UNIT 1

**Definition of AI**

AI is a branch of computer science which is concerned with the study and creation of computer

systems that exhibit

some form of intelligence

OR

those characteristics which we associate with intelligence in human behavior

AI is a broad area consisting of different fields, from machine vision, expert systems to the

creation of machines that can "think". In order to classify machines as "thinking", it is necessary

to define intelligence

**Turing test**

Year 1950: The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes "Computing Machinery and Intelligence" in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a Turing test

**Definition of agent.**

An agent perceives its environment via sensors and acts upon that environment

through its actuators

agent = architecture + program

A diagram of a person with text

Description automatically generated

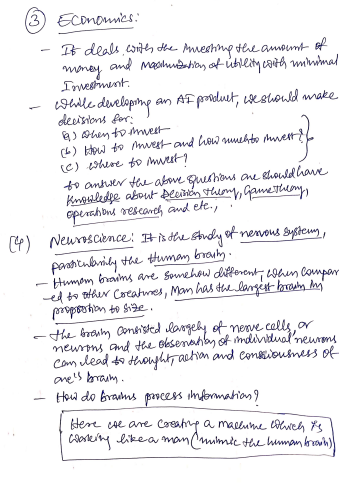
**Foundation of AI and fields supporting AI are same**

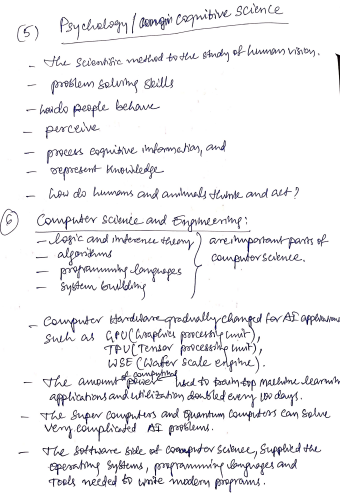
A paper with writing on it

Description automatically generated

A paper with blue writing

Description automatically generated





A screenshot of a computer screen

Description automatically generated

**Structure of agent**

Agents Definition: An agent perceives its environment via sensors and acts upon that environment through its actuators

Rational Agents: An agent should strive to "do the right thing", based on what: – it can perceive and – the actions it can perform.

For each possible percept sequence, a rational agent should select an action that maximizes its performance measure (in expectation) given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

If an agent’s sensors give it access to the complete state of the environment needed to choose an action, the environment is fully observable. ▪ (e.g. Chess) o An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data ▪ (e.g. Kriegspiel chess )

PEAS: Performance measure, Environment, Actuators, Sensors

An agent perceives and acts in an environment, has an architecture, and is implemented by an agent program.

A rational agent always chooses the action which maximizes its expected performance, given its percept sequence so far.

An autonomous agent uses its own experience rather than built-in knowledge of the environment by the designer.

An agent program maps from percept to action and updates its internal state.

 Simple reflex agents

are based on condition-action rules, implemented with an appropriate production system. They are stateless devices which do not have memory of past world

states.

 Agents with memory - Model-based reflex agents

have internal state, which is used to keep track of past states of the world.

 Agents with goals – Goal-based agents

are agents that, in addition to state information, have goal information that

describes desirable situations. Agents of this kind take future events into

consideration.

 Utility-based agents

base their decisions on classic axiomatic utility theory in order to act rationally.

 Learning agents

they have the ability to improve performance through learning.

Representing knowledge is important for successful agent design

**Types of agent**

1. Simple reflex agents

2. Model based reflex agents

3. Goal based agents

4. Utility based agents

5. Learning agents

1. Simple reflex agents:

 Agents do not have memory of past world states or percepts. So, actions depend solely on current percept. In this agent Action becomes a “reflex.” So it uses condition-action rules. Agent selects actions on the basis of current percept only.

function SIMPLE-REFLEX-AGENT(percept ) returns an action

persistent: rules, a set of condition–action rules

state←INTERPRET-INPUT(percept )

rule←RULE-MATCH(state, rules)

action ←rule.ACTION

return action

Example: The agent program for a simple reflex agent in the two-state vacuum

environment.

function REFLEX-VACUUM-AGENT([location,status]) returns an action

if status = Dirty then return Suck

else if location = A then return Right

else if location = B then return Left

2. Model-based reflex agents

They use a model of the world to choose their actions. They maintain an internal state.

Model − knowledge about “how the things happen in the world”.

Internal State − It is a representation of unobserved aspects of current state depending on

percept history.

Updating the state requires the information about −

• How the world evolves.

• How the agent’s actions affect the world.

function MODEL-BASED-REFLEX-AGENT(percept ) returns an action

persistent:

state, the agent’s current conception of the world state

model , a description of how the next state depends on current state and action

rules, a set of condition–action rules

action, the most recent action, initially none

state←UPDATE-STATE(state, action, percept ,model )

rule←RULE-MATCH(state, rules)

action ←rule.ACTION

return action

3. Goal-based agents

They choose their actions in order to achieve goals. Goal-based approach is more flexible than

reflex agent since the knowledge supporting a decision is explicitly modeled, thereby allowing

for modifications.

Goal − It is the description of desirable situations

4. Utility-based agents

They choose actions based on a preference (utility) for each state.

Goals are inadequate when −

• There are conflicting goals, out of which only few can be achieved.

• Goals have some uncertainty of being achieved and you need to weigh likelihood of

success against the importance of a goal

5. Learning agents

 Learning agents are such agents which adapts and improve over time.

