UNIT-4

Knowledge Representation

Knowledge Representation (KR) in Artificial Intelligence refers to the method of storing, organizing, and structuring information about the world in such a way that a computer system can understand, reason, make decisions, and solve problems like a human.
It involves creating formal models that describe objects, facts, concepts, relationships, and rules within a particular domain.

In AI, knowledge representation is not just about storing data; it is about storing it meaningfully, so the machine can use it for inference (logical reasoning), interpretation, and problem-solving

Knowledge Representation (KR) in Artificial Intelligence is a subfield of AI that focuses on designing computer-interpretable structures which describe knowledge about the real world or any problem domain. KR provides symbols, models, and formalisms that enable the system to *represent facts*, *define relationships*, and *apply logical rules* to derive new information from existing data. It forms the foundation for intelligent behavior such as planning, understanding language, diagnosing problems, and learning from past experiences.

KR aims to capture knowledge in such a way that a machine can:

1. **Store** the knowledge efficiently
2. **Retrieve** the knowledge when needed
3. **Understand** the relationships between facts
4. **Reason** logically using rules and constraints
5. **Update** the knowledge as new facts arrive

Thus, knowledge representation transforms real-world information into a structured form that a computer can use to act intelligently.

Approaches in Artificial Intelligence (AI)

Approaches in AI refer to the **different methods, techniques, or strategies** used to design intelligent systems. These approaches describe **how an AI system thinks, learns, represents knowledge, and solves problems**. Each approach follows a different philosophy about how intelligence works—some imitate human thinking, some use mathematical models, and some learn from experience.

Approaches in AI can broadly be divided into **four major categories**:

1. Symbolic / Knowledge-Based Approach (Top-Down Approach)

This approach assumes that intelligence can be created by **representing human knowledge in symbols** and applying **logical rules** to manipulate these symbols.
AI systems are given explicit information in forms such as rules, facts, and logic, and the system makes decisions through **reasoning (inference)**.

Example:

- **Expert System** like MYCIN uses *if–then* rules to diagnose diseases.
- "If fever AND cough → Possible flu"

2. Sub-Symbolic / Machine Learning Approach (Bottom-Up Approach)

This approach focuses on enabling machines to **learn from data** rather than being explicitly programmed with rules. It uses models such as **Neural Networks, Deep Learning, Decision Trees, SVMs**, etc.
The system improves its performance automatically by analyzing patterns in the data.

Example:

- A neural network learns to recognize cat images by seeing thousands of cat pictures.
- Speech recognition systems like Alexa or Google Assistant.

3. Search-Based Approach

This approach treats problem-solving as a process of **searching through possible states** to reach a goal.
AI explores different paths (states) using algorithms like **Breadth-First Search (BFS), Depth-First Search (DFS), A***, etc., until it finds an optimal or acceptable solution.

Example:

- Finding the shortest path in Google Maps
- Puzzle solving (8-Puzzle, Chess moves)

4. Statistical and Probabilistic Approach

This approach uses **probability theory and statistics** to handle uncertainty in AI.
It is used when data is incomplete, noisy, or unpredictable. Models like **Bayesian Networks, Hidden Markov Models (HMMs), Markov Decision Processes (MDPs)** fall under this category.

Example:

- Predicting weather using probability
- Spam detection using statistical patterns

Issues in Artificial Intelligence

Issues in AI refer to the **challenges, limitations, and problems** that arise when developing or deploying AI systems. These issues may be **technical, ethical, social, or practical** and must be handled carefully to build safe and reliable AI.

Major issues in AI are described below:

1. Knowledge Representation Problem

AI needs a large amount of knowledge represented in a structured, logical format. The challenge is deciding **how to represent complex real-world knowledge** so that the machine understands it.
Not all knowledge fits easily into rules or symbols.

Example:

- Representing “common sense” knowledge like “ice is cold” is difficult for machines.

2. Reasoning and Inference Problem

AI systems must draw conclusions from available facts. But real-world information may be **incomplete, uncertain, or contradictory**, making logical reasoning difficult.

Example:

- A medical expert system may have incomplete symptoms but still must make a diagnosis.

3. Learning from Limited Data

Machine Learning systems often need **massive amounts of data** to learn accurately.
When data is limited or biased, AI will produce poor results.

Example:

- Facial recognition systems fail to identify people correctly if trained on limited data.

4. Natural Language Understanding Issue

Human language is complex, ambiguous, and full of context.
Understanding sentences, slang, emotion, sarcasm, and grammar is extremely challenging for AI.

Example:

- “I saw a man with a telescope.”
  AI cannot easily understand who has the telescope.

5. Perception and Vision Issues

AI systems using Computer Vision or Sensors interpret images, sounds, and surroundings.
But real-world environments are unpredictable, causing errors.

Example:

- Self-driving car misidentifies an object due to bad lighting or fog.

6. Ethical and Social Issues

AI raises concerns about privacy, job loss, bias, and misuse.
AI might make unfair decisions if trained on biased data.

Example:

- AI hiring tools rejecting candidates based on biased patterns.
- Surveillance systems violating privacy.

