Unit – 1

1) Define Artificial intelligence and mention the sub areas and applications of AI?

A) Artificial Intelligence is concerned with the design of intelligence in an artificial device. The term was coined by John McCarthy in 1956.

- Intelligence is the ability to acquire, understand and apply the knowledge to achieve goals in the world.
- AI is the study of the mental faculties through the use of computational models
- AI is the study of intellectual/mental processes as computational processes.
- AI program will demonstrate a high level of intelligence to a degree that equals or exceeds the intelligence required of a human in performing some task.
- AI is unique, sharing borders with Mathematics, Computer Science, Philosophy, Psychology, Biology, Cognitive Science and many others.
- Although there is no clear definition of AI or even Intelligence, it can be described as an attempt to build machines that like humans can think and act, able to learn and use knowledge to solve problems on their own.

History of AI:

Important research that laid the groundwork for AI:

In 1931, Goedel layed the foundation of Theoretical Computer Science1920-30s:

He published the first universal formal language and showed that math itself is either

flawed or allows for unprovable but true statements.

In 1936, Turing reformulated Goedel's result and church's extension thereof

In 1956, John McCarthy coined the term "Artificial Intelligence" as the topic of the Dartmouth Conference, the first conference devoted to the subject.

Sub Areas of AI:

1) Game Playing

Deep Blue Chess program beat world champion Gary Kasparov

2) Speech Recognition

PEGASUS spoken language interface to American Airlines' EAASY SABRE reseration system, which allows users to obtain flight information and make reservations over the telephone. The 1990s has seen significant advances in speech recognition so that limited systems are now successful.

3) Computer Vision

Face recognition programs in use by banks, government, etc. The ALVINN system from CMU autonomously drove a van from Washington, D.C. to San Diego (all but 52 of 2,849 miles), averaging 63 mph day and night, and in all weather conditions. Handwriting recognition, electronics and manufacturing inspection, photo interpretation, baggage inspection, reverse engineering to automatically construct a 3D geometric model.

4) Expert Systems

Application-specific systems that rely on obtaining the knowledge of human experts in an area and programming that knowledge into a system.

a. Diagnostic Systems : MYCIN system for diagnosing bacterial infections of the blood and suggesting treatments. Intellipath pathology diagnosis system (AMA approved). Pathfinder medical diagnosis system, which suggests tests and makes diagnoses. Whirlpool customer assistance center.

b. System Configuration

DEC's XCON system for custom hardware configuration. Radiotherapy treatment planning.

c. Financial Decision Making

Credit card companies, mortgage companies, banks, and the U.S. government employ AI systems to detect fraud and expedite financial transactions. For example, AMEX credit check.

d. Classification Systems

Put information into one of a fixed set of categories using several sources of information. E.g., financial decision making systems. NASA developed a system for classifying very faint areas in astronomical images into either stars or galaxies with very high accuracy by learning from human experts' classifications.

5) Mathematical Theorem Proving

Use inference methods to prove new theorems.

6) Natural Language Understanding

AltaVista's translation of web pages. Translation of Catepillar Truck manuals into 20 languages.

7) Scheduling and Planning

Automatic scheduling for manufacturing. DARPA's DART system used in Desert Storm and Desert Shield operations to plan logistics of people and supplies. American Airlines rerouting contingency planner. European space agency planning and scheduling of spacecraft assembly, integration and verification.

8) Artificial Neural Networks:

9) Machine Learning

Application of AI:

AI algorithms have attracted close attention of researchers and have also been applied successfully to solve problems in engineering. Nevertheless, for large and complex problems, AI algorithms consume considerable computation time due to stochastic feature of the search approaches

1) Business; financial strategies

2) Engineering: check design, offer suggestions to create new product, expert systems for all engineering problems

3) Manufacturing: assembly, inspection and maintenance

4) Medicine: monitoring, diagnosing

5) Education: in teaching

6) Fraud detection

7) Object identification

8) Information retrieval

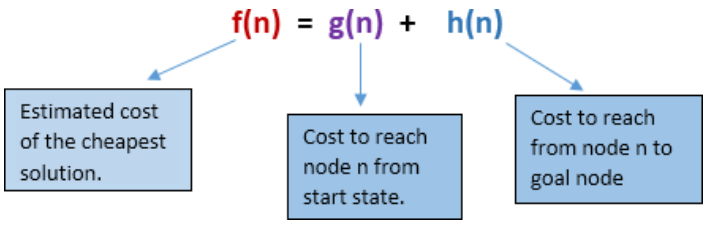
9) Space shuttle scheduling

2. Explain about A* Search with a map example.

Ans: A* Search Algorithm:

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). It has combined features of UCS and greedy best- first search, by which it solve the problem efficiently. 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. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

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 fitnessnumber.



Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

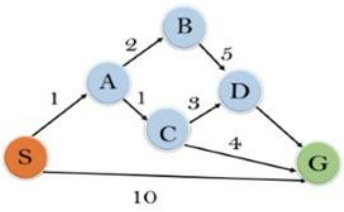
Advantages:

- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

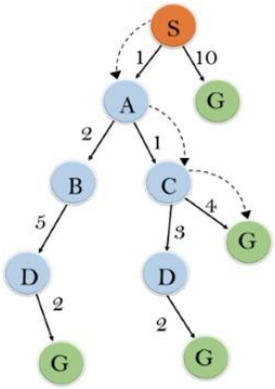
Disadvantages:

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, soit is not practical for various large-scale problems.

Example:



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



3. Describe Problem solving Agentsin detail with an example of taxi driver?

Artificial intelligence is defined as the study of rational agents. A rational agent could be anything that makes decisions, as a person, firm, machine, or software. Itcarries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance).

Types of Agents

- Simple Reflex Agents

- Model- Based Reflex Agents
- Goal- Based Agents
- Utility-

Based Agents

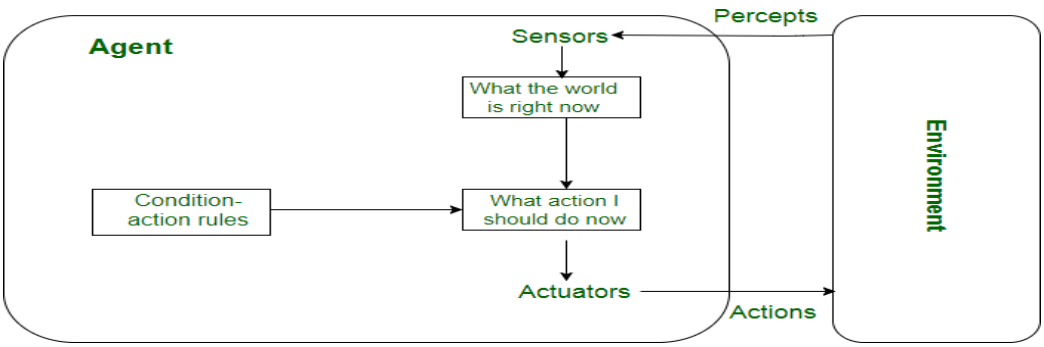
Simple reflex

agents

Simple reflex agents ignore the rest of the percept history and act only on the basis of the current percept. Percept history is the history of all that an agent hasperceived to date. The agent function is based on the condition- action rule.

Problemswith Simple reflex agentsare :

- Very limited intelligence.
- No knowledge of non- perceptual parts of the state.
- Usually too big to generate and store.
- If there occurs any change in the environment, then the collection of rulesneed to be updated.



