PART – A

1a) Definition of 'intelligent agent' in AI.?

An 'intelligent' is an entity that perceives its environment through sensors and acts on it through actuators to achive goals rationally.

b) Definition of heuristic function h(n) ?

A heuristic function h(n) estimates the cost or distance from the current node n to the goal node to guide the search efficiently

c) Mention one way to improve CSP efficiency?

One way to improve CSP efficiency is using constraint propagation (like forward checking)

d) What is the role of Wumpus World in AI?

Wumpus world is used as a testing environment to demonstrate knowledge-based agent reasoning, logic and decision making under uncertainty.

e) How is FOL different from Propositional Logic?

First Order Logic (FOL) uses predicates, variables and quantifiers, while Propositional Logic uses only simple true/false statements without structure.

f) What type of reasoning is used in Backward Chaining?

Backward chaining uses goal-driven (deductive) reasoning, starting from the goal and working backward to known facts.

g) What are events in knowledge Representation?

Events represent actions or occurrences that cause changes in the state of the world.

h) Definition of Mental Events in AI?

Mental events are internal states of an agent such as beliefs, intentions, desires, and decisions.

i) Definition of Dempster-Shafer Theory.

Dempster-Shafer Theory is a mathematical approach for reasoning under uncertainty using belief functions instead of exact probabilities.

j) Definition of Uncertainty in AI?

Uncertainty in AI refer to the lack of complete, accurate, or definite knowledge about the environment or outcomes of actions.

PART B

2a) At least 5 Differences between Informed & Uninformed Search with Examples.

Ans Uninformed Search (Blind Search) : Searches without any knowledge about the goal location. It explores Systematically.

Informed Search :- (Heuristic Search): Uses a heuristic function (h(n)) to estimate how close a node is to the goal and guide the search.

Key Differences

Feature	Uninformed search	Informed Search.
⇒ Knowledge used	* No extra information	* Uses heuristic information.
⇒ Goal direction	* No idea where goal is	* Knows approximate direction of goal
⇒ Speed	* Slow	* Faster
⇒ Memory	* High	* optimized
⇒ Efficiency	* Less ~~High~~ efficient	* More efficient
⇒ optimality	* Sometimes optimal	* often optimal

Examples :-

* Uninformed Search:
  - ⇒ Breadth First Search (BFS)
  - ⇒ Depth First Search (DFS)
  - ⇒ Uniform cost Search.

* Informed Search:
  * A* Algorithm
  * Best First search
  * Hill climbing

uniformed search works without goal knowledge, while informed search uses heuristics to reach the goal faster and more efficiently.

2b) Definition of A* Search? Explain with Example.

Ans: A (A-Star) Search * is an informed search algorithm that finds the optimal path by using:

$$f(n) = g(n) + h(n)$$

Where:

* $g(n)$ = actual cost from start to node n
* $h(n)$ = heuristic estimated cost from n to goal
* $f(n)$ = total estimated cost

Working Principle

1) Start from the intial node.
2) Calculate $f(n)$ for all possible next nodes
3) Select the node with lowest $f(n)$
4) Repeat until the goal is reached

Example (Simple Path Finding)

Suppose we want to go from city A → city D

Node	g(n)	h(n)	f(n)
B	3	5	8
C	4	2	6

A* Select C first because it has the lowest f(n) This continues until the goal is found with the shortest path.

Advantages :-

⇒ Finds optimal solution
⇒ Faster than BFS and DFS
⇒ Used in GPS navigation, robotics, and games

A* Search is the most popular optimal and efficient heuristic search algorithm.

---

3) Definition of Hill-climbing search with pseudo algorithm? Its Types with Examples?

Ans: Hill-climbing Search is a local search algorithm that starts with an intial solution and furs moves step-by-step towards a better neighboring - solution using a heuristic value. until no further improvement is possible.
It always tries to reach the highest peak (best solution) just like climbing a hill.

Working Principle :-

1) Start with an initial State
2) Evaluate its neighboring states using heuristic.
3) Move to the neighbor with the best value.
4) Repeat the process.
5) Stop when no further improvement is possible.