7. Computational Cost and Resources

Advanced AI models require **high processing power, large memory, GPUs, and long training time**, making implementation expensive.

Example:

- Training a large Deep Learning model costing thousands of dollars in electricity and hardware.

8. Explanation and Transparency Issue ("Black Box")

Some AI models, especially Deep Learning networks, make decisions **without explaining how**.
This makes trust and debugging difficult.

Example:

- AI predicts a medical risk but cannot explain which factor caused it.

Frame in Artificial Intelligence

A **Frame** in Artificial Intelligence is a **knowledge representation structure** that organizes information about an object, concept, or situation in the form of an **organized template**.
A frame is essentially a **data structure** consisting of **slots** (attributes) and **slot values** (information about those attributes).

The main purpose of a frame is to represent real-world knowledge in a **structured and hierarchical form** so that an AI system can easily:

- understand the properties of an object,
- apply **inheritance** (child inherits properties from the parent),
- perform reasoning,
- fill missing information using **default values**,
- and process complex information in a simplified unit-based form.

Frames divide complex knowledge into smaller **units**, making it easier for AI systems to store, retrieve, and reason with the information.
Frames are similar to **Semantic Networks**, but they are more structured, organized, and powerful.

A frame generally consists of:

1. **Frame Name** – the name of the object or concept
2. **Slots** – the attributes or properties
3. **Fillers** – the values stored in the slots

Frames are used in various areas of AI such as **expert systems, natural language understanding, scene representation, planning, and vision systems**.

A Frame is a **template-like structure** where all the information related to an object is stored in "slots" so that AI can understand and reason about it.

Example of a Frame

Frame: PERSON

Frame Name: PERSON

Slots:

Name: (Aarti, John, etc.)

Age: (value)

Gender: (Male/Female)

Occupation: (Teacher, Engineer, Student)

Address: (City)

Height: (value)

Weight: (value)

Now let's create a specific frame:

Frame: PERSON–AARTI

Frame Name: PERSON-AARTI

Slots:

Name: Aarti

Age: 25

Gender: Female

Occupation: Computer Teacher

Address: Raipur

From this frame, the AI can reason:

- Aarti is a PERSON → inherits all properties of the PERSON frame
- If Occupation = Teacher → AI can apply teacher-related reasoning

Example of a Frame (Animal Example)

Frame: DOG

Frame Name: DOG

Slots:

Type: Mammal

Sound: Bark

Legs: 4

Color: (variable)

Breed: (variable)

Food: Non-vegetarian

Frame: DOG–TOMMY

Frame Name: DOG-TOMMY

Slots:

Color: Brown

Breed: Labrador

Age: 3

The AI will automatically understand:

- TOMMY is a DOG → inherits all default properties of DOG
- Sound = Bark
- Legs = 4

Conceptual Dependency (CD) in Artificial Intelligence

Conceptual Dependency (CD) is a **theory of natural language understanding** introduced by **Roger Schank (1973)**.
It is a **knowledge representation model** designed to represent the meaning of sentences in a **language-independent**, **unambiguous**, and **concept-level** form.

The main idea behind Conceptual Dependency is that all sentences, regardless of the language used, can be represented in terms of a **small set of primitive actions (ACTs)** and **conceptual structures**.
These primitive acts express the **basic meaning** behind every verb or action.
The representation focuses on **what actually happens** in the real world, not on the exact words used in the sentence.

Conceptual Dependency aims to:

- remove the ambiguity found in natural languages,
- represent meaning in a standardized way,
- make it easier for AI systems to understand, infer, and reason about events,

- allow AI to identify relationships, motivations, outcomes, and consequences of actions,
- and support machine translation, question answering, and story understanding.

In CD theory, every action is broken into a **small set of primitive act categories**, such as:

- **PTRANS** – Physical Transfer (movement)
- **ATRANS** – Abstract Transfer (ownership change)
- **MTRANS** – Mental Transfer (communication, knowledge transfer)
- **MBUILD** – Mental Creation (thinking or planning)
- **INGEST** – Eating, drinking
- **PROPEL** – Applying physical force
- **MOVE** – Movement of body parts
- **SPEAK** – Production of sound or speech

CD also uses **conceptual cases**, such as:

- **Actor** – who performs the action
- **Object** – what is acted upon
- **Instrument** – tool used
- **Recipient** – who receives something
- **Direction / Source / Destination** – movement relations

By representing meaning in this structured form, AI can easily compare, analyze, or infer deeper meaning, even if sentences appear different on the surface.

Simple Meaning

Conceptual Dependency is a method of representing the meaning of sentences using a set of basic action primitives so that AI can understand the **real meaning** behind any sentence, regardless of the language used.

Example of Conceptual Dependency

Sentence:

“John gave a book to Mary.”

This sentence involves a transfer of ownership.
Therefore, the primitive act is: **ATRANS (transfer of possession).**

CD Representation (Conceptual Form):

- **ATRANS**
    - **Actor:** John
    - **Object:** Book
    - **Recipient:** Mary
    - **From:** John’s possession
    - **To:** Mary’s possession

Although the sentence could be written in many different ways (e.g., “John gifted Mary a book”), the CD representation remains the same because it captures the **real meaning**, not the words.

