graphic designers, programmers, and musicians to extend the company from an auxiliary-tool developer to a game development group. The company continued to develop new entries in the Wars and Fire Emblem franchises.

Video Content / Details of website for further learning (if any):

https://link.springer.com/chapter/10.1007/978-3-642-21705-0_1

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 480

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : IV

Date of Lecture:

Topic of Lecture: Genetic Programming- Fuzzy Neural Nets

Introduction :

- Genetic programming is one of the two techniques used in machine learning. The model is based on the testing and selecting the best choice among a set of results

Prerequisite knowledge for Complete understanding and learning of Topic:

- Genetic Algorithm
- Fuzzy Logic
- Neural Networks

Detailed Content of the Lecture:

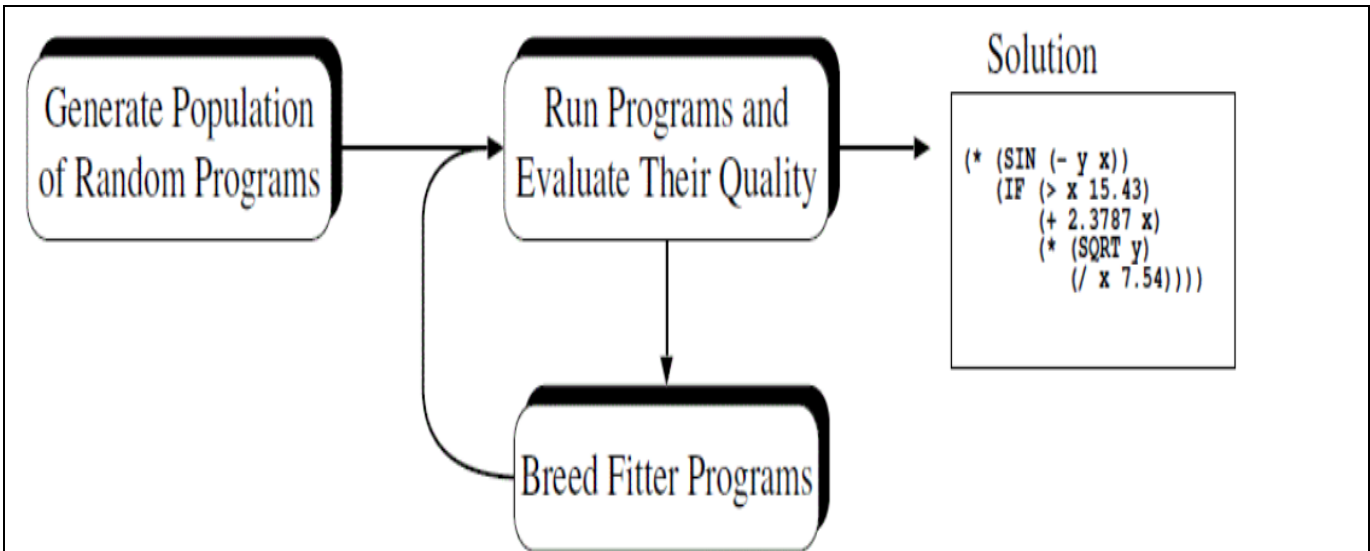
Genetic Programming (GP) is an Evolutionary Computation (EC) technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance.

GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done.

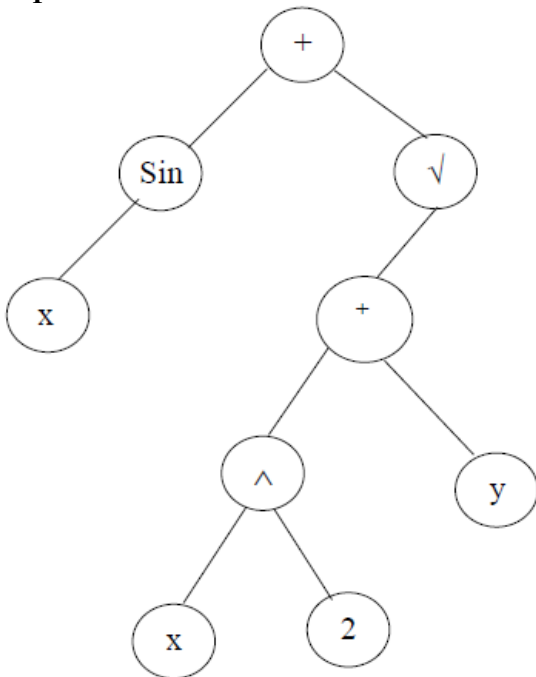
Why Genetic Programming

1. It saves time by freeing the human from having to design complex algorithms
2. Not only designing the algorithms but creating ones that give optimal solutions

How Genetic Programming?



Representation of GP



A **fuzzy neural network** or neuro-fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e., fuzzy sets, fuzzy rules) by exploiting approximation techniques from neural networks.