 More complicated when agent needs to learn utility information: Reinforcement learning

**Types of problems (real world and toy problem)**

A problem can be defined formally by five components:

Initial State:

The state where agents starts to perform the search to reach the goal state

Actions:

Set of applicable actions perform in state S.

Transition model:

What each actions does

Goal Test:

Which determines whether a given state is a goal state or not.

Path Cost:

Assigns a numeric cost to each path

Example: Vacuum-cleaner world state space graph

A diagram of a diagram

Description automatically generated

States: Integer dirt and robot locations

Actions: left, right, suck, noOp

Goal Test: not dirty?

Path Cost: 1 per operation (0 for noOp)

Real-World Problems:

These are practical problems encountered in various domains, and solving them has significant real-world impact.

a. Classification:

Assigning categories to documents, products, images, or any data.

Examples:

Categorizing support tickets by relevant topics.

Identifying defects in images of silicon wafers1.

b. Regression:

Estimating numerical values based on input data.

Examples:

Predicting machine service intervals.

Modeling the effect of drug dosage on blood pressure1.

c. Recommendation:

Providing personalized content or product suggestions.

Examples:

Product recommendations based on user preferences.

Suggesting whom to follow on social media platforms.

d. Search Relevance:

Improving search result rankings for users.

Analyzing search logs to diagnose issues.

May involve machine learning or not1.

e. Information Extraction (IE):

Extracting specific information from large text volumes.

Examples:

Identifying patient symptoms from clinical notes.

Pre-populating application forms from resumes.

f. Text Summarization:

Creating concise summaries of lengthy documents.

Useful for research papers or news articles.

g. Clustering:

Grouping similar data points based on features.

Segmenting customers for targeted marketing.

h. Entity Recognition:

Identifying and extracting entities (names, dates, etc.) from text.

Recognizing people, organizations, or locations.

i. Virtual AI Assistant:

Building conversational agents for tasks like customer support.

j. Sentiment Analysis:

Determining sentiment (positive, negative, neutral) expressed in text.

k. Object Detection:

Locating and identifying objects in images or videos.

l. Document Segmentation Problem:

Dividing lengthy documents into meaningful sections.

Toy Problems:

These are simplified problems used for educational purposes or as benchmarks.

a. Water Jug Problem:

Solving how to measure a specific volume using two jugs of different capacities.

b. Missionaries and Cannibals Problem:

Moving missionaries and cannibals across a river without violating constraints.

c. 8-Queen Problem:

Placing 8 queens on a chessboard so that no two attack each other.

d. 8-Puzzle Problem:

Rearranging tiles in a 3x3 grid to reach a desired configuration.

e. Monkey Banana Problem:

A monkey navigating a grid to collect bananas while avoiding obstacles.

f. Tower of Hanoi Problem:

Moving a stack of disks from one peg to another following specific rules.

g. Cryptarithmetic Problem:

Solving puzzles where letters represent digits in arithmetic operations.

**What is state space of an agent?**

State Space

 Together the initial state, actions and transition model implicitly define the state space of

the problem – the set of all states reachable from the initial state by any sequence of

actions.

 The state space forms a directed network or graph in which the nodes are states and the

links between nodes are actions.

 A path in the state space is a sequence of states connected by a sequence of actions.

**Difference between informed and un informed search strategies**

Types of search algorithms:

Based on the search problems we can classify the search algorithms into uninformed (Blind

search) search and Informed search (Heuristic search) algorithms.

Uninformed/Blind Search:

 The uninformed search does not contain any domain knowledge such as closeness, the

location of the goal. It operates in a brute-force way as it only includes information about

how to traverse the tree and how to identify leaf and goal nodes.

 Uninformed search applies a way in which search tree is searched without any information

about the search space like initial state operators and test for the goal, so it is also called

blind search. It examines each node of the tree until it achieves the goal node.

 So Distance to goal not taken into account

 Ex: DFS, BFS, Iterative deepening DFS

Informed Search:

 Informed search algorithms use domain knowledge. In an informed search, problem

information is available which can guide the search. Informed search strategies can find a

solution more efficiently than an uninformed search strategy. Informed search is also

called a Heuristic search.

 A heuristic is a way which might not always be guaranteed for best solutions but

guaranteed to find a good solution in reasonable time.

 Informed search can solve much complex problem which could not be solved in another

way.

 The informed search algorithm is more useful for large search space. Informed search

algorithm uses the idea of heuristics, so it is also called Heuristic search.

 So Information about cost to goal taken into account

 Ex: Best first search , A\* search

A screenshot of a computer screen

Description automatically generated

**A\*algorithm**

A

\*

search

 A\* search is the most commonly known form of best-first search. It uses heuristic

function h(n), and cost to reach the node n from the start state g(n).

 A\* search algorithm finds the shortest path through the search space using the heuristic

function.

 This search algorithm expands less search tree and provides optimal result faster.

 In A\* search algorithm, we use search heuristic as well as the cost to reach the node.

Hence we can combine both costs as following, and this sum is called as a fitness

Number

A diagram of a graph

Description automatically generated

A diagram of a tree

Description automatically generated

A\*: Difficulties

 It becomes often difficult to use A\* as the OPEN queue grows very large.

 A solution is to use algorithms that work with less memory

 Memory bounded heuristic search algorithms reduce the memory requirements for A\* by

introducing IDA\*.