Simple Algorithm Steps :-

1) Select intial State
2) If current state is goal, Stop.
3) Generate neighbors.
4. Choose the best neighbor.
5. Move to that neighbor.
6. Repeat until no better neighbor exists.

Example of Hill-climbing :-

Example :- Finding the Highest Point
Suppose the height of neighboring points are:

* Start a point with height = 10
* Neighbors: 12, 15, 11
* Best move → 15.
* Next neighbors : 14, 16
* Best move → 16 (Goal reached)

The algorithm always moves upward toward better values.

Types of Hill-Climbing Search :-

1) Simple Hill-climbing :-
   * checks neighbours one by one
   * moves to first better neighbour found
   * Does not check all possible options

Example :-
Robot moves to the first higher step it finds without checking all directions.

Advantages : Fast

Disadvantage : May miss best solution

2. Steepest - Ascent Hill - climbing

* checks all neighbours
* Selects the best among all
* Gives better result than Simple hill-climbing

Example :-

From neighbours 10, 13, 18, 14 → select 18

Advantage : Better solution

Disadvantage : Slower than Simple

3. Stochastic Hill - climbing :

* chooses randomly among better neighbours
* Probability - based selection

Example :-

Instead of picking highest, it randomly picks among better options.

Advantages :- Avoids some local traps

Disadvantages :- Unstable output

Advantages of Hill-climbing :

* Simple to implement
* Requires very less memory
* Faster than blind search
* useful in optimization problems.

Disadvantages of Hill-climbing :

* Can get stuck at local maxima
* No backtracking

* Not guaranteed to find the optimal solution
* Incomplete algorithm

Application

* Game AI
* Robotics path finding
* optimization problems
* scheduling
* Neural network training

Hill-climbing is a fast and simple local search Algorithm that moves continuously toward better solutions but may fail to reach the global optimum due to local maxima and plateaus.

4) Demonstrate the structure of Problem in CSP? Examples of CSP for defining structure?

Ans: A Constraint satisfaction Problem (CSP) is a problem in which:

* We must assign value to variables
* From given domains
* While satisfying all constraints

In AI, many real-world problems like map coloring, Sudoku, scheduling, and time-tabling are modeled as CSPs.

Basic structure of a CSP:

A CSP is formally defined by three main components CSP = (V, D, C)

1. Variables (V)

These are the unknowns of the problem.

Example:

* In map coloring: V = {A, B, C, D}
* In Suduku : V = each cell in the grid

2. Domains (D):

Each variable has a set of possible values.

Example:

* For map coloring:

  D = {Red, Green, Blue}

* For Sudoku :

  D = {1, 2, 3, 4, 5, 6, 7, 8, 9}

3. Constraints (C)

Constraints define the rules or restrications that variable must follow.

Types of Constraints:

* Unary constraints : on one variable

  Example : A ≠ Red

* Binary Constraints : between two variables

  Example A ≠ B

* Global Constraints : on many variables

  Examples All values must be different (Sudoku)

Representation of CSP Structure :-

CSP problems are generally represented using:

1. Constraint Graph
   * Nodes → variables
   * Edges → constraints between variables

Example (map coloring):

* Nodes : A, B, C
* Edge between A and B means $A \neq B$

This graph clearly shows which variables directly affect each other.

2. Constraint Matrix / Table

Constraints can also be represented using tables that show allowed or disallowed value combinations.

Example : Map coloring CSP

Problem :-

Color the regions A, B, C such that no two adjacent regions have the same color.

Variables :

$V = \{A, B, C\}$

Domains :

$D = \{Red, Green, Blue\}$

Constraints :

* $A \neq B$
* $B \neq C$
* $C \neq D$

This fully defines the structure of the CSP.

Solution to a CSP

A solution is a complete assignment of values to all variables such that:

* Every variable gets a value from its domain
* All constraints are satisfied

Example solution:

* A = Red
* B = Green
* C = Blue

CSP Search Process

To solve CSPs, AI uses:

* Backtracking search
* Forward checking
* Arc consistency
* Heuristics (MRV, Degree heuristic)

These improve efficiency.

General structure flow of CSP:

1) Define variables
2) Define Domains
3) Define constraints
4) Apply search + constraint propagation
5) Find valid solution.