Video Content / Details of website for further learning (if any):

<http://geneticprogramming.com/>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 487

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : I V

Date of Lecture:

Topic of Lecture: Cognitive Maps-Stability Analysis

Introduction :

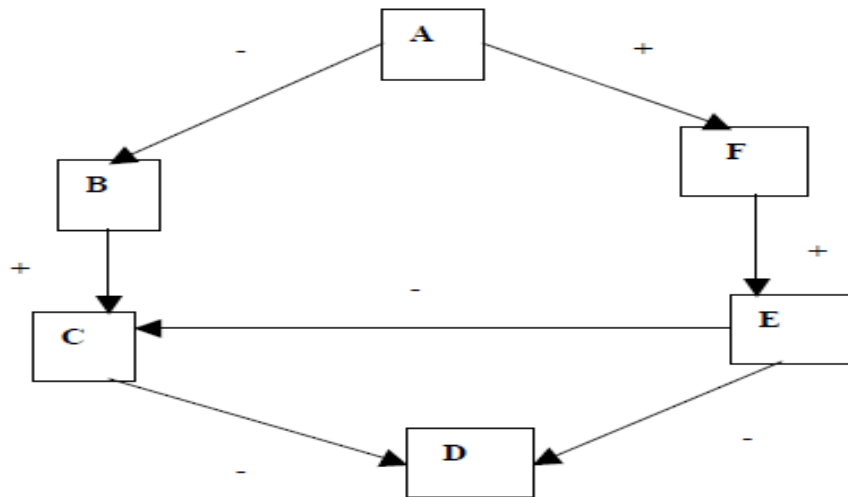
- A cognitive map is a mental picture or image of the layout of one's physical environment mathematics
- stability theory addresses the stability of solutions of differential equations and of trajectories of dynamical systems under small perturbations of initial conditions. The heat equation, for example, is a stable partial differential equation because small perturbations of initial data lead to small variations in temperature at a later time as a result of the maximum principle.
- In partial differential equations one may measure the distances between functions using L_p norms or the sup norm, while in differential geometry one may measure the distance between spaces using the Gromov-Hausdorff distance.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Easy-Mapping Tool
- Fuzzy Logic
- Neural Networks

Detailed Content of the Lecture:

The cognitive map describing the political relationship to hold the middle east peace is presented in figure



- The principles of cognitive mapping for describing relationships among facts. A cognitive map first undergoes a training phase for adaptation of the weights. Once the training phase is completed, the same network may be used for generating inferences from the known beliefs of the starting nodes
- Stability means that the trajectories do not change too much under small perturbations. The opposite situation, where a nearby orbit is getting repelled from the given orbit, is also of interest. In general, perturbing the initial state in some directions results in the trajectory asymptotically approaching the given one and in other directions to the trajectory getting away from it
- Many parts of the qualitative theory of differential equations and dynamical systems deal with asymptotic properties of solutions and the trajectories – what happens with the system after a long period of time
- The simplest kind of behavior is exhibited by equilibrium points, or fixed points, and by periodic orbits. If a particular orbit is well understood, it is natural to ask next whether a small change in the initial condition will lead to similar behavior
- Stability theory addresses the following questions: Will a nearby orbit indefinitely stay close to a given orbit? Will it converge to the given orbit? In the former case, the orbit is called stable; in the latter case, it is called asymptotically stable and the given orbit is said to be attracting

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Stability_theory

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 497

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit IV

Date of Lecture:

Topic of Lecture: Control Command by Cognitive Map

Introduction :

This Control command by cognitive Map uses five nodes to very basically describe the information needed to determine whether or not a particular bridge is to be considered a valid target based upon the current state of the animation. An animated simulation engine was written using the Microsoft Visual Basic language to facilitate rapid development

Prerequisite knowledge for Complete understanding and learning of Topic:

- Easy-Mapping Tool
- Fuzzy Logic
- Neural Networks

Detailed Content of the Lecture:

Design a concept for command and control with Marine Corps applications Marine Corps Simulation Module

- Animation communicates with the Visual C++ FCM Tool Kit using the standard Windows DDEML Dynamic Data Exchange Management Library. The scenario behind the simulation is that a series of bridges is approached by enemy troops. If there is no U.S. Advantage to NOT destroying the bridge, and if the bridge is not currently damaged, then the bridge is a valid target and should be destroyed
- The enemy troop position, bridge locations and U.S. Advantage can all be changed by the user while the simulation is running, by dragging . When running, the simulator selects a target to test, indicated by a rectangle drawn around the bridge and forms the input vector for the Enemy Bridge Use, U.S. Advantage, and Damage State nodes of the FCM
- The relative distance of the Enemy Troops from the bridge being tested is mapped into the range of [-1 0 1] where 1 indicates the troops are possibly using the bridge. The U.S. Advantage state is formed in the same way, where a 1 state indicates some reason NOT to

destroy the given bridge. A function call is then made to the FCM ToolKit, which tells it to perform the Step Forward Inference algorithm

- As the algorithm executes, the ToolKit queries the simulator for the initial Approved for public release; distribution is unlimited. 36 node states. The inferencing engine is then iterated just once to find what the next state will be based upon the current input vectors and the configuration of the FCM
- The FCM ToolKit then returns the vector of the Target Value node to the simulator. If there is no target value to this bridge, (vector of 0 or -1), the simulation advances to the next bridge and the process is started again. Once all bridges have been tested, the simulator loops back to the first bridge and continues iterating through all bridges until the simulation is topped
- If the conditions are such that the Tool Kit returns a Valid Target state (1) to the simulator , the simulation sends a bomb to destroy the bridge and Damage State for that bridge is set to High
- According to the FCM, the next time this bridge is tested, it should not be bombed because the Damage State node for that bridge is now High(1)