Another Example

Sentence:

“Ram told Sita the truth.”

This is mental transfer, so the primitive act is: **MTRANS.**

CD Representation:

- **MTRANS**
    - **Actor:** Ram
    - **Object:** Information (truth)
    - **Recipient:** Sita

Semantic Net in AI

A **Semantic Network (Semantic Net)** is a **knowledge representation technique** used in Artificial Intelligence to represent **objects, concepts, events, and their relationships** in the form of a **graph structure**. It represents knowledge using **nodes (concepts)** and **links/edges (relations)**, making it easier for machines to understand and reason about the relationships between different entities.

In Semantic Nets, each **node** stands for an object, idea, or concept, while each **link** represents the relationship such as **"is-a", "has-a", "part-of", "instance-of", "connected-to", "owns", "lives-in"**, etc. This graphical representation enables an AI system to perform **logical inference**, **knowledge retrieval**, and **natural language understanding**.

Semantic Nets were first introduced by **Ross Quillian (1966)** to model **human memory** and show how humans store and recall information based on conceptual relationships. They provide a **structured, associative network** similar to the working of the human brain, where information is represented in a connected form rather than isolated statements.

Key Characteristics of Semantic Nets

1. **Graph-based structure**
   Knowledge is stored as a graph where:
   - Nodes = objects or concepts
   - Edges = relationships between them
2. **Inheritance mechanism**
   Semantic Nets support **property inheritance**, where lower-level nodes inherit features from their parent nodes (similar to object-oriented programming).
   Example:
   If "Bird" has property "can fly", then "Sparrow", being a type of bird, automatically inherits that property.

3. **Expressive representation**
   Semantic nets can represent **static knowledge (facts)** and **dynamic knowledge (actions)**.
4. **Easy visualization**
   The network form makes understanding relationships intuitive and human-like.
5. **Supports inference**
   AI can perform reasoning by tracing the links between nodes.

Structure of Semantic Networks

A Semantic Net includes:

1. Nodes

These represent:

- Objects (e.g., Dog, Car)
- Concepts (e.g., Animal, Vehicle)
- Events (e.g., Running, Eating)
- Attributes (e.g., Color, Size)

2. Links (Relationships)

These include:

- **IS-A (Category link)** – shows class membership
- **PART-OF** – shows composition
- **HAS-A** – shows possession
- **INSTANCE-OF** – shows specific examples
- **CAUSE-EFFECT** – shows causality
- **AGENT, OBJECT, LOCATION** – action-based relations

Example of a Semantic Net (Detailed)

Consider the sentence:
“A robin is a bird. A bird is an animal. Birds can fly.”

Semantic Network Representation:

- Node: Robin
  → IS-A → Bird
- Node: Bird
  → IS-A → Animal
  → CAN → Fly

Graphically:

Animal

↑

IS-A

|

Bird ---- CAN ----> Fly

↑

IS-A

|

Robin

Inference by Semantic Net:

From this structure, a machine can infer:

- A robin is an animal
- A robin can fly

This is due to **inheritance**.

Applications of Semantic Nets in AI

1. **Natural Language Processing (NLP)**
   Understanding meaning by associating words with concepts.
2. **Expert Systems**
   Representing expert knowledge such as medical diagnosis.
3. **Machine Reasoning**
   Making inferences from interconnected knowledge.
4. **Ontology Design**
   Used in semantic web and knowledge graphs.
5. **Cognitive Modeling**
   Modeling human memory and association patterns.

Advantages of Semantic Nets

- Easy to visualize knowledge
- Supports inheritance (reduces redundancy)
- Good for representing relationships
- Efficient in semantic searching and reasoning
- Useful for concept-based decision making

Limitations of Semantic Nets

- Representation may become too large and complex
- Difficult to represent procedural knowledge
- Not suitable for ambiguous or inconsistent knowledge
- Inference sometimes becomes computationally expensive

Scripts in AI

In Artificial Intelligence, a **Script** is a **structured knowledge representation framework** used to describe **sequences of events** that typically occur in common, everyday situations. The concept of Scripts was introduced by **Roger Schank and Robert Abelson (1977)** as part of **conceptual dependency theory**, to help AI systems understand and predict human behavior in routine scenarios.

A Script provides a **standardized, organized, and stereotyped sequence of actions** that generally take place in a specific context. For example, when a person goes to a restaurant, a predictable series of events occurs: entering, sitting, ordering, eating, paying, and leaving. Scripts capture such sequences in a formal representation so that AI systems can understand, infer, and reason about actions even if certain details are missing in the input text.

Scripts help an AI system **fill in the gaps**, **infer unstated events**, and **interpret natural language** more accurately by using background world knowledge. They act as **templates** that describe "how things normally happen," allowing computers to treat human experiences in a structured, context-aware way.

Key Characteristics of Scripts

1. Predefined Event Sequence

A Script contains a pre-arranged order of events that typically happen in a particular situation.