Model- based reflex agents

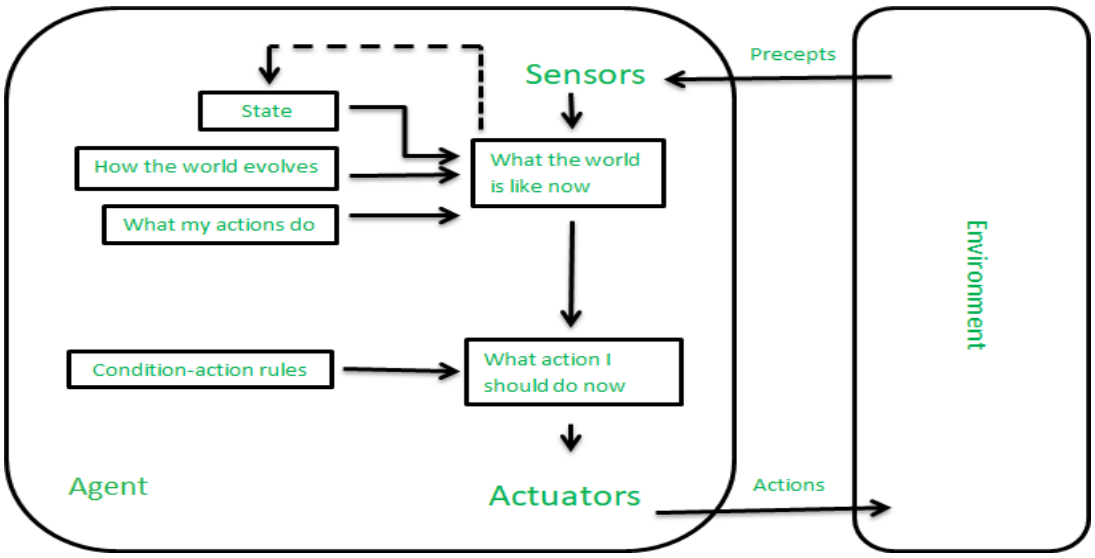
It works by finding a rule whose condition matches the current situation. A model

- based agent can handle partially observable environments by the use of a modelabout the world. The agent has to keep track of the internal state which is adjusted by each percept and that depends on the percept history.

Updating the state requiresinformation about :

- **how the world evolvesindependently from the agent, and**

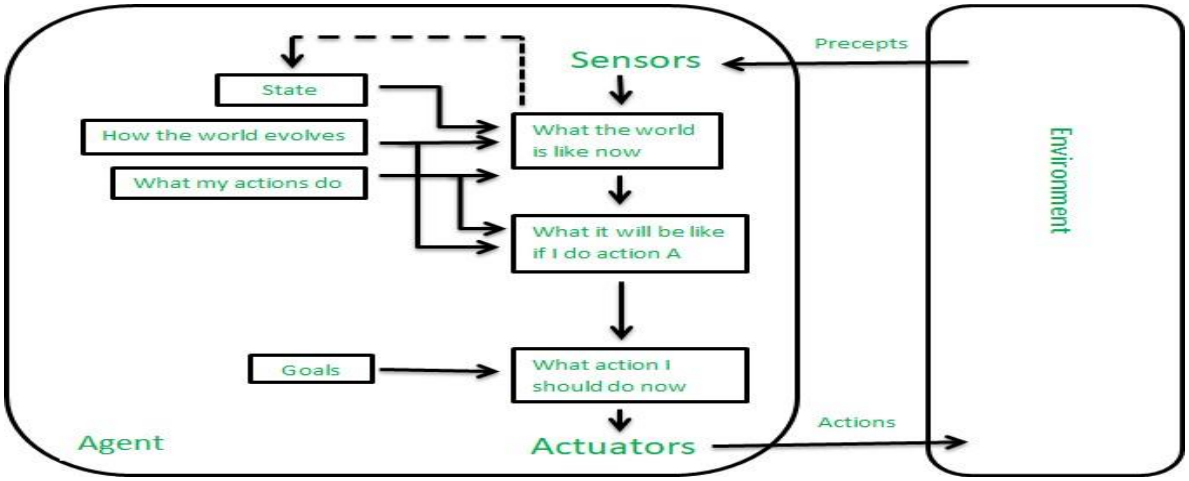
- how the agent'sactionsaffect the world.



Goal- based agents

These kinds of agents take decisions based on how far they are currently from their goal(description of desirable situations). Their every action is intended to reduce its distance from the goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. The

knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require searchand planning. The goal-based agent's behavior can easily be changed.

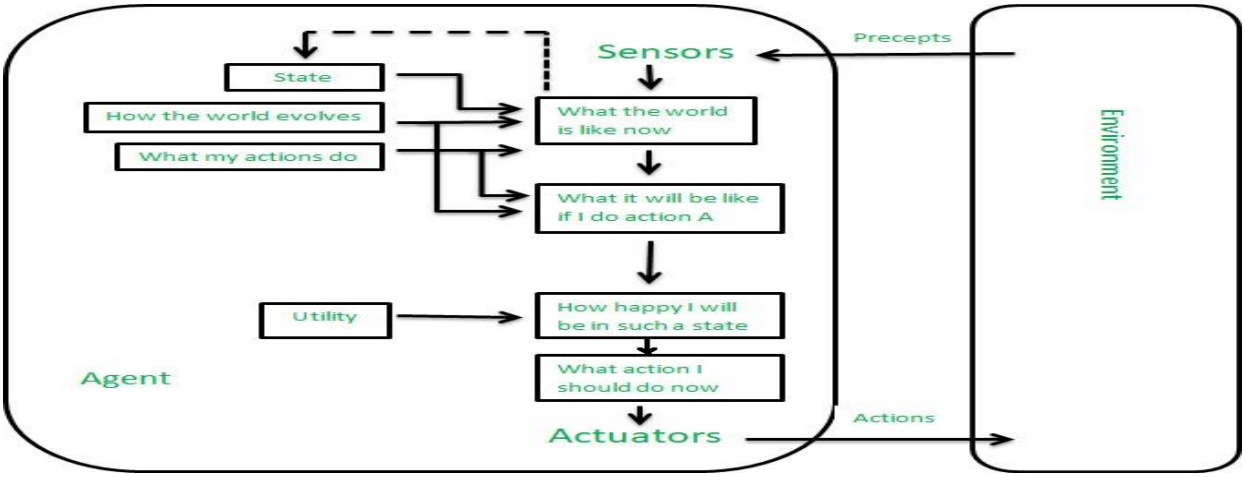


Utility- based agents

The agents which are developed having their end uses as building blocks are called utility-based agents. When there are multiple possible alternatives, then to decide which one is best, utility- based agents are used. They choose actions based on a preference (utility) for

each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a

destination. Agent happiness should be taken into consideration. Utility describeshow “happy” the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.



4)Explain any uninformed search algorithm with complexity analysis?

A)Uninformed Search Algorithms

Uninformed search is a class of general- purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.

Following are the varioustypesof uninformed search algorithms:

1. Breadth- first Search
2. Depth- first Search
3. Depth- limited Search
4. Iterative deepening depth- first search
5. Uniform cost search
6. Bidirectional Search

1. Breadth- first Search:

- Breadth- first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth- first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth- first search algorithm is an example of a general- graph search algorithm.
- Breadth- first search implemented using FIFO queue data structure.

Advantages:

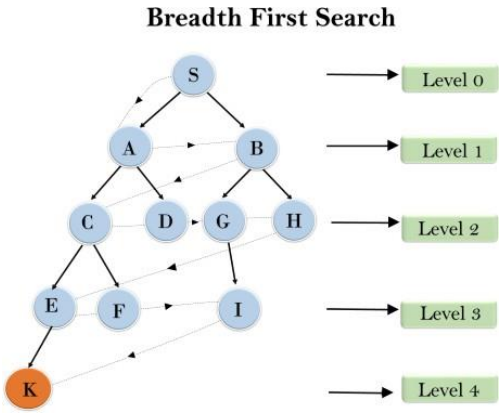
- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

Example:

S- - - > A--- >B- - - - >C- - - >D- - - - >G- - - >H- - - >E- - - - >F- - - - >I >K



Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$T(b) = 1+b^2+b^3+ + b^d = O(b^d)$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non- decreasing function of the depth of the node.

5. Elucidate Uniform cost Search in detail.

A) Uniform- cost Search Algorithm:

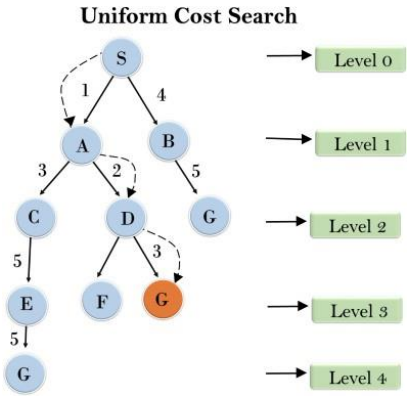
Uniform- cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform- cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform- cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform- cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.