 IDA\* is an iterative deepening algorithm.

 The cut-off for nodes expanded in an iteration is decided by the f-value of the nodes

**Greedy best first search or best first search**

Best-first Search Algorithm (Greedy Search)

 Greedy best-first search algorithm always selects the path which appears best at that

moment.

 It is the combination of depth-first search and breadth-first search algorithms.

 It uses the heuristic function and search. Best-first search allows us to take the advantages

of both algorithms.

 With the help of best-first search, at each step, we can choose the most promising node.

 In the best first search algorithm, we expand the node which is closest to the goal node

and the closest cost is estimated by heuristic function,

i.e. f(n)= g(n).

Where, h(n)= estimated cost from node n to the goal.

 The greedy best first algorithm is implemented by the priority queue.

Greedy best-first tree search is also incomplete even in a finite state space, much like depth-first search.

The worst-case time and space complexity for the tree version is O(b

m),

where b is the maximum branching factor of the search tree

m is the maximum depth of the search space.

 With a good heuristic function, however, the complexity can be reduced substantially.

 The amount of the reduction depends on the particular problem and on the quality of the

Heuristic

**IDA\* algorithm**

IDA\* = A\* + Iterative Deepening DFS

 The key feature of the IDA\* algorithm is that it doesn’t keep a track of each visited node

which helps in saving memory consumption and can be used where memory is

constrained.

 It is path search algorithm which finds shortest path from start node to goal node in

weighted graph.

 It borrows the idea to use a heuristic function from A\*

 Its memory usage is weaker than in A\*, thereby reducing memory problem in A\*

 IDA\* is optimal in terms of space and cost

 Problem: Excessive node generation

Algorithm:

 Set certain threshold/f-bound

 If f(node) > threshold/f-bound, prune the node

 Set threshold/f-bound = minimum cost of any node that is pruned

 Terminates when goal is reached.

IDA\* Search Procedure

 Consider the threshold/f-bound value and cycle through the algorithm

 In every branch visit the depth till the f(node) is greater than the threshold/f-bound and

note down that f(node) value, do this till all branches are explored upto certain depth.

 Then the cycle continues from the starting node again with the new threshold/f-new value

that is the minimum of f(node) values noted down.

 This continues until the goal is found or the time limit is exceeded

Properties of IDA\*:

 Complete: Yes, similar to A\*

 Time: Depends strongly on the number of different values that the heuristic value can

take on. If A\* expands N nodes, IDA\* expands 1+2+……+N=O(N2

) nodes.

 Space: It is DFS, it only requires space proportional to the longest path it explores.

**Difference between perfect and imperfect decisions**

Perfect decision games:

 In game theory, a sequential game has perfect information if each player, when making

any decision, is perfectly informed of all the events that have previously occurred,

including the "initialization event" of the game (e.g. the starting hands of each player in a

card game).

 Perfect information is importantly different from complete information, which

implies common knowledge of each player's utility functions, payoffs, strategies and

"types". A game with perfect information may or may not have complete information.

 Chess is an example of a game with perfect information as each player can see all the

pieces on the board at all times.

 Other examples of games with perfect information include tic-tac-toe, checkers, infinite

chess.

Imperfect decision games:

 In game theory, a game is imperfect game if each player, when making any decision, is

uninformed of all the events that have previously occurred.

 Imperfect information implies no idea of the move taken by the opponent it means the

player is unaware of the actions chosen by other players.

 Card games where each player's cards are hidden from other players such

as poker and bridge are examples of games with imperfect information

**What is evaluation function?**

Static (Heuristic) Evaluation Functions:

An Evaluation Function:

 estimates how good the current board configuration is for a player.

 Typically, one figures how good it is for the player, and how good it is for the opponent,

and subtracts the opponents score from the players

 Tic-tac-toe : Number of successful moves by X – Number of successful moves by O

 Chess: Value of all white pieces - Value of all black pieces

 Typical values from -infinity (loss) to +infinity (win) or [-1, +1].

 If the board evaluation is X for a player, it’s -X for the opponent.

 Many clever ideas about how to use the evaluation function.

e.g. null move heuristic: let opponent move twice.

 Example:

Evaluating chess boards,

Tic-tac-toe

**Min-max algorithm**

MINI - MAX ALGORITHM:

 Mini-max algorithm is a recursive or backtracking algorithm which is used in decisionmaking and game theory. It provides an optimal move for the player assuming that

opponent is also playing optimally.

 Mini-Max algorithm uses recursion to search through the game-tree.

 Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tictac-toe, go, and various tow-players game. This Algorithm computes the minimax

decision for the current state.

 In this algorithm two players play the game, one is called MAX and other is called MIN.

 Both the players fight it as the opponent player gets the minimum benefit while they get

the maximum benefit.

 Both Players of the game are opponent of each other, where MAX will select the

maximized value and MIN will select the minimized value.

 The minimax algorithm performs a depth-first search algorithm for the exploration of the

complete game tree.

 The minimax algorithm proceeds all the way down to the terminal node of the tree, then

backtrack the tree as the recursion

Draw back:

 The main drawback of the minimax algorithm is that it gets really slow for complex

games such as Chess, go, etc.

 This type of games has a huge branching factor, and the player has lots of choices to

decide.

 This limitation of the minimax algorithm can be improved from alpha-beta pruning

**Alpha-beta pruning.**

α-β Pruning

 Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization

technique for the minimax algorithm.

