Unit III: Knowledge Representation and Reasoning (10 Hours)

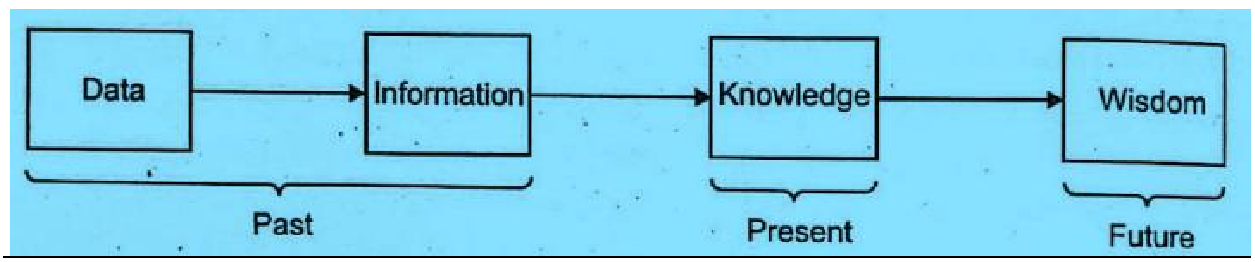
3.1 Definition and importance of Knowledge, Issues in Knowledge Representation
3.2. Knowledge Representation Systems: Semantic Nets, Frames, Conceptual Dependencies, Scripts, Rule Based Systems(Production System), Propositional Logic, Predicate Logic
3.3.Propositional Logic(PL): Syntax, Semantics, Formal logic- connectives, truth tables, tautology, validity, well-formed- formula, Inference using Resolution,
3.4 Backward Chaining and Forward Chaining
3.5 Predicate Logic: FOPL, Syntax, Semantics, Quantification, Inference with FOPL, Inference using resolution
3.6 Bayes' Rule and its use, Bayesian Networks
3.7 Fuzzy Logic

3.1 Definition and importance of Knowledge, Issues in Knowledge Representation

Knowledge in Artificial Intelligence refers to the facts, information, rules, concepts, and relationships about the world that an intelligent system uses to understand, reason, and make decisions.

Knowledge is a progression that starts with data which is of limited utility.

Data when processed becomes information, information when interpreted or evaluated becomes knowledge and understanding, of the principles embodied with the knowledge is wisdom.



Data → Information → Knowledge → Wisdom (DIKW Model)

Imagine a health monitoring system used in a smart hospital.

Step 1 — Data (Past)
Raw, unprocessed facts with no meaning by themselves.

Example (Data):
- Temperature readings: 101.4°F, 99.8°F, 102.1°F
- Heart rate values: 112 bpm, 108 bpm, 118 bpm
- Oxygen level readings: 94%, 92%, 95%

These are just numbers collected by sensors.

They do not tell us what is happening.

Step 2 — Information (Past)
Data processed to give context and meaning.

Example (Information):
• “Patient’s temperature is consistently above 101°F.”
• “Heart rate is higher than normal.”
• “Oxygen level is slightly low.”

Now the system interprets data → useful patterns.

Still no decision, but the system understands what the data represents.

Step 3 — Knowledge (Present)
Using rules, models, and experience to interpret information.

Example (Knowledge):
The AI system applies medical rules:
• IF fever > 101°F AND heart rate > 110 THEN → fever with tachycardia
• IF oxygen < 94% THEN → possible respiratory problem
• IF fever + cough + low oxygen THEN → suspected pneumonia

Here AI combines information + stored rules.

AI “knows” how symptoms relate to illnesses.

Step 4 — Wisdom (Future)
Ability to make sound decisions or recommendations.

Example (Wisdom):
Based on knowledge, AI suggests:
• “Patient may require a chest X-ray.”
• “Start oxygen support immediately.”
• “Alert the doctor with high priority.”
• “Predict 24-hour risk of deterioration: HIGH.”

Wisdom = Choosing the best action based on knowledge.
This is closest to intelligent decision-making.

Types of Knowledge in AI

AI systems rely on different types of knowledge to function efficiently. Each type serves a specific role in reasoning, decision-making, and problem-solving. Below are the primary types of knowledge used in AI:

1. Declarative Knowledge (Descriptive Knowledge)

Declarative knowledge consists of facts and information about the world that AI systems store and retrieve when needed. It represents "what" is known rather than "how" to do something. This type of knowledge is often stored in structured formats like databases, ontologies, and knowledge graphs.

For example, a fact such as "Paris is the capital of France" is declarative knowledge. AI applications like search engines and virtual assistants use this type of knowledge to answer factual queries and provide relevant information.

2. Procedural Knowledge (How-To Knowledge)

Procedural knowledge defines the steps or methods required to perform specific tasks. It represents "how" to accomplish something rather than just stating a fact.

For instance, knowing how to solve a quadratic equation or how to drive a car falls under procedural knowledge. AI systems, such as expert systems and robotics, utilize procedural knowledge to execute tasks that require sequences of actions. This type of knowledge is often encoded in rule-based systems, decision trees, and machine learning models.