Completeness:

Uniform- cost search is complete, such as if there is a solution, UCS will find it..

Time Complexity:

Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

Hence, the worst- case time complexity of Uniform- cost search is$\mathbf{O(b^{1 + [C^*/\varepsilon]})/}$

Space Complexity:

The same logic is for space complexity so, the worst- case space complexity of Uniform- cost search is **O(b1 + [C*/ε])**.

Optimal:

Uniform- cost search is always optimal as it only selects a path with the lowest path cost.

UNIT-2

1. Explain about Hill climbing search algorithm with an example?

A) Hill Climbing Algorithm in Artificial Intelligence

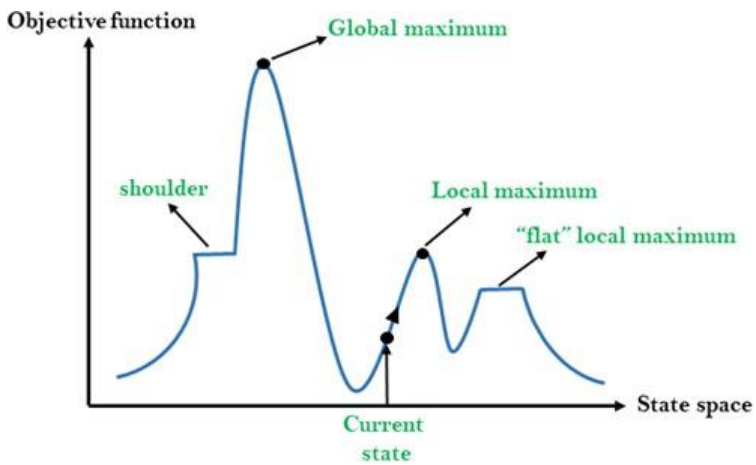
- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and valueHill Climbing is mostly used when a good heuristic is available.

Featuresof Hill Climbing:

Following are some main features of Hill Climbing Algorithm:

- **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
- **Greedy approach:** Hill- climbing algorithm search moves in the direction which optimizes the cost.
- **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

State- space Diagram for Hill Climbing:



Different regions in the state space landscape:

Local Maximum: Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

Global Maximum: Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

Current state: It is a state in a landscape diagram where an agent is currently present.

Flat local maximum: It is a flat space in the landscape where all the neighbor states of current states have the same value.

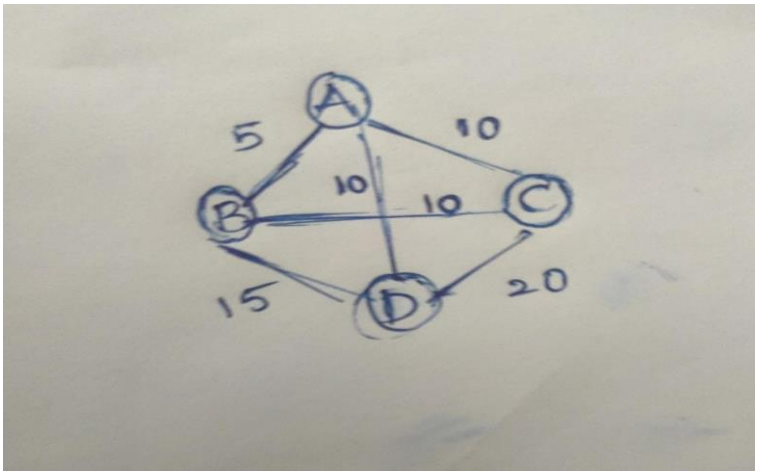
Shoulder: It is a plateau region which has an uphill edge**.**

2) **Describe Travelling Salesman Problem and solve it using heuristic approach? Ans: statement:**

A) In travelling salesman problem (tsp) , one is required to find the shortest route of visiting all the cities once and returning back to starting point. Assume that there are 'n' cities and the distance between each pair of the cities is given

The problem seems to be simple but deceptive. The tsp is one of the most intensely studied problems in computational mathematics and yet no effective method is known for the general case.

In this problem, a simple motion causing and systematic control strategy could, in principle, be applied to solve it. All possible paths of the search tree are explored and the shortest path is



returned. This will require (n- 1)! (i.e.., factorial of n- 1) paths to be examined for 'n' cities. if number of cities grows, then the time required to wait a salesman to get the information about the shortest path is not a practical situation. This phenomenon is called combinatorial explosion.

Therefore, some kind of thumb rules or heuristic techniques may be thought of and applied furthermore, there may be more than one ways of solving the problem and one would like to

exercise a choice between various solution paths based on some criteria of goodness or on some thumb rule. The following sections describe various heuristics search techniques.

Paths explored. Assume A to be start city

	Distance
1. $A \xrightarrow{5} B \xrightarrow{10} C \xrightarrow{20} D \xrightarrow{10} A$	45
2. $A \xrightarrow{5} B \xrightarrow{15} D \xrightarrow{20} C \xrightarrow{A} A$	50
3. $A \xrightarrow{10} D \xrightarrow{10} B \xrightarrow{10} C \xrightarrow{10} A$ (Best path)	40
4. $A \xrightarrow{10} D \xrightarrow{20} C \xrightarrow{10} B \xrightarrow{5} A$	45
5. $A \xrightarrow{10} C \xrightarrow{10} B \xrightarrow{15} D \xrightarrow{10} A$	45
6. $A \xrightarrow{10} C \xrightarrow{20} D \xrightarrow{15} B \xrightarrow{5} A$	50

3) Elucidate Constraint Satisfaction Problem with the given example,

Example: There are four countrieson the map that can be coloured by using just two colours(Red, Green) in such a way that countriessharing an edge have different colours.

Ans: **Constraint satisfaction:**

Many ai problems can be viewed as problems of constraint satisfaction in which the goal is to solve some problem state that satisfies a given set of constraints instead of finding optimal path to the solution. Such problems are called constraints

satisfaction cs problem. Search can be made easier in those cases in which the

solution is required to satisfy local consistency conditions, For example, some of the simple constraint satisfaction problems are cryptography, the n- queen problem, map coloring, crossword puzzle, etc..

4. Elaborate simulated annealing algorithm in detail with an example.

Ans: SIMULATED ANNEALING

A Simulated annealing algorithm is a method to solve bound-constrained and unconstrained optimization parameters models. The method is based on physical annealing and is used to minimize system energy.

In every simulated annealing example, a random new point is generated. The distance between the current point and the new point has a basis of the probability distribution on the scale of the proportion of temperature. The algorithm aims at all those points that minimize the objective with certain constraints and probabilities. Those points that raise the objective are also accepted to explore all the possible solutions instead of concentrating only on local minima. Optimization by simulated annealing is performed by systematically decreasing the temperature and minimising the search's extent.

IMPLEMENTSIMULATED ANNEALING

There are a set of steps that are performed for simulated annealing in ai. These steps can be summarized as follows:

Simulated annealing creates a trial point randomly. The algorithm selects the distance between the current point and the trial point by a probability distribution. The scale of such distribution is temperature. With the annealing function trial, point distribution distance is set. To keep the boundaries intact, the trial point is shifted gradually.

The Simulated Annealing formula then determines if the new point is better than the older or not. If the new point is better, it is made as a next point, while if the new point is worse, it can still be accepted depending upon the simulated annealing acceptance function.

A systematic algorithm gradually reduces the temperature selecting the best point that gets generated in the process. For lowering the values, the annealing parameters are set, raising and reducing the temperatures. The simulated annealing parameters are based on the values of the probable gradients of every dimension of the objective.

The simulated annealing is concluded when it reaches the lowest minima or any of the specific stopping criteria.

SIMULATED ANNEALING WORKED EXAMPLE

To understand how simulated-annealing works, one can take the example of a traveling salesman.

The solution can be created by applying any of the language selections. Let us understand the problem and the solution with simulated-annealing applications.