 As we have seen in the minimax search algorithm that the number of game states it has to

examine are exponential in depth of the tree.

 Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a

technique by which without checking each node of the game tree we can compute the

correct minimax decision, and this technique is called pruning.

 This involves two threshold parameter Alpha and beta for future expansion, so it is

called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.

 Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune

the tree leaves but also entire sub-tree.

The two-parameter can be defined as:

 Alpha: The best (highest-value) choice we have found so far at any point along the path

of Maximizer. The initial value of alpha is -∞.

 Beta: The best (lowest-value) choice we have found so far at any point along the path of

Minimizer. The initial value of beta is +∞.

 The Alpha-beta pruning to a standard minimax algorithm returns the same move as the

standard algorithm does, but it removes all the nodes which are not really affecting the

final decision but making algorithm slow. Hence by pruning these nodes, it makes the

algorithm fast.

 Using α-β Pruning improve search by reducing the size of the game tree

UNIT 2

**What is knowledge base?**

A knowledge base is a set of representations of facts of the world.

Each individual representation is called a sentence.

The sentences are expressed in a knowledge representation language.

Knowledge base = set of sentences in a formal language

 Declarative approach to building an agent (or other system):

Tell it what it needs to know

 Then it can Ask itself what to do - answers should follow from the KB

 Agents can be viewed at the knowledge level - i.e., what they know, regardless of how

implemented

 Or at the implementation level

i.e., data structures in KB and algorithms that manipulate them

**Difference between propositional logic and predicate logic?**

**Difference between Propositional Logic and Predicate Logic**

Logical reasoning forms the basis for a huge domain of computer science and mathematics. They help in establishing mathematical arguments, valid or invalid.

**1. Propositional Logic :**  
A **proposition** is basically a declarative sentence that has a truth value. Truth value can either be true or false, but it needs to be assigned any of the two values and not be ambiguous. The purpose of using propositional logic is to analyze a statement, individually or compositely.

**For example :**  
The following statements :

1. If x is real, then x2 > 0
2. What is your name?
3. (a+b)2 = 100
4. This statement is false.
5. This statement is true.

Are not propositions because they do not have a truth value. They are ambiguous.

But the following statements :

1. (a+b)2 = a2 + 2ab + b2
2. If x is real, then x2 >= 0
3. If x is real, then x2 < 0
4. The sun rises in the east.
5. The sun rises in the west.

Are all propositions because they have a specific truth value, true or false.

The branch of logic that deals with proposition is **propositional logic**.

**2. Predicate Logic :**  
Predicates are properties, additional information to better express the subject of the sentence. A quantified predicate is a proposition , that is, when you assign values to a predicate with variables it can be made a proposition.

**For example :**

In P(x) : x>5, x is the subject or the variable and ‘>5’ is the predicate.

P(7) : 7>5 is a proposition where we are assigning values to the variable x, and it has a truth value, i.e. True.

The set of values that the variables of the predicate can assume is called the Universe or Domain of Discourse or Domain of Predicate.

**Difference between Propositional Logic and Predicate Logic :**

|  | **Propositional Logic** | **Predicate Logic** |
| --- | --- | --- |
| 1 | Propositional logic is the logic that deals with a collection of declarative statements which have a truth value, true or false. | Predicate logic is an expression consisting of variables with a specified domain. It consists of objects, relations and functions between the objects. |
| 2 | It is the basic and most widely used logic. Also known as Boolean logic. | It is an extension of propositional logic covering predicates and quantification. |
| 3 | A proposition has a specific truth value, either true or false. | A predicate’s truth value depends on the variables’ value. |
| 4 | Scope analysis is not done in propositional logic. | Predicate logic helps analyze the scope of the subject over the predicate. There are three quantifiers : Universal Quantifier (∀) depicts for all, Existential Quantifier (∃) depicting there exists some and Uniqueness Quantifier (∃!) depicting exactly one. |
| 5 | Propositions are combined with Logical Operators or Logical Connectives like Negation(¬), Disjunction(∨), Conjunction(∧), Exclusive OR(⊕), Implication(⇒), Bi-Conditional or Double Implication(⇔). | Predicate Logic adds by introducing quantifiers to the existing proposition. |
| 6 | It is a more generalized representation. | It is a more specialized representation. |
| 7 | It cannot deal with sets of entities. | It can deal with set of entities with the help of quantifiers. |

Propositional logic (PL) is the simplest form of logic where all the statements are made by

propositions. A proposition is a declarative statement which is either true or false. It is a

technique of knowledge representation in logical and mathematical form. Propositions can be

either true or false, but it cannot be both. Propositional logic is also called Boolean logic as it

works on 0 and 1.

Examples:

Today is Tuesday.

The Sun rises from West (False proposition)

2+2= 5(False proposition)

Syntax of Propositional Logic:

The syntax of propositional logic defines the allowable sentences. The atomic sentences consist

of a single proposition symbol. Each such symbol stands for a proposition that can be true or

false. We use symbols that start with an uppercase letter and may contain other letters or

subscripts.

Example: P, Q, R, W1, 3 and North

In propositional logic there are two types of propositions. They are:

1. Atomic Propositions

2. Compound Propositions

Atomic Proposition: Atomic propositions are the simple propositions. It consists of a single

proposition symbol. These are the sentences which must be either true or false.