Video Content / Details of website for further learning (if any):

<https://pdfs.semanticscholar.org/1eb4/945545cc8b7e5fa162a4b3e61fbd1cfd6cb.pdf>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 497

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L18



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : Unit IV

Date of Lecture:

Topic of Lecture: Visual perception- Case Study

Introduction :

- Visual perception is the ability to see, organize, and interpret one's environment.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Cognitive Map Analysis
- Fuzzy Logic
- Neural Networks

Detailed Content of the Lecture:

Visual perception is the ability to perceive our surroundings through the light that enters our eyes. The visual perception of colors, patterns, and structures has been of particular interest in relation to graphical user interfaces (GUIs) because these are perceived *exclusively* through vision. An understanding of visual perception therefore enables designers to create more effective user interfaces.

Physiologically, visual perception happens when the eye focuses light on the *retina*. Within the retina, there is a layer of photoreceptor (light-receiving) cells which are designed to change light into a series of electrochemical signals to be transmitted to the brain.

Visual perception occurs in the brain's cerebral cortex; the electrochemical signals get there by traveling through the optic nerve and the thalamus. The process can take a mere 13 milliseconds, according to a 2017 study at MIT in the United States.

Different attributes of visual perception are widely used in GUI design. Many designers apply Gestalt principles (i.e., how humans structure visual stimuli) to the design of GUIs so as to create interfaces that are easy for users to perceive and understand.

The visual perception of affordances (action possibilities in the environment) is another example of how the understanding of visual perception is a critical item in any designer's toolkit.

Video Content / Details of website for further learning (if any):

<https://www.cognifit.com/science/cognitive-skills/visual-perception>

Important Books/Journals for further learning including the page nos.:

Amit Konar, "Artificial Intelligence", CRC Press, 2009. Page No: 513

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Expert systems

Introduction :

- The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise

Prerequisite knowledge for Complete understanding and learning of Topic:

- Knowledge Representation
- Logical Reasoning

Detailed content of the Lecture:

To solve expert-level problems, expert systems will need efficient access to a substantial domain knowledge base, and a reasoning mechanism to apply the knowledge to the problems they are given. Usually they will also need to be able to explain, to the users who rely on them, how they have reached their decisions. They will generally build upon the ideas of knowledge representation, production rules, search, and so on, that we have already covered. Often we use an expert system shell which is an existing knowledge independent framework into which domain knowledge can be inserted to produce a working expert system

Some typical existing expert system tasks include:

1. The interpretation of data such as sonar data or geophysical measurements
2. Diagnosis of malfunctions Such as equipment faults or human diseases
3. Structural analysis or configuration of complex objects such as chemical compounds or computer systems
4. Planning sequences of actions such as might be performed by robots
5. Predicting the future Systems

Characteristics of Expert Systems

- Expert systems can be distinguished from conventional computer systems in that:

1. They simulate human reasoning about the problem domain, rather than simulating the domain itself.
2. They perform reasoning over representations of human knowledge, in addition to doing numerical calculations or data retrieval. They have corresponding distinct modules referred to as the inference engine and the knowledge base.
3. Problems tend to be solved using heuristics (rules of thumb) or approximate methods or probabilistic methods which, unlike algorithmic solutions, are not guaranteed to result in a correct or optimal solution.
4. They usually have to provide explanations and justifications of their solutions or recommendations in order to convince the user that their reasoning is correct.

Note that the term Intelligent Knowledge Based System (IKBS) is sometimes used as a synonym for Expert System.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

Important Books/Journals for further learning including the page nos.:

Elaine Rich, Kevin Knight, Shivashankar.B.Nair, “Artificial Intelligence”, Tata Mc Graw Hill, 2011. Page No:547

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Expert systems- Applications

Introduction

- **Availability** – They are easily available due to mass production of software.
- **Less Production Cost** – Production cost is reasonable. This makes them affordable.
- **Speed** – They offer great speed. They reduce the amount of work an individual puts in.
- **Less Error Rate** – Error rate is low as compared to human errors.
- **Reducing Risk** – They can work in the environment dangerous to humans.
- **Steady response** – They work steadily without getting motional, tensed or fatigued

Prerequisite knowledge for Complete understanding and learning of Topic:

- **Learning**
- **Knowledge Representation**
- **Logical Reasoning**

Detailed Content of the Lecture:

Applications of Expert System

The following table shows where ES can be applied.

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.

Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

Expert System Technology

There are several levels of ES technologies available. Expert systems technologies include –

- **Expert System Development Environment** – The ES development environment includes hardware and tools. They are –
 - Workstations, minicomputers, mainframes.
 - High level Symbolic Programming Languages such as **LIST** Programming (LISP) and **PRO**grammation en **LOG**ique (PROLOG).
 - Large databases.
- **Tools** – They reduce the effort and cost involved in developing an expert system to large extent.
 - Powerful editors and debugging tools with multi-windows.
 - They provide rapid prototyping
 - Have Inbuilt definitions of model, knowledge representation, and inference design.
- **Shells** – A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility. For example, few shells are given below –
 - Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system.
 - *Vidwan*, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules.