3. Meta-Knowledge (Knowledge About Knowledge)

Refers to knowledge about how information is structured, used, and validated. It helps AI determine the reliability, relevance, and applicability of knowledge in different scenarios.

For example, an AI system deciding whether a piece of medical advice comes from a trusted scientific source or a random blog post is using meta-knowledge. This type of knowledge is crucial in AI models for filtering misinformation, optimizing learning strategies, and improving decision-making.

4. Heuristic Knowledge (Experience-Based Knowledge)

Heuristic knowledge is derived from experience, intuition, and trial-and-error methods. It allows AI systems to make educated guesses or approximate solutions when exact answers are difficult to compute.

For example, a navigation system suggesting an alternate route based on past traffic patterns is applying heuristic knowledge. AI search algorithms, such as A* search and genetic algorithms, leverage heuristics to optimize problem-solving processes, making decisions more efficient in real-world scenarios.

5. Common-Sense Knowledge

Common-sense knowledge represents basic understanding about the world that humans acquire naturally but is challenging for AI to learn. It includes facts like "water is wet" or "if you drop something, it will fall."

AI systems often struggle with this type of knowledge because it requires contextual understanding beyond explicit programming.

Researchers are integrating common-sense reasoning into AI using large-scale knowledge bases such as ConceptNet, which helps machines understand everyday logic and improve their interaction with humans.

6. Domain-Specific Knowledge

Domain-specific knowledge focuses on specialized fields such as medicine, finance, law, or engineering. It includes highly detailed and structured information relevant to a particular industry.

For instance, in the medical field, AI-driven diagnostic systems rely on knowledge about symptoms, diseases, and treatments. Similarly, financial AI models use economic indicators, risk assessments, and market trends. Expert systems and AI models tailored for specific industries require domain-specific knowledge to provide accurate insights and predictions.

Importance of Knowledge in Artificial Intelligence

Knowledge is the core component of any intelligent system. Without knowledge, AI systems cannot reason or solve problems.

Importance:

Problem Solving

Knowledge helps AI systems find solutions to complex problems.
Example: medical diagnosis systems use medical knowledge.

Reasoning and Decision Making

AI uses knowledge to draw conclusions and make logical decisions.

Learning

Knowledge provides a base for learning new information.
Learning modifies or adds to existing knowledge.

Understanding the Environment

Knowledge allows AI to interpret real-world situations.

Expert Systems

AI systems rely on expert knowledge to mimic human experts.

Knowledge Representation

Knowledge Representation in AI is the method used to store, organize, and structure information about the world so that a computer system can reason, solve problems, and make decisions effectively.
It allows AI systems to simulate human understanding by representing facts, concepts, and relationships in a form that machines can process.

In simple terms:
"KR is the way an intelligent system understands, organizes, and uses knowledge to solve problems."

key Goals of KR

1. Represent the real world – Objects, people, events, locations, properties, relationships
2. Enable intelligent reasoning – Derive new facts from known facts
3. Support problem solving – Diagnostics, planning, decision making
4. Efficient storage & retrieval – Organized knowledge for fast reasoning
5. Provide a basis for communication – Between AI systems and humans

Issues in Knowledge Representation

While knowledge representation is fundamental to AI, it comes with several challenges:

1. Complexity:
Representing all possible knowledge about a domain can be highly complex, requiring sophisticated methods to manage and process this information efficiently.

2. Ambiguity and Vagueness:
Human language and concepts are often ambiguous or vague, making it difficult to create precise representations.

3. Scalability:
As the amount of knowledge grows, AI systems must scale accordingly, which can be challenging both in terms of storage and processing power.

4. Knowledge Acquisition:
Gathering and encoding knowledge into a machine-readable format is a significant hurdle, particularly in dynamic or specialized domains.

5. Reasoning and Inference:
AI systems must not only store knowledge but also use it to infer new information, make decisions, and solve problems. This requires sophisticated reasoning algorithms that can operate efficiently over large knowledge bases.

3.2. Knowledge Representation Systems: Semantic Nets, Frames, Conceptual Dependencies, Scripts, Rule Based Systems(Production System), Propositional Logic, Predicate Logic

Knowledge Representation (KR) systems provide different ways to store, organize, and reason with knowledge in Artificial Intelligence.
Different KR models suit different types of knowledge.

1. Semantic Networks (Semantic Nets)

A semantic network is a graph-based representation of knowledge, where
Nodes = concepts/objects,
Edges = relationships between concepts.

```
 Animal
   |
 (is-a)
   |
  Dog
   |
(has-property)
   |
 Barks
```

Common Relationships:
• Is-a (inheritance)
• Part-of
• Instance-of
• Has-property

Features:
✔ Easy to visualize
✔ Supports inheritance
✔ Good for hierarchical knowledge
✔ Used in ontologies, WordNet

2. Frames

Frames are structured data objects representing knowledge about a concept, object, or situation using attributes (slots) and their values.