Real-World Applications of CSP

* Sudoku Solving
* Time-table scheduling
* Job Scheduling
* Map coloring
* Resource allocation
* circuit design

The structure of CSP consists of variables, domain and constraints, and solutions are found by assigning values to all variables while satisfying all constrains.

5) What is Propositional Logic ? How Knowledge is Represented using Propositional Logic ?

Ans: Propositional Logic (PL) is the simplest form of logic representation in Artificial Intelligence

Where :

* knowledge is represented using statments called propositions
* Each proposition is either True (T) or False (F)

It does not describe objects or relations, it only works with simple facts

A proposition is a declarative sentence that has a truth value.

Examples:

* "It is raining" → True or false
* "2+2 = 4" → True
* "India is a country" → True

These are represented as:

* P, Q, R, etc.

Basic components of Propositional Logic

1. Logical connectives

Symbol	Name	Meaning
¬P	NOT	Negation
P∧Q	AND	Both are

Symbol	Name	Meaning
P∨Q	OR	Any one true
P→Q	implication	if P then Q
P↔Q	Biconditional	if and only if

Truth values

Each proposition can have:

* True (T)
* False (F)

Syntax of propositional Logic

The Syntax define how valid statement are formed

Examples of well-formed formulas:

* P
* $\neg P$
* $P \wedge Q$
* $(P \vee Q) \rightarrow R$

Semantic of Propositional Logic

Semantics tells the meaning of formulas using truth tables.

Example:

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

Knowledge is represented using the following steps:

Step 1: Identify Facts as Propositions

Example:

* P: "It is raining"
* Q: "The ground is wet"

Step 2: Represent Rules using connectives

Rule:

* "If it is raining, then the ground is wet"
* Represented as:

  $P \rightarrow Q$

Step 3: Create a knowledge Base (KB)

Example KB:

1. P (It is raining)
2. $P \rightarrow Q$ (If it is raining, the ground is wet)

Step 4: Infer New knowledge

Using inference:

* From P and $P \rightarrow Q$
* We conclude: Q (The ground is wet)

Real-Time Example of knowledge Representation

- Banking Example:
  * P: "User enters correct PIN"
  * Q: "User is allowed to withdraw money"

Rule:

* $P \rightarrow Q$

if P is true, then Q becomes true.

⇒ Weather Example:

* P: "It is cloudy"
* Q: "It may rain"

Rule:

* $P \rightarrow Q$

Inference in Propositional Logic

Some common inference rules:

* modus Ponens:
  If $P \rightarrow Q$ and P is true, then Q is true

* Modus Tollens:
  If $P \rightarrow Q$ and Q is false, then P is false.

⇒ Propositional Logic is a basic and Powerful method of representing knowledge using true or false statements, and knowledge is represented using propositions, logical connectives, and inference rules.

---

6) Reasoning Systems for categories and objects in knowledge Representation.

Ans: In Artificial Intelligence, resoning about categories and objects helps the system:

* classify objects
* understand relationships
* Inherit properties
* Make logical decisions

These reasoning systems answer questions like:

* What type of objects is this?
* What are its properties?
* what actions can it perform?

Categories in Knowledge Representation

A category is a group or class of similar objects

Example:

* Category : Bird
* objects : Sparrow, Crow, Parrot

All birds share common properties like:

* Has wings
* can fly (generally)
* Lays eggs

This type of organization helps in fast reasoning and reduced memory usage.

objects in Knowledge Representation

An object is a specific instance of a category.

Example;

* category : Student
* object : Ravi

Ravi may have properties;

* Roll number = 23
* Branch = CSE
* Age = 20

Ravi may have properties;

* Roll number = 23
* Branch = CSE
* Age = 20

We Need Reasoning About categories and objects

* To organize knowledge systematically
* To avoid repeating information
* To support property inheritance
* To perform logical classification
* To enable intelligent decision making

Main Reasoning System for categories and objects.

1. Semantic Networks

A Semantic Network represent knowledge in the form of;

* Nodes → objects or categories
* Links → Relationships

Example:

* Bird → has Wings
* Sparrow → is-a → Bird

Here, Sparrow inherits properties of Bird.