2. Slots and Values

Scripts are made up of **slots** (specific roles) such as:

- Entry conditions
- Props (objects involved)
- Roles (people involved)
- Scenes (sub-events)
- Results or outcomes

Each slot can have defined or default values.

3. Predictive Inference

Even if a user does not mention all events, the AI can infer them using the script.

Example:
If a sentence says, “John paid the bill and left the restaurant,” the AI can infer that he must have eaten even if it is not explicitly stated.

4. Hierarchical Structure

Scripts often contain **scenes**, which break down the main script into smaller sub-events.
E.g., Restaurant Script → Ordering Scene → Eating Scene → Payment Scene.

5. Domain-Specific

Different scenarios have different scripts:

- Airport script
- Classroom script
- Movie theatre script
- Doctor’s visit script

Components of a Script

A script is typically divided into the following parts:

1. **Entry Conditions**
   Conditions that must be true before the script begins.
2. **Props**
   Objects used in the script (menu, table, food, bill, etc.)
3. **Roles**
   People involved (customer, waiter, cashier).
4. **Scenes**
   Sub-events with a sequence of actions.

5. **Results**
   What happens after the script completes.

Example: Restaurant Script (Detailed)

A typical **Restaurant Script** includes:

Entry Conditions

- Customer is hungry
- Customer has money

Props

- Menu
- Table
- Food
- Bill

Roles

- Customer
- Waiter
- Cook
- Cashier

Scenes

1. **Entering and Being Seated**
2. **Ordering Food**
3. **Eating Food**
4. **Requesting and Paying Bill**
5. **Leaving**

Result

- Customer is no longer hungry
- Customer has less money

Why Scripts Are Important in AI

1. **Natural Language Understanding (NLU)**
   Helps AI interpret text by using real-world context.
2. **Inference Generation**
   If some actions are missing in a story, scripts help fill the missing steps.
3. **Disambiguation**
   Helps resolve ambiguous sentences using expected event sequences.
4. **Knowledge Organization**
   Stores experiences in a structured and reusable form.
5. **Understanding Human Behavior**
   Useful in cognitive modeling and psychology-based AI.

Advantages of Scripts

- Efficient representation of common events
- Reduces ambiguity in understanding text
- Helps AI infer unstated facts
- Enables story understanding and narrative reasoning

Limitations of Scripts

- Limited to stereotypical events
- Cannot handle unusual or exceptional cases
- Hard to scale for all real-world scenarios
- Too rigid for dynamic or unpredictable events

Propositional Logic in AI

Propositional Logic—also known as **Propositional Calculus** or **Boolean Logic**—is a fundamental branch of logic used in Artificial Intelligence for representing **facts**, **truth statements**, and **logical reasoning**. In Propositional Logic, the basic building blocks are **propositions**, which are simple declarative statements that are either **true (T)** or **false (F)** but **not both**. These propositions are represented using **propositional variables** such as *P, Q, R*, and they are combined using **logical connectives** like AND, OR, NOT, IF–THEN, and IF AND ONLY IF to express more complex logical statements.

Propositional Logic provides AI systems with a **formal, mathematical, and syntax-based method** for representing information and performing **deductive reasoning**. It enables machines to derive new facts from known facts using rules of inference. Since propositional logic deals only with complete statements (not internal structure of statements), it is considered a **simple yet powerful** form of logic that forms the foundation of more advanced systems such as **Predicate Logic**, **Resolution-based reasoning**, **Expert Systems**, and **Automated Theorem Proving**.

Key Characteristics of Propositional Logic

1. Truth-valued Statements

A proposition must have a definite truth value:

- TRUE (T)
- FALSE (F)

Example:
“Sky is blue” → True
“4 is greater than 10” → False

2. Propositional Variables

Statements are replaced with symbols like P, Q, R for easy manipulation.

Example:
Let P = “It is raining”
Let Q = “I will carry an umbrella”

3. Logical Connectives

Propositions can be combined using connectives:

Connective	Symbol	Meaning
AND	∧	Both statements true
OR	∨	At least one is true
NOT	¬	Negation
IMPLICATION	→	If...then
BICONDITIONAL	↔	If and only if

Example:
P → Q
“If it is raining, then I will carry an umbrella.”

4. Syntax and Semantics

- **Syntax** defines the rules for forming valid expressions.
- **Semantics** assigns truth values to expressions.

5. Truth Tables

Truth tables are used to compute the truth values of compound propositions.

Inference in Propositional Logic

AI uses rules of inference to derive new facts:

1. Modus Ponens

If P → Q and P is true, then Q must be true.

2. Modus Tollens

If P → Q and Q is false, then P is false.

3. Resolution Rule

Used in automated theorem proving.

Propositional Logic Example (Detailed)

Consider:
P = “It is sunny.”
Q = “We will go to the park.”

Statement:
If it is sunny, then we will go to the park.
This is written as:

P → Q

Now suppose P is true.
Using **Modus Ponens**, we infer:

Since P → Q and P is true → therefore Q is true.

This is how AI uses propositional logic to reason.