Structure:
Frame = Concept
Slots = Attributes
Fillers = Values

Example (Car Frame):

```yaml
Frame: Car
  Slots:
    Color: Red
    Engine: 1500cc
    Fuel: Petrol
    Owner: Sanjeev
```

Features:
✔ Very organized and hierarchical
✔ Supports defaults
✔ Supports inheritance
✔ Suitable for object-oriented representation

3. Conceptual Dependency (CD Theory)

Introduced by Roger Schank, conceptual dependency represents the meaning of sentences in a language-independent, conceptual form.

Purpose:
✔ Avoid ambiguity
✔ Capture true meaning of actions

CD uses primitive actions, such as:
• PTRANS – physical transfer
• ATRANS – abstract transfer (give, receive)
• MTRANS – mental transfer (informing)
• INGEST – eating
• EXPEL – excretion

Example Sentence:

"Ram gave Sita a book."

CD Representation:
• ATRANS (possession transfer)
• From: Ram
• To: Sita
• Object: Book

4. Scripts

Scripts represent event sequences or stereotypical situations.
Used for understanding stories, dialogues, and events.

Example:

Restaurant Script

markdown

```
1. Enter restaurant
2. Sit at table
3. Order food
4. Eat food
5. Pay bill
6. Leave
```

Use-Cases:
✔ Story understanding
✔ Natural language processing
✔ Event prediction

5. Rule-Based Systems (Production Systems)

Knowledge is represented as IF THEN rules.

Structure:
• IF <condition>
• THEN <action>
Rules + Working Memory + Inference Engine = Production System

Example:

java

```java
IF temperature > 101°F AND cough = true
THEN diagnose = Flu
```

Features:
✔ Easy to update
✔ Human-readable
✔ Used in expert systems (MYCIN, DENDRAL)

Two Main Reasoning Methods:
• Forward Chaining (data-driven)
• Backward Chaining (goal-driven)

6. Propositional Logic (PL)

Represents knowledge using simple statements (propositions) that are either true or false.

Example Propositions:
• P: “It is raining.”
• Q: “The road is wet.”
Compound Statements:
• AND (P ∧ Q)
• OR (P ∨ Q)
• NOT (¬P)
• Implication (P → Q)
• Biconditional (P ↔ Q)

Example:

“If it is raining, the road is wet”:

css

```css
P → Q
```

Features:
✔ Simple
✔ Easy for computers
But ❌ cannot represent relations or quantifiers.

7. Predicate Logic (First-Order Logic, FOL)

Extends propositional logic by allowing statements about objects, properties, and relations.

Syntax:

- Predicates: Loves(Ram, Sita)
- Constants: Ram, Nepal
- Variables: x, y
- Functions: FatherOf(x)
- Quantifiers:
  - o ∀ (For all)
  - o ∃ (There exists)

Features:

✔ Powerful and expressive
✔ Represents relationships
✔ Supports inference

Example:

"All humans are mortal."

```css
∀x: Human(x) → Mortal(x)
```

"Ram is human."

```scss
Human(Ram)
```

3.3.Propositional Logic(PL): Syntax, Semantics, Formal logic-connectives, truth tables, tautology, validity, well-formed- formula, Inference using Resolution

Propositional Logic (PL) is a formal logic system used to represent and reason about facts that are either true or false.
It is the foundation of logical reasoning in AI, digital circuits, and mathematical logic.

Each statement is called a proposition and has only one truth value: True (T) or False (F).
Propositional Logic (PL) is the simplest form of logic used in AI to represent knowledge using propositions (statements) that are either: TRUE, or FALSE

Examples of Propositions:

- P: "It is raining."
- Q: "The road is wet."
- R: "The light is ON."

In 1976, Robert Kowalski came up with an equation
Algorithm = Logic + Control
The logic component specifies the knowledge to be used in solving problems.

Syntax of Propositional Logic
Meaning : Syntax defines the structure and grammar of valid logical expressions.

Basic Elements
1. Propositional symbols: P, Q (atomic propositions)
Example: P = "It is raining" Q = "The ground is wet"
2. Logical connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
3. Parentheses: Used to avoid ambiguity

Syntax Rules
- An atomic symbol is a valid formula
- If P is a formula, then $\neg P$ is a formula
- If P and Q are formulae, then

$(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, $(P \leftrightarrow Q)$ are formulae
Any expression following these rules is a Well-Formed Formula (WFF).

Semantics of Propositional Logic
Semantics deals with the meaning of logical expressions by assigning truth values.
Each proposition is assigned True (T) or False (F)
The truth of a compound statement depends on its connectives

Formal Logic Connectives

1. NOT Operator (Unary Operator)
Definition:
Negates (reverses) the truth value of a proposition.
Unary means it operates on one proposition only.

Symbol: $\neg P$ or $\sim P$

Example:
P = "It is raining."

¬P = “It is NOT raining.”

If P = True → ¬P = False.