Used For;

* Property inheritance
* Relationship-based reasoning

2. Frames

A Frame is a structured data format for representing an objects or category using slots and values.

Example (Frame for student):

Slot	value
Name	Ravi
Age	20
Branch	CSE
College	CMR

Frames also support:

* Default values
* Inheritance
* Procedural attachments

Taxonomies (Hierarchical Structure)

A taxonomy is a tree-like structure where:

* General categories are at the top
* Specific objects are at the bottom

Example:

* Living Being
  - Animal
    - Mammal
      - Dog
        - Tommy

This structure supports:

* classification
* subsumption reasoning (finding super or sub-class)

Description Logic (DL)

Description Logic is a formal logic system used to describe :

* categories
* objects
* Relationships

It is mainly used in :

* ontologies
* Semantic Web
* Expert Systems

It performs :

* Subsumption checking (Is cat an Animal?)
* Instance Checking (Is Tommy a Dog?)

Types of Reasoning in category and object Systems

1. Inheritance Reasoning

Objects inherit properties from their parent category.

Example :

* Bird → can fly
* Sparrow → is a Bird
* Therefore : Sparrow can fly

2. classification Reasoning

Assigning an object to a category.

Example:

* If an object has feathers and lays eggs → classify as Bird

3. Consistency Checking

Ensures there are no logical conflicts.

Example:

* If Bird can fly
* Penguin is Bird
* But penguin cannot fly
* This exception must be handled carefully.

Real-Life Example

⇒ Banking Example:

category: Account

* Saving Account
* current Account

Objects: Ravi's Savings Account

Properties like;

* minimum balance
* Interest rate
* are inherited from Account category

Reasoning Systems for categories and objects organize knowledge using classes and instances and support inheritance, classification, and logical decision making using structure like Semantic network, frames taxonomies, and description logic.

7a) Unification and Lifting in Inference in First order Logic

Ans: Unification is the process of making two logical expression identical by finding suitable substitution for variables.

It is mainly used in resolution and inference in Fqost order Logic.

Example :

* P(X,a)
* P(b,y)

After unification:

* X=b
* y=a

so both become P(b,a)

Purpose of Unification :

* Helps in matching predicates
* Makes logical inference possible
* Used in resolution - based proofs

Lifting (Lifted Inference)

Lifting is the process of applying inference rules directly at the variable level, instead of on ground (constant) facts,

It allows reasoning with general rules, not just specific values,

Example :

* Rule : $\forall x$ (Student (x) $\rightarrow$ Intelligent (x))
* Fact : Student (Ravi)

By lifting :

* we infer : Intelligent (Ravi)

Key Difference :-

* Unification finds substitutions to match expression
* Lifting applies inference at the variable level instead of constants.

Unification and lifting together make automated reasoning efficient and powerful in First order Logic.

---

7)b) Backward Chaining in First order Logic ?

Ans: Backward chaining is a goal-driven inference method in which the system:

* Starts with the goal
* Work backwards to find supporting facts or rules

It is mainly used in expert systems and Prolog.

Working Principle :-

1) Start with the query (goal)
2) Find rules that can produce this goal
3) Make their conditions as new sub-goals
4) Continue until :
    * known facts are reached (success), or
    * No rule applies (failure)

Example

Goal: Will Ravi get a job?

Rules:

* If a person is skilled AND certified → get a job
* Ravi is skilled
* Ravi is certified

using backward chaining:

* Start with: gets a job
* checks: skilled AND certified
* Both facts are true

Therefore, Ravi gets a job

features of Backward chaining

* Goal- oriented
* search only relevent rules
* Works well when the number of goals is small

Application

* Expert Systems
* Medical diagnosis
* Fault detection
* Logic programming (Prolog)

Backward chaining is a goal-directed reasoning method that starts from the goal and moves backward to known facts using rules.

8) Reasoning with Default Information for knowledge Representation. examples.

Ans: In many real-world situations, complete information is not always available. So, AI systems use default reasoning where;

* The system assumes something to true by default
* until contridicted by new information

This type of reasoning is called Reasoning with Default Information.