Applications of Propositional Logic in AI

1. **Expert Systems**
   For rule-based reasoning:
   Example:
   IF symptoms → disease.
2. **Automated Theorem Proving**
3. **Knowledge-Based Systems**
4. **Model Checking**
5. **Digital Circuit Design**
   Boolean expressions represent circuits.

6. **AI Planning and Decision Making**
7. **Game Playing and State-Space Reasoning**

Advantages of Propositional Logic

- Simple and easy to understand
- Mathematically precise
- Efficient for small problem domains
- Forms the foundation of more advanced logic systems
- Supports automated reasoning via truth tables and resolution

Limitations of Propositional Logic

- Cannot represent internal structure of statements
  ("All humans are mortal" cannot be represented easily)
- Cannot express relationships between objects
- Large problems require huge numbers of variables
- Not expressive enough for complex real-world knowledge
- No quantifiers (like ∀ or ∃) available

These limitations are addressed using **Predicate Logic**, which is more expressive.

First-Order Propositional Logic (FOPL)

First-Order Propositional Logic (FOPL)—more commonly called **First-Order Logic (FOL)**—is an advanced form of logic used in Artificial Intelligence to represent complex knowledge about objects, their properties, and the relationships between them. Unlike *propositional logic*, which can only express simple true/false statements, **FOPL introduces**

variables, quantifiers, predicates, and functions, enabling it to represent general statements about a whole class of objects rather than specific instances.

Detailed Technical Definition

First-Order Propositional Logic is a formal system of logic that extends propositional logic by allowing statements to include:

1. **Objects (Constants)**
   These are specific entities like Ram, Ravi, Apple.
2. **Variables**
   Symbols like x, y, z that can refer to any object in a domain.
3. **Predicates**
   Functions that return true or false.
   Example:
   - Student(x) → "x is a student"
   - Loves(x, y) → "x loves y"
4. **Functions**
   Mappings from objects to objects, such as:
   - FatherOf(x) → father of x
   - Height(x) → height of x
5. **Quantifiers**
   - **Universal quantifier (∀)** → "for all"
   - **Existential quantifier (∃)** → "there exists"

Using these elements, FOPL can describe **general truths**, **rules**, **relationships**, and **constraints**, making it far more expressive than propositional logic.

Why FOPL is Important in AI

FOPL allows machines to:

- Represent real-world facts in a structured way

- Perform reasoning and inference
- Automate logical deduction
- Build expert systems and knowledge bases

Because FOPL can represent universal statements (true for all objects) and existential statements (true for at least one object), it is widely used in AI fields such as **knowledge representation, planning, expert systems, semantic networks, and natural language understanding**.

Examples of FOPL

1. Universal Statement

English: "All humans are mortal."
FOPL:

$\forall x$ (Human(x) → Mortal(x))

2. Existential Statement

English: "There exists a student who studies AI."
FOPL:

$\exists x$ (Student(x) $\wedge$ Studies(x, AI))

3. Relationship Example

English: "Ravi is the father of Mohan."
FOPL:

Father(Ravi, Mohan)

4. Rule-Based Example

English: "If a person is a mother, then the person is female."
FOPL:

$\forall x$ (Mother(x) → Female(x))

5. Using Functions

English: “The mother of Rhea is Sita.”
FOPL:

MotherOf(Rhea) = Sita

First-Order Propositional Logic (FOPL) – Long Definition (Hindi with English Technical Terms)

First-Order Propositional Logic (FOPL) — जिसे आमतौर पर **First-Order Logic (FOL)** भी कहा जाता है – AI में knowledge representation के लिए इस्तेमाल होने वाली एक powerful logical system है। यह **propositional logic** से कहीं ज्यादा expressive होती है, क्योंकि यह सिर्फ True/False statements पर ही निर्भर नहीं रहती, बल्कि **variables, predicates, quantifiers, objects, और functions** का उपयोग करके complex knowledge को represent कर सकती है।

लंबी व तकनीकी परिभाषा

First-Order Propositional Logic एक ऐसा formal logical framework है जिसमें knowledge को represent करने के लिए निम्न components होते हैं:

1. Objects (या Constants)

ये वास्तविक entities को represent करते हैं।
जैसे: Ram, Apple, Ravi

2. Variables

Symbols जैसे x, y, z
ये किसी भी object को represent कर सकते हैं।

3. Predicates

ये ऐसे functions होते हैं जो True या False return करते हैं। ये किसी object की property या objects के बीच relationship को दर्शाते हैं।
Examples:

- Student(x) → “x एक student है”
- Loves(x, y) → “x y को love करता है”

4. Functions

ये किसी object को input लेकर दूसरा object output में देते हैं।
Examples:

- FatherOf(x) → "x का पिता कौन है"
- Height(x) → "x की height"

5. Quantifiers

FOPL में दो main quantifiers होते हैं:

- **Universal Quantifier (∀)** → "for all"
- **Existential Quantifier (∃)** → "there exists"

इन quantifiers की वजह से हम ऐसे statements लिख सकते हैं जो पूरी class पर लागू होते हैं, न कि केवल किसी एक specific object पर।