2. AND Operator (Conjunctive Operator)
Definition:
P ∧ Q is true only if both P and Q are true.

Symbol: ∧ (and)

Example:
P = “It is sunny.”
Q = “It is warm.”
P ∧ Q = “It is sunny AND warm.”

Truth table:

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

3. OR Operator (Disjunctive Operator)
Definition:
P ∨ Q is true if at least one of P or Q is true.

Symbol: ∨

Example:
P = “I will study.”
Q = “I will play.”
P ∨ Q = “I will study OR play.”

Truth table:

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

4. IMPLICATION (Conditional Operator)
Definition:
P → Q means: If P happens, then Q happens.
Symbol: →

Example:
P = "It rains."
Q = "Road becomes wet."
P → Q = "If it rains, then the road becomes wet."

Truth Table:

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

The only false case is True → False.

4. Biconditional (IF AND ONLY IF)
Definition:
P ↔ Q means:P happens if and only if Q happens.

Symbol: $\leftrightarrow$

Example:
P = “It rains.”
Q = “Road becomes wet.”
$P \leftrightarrow Q$ = “If it rains, then the road becomes wet and if the road becomes wet, then it rains.”

Truth Table:

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

5. Precedence of Operators
When evaluating complex logical expressions, operator precedence is:

1. $\neg$ (NOT)
2. $\land$ (AND)
3. $\lor$ (OR)
4. $\rightarrow$ (Implication)
5. $\leftrightarrow$ (Biconditional)

Example: Evaluate:

$\neg P \lor Q \land R$

Step order:
1. $\neg P$
2. $Q \land R$
3. $\neg P \lor (Q \land R)$

6. **Tautology**
Definition:
A proposition that is always true, for all truth values of variables.

Example:
$P \lor \neg P$
Always TRUE (law of excluded middle).

7. Contradiction
Definition:
A proposition that is always false.

Example:
$P \land \neg P$
Always FALSE.

Validity

An argument is valid if the conclusion is true whenever premises are true.

Example (Modus Ponens)
1. $P \rightarrow Q$
2. P

$\therefore Q$
This argument is valid.
Validity depends on structure, not real-world truth.

Topic: Medical Diagnosis (Simple Logic)
1. $P \rightarrow Q$: “If fever, then illness”
2. P : “Patient has fever”

$\therefore Q$: “Patient is ill”
✔ Valid argument (Modus Ponens)

Well-Formed Formula (WFF)

A WFF is a syntactically correct logical expression.

Examples
✔ $(P \land Q) \rightarrow R$
✔ $\neg(P \lor Q)$

$P \land \rightarrow Q$ (invalid syntax)
$(P\ Q \land)$

Topic: Library Management Rule
Valid WFF:
$(P \land Q) \rightarrow R$
Where:
• P: “Book issued”
• Q: “Due date passed”
• R: “Fine imposed”
Syntax correctness ensures logical clarity.

Inference in Propositional Logic

Inference is the process of deriving new conclusions from known facts.

Common Inference Rules
• Modus Ponens
• Modus Tollens
• Resolution

Topic: Cybersecurity Alert
1. $P \rightarrow Q$: “If malware detected → system alert”
2. P : “Malware detected”
∴ Q : “System alert raised”
Demonstrates logical reasoning.

Resolution

Resolution is a rule of inference used to prove statements by contradiction.

Resolution Rule
From:
$(P \lor Q)$
$(\neg P \lor R)$

We can infer:
$(Q \lor R)$

Steps in Resolution
1. Convert formula to Conjunctive Normal Form (CNF)
2. Apply resolution rule
3. Derive empty clause (contradiction)
→ proof complete

Importance
• Core method in automated theorem Proving
• Used in AI reasoning systems

Topic: Access Control System
Clauses:
• $(\neg P \lor Q)$ → “If logged in, then access”
• (P) → “User logged in”
Resolving:
→ Q (Access granted)

Used in AI theorem proving & automated reasoning.

Numerical
Given Propositions
P: It is raining Q: The road is wet R: Traffic is slow S: I reach office late

Ans

Q1 “If it is raining, then the road is wet.”

$P \rightarrow Q$

Q2 “It is raining and traffic is slow.”

$P \land R$

Q3 “The road is wet or traffic is slow.”

$Q \lor R$

Q4 “I reach office late iff traffic is slow.”

$S \leftrightarrow R$ (same as $R \leftrightarrow S$)

Q5 Find truth of $(P \land Q)$ when P=T, Q=F

$T \land F$ = False

Q6 Construct truth table for $\neg P$

P	¬P
T	F
F	T

Q7 $P \lor \neg P$ is?

Tautology

Q8 Validity of: $P \rightarrow Q, P \therefore Q$

Valid (Modus Ponens)

Q9 Convert $R \rightarrow S$ using only $\neg$ and $\lor$

$\neg R \lor S$

Q10 From $(P \land R) \rightarrow S$ and $P \land R$, derive S

S (Modus Ponens)

3.4 Backward Chaining and Forward Chaining

Forward chaining and backward chaining are two fundamental reasoning techniques used in artificial intelligence and rule-based systems. They differ in their approach to problem-solving and are suited for different scenarios.