Video Content/ Details of website for further learning (if any):

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

Important Books/Journals for further learning including the page nos.:

Elaine Rich, Kevin Knight, Shivashankar.B.Nair, “Artificial Intelligence”, Tata Mc Graw Hill, 2011.
Page No:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Architecture of Expert Systems

Introduction :

The knowledge base contains the specific domain knowledge that is used by an expert to derive conclusions from facts

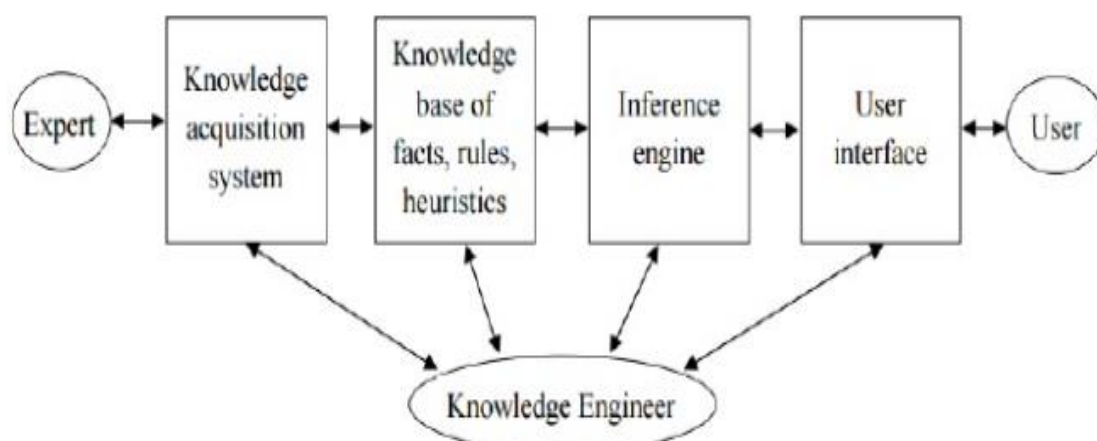
Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

Detailed Content of the Lecture:

The Architecture of Expert Systems

The process of building expert systems is often called knowledge engineering. The knowledge engineer is involved with all components of an expert system:



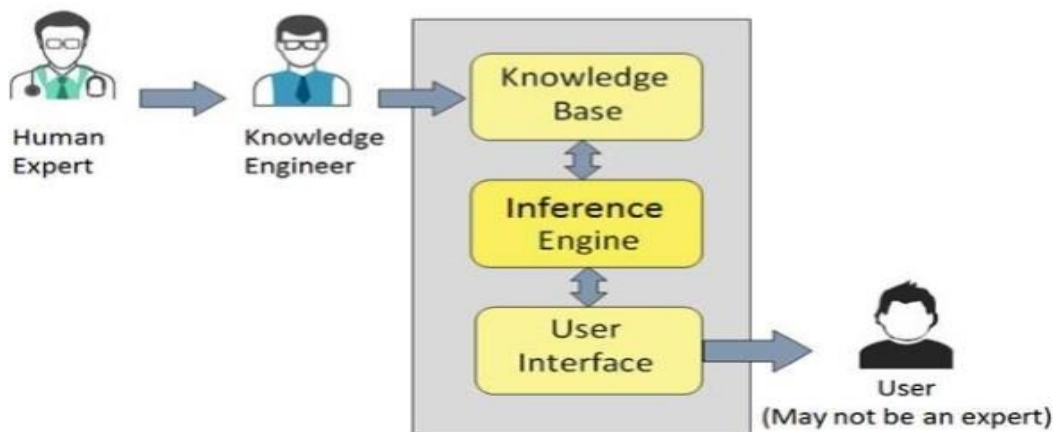
- Building expert systems is generally an iterative process. The components and their interaction will be refined over the course of numerous meetings of the knowledge engineer with the experts and users. We shall look in turn at the various components

Components of Expert Systems

The components of ES include –

- Knowledge Base
- Inference Engine
- User Interface

Let us see them one by one briefly –



Knowledge Base

It contains domain-specific and high-quality knowledge.

Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

What is Knowledge?

The data is collection of facts. The information is organized as data and facts about the task domain. **Data**, **information**, and **past experience** combined together are termed as knowledge.

Components of Knowledge Base

The knowledge base of an ES is a store of both, factual and heuristic knowledge.

- **Factual Knowledge** – It is the information widely accepted by the Knowledge Engineers and scholars in the task domain.
- **Heuristic Knowledge** – It is about practice, accurate judgement, one's ability of evaluation, and guessing.

Knowledge representation

It is the method used to organize and formalize the knowledge in the knowledge base. It is in the form of IF-THEN-ELSE rules.

Knowledge Acquisition

The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledge base.

The knowledge base is formed by readings from various experts, scholars, and the **Knowledge Engineers**. The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills.

He acquires information from subject expert by recording, interviewing, and observing him at work, etc. He then categorizes and organizes the information in a meaningful way, in the form of IF-THEN-ELSE rules, to be used by interference machine. The knowledge engineer also monitors the development of the ES.

Inference Engine

Use of efficient procedures and rules by the Inference Engine is essential in deducing a correct, flawless solution.

In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

In case of rule based ES, it –

- Applies rules repeatedly to the facts, which are obtained from earlier rule application.
- Adds new knowledge into the knowledge base if required.
- Resolves rules conflict when multiple rules are applicable to a particular case.