FOPL की AI में ज़रूरत

AI में FOPL का उपयोग किया जाता है क्योंकि यह machines को:

✓ complex real-world facts समझने में
✓ logical reasoning करने में
✓ inference generate करने में
✓ expert systems और knowledge bases बनाने में

मदद देता है।

Propositional logic सिर्फ simple facts बताती है, जबकि FOPL general rules, relations, constraints, और object-based knowledge represent कर सकता है।

Examples of FOPL

1. Universal Rule Example

English Meaning: "All humans are mortal."
FOPL:

∀x (Human(x) → Mortal(x))

2. Existential Example

English Meaning: “There exists a student who studies AI.”
FOPL:

$\exists x$ (Student(x) $\wedge$ Studies(x, AI))

3. Relationship Example

English Meaning: “Ravi is the father of Mohan.”
FOPL:

Father(Ravi, Mohan)

4. Rule-Based Example

English Meaning: “If a person is a mother, then she is female.”
FOPL:

$\forall x$ (Mother(x) $\rightarrow$ Female(x))

5. Function Example

English Meaning: “The mother of Rhea is Sita.”
FOPL:

MotherOf(Rhea) = Sita

Conversion to Clausal Form in AI

Conversion to Clausal Form is a systematic, multi-step process used in predicate logic (especially First-Order Logic) to transform complex logical expressions into a standardized format called **clausal form** or **conjunctive normal form for clauses**. Clausal form represents a formula as a **set of disjunctions (ORs) of literals**, where each clause contains only literals (a literal is either an atomic proposition or its negation). This format is essential for automated reasoning methods such as **Resolution**, **Theorem Proving**, **Refutation**, and **Logic Programming**.

In AI, clausal form is extremely important because it provides a **uniform, machine-readable, and computation-friendly format** that allows inference algorithms to operate in a simple and standardized way. The transformation preserves the logical meaning (i.e., the formula remains

logically equivalent) while making the structure compatible with the resolution principle.

Key Characteristics of Clausal Form

A formula is in **clausal form** if:

1. It is a conjunction (AND) of clauses.
2. Each clause is a disjunction (OR) of literals.
3. Every quantifier is removed by Skolemization.
4. No implications or equivalence signs are present.
5. Negations appear only in front of atomic propositions (i.e., in Negation Normal Form).
6. The entire formula is represented as a **set of clauses**.

Steps for Converting to Clausal Form (Detailed)

The standard steps are:

Step 1: Eliminate Implications (→) and Biconditionals (↔)

Replace:

- $P \rightarrow Q$with $\neg P \vee Q$
- $P \leftrightarrow Q$with $(P \rightarrow Q) \wedge (Q \rightarrow P)$

Step 2: Move NOT (¬) Inward using De-Morgan's Laws

Using rules:

- ¬∀x P becomes ∃x ¬P
- ¬∃x P becomes ∀x ¬P
- ¬(P ∧ Q) becomes ¬P ∨ ¬Q

- ¬(P ∨ Q) becomes ¬P ∧ ¬Q

This step yields **Negation Normal Form (NNF)**.

Step 3: Standardize Variables

Rename variables so that no two quantifiers use the same variable.

Example:

∀x P(x) ∧ ∀x Q(x)

becomes

∀x P(x) ∧ ∀y Q(y)

Step 4: Skolemization (Remove Existential Quantifiers)

Convert existential quantifiers (∃) into Skolem constants or Skolem functions.

- If ∃x does **not** depend on a universal quantifier → use a **Skolem constant**
- If ∃x depends on universal variables → use a **Skolem function**

Example:

∀x ∃y Loves(x, y)

Skolemized form:

∀x Loves(x, f(x))

Step 5: Drop Universal Quantifiers

After Skolemization, only ∀ quantifiers remain; they can be omitted because clauses assume all variables are universally quantified.

Step 6: Distribute ∧ and ∨ (Convert to Conjunctive Normal Form)

Bring formula into **CNF**:

- Apply distribution:
  $P \vee (Q \wedge R)$becomes $(P \vee Q) \wedge (P \vee R)$

Step 7: Split into Clauses

Break the CNF formula into a **set of clauses**, each clause a disjunction of literals.

Complete Worked Example

Convert the following to clausal form:

Given:

∀x (P(x) → ∃y Q(x, y))

Step 1: Remove implication

∀x (¬P(x) ∨ ∃y Q(x, y))

Step 2: Negation is already inside

No change needed.

Step 3: Standardize variables

Already standard.

Step 4: Skolemization

Convert ∃y to a Skolem function f(x):

∀x (¬P(x) ∨ Q(x, f(x)))

Step 5: Drop universal quantifier

¬P(x) ∨ Q(x, f(x))

Step 6: Already in CNF

Step 7: Write as a clause

Final Clausal Form:

{ ¬P(x) , Q(x, f(x)) }

Inference Rules in AI

Inference rules in Artificial Intelligence are formal, logically valid procedures that specify how new knowledge (conclusions) can be derived from existing knowledge (premises). These rules define the *legal steps* a reasoning system can use to infer new statements that are guaranteed to be logically true if the original statements are true. In other words, inference rules allow AI systems to **perform logical reasoning**, **automatically derive new facts**, and **extend knowledge bases** in a sound and consistent manner.