Ex: The sun rises from west . (False Atomic Propositions)

The sun rises from east (True Atomic Propositions)

Compound proposition: Compound propositions are constructed by combining simpler or

atomic propositions, using parenthesis and logical connectives.

Ex: Arsalan is an engineer and arsalan lives in dubai.

Arsalan is an engineer or arsalan is a doctor.

Propositional logic is the simplest logic – illustrates basic ideas

The proposition symbols P1, P2 etc are (atomic) sentences

 If S is a sentence, (S) is a sentence (negation)

 If S1 and S2 are sentences, (S1  S2) is a sentence (conjunction)

 If S1 and S2 are sentences, (S1  S2) is a sentence (disjunction)

 If S1 and S2 are sentences, (S1  S2) is a sentence (implication)

 If S1 and S2 are sentences, (S1  S2) is a sentence (biconditional)

A table with black text

Description automatically generated with medium confidenceLimitations of Propositional Logic:

We cannot represent relations like All, some, or none with propositional logic.

Examples:

All boys are smart.

All girls are hardworking.

Some mangoes are sweet.

Propositional logic has limited expressive power and in propositional logic, we cannot describe

statements in terms of their properties or logical relationships.

**What is inference?**

Inference: In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, so generating the conclusions from evidence and facts is termed as Inference

Techniques:

 Forward Chaining

 Backward Chaining

 Resolution

Forward chaining: Forward chaining is the process where it matches the set of conditions and

infer results from these conditions. It is data driven approach using which it reaches (infers) to

the goal condition.

Example: Given A (is true)

B->C

A->B

C->D

Prove D is also true.

Solution: Starting from A, A is true then B is true (A->B)

B is true then C is true (B->C)

C is true then D is true Proved (C->D)

Backward chaining: Backward chaining is the process where it performs backward search from

goal to the conditions used to get the goal. It is goal driven approach using which it reaches to

the initial condition.

Example: Given A (is true)

B->C

A->B

C->D

Prove D is also true.

Solution: Starting from D,

Let D is true then C is true (C->D)

C is true then B is true (B->C)

B is true then A is true Proved (A->B)

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form

Steps in Resolution:

 Conversion of facts into first-order logic.

 Negate the statement which needs to prove (proof by contradiction)

 Convert FOL statements into CNF

 Draw resolution graph (unification).

**What is inference engine?**

Inference Engine(Rules of Engine)

➢ The inference engine is known as the brain of the expert

system as it is the main processing unit of the system. It

applies inference rules to the knowledge base to derive a

conclusion or deduce new information. It helps in deriving

an error-free solution of queries asked by the user.

➢ With the help of an inference engine, the system extracts

the knowledge from the knowledge base.

There are two types of inference engine:

➢ Deterministic Inference engine: The conclusions drawn

from this type of inference engine are assumed to be true. It

is based on facts and rules

Probabilistic Inference engine: This type of inference

engine contains uncertainty in conclusions, and based on

the probability.

Inference engine uses the below modes to derive the

solutions:

➢ Forward Chaining: It starts from the known facts and

rules, and applies the inference rules to add their conclusion

to the known facts.

➢ Backward Chaining: It is a backward reasoning method

that starts from the goal and works backward to prove the

known facts

**What is resolution?**

Resolution in FOL:

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e.,

proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year

1965.Resolution is used, if there are various statements are given, and we need to prove a

conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution

is a single inference rule which can efficiently operate on the conjunctive normal form or clausal

form.

 Clause: Disjunction of literals (an atomic sentence) is called a clause. It is also known as

a unit clause.

 Conjunctive Normal Form: A sentence represented as a conjunction of clauses is said

to be conjunctive normal form or CNF.

Steps in Resolution:

 Conversion of facts into first-order logic.

 Negate the statement which needs to prove (proof by contradiction)

 Convert FOL statements into CNF

 Draw resolution graph (unification).

Example:

Facts:

All peoples who are graduating are happy

All happy people smile

Some one is graduating

Prove that “Is someone smiling?” using resolution

1. Conversion of Facts into first order logic:

 All peoples who are graduating are happy

∀x(Graduating(x) ⇒ happy(x))

 All happy people smile

∀x(happy(x) ⇒ smile(x))

 Some one is graduating

∃x graduating(x)

 “Is someone smiling”

∃x smile(x)

2. Negate the statement which needs to prove (proof by contradiction)

 We need to proof “Is someone smiling”.

 So negate the statement (proof by contradiction)

¬ ∃x smile(x)

3. Convert FOL statements into CNF

Steps to convert FOL to CNF

 Eliminate Implication

 Standardize variable

 Move Negation inwards

 Skolemization

 Drop Universal Quantifier

Eliminate Implication

α ⇒ β ≡ ¬ α ∨ β

α ⇔ β ≡ (α ⇒ β) ∧ (β ⇒ α )

Example:

Eliminate Implication: α ⇒ β ≡ ¬ α ∨ β

 ∀x(¬ Graduating(x) ∨ happy(x))

 ∀x(¬happy(x) ∨ smile(x))

 ∃x graduating(x)

 ¬∃x smile(x)

Standardize variable

∀x regular(x) ∀x regular(x)

∃x busy(x) ∃y busy(y)

∃x attentive(x) ∃z attentive(z)

Example:

 ∀x(¬ Graduating(x) ∨ happy(x))

 ∀y(¬happy(y) ∨ smile(y))

 ∃z graduating(z)

 ¬ ∃w smile(w)