Default reasoning is a type of non-monotonic reasoning wheree;

* conclusions can be withdrawn when the new facts arrive
* The system uses common-sense assumptions

Example;

* Birds can fly" is taken as a default rule
* If we later learn "Penguin is a bird but cannot fly" that default is canceled

Default Reasoning is Needed

* Real-World knowledge is incomplete
* Humans naturally use assumptions
* Reduce the need to specify every exception
* Helps in faster decision making

Characteristics of Default Reasoning

* Based on assumption
* conclusions are retractable
* supports exceptions
* works with incomplete and uncertain data
* used in common-sense reasoning

Default Rules

A default Rules has the form:

If condition is true AND there is no evidence against it, then assume the conclusion is true.

Example:

=> If x is a Bird → assume x can fly

=> If x is a Student → assume x attends classes

Non-Monotonic Nature of default Reasoning

In monotonic reasoning, once something is true, it always remain true.

In default reasoning (non-monotonic):

* New knowledge can cancel previous conclusions

Example:

1) Bird (Tweety) → assume Tweety can fly
2) New fact: Penguin (Tweety)
3) New conclusion: Tweety cannot fly

So, the earlier conclusion is removed

Methods to Represent Default Knowledge

1) Default Logic (Reiter's Default Logic)

A default rule is written as:

$$\frac{\text{Prerequisite} : \text{justification}}{\text{conclusion}}$$

Example:

$$\frac{Bird(x) : canfly(x)}{canFly(x)}$$

Meaning:
if x is a bird and it is consistent to assume it can fly, then conclude it can fly.

2. circumscription:

circumscription assumes that:

* Only the minimum number of facts are true
* Everything else is assumed false by default

Used to represent:

* "only these are exceptions"

3. Inheritance with exceptions

used in:

* Semantic networks
* Frame Systems

Example:

* Bird → can fly
* Penguin → cannot fly

* Penguin overrides Bird's default rule

Example of Default Reasoning (Detailed)

Example: Bird and Penguin

Rules:

* All birds can fly (default)
* Penguin is a bird
* Penguins cannot fly (exception)

Facts:-

* Tweety is a bird
* Tweety is a Penguin

Default reasoning:

* From Bird → assume tweety can fly
* From Penguin → override → Tweety cannot fly

Advantages of Default Reasoning

* Handles incomplete information
* Models human common-sense reasoning
* Reduces rule complexity
* Easy to represent exceptions
* useful in real-world applications

Application of Default Reasoning:

* Expert Systems
* Medical diagnosis
* Legal Reasoning
* Natural language understanding

* Intelligent agents
* Knowledge-based Systems

Reasoning with default information allows AI system to draw conclusion based on assumption in the absence of complete knowledge, making it an important technique for common-sense and real-world reasoning.

---

9) Explain Planning Graphs, and Its Applications.?

Ans: Planning in AI is the process of finding a sequence of actions that transforms the initial state into a goal state. A planning Graph is a powerful data structure used to efficiently solve planning problems It was introduced in the graph plan algorithm.

A Planning Graph is a layered graph structure that represents:

* All possible actions
* All possible states (facts)
* And their relationships over time.

It helped in:

* Fast plan generation
* Heuristic estimation
* checking goal reachability

Structure of a Planning Graph

A planning Graph consists of alternating levels (layers):

1) State Levels (S)

* Represent facts (states) that could be true
* Written as: S0, S1, S2 ....

2. Action Levels (A):

* Represent actions that can be applied
* Written as: A0, A1, A2, ...

So the structure is:

$$S_0 \rightarrow A_0 \rightarrow S_1 \rightarrow A_1 \rightarrow S_2 \rightarrow A_2 \rightarrow \dots$$

Components of a Planning Graph:

1. Initial State (S0)
   * Contains all facts that are true initially

2. Actions (A)
   Each action has:
   * Preconditions
   * Effects (Add list and Delete list)

3. Goals
   * The target facts that we want to achieve

Mutex (Mutual Exclusion) in Planning Graphs:
Mutex relations show conflicts between actions or states.

1) Action Mutex:
Two actions are mutex if:
* one action deletes the effect of another
* one action deletes the precondition of another
* They have competing needs.