To recommend a solution, the Inference Engine uses the following strategies –

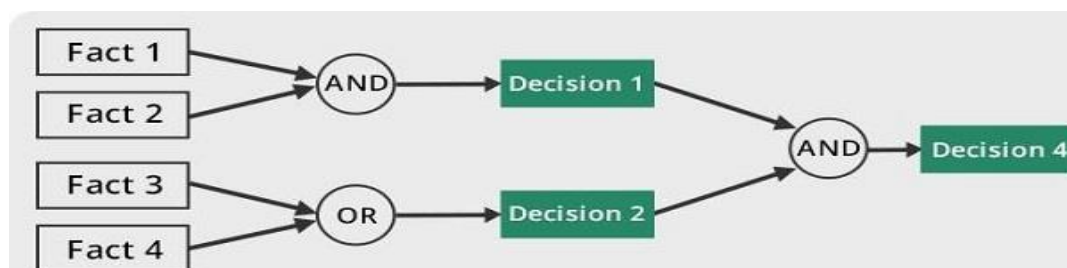
- Forward Chaining
- Backward Chaining

Forward Chaining

It is a strategy of an expert system to answer the question, **“What can happen next?”**

Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

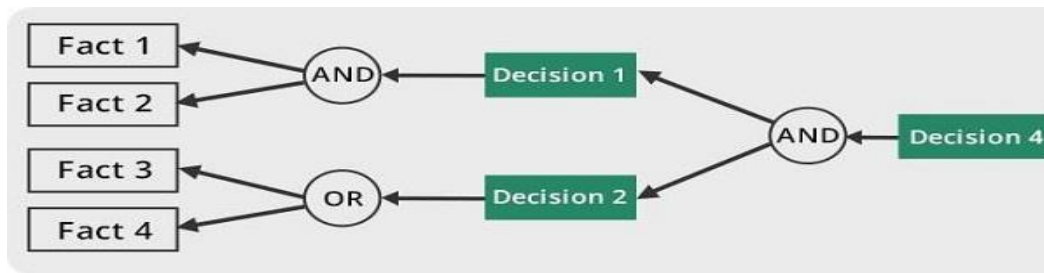
This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

With this strategy, an expert system finds out the answer to the question, **“Why this happened?”**

On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



User Interface

User interface provides interaction between user of the ES and the ES itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.

It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms –

- Natural language displayed on screen.
- Verbal narrations in natural language.
- Listing of rule numbers displayed on the screen.

The user interface makes it easy to trace the credibility of the deductions.

Requirements of Efficient ES User Interface

- It should help users to accomplish their goals in shortest possible way.
- It should be designed to work for user's existing or desired work practices.
- Its technology should be adaptable to user's requirements; not the other way round.
- It should make efficient use of user input.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

Important Books/Journals for further learning including the page nos.:

Elaine Rich, Kevin Knight, Shivashankar.B.Nair, "Artificial Intelligence", Tata Mc Graw Hill, 2011. Page No:550

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L

LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Roles of expert systems

Introduction :

- Instructing and assisting human in decision making
- Demonstrating

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

Detailed Content of the Lecture:

Development of Expert Systems: General Steps

The process of ES development is iterative. Steps in developing the ES include –

Identify Problem Domain

- The problem must be suitable for an expert system to solve it.
- Find the experts in task domain for the ES project.
- Establish cost-effectiveness of the system.

Design the System

- Identify the ES Technology
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Develop the Prototype

From Knowledge Base: The knowledge engineer works to –

- Acquire domain knowledge from the expert.
- Represent it in the form of If-THEN-ELSE rules.

Test and Refine the Prototype

- The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance.
- End users test the prototypes of the ES.

Develop and Complete the ES

- Test and ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems.
- Document the ES project well.
- Train the user to use ES.

Maintain the System

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.
- Three fundamental roles in building expert systems are:
 1. Expert - Successful ES systems depend on the experience and application of knowledge that the people can bring to it during its development. Large systems generally require multiple experts
 2. Knowledge Engineer - The knowledge engineer has a dual task. This person should be able to elicit knowledge from the expert, gradually gaining an understanding of an area of expertise. Intelligence, tact, empathy, and proficiency in specific techniques of knowledge acquisition are all required of a knowledge engineer.
 3. Knowledge-acquisition techniques include conducting interviews with varying degrees of structure, protocol analysis, observation of experts at work, and analysis of cases.

Capabilities of Expert Systems

The expert systems are capable of –

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results

- Justifying the conclusion
- Suggesting alternative options to a problem

They are incapable of –

- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Knowledge Acquisition

Introduction :

- The knowledge acquisition component allows the expert to enter their knowledge or expertise into the expert system, and to refine it later as and when required. Historically, the knowledge engineer played a major role in this process, but automated systems that allow the expert to interact directly with the system are becoming increasingly common