Move Negation inwards

 ¬ (∀x P(x)) ≡ ∃x ¬P(x)

 ¬ (∃ x P(x)) ≡ ∀x ¬P(x)

 ¬ (α ∨ β) ≡ ¬ α ∧ ¬ β

 ¬ (α ∧ β) ≡ ¬ α ∨ ¬ β

 ¬ (¬ α) ≡ α

Example:

 ∀x(¬ Graduating(x) ∨ happy(x))

 ∀y(¬happy(y) ∨ smile(y))

 ∃z graduating(z)

 ∀w ¬ smile(w)

Skolemization: Removing of Existential Quantifier and replacing it by Skolem Constants

∃(y) busy(y)

∃(x) attentive(x)

After Skolemization it changes into:

busy(A)

attentive(B)

Example:

∀x(¬ Graduating(x) ∨ happy(x))

∀y(¬happy(y) ∨ smile(y))

After Skolemization it changes into:

graduating(A)

∀w ¬ smile(w)

Drop Universal Quantifier:

∀(y) busy(y)

∀(x) attentive(x)

After dropping it changes into:

busy(y)

attentive(x)

Example:

 ¬ Graduating(x) ∨ happy(x)

 ¬happy(y) ∨ smile(y)

 graduating(A)

 ¬ smile(w)

Now the sentences are in CNF

4. Resolution Graph

 If fact F is to be proved then it start with ¬F

 It contradicts all other rules in KB

 The process stop when it returns Null clause

Example:

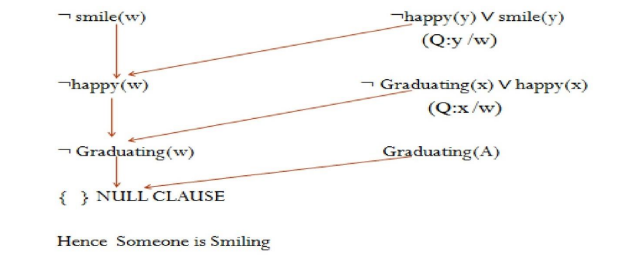
 ¬ Graduating(x) ∨ happy(x)

 ¬happy(y) ∨ smile(y)

 graduating(A)

 ¬ smile(w)

Resolution Graph:



**What are the types of knowledge representation?**

Following are the kind of knowledge which needs to be represented in AI systems:

* **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
* **Events:** Events are the actions which occur in our world.
* **Performance:** It describe behavior which involves knowledge about how to do things.
* **Meta-knowledge:** It is knowledge about what we know.
* **Facts:** Facts are the truths about the real world and what we represent.
* **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

**Knowledge:** Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

**1.Declarative Knowledge:**

* Declarative knowledge is to know about something.
* It includes concepts, facts, and objects.
* It is also called descriptive knowledge and expressed in declarativesentences.
* It is simpler than procedural language.

**2. Procedural Knowledge**

* It is also known as imperative knowledge.
* Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
* It can be directly applied to any task.
* It includes rules, strategies, procedures, agendas, etc.
* Procedural knowledge depends on the task on which it can be applied.

**3. Meta-knowledge:**

* Knowledge about the other types of knowledge is called Meta-knowledge.

**4. Heuristic knowledge:**

* Heuristic knowledge is representing knowledge of some experts in a filed or subject.
* Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

**5. Structural knowledge:**

* Structural knowledge is basic knowledge to problem-solving.
* It describes relationships between various concepts such as kind of, part of, and grouping of something.
* It describes the relationship that exists between concepts or objects.

**What are frames?**

A frame is a record like structure which consists of a collection of attributes and its values to

describe an entity in the world. A frame is analogous to a record structure, corresponding to the

fields and values of a record are the slots and slot fillers of a frame

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

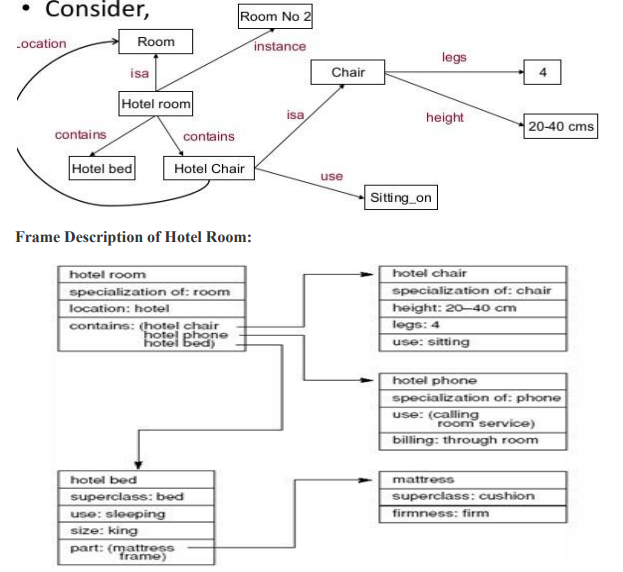
objects. A single frame is not much useful. Frames system consist of a collection of frames

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.

Graphs Representation of Hotel Room:



Frame Description of Hotel Room:

Advantages of Frame representation:

 The frame knowledge representation makes the programming easier by grouping the

related data.

 The frame representation is comparably flexible and used by many applications in AI.

 It is very easy to add slots for new attribute and relations.

 It is easy to include default data and to search for missing values.

 Frame representation is easy to understand and visualize.