2. State Mutex:

Two states are mutex if:

* They are achived by mutex actions

Mutex helps in:

* Proning impossible combinations
* Speeding up planning

Planning Graph is constructed:

1. Start with intial State S0
2. Generate all applicable actions A0
3. Generate resulting State S1
4. Add mutex relations
5. Repeat until:
    * Goal appears, or
    * No new states can be added (graph levels off)

Working of Planning Graph (simple explanation)

* The graph is expanded level by level
* At each level:
    * More actions and states are added
* The graph never shrinks
* When all goals appear in the same State level without mutex, a solutions may exist.

Advantages of Planning Graphs

* Faster then Traditional planning
* Avoid Repeated Searching
* Reduce complexity using mutex
* works well for large planning problems
* Provides good heuristic estimates

Applications of Planning Graphs:

1. Robotics
   * Planning Robot movements
   * Pick and place operations
   * path planning
2. Automated Planning Systems
   * Generating actions sequence in Intelligent agents
3. Game AI
   * Strategy planning
   * Multi-step action decisions
4. Scheduling Problems:
   * Task Scheduling
   * Resource planning
   * Project Management
5. Logistics and Supply chain
   * Transport planning
   * warehouse optimization

* Delivery route planning

6. Web Services Composition

* Selecting and sequencing services automatically

Limitations of planning Graphs

* High memory usage for very large problems
* Not suitable for continuous environments
* Works mainly with deterministic actions
* Needs well-defined actions and effects

A planning graph is a layered structure that efficiently actions and states for solving AI planning problems, and it is widely used in robotics, scheduling, games, and automated decision systems.

10g) Explain the Baye's Rule. Its Uses.

Ans: Baye's Rule is a mathematical formula used to update the probability of an event based on new evidence.

It is written as;

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

* $P(A|B)$ = Probability of A given B
* $P(B|A)$ = Probability of B given A
* $P(A)$ = Prior probability of A
* $P(B)$ = Probability of evidence

Bayes' Rule tell us how to revise our belief after getting new information.

Simple Example

Let :

* A = Person has a disease
* B = Medical test is positive

Baye's Rule helps to find :

* What is the actual probability that the person has the disease after the test is positive.

Uses of Baye's Rule :-

* Medical Diagnosis
* Spam email filtering
* Weather prediction
* Machine learning (Naive Bayes classifier)
* Fault detection
* Expert systems

Baye's Rule is a powerful method used to update probabilities using new evidence, and is widely applied in AI decision-making systems.

10b) How does uncertainty Arise in Artificial Intelligence ?

Ans: Uncertainty in AI refers to a situation where:

* The System does not have complete or exact information
* outcomes cannot be predicted with full confidence.

Major Reasons for uncertainty in AI

1. Incomplete Information

The System does not have full data.

Example:
A doctor does not known all Symptoms of a patient.

2. Noisy or Inaccurate Data

Sensors or inputs may give wrong values.

Example:
A camera giving blurred images.

3. Unpredictable Environment

Some environments changed dynamically.

Example:
Traffic conditions on roads.

4. Ambiguity in Data

Same data may have multiple meanings

Examples:
Word "bank" can mean river bank or money bank.

5. Complex Real-World Problems

To many variables make exact reasoning difficult

Example:
Stock market prediction.

Effects of uncertainty :-

⇒ Wrong decision

⇒ Incomplete conclusions

⇒ Need for probabilistic reasoning

Uncertainty in AI arises due to incomplete, inaccurate, ambiguous, or unpredictable information in real-world environments.

---

11) Inference using Full Joint Distributions in Uncertain Knowledge with formula / Example.

Learning uncertainty with methods, examples.

Ans:- In real-life situations, AI systems do not always have complete or exact information. Because of this probability theory is used to handle uncertain knowledge and make decisions.

One important method for handling uncertainty is Inference using Full Joint Distributions.

A Full Joint Distribution (FJD) gives the probabilities of all possible combinations of random variables in a system.

If we have variables:

* A, B, C

Then the full joint distribution gives:

* P(A, B, C)

It tells us:
The probability that A, B and C occur together.

Purpose of using Full Joint Distributions

* To answer any probability query
* To find conditional probabilities
* To support accurate inference under uncertainty
* To model real-world uncertain situation

Inference using Full Joint Distributions:

Inference means finding the probability of a query variable based on given evidence.