At the onset, a city class needs to be created to specify several destinations the travelling salesman would visit.

After that, a class has to be created that keeps track of the cities.

Then a class is created that models the tour of the travelling salesman.

With all the different classes and the information in hand, a simulated-annealing algorithmis created.

Thus with the types of optimization problems, a relatively simpler algorithm is created, and

the solution is sought.

5) Explain the different typesof consistency used in Constraint Propagation.

Ans:Constraint propagation: Inference in CSPs

A number of inference techniques use the constraints to infer which variable/value pairs areconsistent and which are not. These include node, arc, path, and kconsistent.

constraint propagation: Using the constraints to reduce the number of legal values for a variable, which in turn can reduce the legal values for another variable,and so on.
local consistency: If we treat each variable as a node in a graph and each binary constraint as anarc, then the process of enforcing local consistency in each part of the graph causes inconsistent values to be eliminated throughout the graph. There are different types of local consistency:
Node consistency
A single variable (a node in the CSP network) is node- consistent if all the values in the variable's domain satisfy the variable's unary constraint.
We say that a network is node-consistent if every variable in the network is nodeconsistent.
Arc consistency
A variable in a CSP is arc-consistent if every value in its domain satisfies the variable's binary constraints.
Xi is arc- consistent with respect to another variable Xj if for every value in the current domain Di there is some value in the domain Dj that satisfies the binary constraint on the arc (Xi, Xj).
A network is arc-consistent if every variable is arc-consistent with every other variable.

Arc consistency tightens down the domains (unary constraint) using the arcs (binary constraints).**Path**

consistency
Path consistency: A two-variable set {Xi, Xj} is path-consistent with respect to a third variable Xm if, for every assignment {Xi = a, Xj = b} consistent with the constraint on {Xi, Xj}, there is an assignment to Xm that satisfies the constraints on {Xi, Xm} and {Xm, Xj}.
Path consistency tightens the binary constraints by using implicit constraints that are inferred by looking at triples of variables.

K-consistency
K- consistency: A CSP is k- consistent if, for any set of k- 1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any kth variable.1-consistency = node consistency;
2-consisency = arc consistency;
3-consistensy = path consistency.
A CSP is strongly k-consistent if it is k-consistent and is also (k - 1)-consistent, (k–2)-consistent, …all the way down to 1-consistent.
A CSP with n nodes and make it strongly n- consistent, we are guaranteed to find a solution in time O(n2d). But algorithm for establishing n- consitentcy must take time exponential in n in the worse case, also requires space that is exponential in n.

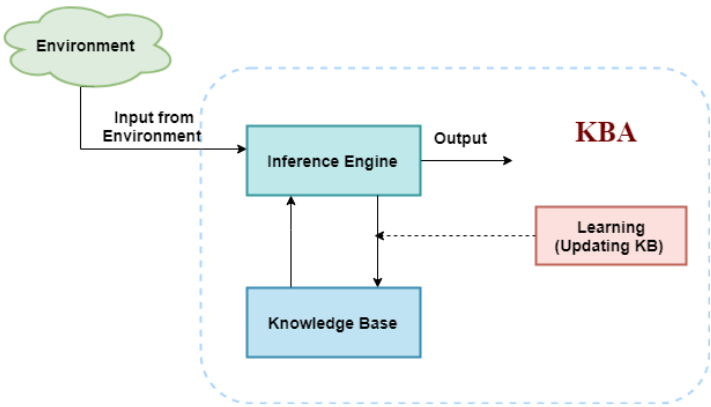
AI Q AND A

UNIT-3

1. **Explain about Knowledge based agents with an example of Wumpus World?**

Ans: Knowledge-Based Agent in Artificial intelligence

- An intelligent agent needs knowledge about the real world for taking decisions and reasoning to act efficiently.
- Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.
- Knowledge-based agents are composed of two main parts:
  - Knowledge-base and
  - Inference system.

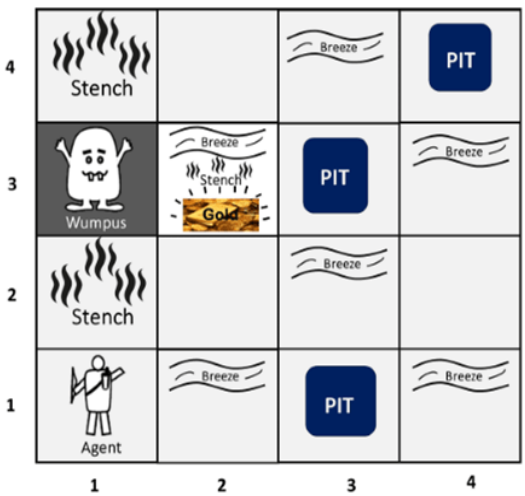


Knowledge base: Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences (here 'sentence' is a technical term and it is not identical to sentence in English). These sentences are expressed in a language which is called a knowledge representation language. The Knowledge-base of KBA stores fact about the world.

Inference system: Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information.

A knowledge-based agent must able to do the following:

- An agent should be able to represent states, actions, etc.
- An agent Should be able to incorporate new percepts
- An agent can update the internal representation of the world
- An agent can deduce the internal representation of the world
- An agent can deduce appropriate actions.



WUMPUS WORLD PROBLEM

Performance Measure

Gold: 1000
Death: -1000
Step: -1
Using arrow: -10

Edit with WPS Office

Environment

- Square adjacent to Wumpus are stenchy.
- Squares adjacent to pit are breezy.
- Glitter iff gold the same square.
- Shooting kills Wumpus if you are facing it.
- Shooting uses up the arrow.
- Grabbing picks up gold if in same square.
- Releasing drops the gold in same square.

Actuators:
Left turn , Right turn , Forward, Grab, Release, Shoot.

Sensors:
Stench , Breeze, Glitter, Bump, Scream.

2. **Describe propositional Logic along with syntax, semantics and inference Procedure?**

Ans:

Propositional logic in Artificial intelligence

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

Example:

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

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and **logical connectives**.
- These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.
- A proposition formula which has both true and false values is called
- Statements which are questions, commands, or opinions are not propositions such as "**Where is Rohini**", "**How are you**", "**What is your name**", are not propositions.

Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge

Edit with WPS Office

representation. There are two types of 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.

  Example:

1. a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.
2. b) "The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

  Example:

1. a) "It is raining today, and street is wet."
2. b) "Ankit is a doctor, and his clinic is in Mumbai.

Logical Connectives:

Negation: A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
Conjunction: A sentence which has ∧ connective such as, P ∧ Q is called a conjunction.
Example: Rohan is intelligent and hardworking. It can be written as,
P= Rohan is intelligent,
Q= Rohan is hardworking. → P∧ Q.
Disjunction: A sentence which has ∨ connective, such as P ∨ Q. is called disjunction, where P and Q are the propositions.
Example: "Ritika is a doctor or Engineer",
Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as P ∨ Q.
Implication: A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as

If it is raining, then the street is wet.

Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
Biconditional: A sentence such as P ⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive

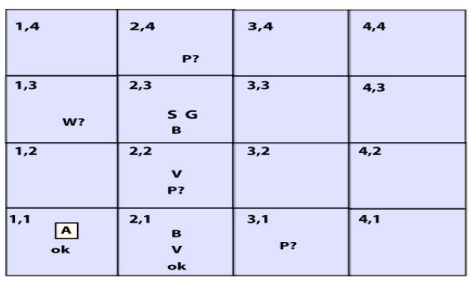
P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.

3. **Explain Agents Based on Propositional Logic with an example of Wumpus World.**

Ans: Knowledge-base for Wumpus world

As in the previous topic we have learned about the wumpus world and how a knowledge-based agent evolves the world. Now in this topic, we will create a knowledge base for the wumpus world, and will derive some proves for the Wumpus-world using propositional logic.

The agent starts visiting from first square [1, 1], and we already know that this room is safe for the agent. To build a knowledge base for wumpus world, we will use some rules and atomic propositions. We need symbol [i, j] for each location in the wumpus world, where i is for the location of rows, and j for column location



Atomic proposition variable for Wumpus world:

- Let $P_{i,j}$ be true if there is a Pit in the room [i, j].