Inference rules are foundational to many AI systems, including:

- Expert systems
- Knowledge-based systems
- Automated theorem provers
- Logic programming languages (e.g., Prolog)
- Deductive databases
- Semantic reasoning systems

These rules ensure that an AI system derives only **logically valid** conclusions and avoids contradictions.

Why Inference Rules Are Important in AI

AI must often make conclusions without being explicitly told every fact. Inference rules allow:

✓ Automated reasoning
✓ Deduction from incomplete information
✓ Checking consistency of knowledge bases
✓ Generating new knowledge from old
✓ Proving theorems
✓ Making decisions based on logical outcomes

They are the mathematical backbone of **intelligent behavior**.

Main Types of Inference Rules in AI

1. Modus Ponens (Most Common Rule)

Definition: If "P implies Q" is true, and P is true, then Q must be true.
Formula:

P → Q

P

———————

∴ Q

Example:
If "It is raining → Road is wet"
and "It is raining"
⇒ "Road is wet."

2. Modus Tollens

If P implies Q, and Q is false (¬Q), then P must also be false.
Formula:

P → Q

¬Q

———————

∴ ¬P

3. Hypothetical Syllogism

Combines two implications.

P → Q

Q → R

———————

∴ P → R

4. Disjunctive Syllogism

If we know P ∨ Q is true, and P is false, then Q must be true.

P ∨ Q

¬P

———————

∴ Q

5. Addition Rule

P

———————

∴ P ∨ Q

6. Simplification Rule

$P \land Q$

———————

$\therefore P$

7. Conjunction Rule

P

Q

———————

$\therefore P \land Q$

8. Resolution Rule (MOST IMPORTANT in AI)

This is the main inference rule used in **automated theorem proving**, **predicate logic**, and **Prolog**.

Definition:
If a clause contains a literal and another clause contains its negation, they can be resolved to form a new clause.

Formula (Propositional):

$(P \lor A), (\neg A \lor Q)$

———————————————

$P \lor Q$

Example:

$(\neg Rain \lor Wet)$

(Rain)

———————————

Wet

Importance: Resolution is complete and can derive all possible logical conclusions from clausal form.

9. Generalized Resolution (Predicate Logic)

Used for First-Order Logic. It includes:

- Unification
- Substitution
- Variable standardization

Example:

¬P(x) ∨ Q(x)

P(a)

———————

Q(a)

10. Universal Instantiation

From a universal rule, you infer a specific case.

∀x P(x)

———————

P(a)

11. Existential Instantiation

If something exists, assume a new constant represents it.

∃x P(x)

———————

P(c)

Long Conceptual Explanation

Inference rules operate on the principle of **logical entailment**, which means:

A conclusion C is entailed by premises P if and only if C is true in every situation where P is true.

Inference rules do **not** invent new knowledge; they derive knowledge that is *implicitly contained* in the original facts. By applying inference rules repeatedly, an AI system builds more complex conclusions, performs reasoning, and solves logical problems.

Inference rules must be:

1. Sound

They must never derive a false conclusion from true premises.

2. Complete

They should be strong enough to derive every conclusion that logically follows.

Resolution is both **sound and complete**, which is why it is heavily used in AI.

Example: Using Multiple Inference Rules

Given:

1. ∀x (Human(x) → Mortal(x))
2. Human(Socrates)

Step 1: Universal Instantiation

Human(Socrates) → Mortal(Socrates)

Step 2: Modus Ponens
Since Human(Socrates) is true:

Mortal(Socrates)

Thus, the system infers Socrates is mortal.

Inference Rules in AI

Inference rules in Artificial Intelligence are formal, logically valid procedures that specify how new knowledge (conclusions) can be derived from existing knowledge (premises). These rules define the *legal steps* a reasoning system can use to infer new statements that are guaranteed to be logically true if the original statements are true. In other words, inference rules allow AI systems to **perform logical reasoning**, **automatically derive new facts**, and **extend knowledge bases** in a sound and consistent manner.

Inference rules are foundational to many AI systems, including:

- Expert systems
- Knowledge-based systems
- Automated theorem provers
- Logic programming languages (e.g., Prolog)
- Deductive databases
- Semantic reasoning systems

These rules ensure that an AI system derives only **logically valid** conclusions and avoids contradictions.

Why Inference Rules Are Important in AI

AI must often make conclusions without being explicitly told every fact. Inference rules allow:

✓ Automated reasoning
✓ Deduction from incomplete information
✓ Checking consistency of knowledge bases

✓ Generating new knowledge from old

✓ Proving theorems

✓ Making decisions based on logical outcomes

They are the mathematical backbone of **intelligent behavior**.

Main Types of Inference Rules in AI

1. Modus Ponens (Most Common Rule)

Definition: If "P implies Q" is true, and P is true, then Q must be true.
Formula:

P → Q

P

———————

∴ Q

Example:
If "It is raining → Road is wet"
and "It is raining"
⇒ "Road is wet."