We use the formula:

$$P(\text{Query} \mid \text{Evidence}) = \frac{P(\text{Query} \cap \text{Evidence})}{P(\text{Evidence})}$$

This is computed using values from the full joint distribution table.

Example of inference using Full Joint Distribution

Let variable be;

* R = It is raining
* W = The ground is Wet

Full Joint Distribution:

R	W	P(R, W)
T	T	0.4
T	F	0.1
F	T	0.2
F	F	0.3

Now find :

$$P(R|W)$$

Step 1 :

$$P(R \cap W) = 0.4$$

Step 2 :

$$P(W) = 0.4 + 0.2 = 0.6$$

Final Answer :

$$P(R|W) = \frac{0.4}{0.6} = 0.67$$

So, the probability that it is raining when the ground is wet is 0.67

Limitations of Full Joint Distributions :

* Requires huge memory
* Number of probabilities grows exponentially
* Not practical for large AI systems
* Computationally very expensive

This is why systems use :

* Bayesian Networks
* Markov Models

Instead of full joint distributions.

Learning uncertainty means :

* The AI System learns probabilities from data
* Instead of being manually programmed

It allows the system to :
* improve with experience
* Adapt to new environments
* Predict uncertain outcomes more accurately

Methods of Learning Uncertainty

1) Supervised Learning
Learns probabilities from labelled data
Example :
* Spam or Not spam emails

2. Bayesian Learning
Updates probabilities using Bayes' Rule
Used in :
* Medical diagnosis
* Fault detection

3. Maximum Likelihood Estimation (MLE)
Finds probabilities that best fit the observed data

4. Naive Bayes classifier :
A probabilistic learning model using :
* Baye's theorem
* Feature independence assumption

Used for :
* Text classification
* Email Spam filtering.

Applications of Inference and Learning under Uncertainty:

* Medical diagnosis
* Speech recognition
* Weather forecasting
* Stock market prediction
* Robotics
* Fraud detection
* Recommendation Systems.

Inference using full joint distributions provides a theoretical for reasoning under uncertainty, while learning uncertainty allows AI systems to automatically learn and improve probability models from real-world data.

B.TECH III-I SEM END EXAM (Regular) DEC-2025

SCHEME OF EVALUATION

ARTIFICIAL INTELLEGENCE (AI)-R22

PART A

SL NO	THEORY	MARKS	TOTAL
a	Definition of intelligent agent in AI.	1	1
b	Definition of heuristic function h(n).	1	1
c	Mention one way to improve CSP efficiency.	1	1
d	What is the role of Wumpus World in AI?	1	1
e	How is FOL different from Propositional Logic?	1	1
f	What type of reasoning is used in Backward Chaining?	1	1
g	What are events in Knowledge Representation?	1	1
h	Definition of Mental Events in AI.	1	1
i	Definition of Dempster-Shafer Theory.	1	1
j	Definition of Uncertainty in AI?	1	1

PART B

SL NO	THEORY	MARKS	TOTAL
2.	a) At least 5 Differences between Informed & Uninformed Search with Examples.	05	10
	b) Definition of A* Search?	02	
	Explain with Example.	03	
3.	Definition of Hill-Climbing Search with Pseudo algorithm.	05	10
	Its Types with Examples.	05	
4.	Demonstrate the Structure of Problems in CSP.	07	10
	Examples of CSP for defining Structure.	03	
5.	What is Propositional Logic?	03	10
	How Knowledge is Represented Using Propositional Logic	07	
6.	Reasoning Systems for Categories	05	10
	and Objects in Knowledge Representation	05	
7.	a) Unification and Lifting in Inference in First Order Logic	05	10
	b) Backward Chaining in First Order Logic.	05	
8.	Reasoning with Default Information for Knowledge Representation.	07	10
	Examples	03	

9.	Explain Planning Graphs. and Its Applications.	05 05	10
10.	a) Explain the Bayes' Rule. Its Uses. b) How Does Uncertainty Arise in Artificial Intelligence?	03 02 05	10
11.	Inference using Full Joint Distributions in Uncertain Knowledge with Formula/ Example Learning Uncertainty with methods, Examples.	07 03	10