Edit with WPS Office

- Let $B_{i,j}$ be true if agent perceives breeze in [i, j], (dead or alive).
- Let $W_{i,j}$ be true if there is wumpus in the square[i, j].
- Let $S_{i,j}$ be true if agent perceives stench in the square [i, j].
- Let $V_{i,j}$ be true if that square[i, j] is visited.
- Let $G_{i,j}$ be true if there is gold (and glitter) in the square [i, j].
- Let $OK_{i,j}$ be true if the room is safe.

Some Propositional Rules for the wumpus world:

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

Representation of Knowledgebase for Wumpus world:

Following is the Simple KB for wumpus world when an agent moves from room [1, 1], to room [2,1]:

$\neg W_{11}$	$\neg S_{11}$	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	V_{11}	OK_{11}
$\neg W_{12}$	----	$\neg P_{12}$	-----	----	$\neg V_{12}$	OK_{12}
$\neg W_{21}$	$\neg S_{21}$	$\neg P_{21}$	B_{21}	$\neg G_{21}$	V_{21}	OK_{21}

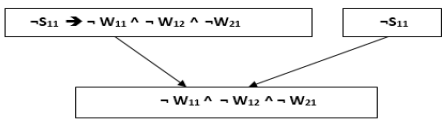
Here in the first row, we have mentioned propositional variables for room[1,1], which is showing that room does not have wumpus($\neg W_{11}$), no stench ($\neg S_{11}$), no Pit($\neg P_{11}$), no breeze($\neg B_{11}$), no gold ($\neg G_{11}$), visited (V_{11}), and the room is Safe(OK_{11}).

In the second row, we have mentioned propositional variables for room [1,2], which is showing that there is no wumpus, stench and breeze are unknown as an agent has not visited room [1,2], no Pit, not visited yet, and the room is safe.

In the third row we have mentioned propositional variable for room[2,1], which is showing that there is no wumpus($\neg$ W21), no stench ($\neg S_{21}$), no Pit ($\neg P_{21}$), Perceives breeze(B_{21}), no glitter($\neg G_{21}$), visited (V_{21}), and room is safe (OK_{21}).

Prove that Wumpus is in the room (1, 3)

We can prove that wumpus is in the room (1, 3) using propositional rules which we have derived for the wumpus world and using inference rule.



Apply Modus Ponens with ¬S11 and R1:

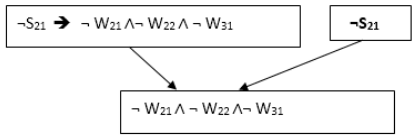
We will firstly apply MP rule with R1 which is $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$, and $\neg S_{11}$ which will give the output $\neg W_{11} \wedge W_{12} \wedge W_{12}$.

Apply And-Elimination Rule:

Edit with WPS Office

After applying And-elimination rule to $\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$, we will get three statements: $\neg W_{11}$, $\neg W_{12}$, and $\neg W_{21}$.

Apply Modus Ponens to $\neg S_{21}$, and R2:

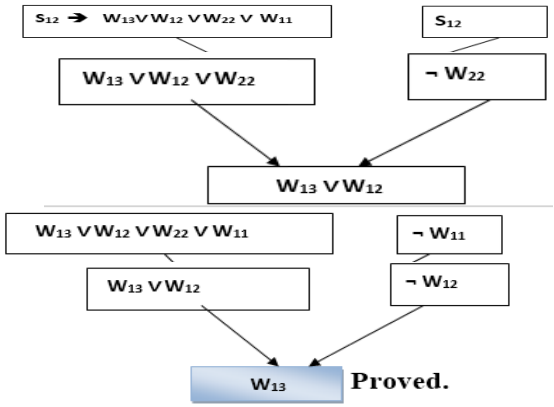


Now we will apply Modus Ponens to $\neg S_{21}$ and R2 which is $\neg S_{21} \rightarrow \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$, which will give the Output as $\neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

Apply And -Elimination rule:

Now again apply And-elimination rule to $\neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$, We will get three statements: $\neg W_{21}$, $\neg W_{22}$, and $\neg W_{31}$.

Apply MP to S_{12} and R4:



Apply Modus Ponens to S_{12} and R_4 which is $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$, we will get the output as $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$.

Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ and $\neg W_{11}$:

After applying Unit resolution formula on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ and $\neg W_{11}$ we will get $W_{13} \vee W_{12} \vee W_{22}$.

Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$ and $\neg W_{22}$:

After applying Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$, and $\neg W_{22}$, we will get $W_{13} \vee W_{12}$ as output.

Apply Unit Resolution on $W_{13} \vee W_{12}$ and $\neg W_{12}$:

After Applying Unit resolution on $W_{13} \vee W_{12}$ and $\neg W_{12}$, we will get W_{13} as an output, hence it is proved that the Wumpus is in the room [1, 3].

4. **Describe Horn clauses and definite clauses with examples.**

Ans:

Horn Clause and Definite clause:

Horn clause and definite clause are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the **first-order definite clause**.

Definite clause: A clause which is a disjunction of literals with **exactly one positive literal** is known as a definite clause or strict horn clause.

Horn clause: A clause which is a disjunction of literals with **at most one positive literal** is known as horn clause. Hence all the definite clauses are horn clauses.

Edit with WPS Office

Example: (¬ p V ¬ q V k). It has only one positive literal k.

It is equivalent to p ∧ q → k.

5.Elucidate Forward and Backward chaining with algorithms in Propositional logic.

Ans:

Propositional logic in Artificial intelligence

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

Forward Chaining and backward chaining in AI

In artificial intelligence, forward and backward chaining is one of the important topics, but before understanding forward and backward chaining lets first understand that from where these two terms came.

Inference engine:

The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:

a. **Forward chaining**
b. **Backward chaining**

A. Forward Chaining

Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

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

Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Edit with WPS Office

B. Backward Chaining:

Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties of backward chaining:

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- The backward-chaining method mostly used a **depth-first search** strategy for proof.

UNIT -4

1. **Explain in detail about first order logic and it's Syntax and Semantics.**

Ans: **First-Order logic:**First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.

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

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

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

Constant	1, 2, A, John, Mumbai, cat,....

Edit with WPS Office

Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	∧, ∨, ¬, ⇒, ⇔
Equality	==
Quantifier	∀, ∃

Atomic sentences:

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

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

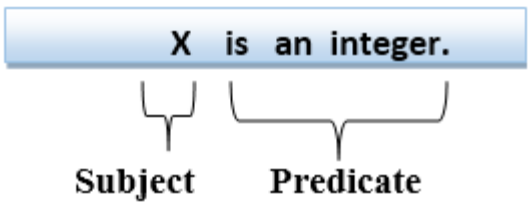
Complex Sentences:

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

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

- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:

  1. **Universal Quantifier, (for all, everyone, everything)**
  2. **Existential quantifier, (for some, at least one).**

2. **Differentiate Propositional vs. First-Order Inference.**

Ans:

Edit with WPS Office

First order logic vs propositional

- basis of reasoning
  - propositional logic: statements
  - first order logic: OBJECT-ORIENTED
    - objects
    - relations
    - functions
    - statements about objects, relations and functions
- possible values of statements
  - true, false, unknown

Propositional logic	First-Order logic
It converts a full sentence into a symbol and makes it logical.	First-Order logic relation of a particular sentence will be made which involves constants, relations, and functions.
The limitation of propositional logic is that it does not represent any individual entities.	It can easily represent the individual establishment which means if the user is writing a single sentence then it can be easily represented in First-Order logic.
It does not signify the generalization, pattern, or specialization.	It does express the generalization, pattern, or specialization.