Forward Chaining

Forward chaining is a data-driven inference technique. It begins with known facts and applies rules to infer new facts, moving step-by-step toward a conclusion. This approach is ideal for scenarios where real-time data is abundant, and the system needs to process all available information to derive conclusions.

Example:
In a smart home system:

Fact: Room temperature exceeds 25°C.

Rule: If the temperature is high, turn on the AC.

Action: The AC activates.

Advantages

Simplicity: Easy to implement and understand.
Real-time adaptability: Processes data as it arrives, making it suitable for dynamic environments.
Comprehensive: Explores all possible inferences, ensuring thorough analysis.

Disadvantages

Inefficiency for specific goals: May generate irrelevant inferences when only a specific outcome is needed.
Memory-intensive: Requires significant memory to store intermediate facts.
Complexity with large rule sets: Performance can degrade as the number of rules increases.

Backward Chaining

Backward chaining is a goal-driven inference technique. It starts with a specific goal or hypothesis and works backward to determine the facts and rules needed to support it. This method is particularly effective for diagnostic systems and targeted problem-solving.

Example:
In medical diagnosis:

Goal: Confirm diabetes.

Steps: Check blood sugar levels, symptoms, and family history.

Conclusion: If all conditions align, the diagnosis is confirmed.

Advantages

Goal-oriented: Focuses only on the facts needed to achieve the goal, making it efficient.
Resource-efficient: Requires less memory compared to forward chaining.
Interactive: Suitable for systems that need to answer specific queries or solve particular problems.

Disadvantages

Complex implementation: Requires sophisticated strategies to manage its recursive nature.
Predefined goals required: Not suitable for dynamic environments where goals are unknown.
Inefficiency with multiple goals: Needs to be repeated for each goal, which can be time-consuming.

3.5 Predicate Logic: FOPL, Syntax, Semantics, Quantification, Inference with FOPL, Inference using resolution

Predicate Logic is an extension of Propositional Logic that allows us to represent:
• Objects
• Properties of objects
• Relations among objects
• Quantification (for all / there exists)

It is much more expressive and suitable for real-world knowledge representation.

Why Predicate Logic is Needed
Propositional Logic cannot express:
• “All students passed the exam”
• “Some patients have fever”
• “Every human is mortal”
Predicate Logic solves this by introducing predicates, variables, and quantifiers.

First-Order Predicate Logic (FOPL) is an extension of propositional logic that introduces quantifiers, predicates, and variables, allowing for reasoning about objects and their properties.

FOPL uses constants, variables, predicates, and functions to express relationships and properties. For example, "Loves (John, Mary)" indicates a relationship between John and Mary.

Constants
Represent specific objects.
Examples: Ram, Sita, Kathmandu, Car1

Variables
Represent arbitrary objects.
Examples: x, y, z

Predicates
Represent properties or relations.
Student(x) → x is a student

Teaches(x, y) → x teaches y
Greater(x, y) → x is greater than y

Functions
Map objects to objects.
father(x)
age(x)
Example:
age(Ram) = 20

Syntax of Predicate Logic
Syntax defines how valid formulas are written.
Atomic Formula
    A predicate applied to terms:
    Student(Ram)
    Teaches(Teacher, AI)

Compound Formula
    Formed using logical connectives:
    Student(x) ∧ Intelligent(x)
Rules
• If P is a formula → ¬P is a formula
• If P and Q are formulae → (P ∧ Q), (P ∨ Q), (P → Q) are formulae
✔ Any correctly formed expression is a Well-Formed Formula (WFF).

Semantics of Predicate Logic
Semantics gives meaning to formulas.
It depends on:
• Domain of discourse (set of objects)
• Interpretation of predicates and functions
• Assignment of variables

Example:
Domain = {Ram, Sita}
Student(Ram) = True
Student(Sita) = True

Quantifiers

FOPL includes quantifiers like universal quantifiers ($\forall$) and existential quantifiers ($\exists$) to express statements about all or some objects.

1. Universal Quantifier ($\forall$)

Means "for all".

Example:

$\forall x\ Human(x) \rightarrow Mortal(x)$

Meaning: All humans are mortal.

2. Existential Quantifier ($\exists$)

Means "there exists".

Example:

$\exists x\ Student(x) \land Intelligent(x)$

Meaning: There exists a student who is intelligent

Inference in Predicate Logic

1. Universal Instantiation (UI)

From:

$\forall x\ P(x)$

Infer:

P(Ram)

2. Existential Instantiation (EI)

From:

$\exists x\ P(x)$

Infer:

P(c)   (c is a new constant)

3. Modus Ponens

From:

$P \rightarrow Q$

P

Infer:

Q

Resolution in Predicate Logic
Resolution is a mechanical inference method used in AI.