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

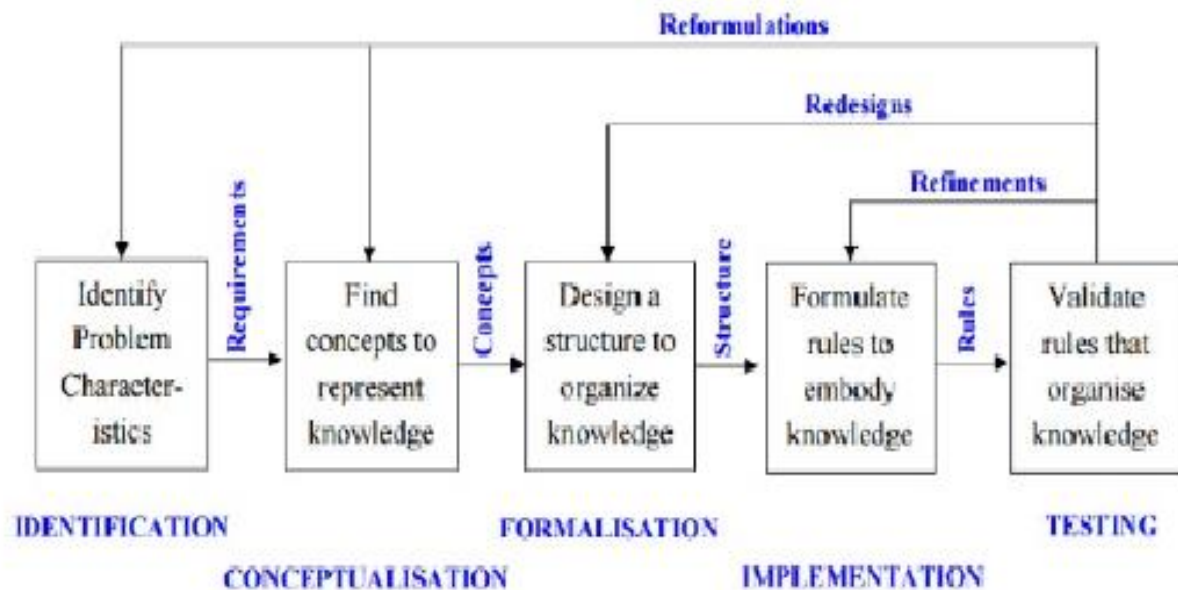
Detailed Content of the Lecture:

The knowledge acquisition process is usually comprised of three principal stages:

1. Knowledge elicitation is the interaction between the expert and the knowledge engineer/program to elicit the expert knowledge in some systematic way.
2. The knowledge thus obtained is usually stored in some form of human friendly intermediate representation.
3. The intermediate representation of the knowledge is then compiled into an executable form

In practice, much iteration through these three stages is usually required

The iterative nature of the knowledge acquisition process can be represented in the following diagram.



Inference Engine

Use of efficient procedures and rules by the Inference Engine is essential in deducing a correct, flawless solution.

In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

In case of rule based ES, it –

- Applies rules repeatedly to the facts, which are obtained from earlier rule application.
- Adds new knowledge into the knowledge base if required.
- Resolves rules conflict when multiple rules are applicable to a particular case.

Video Content / Details of website for further learning (if any):

<https://www.knowledge-management-tools.net/knowledge-acquisition.php>

Important Books/Journals for further learning including the page nos.:

Elaine Rich, Kevin Knight, Shivashankar.B.Nair, “Artificial Intelligence”, Tata Mc Graw Hill, 2011. Page No:553

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Knowledge Acquisition

Introduction :

The knowledge acquisition component allows the expert to enter their knowledge or expertise into the expert system, and to refine it later as and when required. Historically, the knowledge engineer played a major role in this process, but automated systems that allow the expert to interact directly with the system are becoming increasingly common

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

Detailed Content of the Lecture:

To recommend a solution, the Inference Engine uses the following strategies –

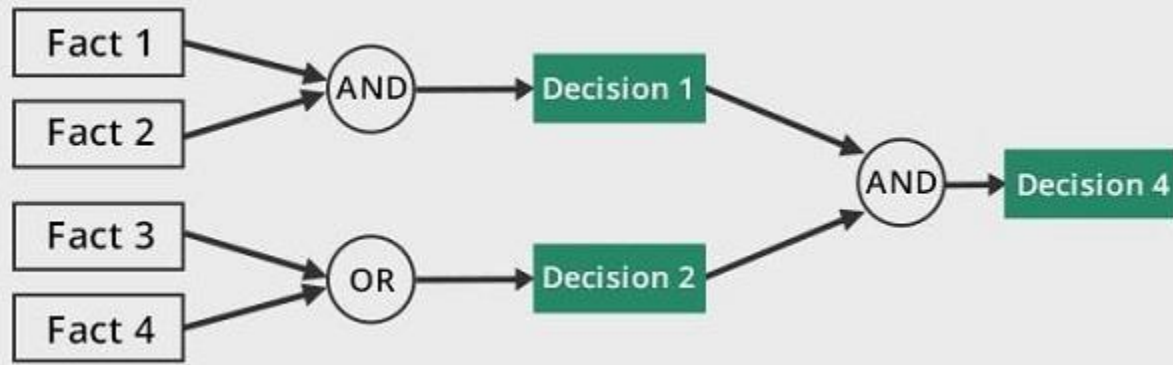
- Forward Chaining
- Backward Chaining

Forward Chaining

It is a strategy of an expert system to answer the question, **“What can happen next?”**

Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

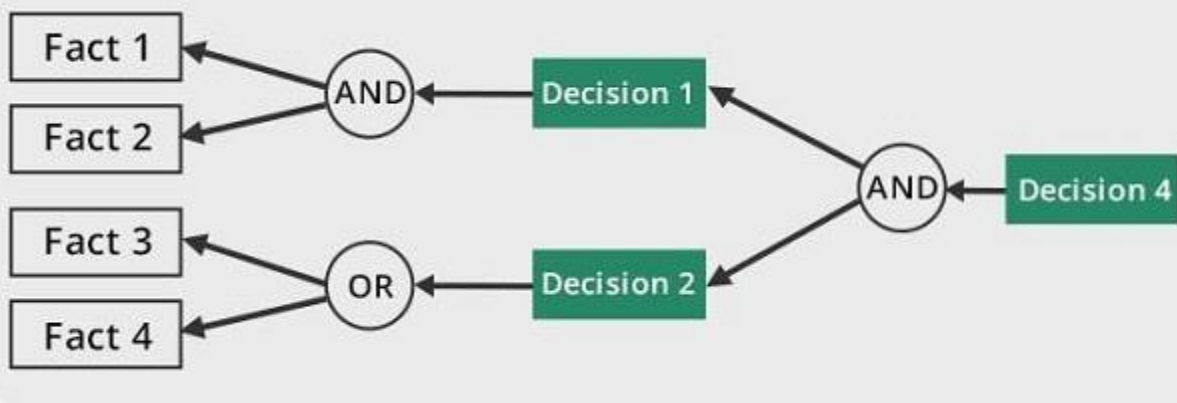
This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

With this strategy, an expert system finds out the answer to the question, “Why this happened?”

On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



User Interface

User interface provides interaction between user of the ES and the ES itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.

It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms –

- Natural language displayed on screen.
- Verbal narrations in natural language.
- Listing of rule numbers displayed on the screen.

The user interface makes it easy to trace the credibility of the deductions.

Requirements of Efficient ES User Interface

- It should help users to accomplish their goals in shortest possible way.
- It should be designed to work for user’s existing or desired work practices.
- Its technology should be adaptable to user’s requirements; not the other way round.
- It should make efficient use of user input.

Video Content / Details of website for further learning (if any):

<https://www.knowledge-management-tools.net/knowledge-acquisition.php>

Important Books/Journals for further learning including the page nos.:

Elaine Rich, Kevin Knight, Shivashankar.B.Nair, "Artificial Intelligence", Tata Mc Graw Hill, 2011.
Page No:555

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L16



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Meta knowledge, Heuristics

Introduction :

A heuristic (to find) approach to a problem is an empirical search or optimization method that often works at solving the problem, but doesn't have any of the rigorous proof that people like physicists and mathematicians expect. Nobody knows if it will always give the best answer to the problem. To put it simply, it is a short cut to solving difficult problems.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

Detailed Content of the Lecture:

- Knowledge about knowledge
- Meta knowledge can be simply defined as knowledge about knowledge.
- Meta knowledge is knowledge about the use and control of domain knowledge in an expert system.

Meta-heuristics are heuristic procedures used to tune, control, guide, allocate computational resources or reason about object-level problem solvers in order to improve their quality, performance, or efficiency.

Offline meta-heuristics define the best structural and/or parametric configurations for the object-level model, while on-line heuristics generate run-time corrections for the behavior of the same object-level solvers.

Soft Computing is a framework in which we encode domain knowledge to develop such meta-heuristics. We explore the use of meta-heuristics in three application areas:

a) control

b) optimization

c) classification

- In the context of control problems, we describe the use of evolutionary algorithms to perform offline parametric tuning of fuzzy controllers, and the use of fuzzy supervisory controllers to perform on-line mode-selection and output interpolation.
- In the area of optimization, we illustrate the application of fuzzy controllers to manage the transition from exploration to exploitation of evolutionary algorithms that solve the optimization problem.
- In the context of discrete classification problems, we have leveraged evolutionary algorithms to tune knowledge-based classifiers and maximize their coverage and accuracy.

Video Content / Details of website for further learning (if any):

https://www.researchgate.net/post/What_are_the_differences_between_heuristics_and_metaheuristics

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Typical expert systems MYCIN, DART, XOON

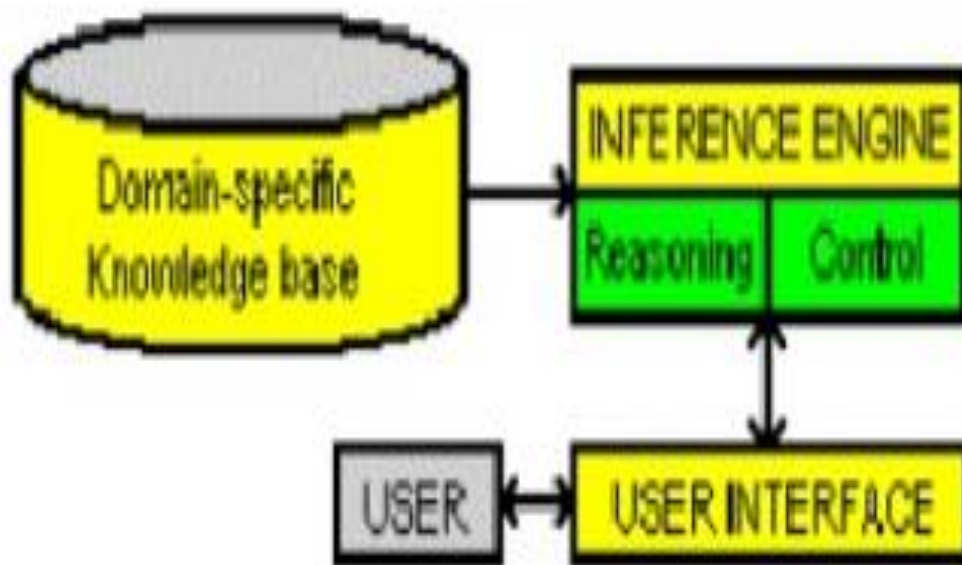
Introduction :