3. **Discuss the forward chaining and differentiate the efficient forward chaining in first order Logic.**

Ans:

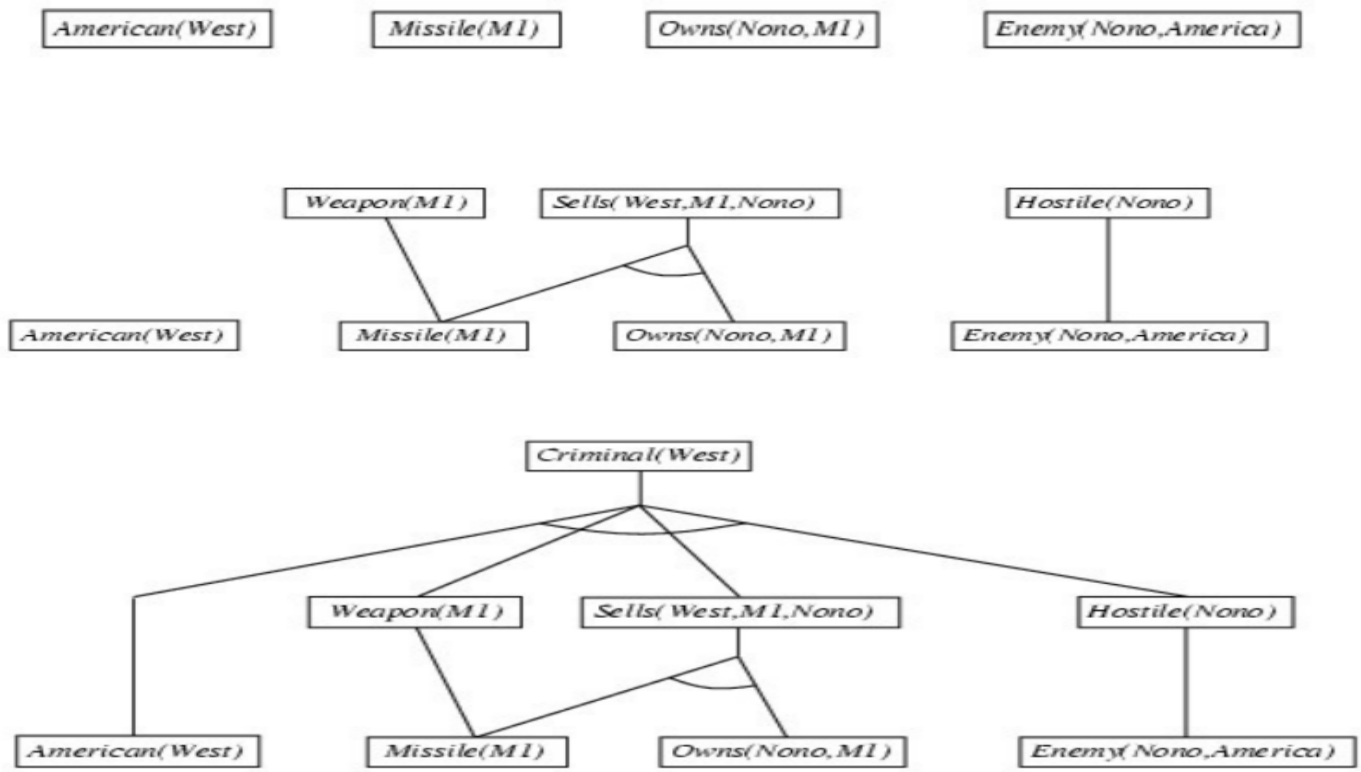
Forward Chaining

- Starting from known facts, it triggers all the rules whose premises are satisfied, adding their conclusions to the known facts. This process will continue until query is answered.
- When a new fact P is added to the KB:
- For each rule such that P unifies with a premise
- if the other premises are already known
- then add the conclusion to the KB and continue chaining
- Forward chaining is data-driven:
- inferring properties and categories from percepts

Forward Chaining Algorithm:

Edit with WPS Office

```
function FOL-FC-ASK(KB, α) returns a substitution or false
   repeat until new is empty
      new ← { }
      for each sentence r in KB do
         ( p1 ∧ . . . ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
         for each θ such that (p1 ∧ . . . ∧ pn)θ = (p'1 ∧ . . . ∧ p'n)θ
                  for some p'1, . . . , p'n in KB
            q' ← SUBST(θ, q)
            if q' is not a renaming of a sentence already in KB or new then do
               add q' to new
               φ ← UNIFY(q', α)
               if φ is not fail then return φ
      add new to KB
   return false
```



Analysis of Forward Chaining

- A fixed point of the inference process can be reached
- The algorithm is sound and complete
- Its efficiency can be improved

4. **Elucidate standard quantifiers and resolution in First Order logic.**

Ans:

Edit with WPS Office

QUANTIFIERS

- A language element which generates a quantification (such as "every") is called a quantifier.
- First-order logic contains two standard quantifiers, called universal and existential.
- **Universal quantification (∀):**
- The expression: ∀ x P(x), denotes the universal quantification of the atomic formula P(x).
- the expression is understood as: "For all x, P(x) holds", "for each x, P(x) holds" or
- "for every x, P(x) holds". is called the universal quantifier, and x means all the objects x
- in the universe.
- for eg. : " All kings are persons,"
- is written in first-order logic as ∀ x King(x) ⇒ Person(x) .
- ∀ is usually pronounced "For all ...". (Remember that the upside-down A stands for "all.")
- Thus, the sentence says, "For all x, if x is a king, then x is a person."
- The symbol x is called a variable. By convention, variables are lowercase letters.
- A term with no variables is called a ground term.

Existential quantification (∃):

- Universal quantification makes statements about every object.
- Similarly, we can make a statement about some object in the universe without naming it, by using an existential quantifier.
- To say, for example, that King John has a crown on his head, we write
- ∃ x Crown(x) ∧ OnHead(x, John) .
- ∃x is pronounced "There exists an x such that ..." or "For some x...".
- Intuitively, the sentence ∃ x P says that P is true for at least one object x. More
- precisely, ∃ x P is true in a given model if P is true in at least one extended interpretation
- that assigns x to a domain element. That is, at least one of the following is true:
- Richard the Lionheart is a crown ∧ Richard the Lionheart is on John's head;
- King John is a crown ∧ King John is on John's head;
- Richard's left leg is a crown ∧ Richard's left leg is on John's head;
- John's left leg is a crown ∧ John's left leg is on John's head;
- The crown is a crown ∧ the crown is on John's head.

Resolution in FOL

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 for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

To better understand all the above steps, we will take an example in which we will apply resolution.

Example:

a. **John likes all kind of food.**
b. **Apple and vegetable are food**

Edit with WPS Office

c. Anything anyone eats and not killed is food.
d. Anil eats peanuts and still alive
e. Harry eats everything that Anil eats. Prove by resolution that:
f. John likes peanuts.

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

a. ∀x: food(x) → likes(John, x)

b. food(Apple) ∧ food(vegetables)

c. ∀x ∀y: eats(x, y) ∧ ¬ killed(x) → food(y)

d. eats (Anil, Peanuts) ∧ alive(Anil).

e. ∀x : eats(Anil, x) → eats(Harry, x)

f. ∀x: ¬ killed(x) → alive(x) } added predicates.

g. ∀x: alive(x) →¬ killed(x)

h. likes(John, Peanuts)

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

- Eliminate all implication (→) and rewrite
  1. ∀x ¬ food(x) V likes(John, x)
  2. food(Apple) Λ food(vegetables)
  3. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
  4. eats (Anil, Peanuts) Λ alive(Anil)
  5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
  6. ∀x¬ [¬ killed(x)] V alive(x)
  7. ∀x ¬ alive(x) V ¬ killed(x)
  8. likes(John, Peanuts).
- Move negation (¬)inwards and rewrite

1. ∀x ¬ food(x) V likes(John, x)
   2. food(Apple) Λ food(vegetables)
   3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
   4. eats (Anil, Peanuts) Λ alive(Anil)
   5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
   6. ∀x ¬killed(x)] V alive(x)
   7. ∀x ¬ alive(x) V ¬ killed(x)
   8. likes(John, Peanuts).

- Rename variables or standardize variables

1. ∀x ¬ food(x) V likes(John, x)
   2. food(Apple) Λ food(vegetables)
   3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
   4. eats (Anil, Peanuts) Λ alive(Anil)
   5. ∀w¬ eats(Anil, w) V eats(Harry, w)
   6. ∀g ¬killed(g)] V alive(g)
   7. ∀k ¬ alive(k) V ¬ killed(k)
   8. likes(John, Peanuts).