Steps
1. Convert formulas to Clause Form
2. Remove quantifiers (Skolemization)
3. Apply Unification
4. Apply Resolution Rule

Example

Clauses:

$(\neg Human(x) \lor Mortal(x))$

$Human(Ram)$

Resolve $\Rightarrow$

$Mortal(Ram)$

Example (Real-World Representation)
Statement
"All students who study AI are intelligent. Ram studies AI."
Predicate Logic

$\forall x\ (StudiesAI(x) \rightarrow Intelligent(x))$

$StudiesAI(Ram)$

Inference

$Intelligent(Ram)$

Case Study: Smart University Academic Decision System

A large public university has implemented an AI-driven Academic Decision Support System to assist the examination department, faculty advisors, and scholarship committee. The system uses First Order Predicate Logic (FOPL) to represent academic rules, student conditions, and institutional policies so that decisions are consistent, transparent, and logically provable.
The university follows these academic policies:
• Every student must maintain minimum attendance to be eligible for examinations.
• A student who has attendance ≥ 80% and has submitted all assignments is considered exam eligible.
• Any student who is exam eligible and has a GPA ≥ 3.6 qualifies for a merit scholarship.

• A student who has failed a subject must attend remedial classes.
• Any student who attends remedial classes and passes the re-exam is considered to have cleared the subject.
• Students who clear all subjects and complete the final project are eligible for degree award.

The system stores real-time data from attendance records, assignment submissions, exam results, and project evaluations.

The following facts are currently stored in the system:
• Ram is a registered student of the university.
• Ram has an attendance of 85%.
• Ram has submitted all required assignments.
• Ram has achieved a GPA of 3.8.
• Ram has passed all subjects.
• Ram has successfully completed the final project.

The university wants the AI system to determine, using predicate logic inference, whether Ram qualifies for a merit scholarship and degree award.

(Given)

Constants

- Ram

Predicates

- Student(x) – x is a student
- Attendance80(x) – x has attendance ≥ 80%
- SubmittedAllAssignments(x) – x submitted all assignments
- ExamEligible(x) – x is eligible for exam
- GPAHigh(x) – x has GPA ≥ 3.6
- MeritScholarship(x) – x qualifies for merit scholarship
- FailedSubject(x) – x has failed a subject
- RemedialClass(x) – x attends remedial classes
- ClearedSubject(x) – x has cleared subject
- CompletedProject(x) – x completed final project
- DegreeEligible(x) – x is eligible for degree award

Knowledge Base (First Order Predicate Logic Rules)

1. $\forall x \quad [(Student(x) \wedge Attendance80(x) \wedge SubmittedAllAssignments(x)) \rightarrow ExamEligible(x)]$

2. $\forall x\ [(ExamEligible(x) \wedge GPAHigh(x)) \rightarrow MeritScholarship(x)]$
3. $\forall x\ [FailedSubject(x) \rightarrow RemedialClass(x)]$
4. $\forall x\ [(RemedialClass(x) \wedge PassedReExam(x)) \rightarrow ClearedSubject(x)]$
5. $\forall x\ [(ClearedSubject(x) \wedge CompletedProject(x)) \rightarrow DegreeEligible(x)]$

Facts (Ground Predicates)
6. $Student(Ram)$
7. $Attendance80(Ram)$
8. $SubmittedAllAssignments(Ram)$
9. $GPAHigh(Ram)$
10. $ClearedSubject(Ram)$
11. $CompletedProject(Ram)$

3.6 Bayes' Rule and its use, Bayesian Networks

Bayes' Rule is a fundamental principle in probability and statistics that relates the conditional probability of two random events and their marginal probabilities.
Bayes' Rule is a fundamental principle of probability theory used in AI to update the probability of a hypothesis when new evidence is observed.
It is used to calculate the value of P(B|A) by using the knowledge of P(A|B).

Bayes' theorem is the name given to the formula used to calculate conditional probability. The formula is as follows:

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

where,

P(A) is the probability that event A occurs.
P(B) defines the probability that event B occurs.
P(A|B) is the probability of the occurrence of event A given that event B has already occurred.
P(B∣A) can now be read as: Probability of event B occurring given that event A occurred.
p(A∩B) is the probability events A and B will happen together.

The Bayes' Theorem encompasses four major elements:
1. Prior Probability (P(A)): The probability or belief in an event A prior to considering any additional evidence, it represents what we know or believe about A based on previous knowledge.

2. Likelihood P(B|A): the probability of evidence B given the occurrence of event A. It determines how strongly the evidence points toward the event.

3. Evidence (P(B)): Evidence is the probability of observing evidence B regardless of whether A is true. It serves to normalize the distribution so that the posterior probability is a valid probability distribution.

4. Posterior Probability P(A|B): The posterior probability is a revised belief regarding event A, informed by some new evidence B. It answers the question, "What is the probability that A is true given evidence B observed?"

Using these components, Bayes' Theorem computes the posterior probability P(A|B), which represents our updated belief in A after considering the new evidence.

How Bayes theorem is relevant in AI?
Bayes' theorem is highly relevant in AI due to its ability to handle uncertainty and make decisions based on probabilities. Here's why it's crucial:

Probabilistic Reasoning: In many real-world scenarios, AI systems must reason under uncertainty. Bayes' theorem allows AI systems to update their beliefs based on new evidence. This is essential for applications like autonomous vehicles, where the environment is constantly changing and sensors provide noisy information.