Limitations of Frame representation:

 In frame system inference mechanism is not be easily processed.

 Inference mechanism cannot be smoothly proceeded by frame representation.

 Frame representation has a much generalized approach

**What are semantics networks?**

Semantic nets were originally proposed in the early 1960 by M. Ross Quillian to represent the

meaning of English words. 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.

Representation:

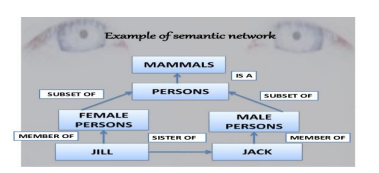
Semantic network representation consists of mainly two types of relations:

1. IS-A relation (Inheritance)

2. Kind-of-relation

Example 1: Following are some statements which we need to represent in the form of nodes and

arcs.



Example 2:

Statements

 Jerry is a cat.

 Jerry is a mammal

 Jerry is owned by Priya.

 Jerry is white colored.

 All Mammals are animal.

A diagram of a cat

Description automatically generated

Advantages of Semantic Networks:

 Semantic networks are a natural representation of knowledge.

 Semantic networks convey meaning in a transparent manner because it is easy to

visualize and understand

 These networks are simple and easily understandable.

 The knowledge engineer can arbitrarily defined the relationships.

 Related knowledge is easily categorised.

 Efficient in space requirements.

 Node objects represented only once.

Limitations of Semantic Networks:

 The limitations of conventional semantic networks were studied extensively by a number

of workers in AI.

 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.

 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.

 These types of representations are inadequate as they do not have any equivalent

quantifier, e.g., for all, for some, none, etc.

 Many believe that the basic notion is a powerful one and has to be complemented by, for

example, logic to improve the notion’s expressive power and robustness.

 Others believe that the notion of semantic networks can be improved by incorporating

reasoning used to describe events.

 Binary relations are usually easy to represent, but sometimes is difficult.

---Problem on semantic network is important----

How planning is different from problem solving?

Planning:

Planning is arranging a sequence of actions to achieve a goal

o or

Planning is the task of coming up with a sequence of actions that will achieve the goal

Planning Problem;

 Find a sequence of actions that achieves a given goal when executed from a given initial

world state

 That is, given

o a set of operator descriptions defining the possible primitive actions by the agent,

o an initial state description, and

o a goal state description or predicate,

o compute a plan, which is

o a sequence of operator instances which after executing them in the initial state

changes the world to a goal state

 Goals are usually specified as a conjunction of goals to be achieved

2.3.2: Planning Vs Problem Solving:

 Planning and problem solving methods can often solve the same sorts of problems

 Planning is more powerful and efficient because of the representations and methods used

 States, goals, and actions are decomposed into sets of sentences (usually in first-order

logic)

 Search often proceeds through plan space rather than state space (though there are also

state-space planners)

 Sub goals can be planned independently, reducing the complexity of the planning

problem

Example: Block World problem

Block world problem assumptions

 Square blocks of same size

 Blocks can be stacked one upon another.

 Flat surface (table) on which blocks can be placed.

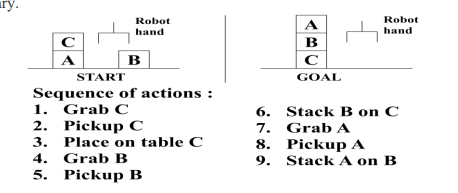
 Robot arm that can manipulate the blocks. It can hold only one block at a time.

In block world problem, the state is described by a set of predicates representing the facts that

were true in that state. One must describe for every action, each of the changes it makes to the

state description. In addition, some statements that everything else remains unchanged is also

necessary.



**What is planning agent?**

Planning Agent:

 An agent interacts with the world via perception and actions

 Perception involves sensing the world and assessing the situation and creating some

internal representation of the world

 Actions are what the agent does in the domain. Planning involves reasoning about actions

that the agent intends to carry out

 Planning is the reasoning side of acting

 This reasoning involves the representation of the world that the agent has, as also the

representation of its actions.

 Hard constraints where the objectives have to be achieved completely for success

 The objectives could also be soft constraints, or preferences, to be achieved as much as

Possible

**Difference between partial order planning and hierarchical planning?**

Partial Order planning:

 A linear planner builds a plan as a totally ordered sequence of plan steps

 A non-linear planner (aka partial-order planner) builds up a plan as a set of steps

with some temporal constraints

o constraints like S1<S2 if step S1 must come before S2.

 One refines a partially ordered plan (POP) by either:

o adding a new plan step, or

o adding a new constraint to the steps already in the plan.

 Works on several sub goals independently

 Solves them with sub plans

 Combine the sub plans

 Flexibility in ordering the sub plans

 Least commitment strategy:

delaying a choice during search

POP Example- putting on a pair of shoes

 Goal( RightShoeOn, LeftShoeOn)

 Init()

 Actions: RightShoe

o PRECOND: RightSockOn

o EFFECT: RightShoeOn

 Actions: RightSock

o PRECOND: None

o EFFECT: RightSockOn

 Actions: LeftShoe

o PRECOND: LeftSockOn

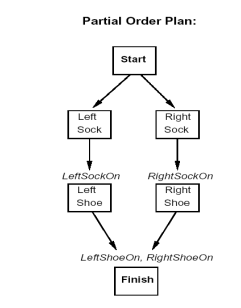
o EFFECT: LeftShoeOn

 Actions: LeftSock

o PRECOND: None

o EFFECT: LeftSockOn

A simple graphical notation:



Hierarchical (HTN) Planning:

In order to solve hard problems, a problem solver may have to generate long plans, it is

important to be able to eliminate some of the details of the problem until a solution that

addresses the main aspect is found. Then an attempt can be made to fill the appropriate details.