Eliminate existential instantiation quantifier by elimination.

In this step, we will eliminate existential quantifier ∃, and this process is known as Skolemization. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

Drop Universal quantifiers.

In this step we will drop all universal quantifier since all the statements are not implicitly quantified

Edit with WPS Office

so we don't need it.

9. ¬ food(x) V likes(John, x)
10. food(Apple)
11. food(vegetables)
12. ¬ eats(y, z) V killed(y) V food(z)
13. eats (Anil, Peanuts)
14. alive(Anil)
15. ¬ eats(Anil, w) V eats(Harry, w)
16. killed(g) V alive(g)
17. ¬ alive(k) V ¬ killed(k)
18. likes(John, Peanuts).

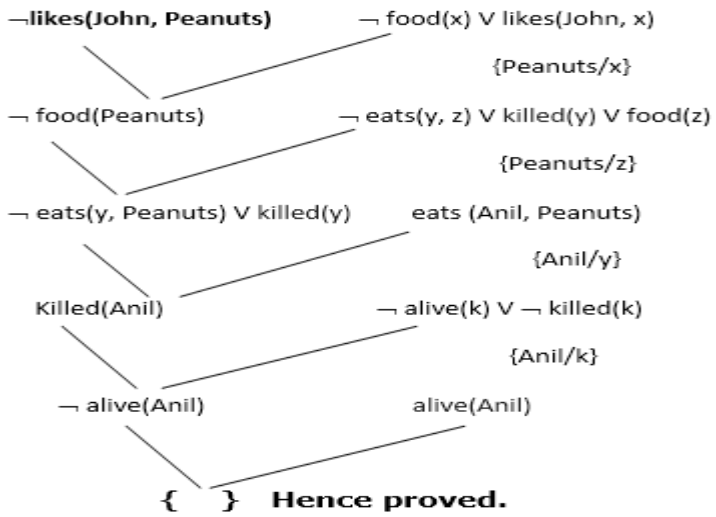
Distribute conjunction ∧ over disjunction ¬.

This step will not make any change in this problem.

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:



5. **Describe the steps involved in Knowledge engineering process in first order logic with an example of electronic circuits domain.**

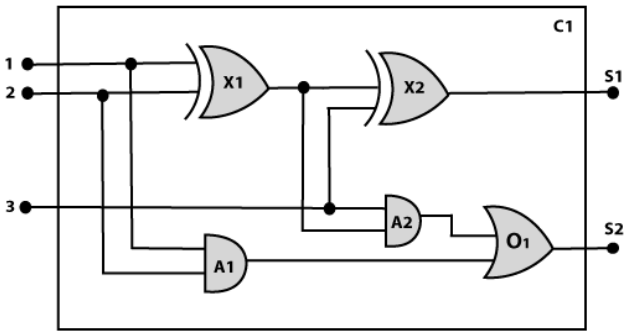
Ans: The process of constructing a knowledge-base in first-order logic is called as knowledge-engineering. In **knowledge-engineering**, someone who investigates a particular domain, learns important concept of that domain, and generates a formal representation of the objects, is known as **knowledge engineer**.

In this topic, we will understand the Knowledge engineering process in an electronic circuit domain, which is already familiar. This approach is mainly suitable for creating **special-purpose knowledge base**.

The knowledge-engineering process:

Following are some main steps of the knowledge-engineering process. Using these steps, we will develop a knowledge base which will allow us to reason about digital circuit (**One-bit full adder**) which is given below

Edit with WPS Office



1. Identify the task:

The first step of the process is to identify the task, and for the digital circuit, there are various reasoning tasks.

At the first level or highest level, we will examine the functionality of the circuit:

- **Does the circuit add properly?**
- **What will be the output of gate A2, if all the inputs are high?**

At the second level, we will examine the circuit structure details such as:

- **Which gate is connected to the first input terminal?**
- **Does the circuit have feedback loops?**

2. Assemble the relevant knowledge:

In the second step, we will assemble the relevant knowledge which is required for digital circuits. So for digital circuits, we have the following required knowledge:

- Logic circuits are made up of wires and gates.
- Signal flows through wires to the input terminal of the gate, and each gate produces the corresponding output which flows further.
- In this logic circuit, there are four types of gates used: **AND, OR, XOR, and NOT**.
- All these gates have one output terminal and two input terminals (except NOT gate, it has one input terminal).

3. Decide on vocabulary:

The next step of the process is to select functions, predicate, and constants to represent the circuits, terminals, signals, and gates. Firstly we will distinguish the gates from each other and from other objects. Each gate is represented as an object which is named by a constant, such as, **Gate(X1)**. The functionality of each gate is determined by its type, which is taken as constants such as **AND, OR, XOR, or NOT**. Circuits will be identified by a predicate: **Circuit (C1)**.

4. Encode general knowledge about the domain:

To encode the general knowledge about the logic circuit

Edit with WPS Office

5. Encode a description of the problem instance:

Now we encode problem of circuit C1, firstly we categorize the circuit and its gate components. This step is easy if ontology about the problem is already thought. This step involves the writing simple atomics sentences of instances of concepts, which is known as ontology.

6. Pose queries to the inference procedure and get answers:

In this step, we will find all the possible set of values of all the terminal for the adder circuit.

7. Debug the knowledge base:

Now we will debug the knowledge base, and this is the last step of the complete process. In this step, we will try to debug the issues of knowledge base.

UNIT -5

1. Discuss the mental objects and Reasoning with Default Information.

Ans:

Mental Events and Mental Objects

For a single- agent domains, knowledge about one's own knowledge and reasoning processes is useful for controlling inference.

• In a multiagent domains, it becomes important for an agent to reason about the mental states of the other agents.

• Example Bob and Alice scenario

• It requires a model of the mental objects in someone's head(knowledge base) and the processes that manipulate these mental objects.

Reasoning in Artificial intelligence

In previous topics, we have learned various ways of knowledge representation in artificial intelligence. Now we will learn the various ways to reason on this knowledge using different logical schemes.

Reasoning: The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or we can say, "**Reasoning is a way to infer facts from existing data.**" It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

Types of Reasoning

In artificial intelligence, reasoning can be divided into the following categories:

Edit with WPS Office

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning
- Common Sense Reasoning
- Monotonic Reasoning
- Non-monotonic Reasoning

1. Deductive reasoning: Deductive reasoning is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.

2. Inductive Reasoning: Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with the series of specific facts or data and reaches to a general statement or conclusion.

3. Abductive reasoning: Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.

4. Common Sense Reasoning: Common sense reasoning is an informal form of reasoning, which can be gained through experiences. Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day. It relies on good judgment rather than exact logic and operates on **heuristic knowledge** and **heuristic rules**.

5. Monotonic Reasoning: In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

6. Non-monotonic Reasoning: In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base. Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

3. **Elucidate Knowledge representation using categories, objects and events.**

Ans:

KR requires the organisation of objects into categories

- interaction at the level of the object
- reasoning at the level of categories
- ex: typically we want to buy a basketball, rather than a particular basketball instance

Categories play a role in predictions about objects

- agent infers the presence of certain objects from perceptual input
- infers category from the perceived properties of the objects,
- uses category information to make predictions about the objects

Categories can be represented in two ways by FOL

- predicates (ex Basketball(x)): relations
- reification of categories into objects (ex Basketballs): sets
- = ⇒ allows categories to be argument of predicates/functions

Edit with WPS Office

Membership of a category as set membership

- ex: Member(b, Basketballs) (abbr. b ∈ Basketballs)

Subcategories (aka subclasses) are (strict) subsets

- ex: Subset(Basketballs, Balls) (abbr Basketballs ⊂ Balls)