Machine Learning: Bayes' theorem serves as the foundation for Bayesian machine learning approaches. These methods allow AI models to incorporate prior knowledge and update their beliefs as they see more data. This is particularly useful in scenarios with limited data or when dealing with complex relationships between variables.

Classification and Prediction: In classification tasks, such as spam email detection or medical diagnosis, Bayes' theorem can be used to calculate the probability that a given input belongs to a particular class. This allows AI systems to make more informed decisions based on the available evidence.

Anomaly Detection: Bayes' theorem is used in anomaly detection, where AI systems identify unusual patterns in data. By modeling the normal behavior of a system, Bayes' theorem can help detect deviations from this norm, signaling potential anomalies or security threats.

Example of Bayes' Rule Application in AI

One of the good old example of Bayes' Rule in AI is its application in spam email classification. This example demonstrates how Bayes' Theorem is used to classify emails as spam or non-spam based on the presence of certain keywords.

Consider an email filtering system that needs to determine whether an incoming email is spam or not based on the presence of the word "win" in the email. We are given the following probabilities:

$P(S)$: The prior probability that any given email is spam.
$P(H)$: The prior probability that any given email is not spam (ham).
$P(W|S)$: The probability that the word "win" appears in a spam email.
$P(W|H)$: The probability that the word "win" appears in a non-spam email.
$P(W)$: The probability that the word "win" appears in any email.

Given Data

$P(S)=0.2$ (20% of emails are spam)
$P(H)=0.8$ (80% of emails are not spam)
$P(W|S)=0.6$ (60% of spam emails contain the word "win")
$P(W|H)=0.1P$ (10% of non-spam emails contain the word "win")

We want to find $P(S|W)$, the probability that an email is spam given that it contains the word "win".

Applying Bayes rule we get:

$$P(S|W)= P(W|S)/(P(S)P(W))$$

First, we need to calculate $P(W)$, the probability that any email contains the word "win". Using the law of total probability:

$$P(W)=P(W|S)\cdot P(S)+P(W|H)\cdot P(H)$$

Substituting the given values:
$P(W)=(0.6 \cdot 0.2)+(0.1 \cdot 0.8)=0.2$

Now, we can use Bayes' Rule to find $P(S|W)$:

$$P(S|W)= P(W)/(P(W|S) \cdot P(S))$$

,

substituting the values:

$$P(S|W)= 0.2/(0.6 \cdot 0.2)$$
$$=0.6$$

Thus we can conclude that the probability that an email is spam given that it contains the word "win" is 0.6, or 60%. This means that if an email contains the word "win," there is a 60% chance that it is spam.

In a real-world AI system, such as an email spam filter, this calculation would be part of a larger model that considers multiple features (words) within an email. The filter uses these probabilities, along with other algorithms, to classify emails accurately and efficiently. By continuously updating the probabilities based on incoming data, the spam filter can adapt to new types of spam and improve its accuracy over time.

Bayesian Networks

Bayesian networks are graphical models that use Bayes' theorem to represent and predict probabilistic relationships between variables.
They are used in a variety of AI applications, such as medical diagnosis, fault detection, and decision support systems.

Bayesian Networks, also known as Bayesian Belief Networks (BBNs) or Bayes Nets, are graphical models that use Bayes' theorem to represent and predict probabilistic relationships between variables.
They are widely applied in various fields, including artificial intelligence, machine learning, and data analysis.

A Bayesian Network (also known as a Bayes network, belief network, or decision network) is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG).
These networks are ideal for modeling the probabilistic relationships between variables and for performing inference and learning tasks.

Key Principles

Bayesian networks consist of two main components:

Directed Acyclic Graph (DAG): Nodes represent variables, and directed edges represent conditional dependencies between these variables.

Conditional Probability Tables (CPTs): Each node has a CPT that quantifies the effect of the parent nodes on the node.

Example

Consider a Bayesian network with three variables: Burglary (B), Earthquake (E), and Alarm (A). The alarm can be triggered by either a burglary or an earthquake. The network structure is as follows:

B and E are parent nodes of A.

The CPT for A depends on the states of B and E.