- A problem-domain-specific knowledge base that stores the encoded knowledge to support one problem domain such as diagnosing why a car won't start. In a rule-based expert system, the knowledge base includes the if-then rules and additional specifications that control the course of the interview.
- 2. An inference engine a set of rules for making deductions from the data and that implements the reasoning mechanism and controls the interview process. The inference engine might be generalized so that the same software is able to process many different knowledge bases.
- 3. The user interface requests information from the user and outputs intermediate and final results. In some expert systems, input is acquired from additional sources such as data bases and sensors.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Rule based system

- An expert system shell consists of a generalized inference engine and user interface designed to work with a knowledge base provided in a specified format. A shell often includes tools that help with the design, development and testing of the knowledge base. With the shell approach, expert systems representing many different problem domains may be developed and delivered with the same software environment.
- There are special high level languages used to program expert systems (e.g. PROLOG. The user interacts with the system through a user interface which may use menus, natural language or any other style of interaction).
- Then an inference engine is used to reason with both the expert knowledge (extracted from our friendly expert) and data specific to the particular problem being solved. The expert knowledge will typically be in the form of a set of IF-THEN rules.
- The case specific data includes both data provided by the user and partial conclusions



MYCIN

- **Tasks and Domain**
- **Disease DIAGNOSIS and Therapy SELECTION**
- **Advice for non-expert physicians with time considerations and incomplete evidence on:**
- **Bacterial infections of the blood**
- **Expanded to meningitis and other ailments**
- **System Goals**

1.Utility

- Be useful, to attract assistance of experts

- Demonstrate competence
- Fulfill domain need (i.e. penicillin)

2.Flexibility

- Domain is complex, variety of knowledge types
- Medical knowledge rapidly evolves, must be easy to maintain K.B.

3.Interactive Dialogue

- Provide coherent explanations (symbolic reasoning paradigm)
- Allow for real-time K.B. updates by experts

4.Fast and Easy

- Meet time constraints of the medical field

DART

- The Dynamic Analysis and Replanning Tool, commonly abbreviated to DART, is an artificial intelligence program used by the U.S. military to optimize and schedule the transportation of supplies or personnel and solve other logistical problems.
- DART uses intelligent agents to aid decision support systems located at the U.S. Transportation and European Commands. It integrates a set of intelligent data processing agents and database management systems to give planners the ability to rapidly evaluate plans for logistical feasibility.
- By automating evaluation of these processes DART decreases the cost and time required to implement decisions. DART achieved logistical solutions that surprised many military planners. Introduced in 1991,DART had by 1995 offset the monetary equivalent of all funds DARPA had channeled into for the previous 30 years combined
- DART's success led to the development of other military planning agents such as:
 - RDA - Resource Description and Access system
 - DRPI - Knowledge-Based Planning and Scheduling Initiative, a successor of DART

Video Content / Details of website for further learning (if any):

<https://prezi.com/p/jwxx-3udrnkp/meta-knowledge-heuristics/>

Important Books/Journals for further learning including the page nos.:

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L



LECTURE HANDOUTS

CSE

III/ VI

Course Name with Code : 16CSE04 ARTIFICIAL INTELLIGENCE

Course Faculty : G.SUMATHI

Unit : V

Date of Lecture:

Topic of Lecture: Expert systems shells

Introduction : An **expert system shell** is a software development environment containing the basic components for building **expert systems**. It does not contain knowledge base. For every domain specific **system**, a knowledge engineer prepares knowledge base with the help of domain **experts** in a particular area.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Learning
- Knowledge Representation
- Logical Reasoning

Detailed Content of the Lecture:

3. Initially each expert system is build from scratch (LISP). Systems are constructed as a set of declarative representations (mostly rules) combined with an interpreter for those representations
 4. It helps to separate the interpreter from domain-specific knowledge and to create a system that could be used construct new expert system by adding new knowledge corresponding to the new problem domain
 5. The resulting interpreters are called shells. Example of shells is EMYCIN (for Empty MYCIN derived from MYCIN)
 6. Shells – A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility
- For example, few shells are given below Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system

- Vidwan, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules. Shells provide greater flexibility in representing knowledge and in reasoning than MYCIN. They support rules, frames, truth maintenance systems and a variety of other reasoning mechanisms
- Early expert system shells provide mechanisms for knowledge representation, reasoning and explanation. Later these tools provide knowledge acquisition. Still expert system shells need to integrate with other programs easily
- Expert systems cannot operate in a vacuum. The shells must provide an easy-to-use interface between an expert system written with the shell and programming environment

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm

Important Books/Journals for further learning including te page nos.:

Course Faculty

Verified by HOD