Inheritance and Taxonomies

A subcategory inherits the properties of the category

- ex:
  if ∀x.(x ∈ Food → Edible(x)), Fruit ⊂ Food, Apples ⊂ Fruit
  then ∀x.(x ∈ Apple → Edible(x))
- A member inherits the properties of the category
- if a ∈ Apples, then Edible(a)
- Subclass relation organize categories into taxonomies
  (aka taxonomic hierarchies)

ex: taxonomy of >10M living&extinct species

ex: Dewey Decimal System: taxonomy of all fields of knowledge

FOL Reasoning about Categories

- FOL allows to state facts about categories:
- an object is a member of a category
  BB9 ∈ Basketballs
- a category is a subclass of another category
  Basketballs ⊂ Balls
- all members of a category have some properties
  ∀x.(x ∈ Basketballs → Spherical(x))
- members of a category can be recognized by some properties
  ∀x.((Orange(x) ∧ Round(x) ∧ Diameter(x) = 9.5” ∧ x ∈ Balls)
  → x ∈ Basketballs)
- category as a whole has some properties
  Dogs ∈ DomesticatedSpecies
- New categories can be defined by providing necessary and
  sufficient conditions for membership
- ∀x.(x ∈ Bachelors ↔ (Unmarried(x) ∧ x ∈ Adults ∧ x ∈ Males))
  Derived relations
- Two or more categories in a set s are disjoint iff they have no
  members in common
  Disjoint(s) ↔ (∀c1c2. ((c1 ∈ s ∧ c2 ∈ s ∧ c1 6= c2)
  → Intersection(c1, c2) = ∅)
- ex:
  Disjoint({Animals, Vegetables}),
  Disjoint({Insects, Birds, Mammals, Reptiles}),
- A set of categories s is an exhaustive decomposition of a
  category c iff all members of c are covered by categories in s
  ExaustiveDecomposition(s, c) ↔ ∀i.(i ∈c ↔ (∃c2.(c2∈s ∧ i ∈c2)))
  ex: E.D.({Americans, Canadians, Mexicans}, NorthAmericans)
- A disjoint exhaustive decomposition is a partition
  Partition(s, c) ↔ (Disjoint(s) ∧ ExhaustiveDecomposition(s, c))
  ex: Partition({Males, Females}, Animals)

4. **Describe the reasoning systems for categories with the proper example.**

Ans:

Semantic Networks

- allow to visualize knowledge bases
- efficient algorithms for category membership inference
- limited expressivity

Edit with WPS Office

- many variants

Description Logics (DLs)

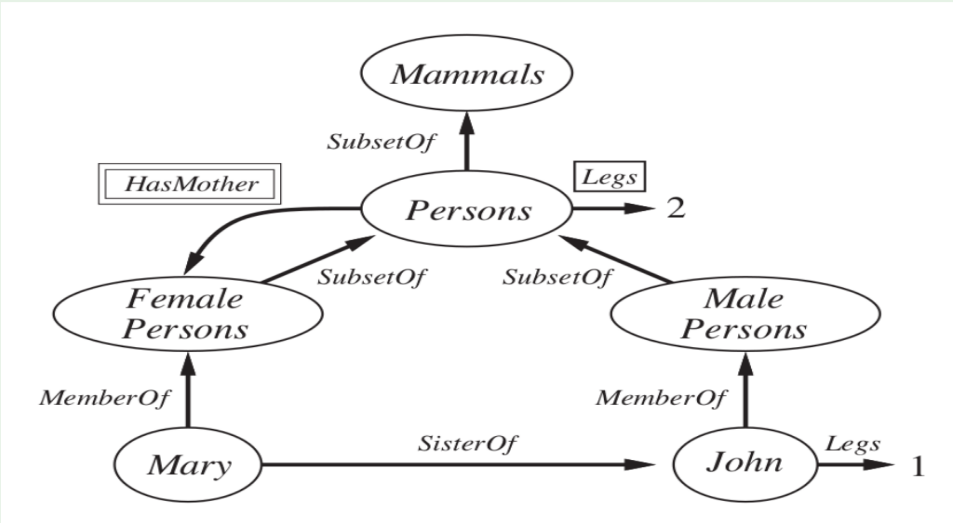
- formal language for constructing and combining category definitions
- (relatively) efficient algorithms to decide subset and superset relationships between categories
- many DLs
  up to very high expressivity
  up to very high complexity (e.g., DOUBLY-EXPTIME)

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals

Two special relations are always present, with different names

- IS-A, aka SubsetOf/SubclassOf (subclass)
- InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification



Semantic Networks: Example

Notice

- "HasMother" is a relation between persons (individuals) (categories do not have mothers)
- "HasMother" (double-boxed notation) means
  $\forall x.(x \in Persons \rightarrow [\forall y.(HasMother(x, y) \rightarrow y \in FemalePersons)])$
- similar for "Legs"

4. Explain the Bayes' Rule and its use. A doctor knows that the disease meningitis causes a patient to have a stiff neck, say, 70% of the time. The doctor also knows some unconditional facts: the prior probability that any patient has meningitis is 1/50,000, and the prior probability that any patient has a

Edit with WPS Office

stiff neck is 1%. Then find the probability of patients with a stiff neck tohave meningitis.

Ans:

letting s be the proposition that the patient has a stiff neck and m be the proposition that the patient has meningitis, we have

$$P(s/m) = 0.7$$

$$P(m) = 1/50000$$

$$P(s) = 0.01$$

$$P(m/s) = \frac{P(m/s)\,P(m)}{P(s)} = \frac{0.7 \times 1/50000}{0.01} = 0.0014$$

That is, we expect only 0.14% of patient with a stiff neck to have meningitis.

Edit with WPS Office

1) Bayes' Rule and its use :-

product rule : It can actually be written in two forms:

$P(a \wedge b) = P(a|b)P(b)$ and $P(a \wedge b) = P(b|a)P(a)$.

Equating the two right-hand sides and dividing by $P(a)$, we get

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

This equation is known as Bayes' Rule (also Bayes' law or Bayes' theorem) This simple equation underlies most modern AI systems for probabilistic inference.

The more general case of Bayes' rule for multivalued variables can be written in the P notation as follows:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

As before, this is to be taken as representing a set of equation, each dealing with specific values of a variables. we will also have occasion to use a more general version conditionalized on some background evidence e:

$$P(Y|X) = \frac{P(X|Y,e)\,P(Y|e)}{P(X|e)}$$

Edit with WPS Office

5. Discuss a simple method for probabilistic inference.

	toothache		*¬toothache*	
	catch	*¬catch*	*catch*	*¬catch*
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

A domain consisting of just the three Boolean variables Toothache, Cavity, and Catch, The full joint distribution of the above variables. Compute the probability that there is no cavity, given a toothache?

Ans:

$P(\text{cavity} \vee \text{toothache}) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$

$P(\text{cavity}) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$

$P(\text{cavity}) = P(\text{cavity}, \text{toothache}, \text{catch}) + P(\text{cavity}, \text{toothache}, \sim\text{catch}) + P(\text{cavity}, \sim\text{toothache}, \text{catch}) + P(\text{cavity}, \sim\text{toothache}, \sim\text{catch})$

$= (0.108, 0.016) + (0.012, 0.064) + (0.072, 0.144) + (0.008, 0.576)$

$= (0.2, 0.8)$

$\therefore$ we can compute the probability of a cavity, given evidence of a toothache:

$$P(\text{cavity} \mid \text{toothache}) = \frac{P(\text{cavity} \wedge \text{toothache})}{P(\text{toothache})}$$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064}$$

$= 0.6.$

Edit with WPS Office

Edit with WPS Office

∴ we can also compute the probability that there is no cavity, given a toothache:

$$P(\sim cavity \mid toothache) = \frac{P(\sim cavity \wedge toothache)}{P(toothache)}$$

$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064}$$

$$= 0.4$$

$$P(cavity \mid toothache) = \alpha P(cavity, toothache)$$

$$= \alpha [P(cavity, toothache, catch) + P(cavity, toothache, \sim catch)]$$

$$= \alpha [(0.108, 0.016) + (0.012, 0.064)]$$

$$= \alpha (0.12, 0.08)$$

$$= (0.6, 0.4)$$

Simple method for probabilistic inference:-

In this section we describe a simple method for probabilistic inference - that is, the computation of posterior probabilities for query propositions given observed evidence. we use the full joint distributi as the "knowledge base" from which answers to all questions may be derived. Along the way we also introduce several useful techniques for manipulating equations involving probabilities.

Edit with WPS Office