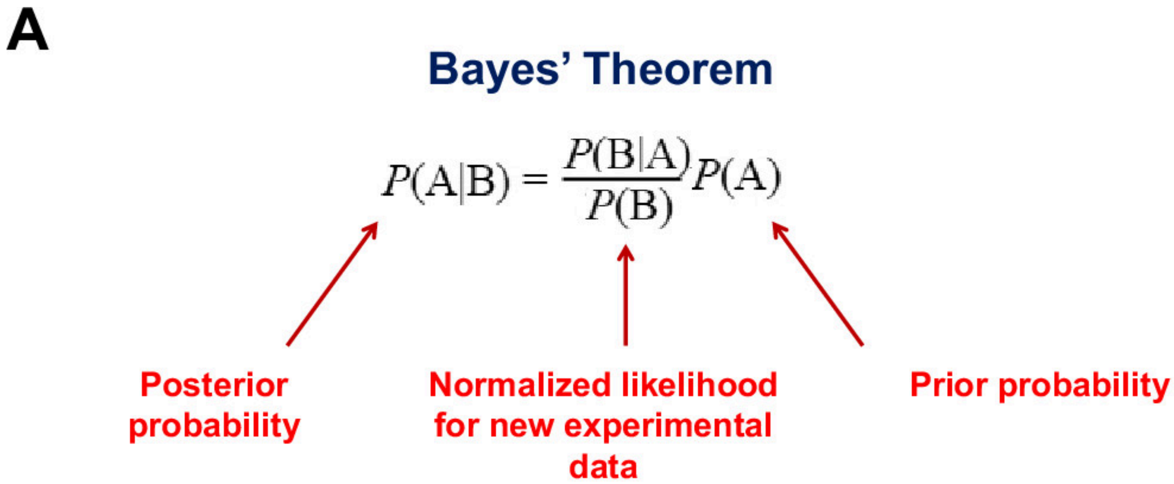
The joint probability distribution can be written as: [$P(B, E, A) = P(A|B, E) .P(B) .P(E)$]

Inference
Bayesian networks can perform three main inference tasks:

Inferring Unobserved Variables: Given evidence, compute the posterior distribution of other variables.

Parameter Learning: Estimate the parameters of the CPTs from data.

Structure Learning: Learn the network structure from data.

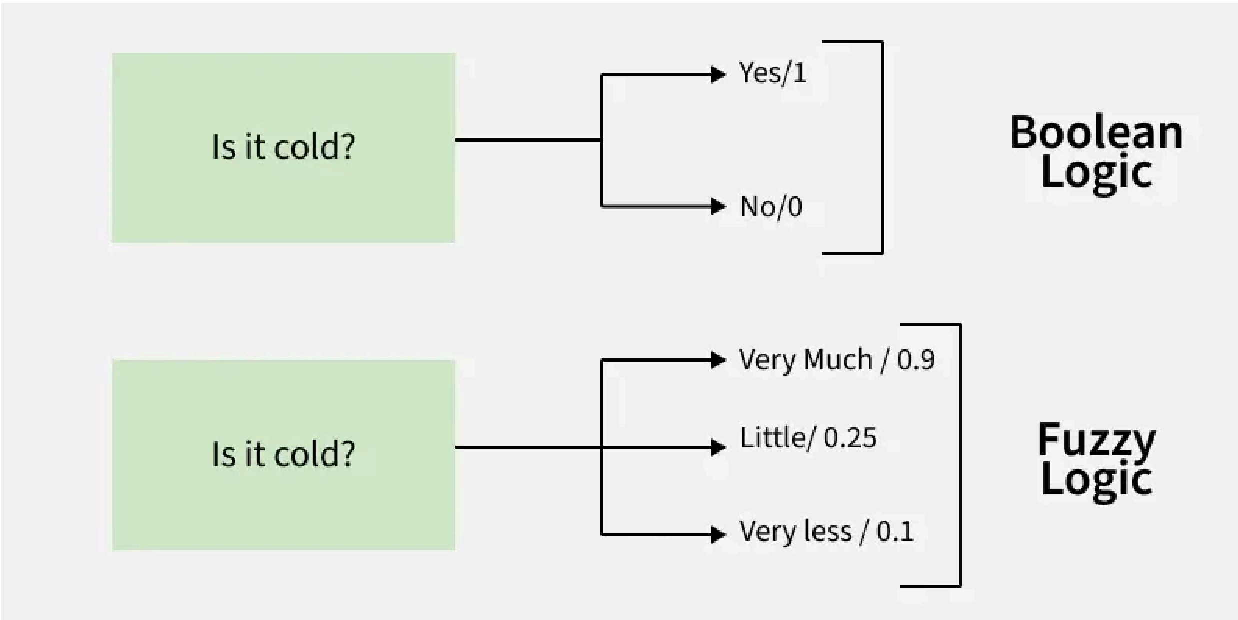


3.7 Fuzzy Logic

Fuzzy Logic is a reasoning technique that handles imprecision and uncertainty by allowing partial truth values between 0 and 1, unlike classical logic which uses only true or false.
Fuzzy Logic helps work with situations where the information is unclear or partly true. Instead of only 0 or 1 like traditional logic, it allows values between 0 and 1 to represent partial truth. This makes it useful in real-world decision-making where data is not exact.

Handles uncertainty and vague information.
Works with partial truth values between 0 and 1.
Useful in control systems, medical diagnosis, AI and image processing.
Helps systems take flexible and human-like decisions.

Fuzzy Logic | Introduction - GeeksforGeeks


Is it cold?
Yes/1
No/0
Boolean Logic
Is it cold?
Very Much / 0.9
Little/ 0.25
Very less / 0.1
Fuzzy Logic