Early attempts to do this involved the use of macro operators.

But in this approach, no details were eliminated from actual descriptions of the operators.

 As an example, suppose you want to visit a friend in London but you have a limited

amount of cash to spend. First preference will be to find the airfares, since finding an

affordable flight will be the most difficult part of the task. You should not worry about of

about your drive way, planning a route to the airport etc, until you are sure you have a

flight.

Structure of Hierarchical (HTN) Planning :

Hierarchical Task Networks (from here on abbreviated as HTN) is a hierarchical planning

technique where actions are divided into different levels, or hierarchies. There are different types

of actions residing within these hierarchies, one of them being High Level Actions (from here on

abbreviated as HLA). These HLAs could be any action that you can divide into smaller actions

called refinements. These refinements may be sequences of actions or even other HLAs.

Refinements in turn are broken down into implementations (also called primitive actions) which

refers to any action that has no further refinements or implementations.

1. **Partial Order Planning (POCL)**:
   * **Definition**: POCL is a planning approach that deals with **partially ordered plans**. In such plans, actions are not strictly sequenced; instead, they can be executed in a flexible order.
   * **Representation**: POCL plans consist of actions and causal links. Causal links protect preconditions of actions, ensuring that they can be executed in a valid order.
   * **Example**: Consider a POCL plan where some actions are connected by causal links. However, not all preconditions are protected, making it an incomplete solution. Further reasoning is needed to address any missing links or threats.
   * [**Complexity Studies**: Researchers have explored the computational complexity of optimizing POCL plans by adding or changing causal links and ordering constraints1](https://interactiveaimag.org/columns/extended-abstracts/hierarchical-planning-and-reasoning-about-partially-ordered-plans-from-theory-to-practice-new-faculty-highlights-extended-abstract/).
   * **Use Cases**: POCL planning finds applications in scenarios where strict linear execution is not necessary, and flexibility in action ordering is beneficial.
2. **Hierarchical Planning**:
   * **Objective**: Hierarchical planning aims to refine an initial **partial plan** into a complete plan without abstract tasks or flaws.
   * **Approach**: Instead of finding a linear sequence of operators, hierarchical planning focuses on breaking down the problem into sub-problems. These sub-problems are interdependent, forming a hierarchy.
   * **Representation**: Hierarchical plans involve abstract tasks and their refinements. The goal is to transform the initial state into the desired goal state.
   * **Example**: Imagine a complex problem where reaching the goal state from the initial state requires breaking it down into manageable sub-tasks.
   * [**Application**: Hierarchical planning is useful for solving challenging problems by organizing them hierarchically and refining the plan step by step](https://interactiveaimag.org/columns/extended-abstracts/hierarchical-planning-and-reasoning-about-partially-ordered-plans-from-theory-to-practice-new-faculty-highlights-extended-abstract/)[2](https://www.codingninjas.com/studio/library/non-linear-planning-in-ai)[3](https://hal.science/hal-03478020/document).

In summary, POCL planning deals with partially ordered plans, while hierarchical planning breaks down complex problems into manageable sub-tasks. Both approaches offer unique advantages based on the specific problem domain and requirements.

UNIT 3

**How expert system helps for building AI models.?**

Definition of Expert Systems:

An expert system is AI software that simulates the behavior and judgment of a human expert in a specific domain.

It acquires relevant knowledge from its knowledge base and interprets it to solve problems, just as a human expert would.

Expert systems are widely used in areas such as medical diagnosis, accounting, coding, and games.

Components of an Expert System:

Knowledge Base:

Represents facts, rules, procedures, and intrinsic data relevant to a specific domain.

Contains knowledge contributed by human experts.

Inference Engine:

Fetches relevant knowledge from the knowledge base.

Interprets the knowledge to find solutions for user problems.

Infers new facts by applying rules to known information.

May include explanation and debugging abilities.

Knowledge Acquisition and Learning Module:

Allows the expert system to acquire knowledge from various sources.

Stores acquired knowledge in the knowledge base.

Role of Knowledge Engineers:

Knowledge Engineering is the process of building an expert system.

Knowledge Engineers ensure that the computer possesses all necessary knowledge to solve problems.

They choose suitable forms to represent knowledge in the computer’s memory.

Examples of Expert Systems:

MYCIN:

Early expert system based on backward chaining.

Identifies bacteria causing severe infections and recommends drugs based on weight.

DENDRAL:

Used for chemical analysis.

Predicts molecular structures from spectrographic data.

R1/XCON:

Selects software to generate desired computer systems.

CaDet:

Identifies lung cancer type and degree based on patient data.

DXplain:

Suggests diseases based on doctor findings.

In summary, expert systems preserve human experts’ knowledge in their knowledge base and provide intelligent solutions to complex problems. They bridge the gap between human expertise and AI capabilities, making them valuable tools in building AI models

**Explain the phases in building an Expert system?**

Architecture of expert system?

Application of Expert system

What is uncertainty?

How bayes rule help in decision making?

What is fuzzy logic?

Fuzzy logic system architecture?

What is utility function?

Write a short note on utility theory?