2. Modus Tollens

If P implies Q, and Q is false (¬Q), then P must also be false.
Formula:

P → Q

¬Q

———————

∴ ¬P

3. Hypothetical Syllogism

Combines two implications.

P → Q

Q → R

———————

∴ P → R

4. Disjunctive Syllogism

If we know P ∨ Q is true, and P is false, then Q must be true.

P ∨ Q

¬P

———————

∴ Q

5. Addition Rule

P

———————

∴ P ∨ Q

6. Simplification Rule

P ∧ Q

———————

∴ P

7. Conjunction Rule

P

Q

———————

∴ P ∧ Q

8. Resolution Rule (MOST IMPORTANT in AI)

This is the main inference rule used in **automated theorem proving**, **predicate logic**, and **Prolog**.

Definition:
If a clause contains a literal and another clause contains its negation, they can be resolved to form a new clause.

Formula (Propositional):

(P ∨ A), (¬A ∨ Q)

———————————————

P ∨ Q

Example:

(¬Rain ∨ Wet)

(Rain)

———————————

Wet

Importance: Resolution is complete and can derive all possible logical conclusions from clausal form.

9. Generalized Resolution (Predicate Logic)

Used for First-Order Logic. It includes:

- Unification
- Substitution
- Variable standardization

Example:

¬P(x) ∨ Q(x)

P(a)

———————————

Q(a)

10. Universal Instantiation

From a universal rule, you infer a specific case.

∀x P(x)

———————

P(a)

11. Existential Instantiation

If something exists, assume a new constant represents it.

∃x P(x)

———————

P(c)

Inference rules operate on the principle of **logical entailment**, which means:

A conclusion C is entailed by premises P if and only if C is true in every situation where P is true.

Inference rules do **not** invent new knowledge; they derive knowledge that is *implicitly contained* in the original facts. By applying inference rules repeatedly, an AI system builds more complex conclusions, performs reasoning, and solves logical problems.

Inference rules must be:

1. Sound

They must never derive a false conclusion from true premises.

2. Complete

They should be strong enough to derive every conclusion that logically follows.

Resolution is both **sound and complete**, which is why it is heavily used in AI.

Example: Using Multiple Inference Rules

Given:

1. ∀x (Human(x) → Mortal(x))
2. Human(Socrates)

Step 1: Universal Instantiation

Human(Socrates) → Mortal(Socrates)

Step 2: Modus Ponens
Since Human(Socrates) is true:

Mortal(Socrates)

Thus, the system infers Socrates is mortal.

Resolution Principle in AI

The **Resolution Principle** is a fundamental inference rule used in **Automated Reasoning**, **Predicate Logic**, and especially in **Propositional Logic** and **First-Order Logic (FOL)** to derive conclusions from a set of statements. It was introduced by **Alan Robinson in 1965** and is the basis for many theorem provers in AI.

Resolution works by converting all statements into a standardized format called **Clausal Form (CNF)**—a conjunction of disjunctions—and then applying a **single inference rule** that eliminates complementary literals to produce a new clause.
Complementary literals are pairs like **P** and **¬P**, or **likes(John, X)** and **¬likes(John, X)**.

The idea behind the resolution principle is that:

If two clauses contain complementary literals, they can be resolved to produce a new clause that represents the logical consequence of the two.

The resolution principle is particularly powerful because it is both:

- **sound** (every derived conclusion is logically valid), and
- **complete** (all valid conclusions can be derived using resolution).

This makes it a preferred method for automated theorem proving, expert systems, and logic programming frameworks like **Prolog**, which internally uses a form of resolution called **SLD-resolution**.

Resolution is mainly used for **proof by contradiction**. To prove a conclusion, we add its **negation** to the knowledge base and apply resolution. If we derive an **empty clause (□)**, it means a contradiction has occurred, and the original conclusion must be true.

Steps of Resolution Principle

1. **Convert all statements into CNF (Clausal Normal Form).**
2. **Identify clauses containing complementary literals.**
3. **Apply the resolution rule** to eliminate complementary pairs.
4. **Generate new clauses** from the remaining literals.
5. **Repeat** until either:
    - The **empty clause (□)** is derived → proof successful
    - No new clause can be generated → proof fails

General Resolution Rule

Given two clauses:

- **(A ∨ P)**
- **(B ∨ ¬P)**

Resolution produces a new clause:

- **(A ∨ B)**

(P and ¬P cancel out)

Example of Resolution Principle

Problem:

Given the statements:

1. **If it is raining, the ground is wet.**
   $$Raining \rightarrow Wet$$
2. **It is raining.**
   $$Raining$$
   Conclusion: The ground is wet.

Step 1: Convert statements into CNF

1. $Raining \rightarrow Wet$
   becomes
   $$\neg Raining \vee Wet$$
2. $Raining$is already a clause.

Step 2: Apply Resolution

Clauses:

- $\neg Raining \vee Wet$
- $Raining$

Complementary literals:

- $Raining$and $\neg Raining$

Step 3: Resolve

New clause:

- Wet

Thus, using resolution, we conclude:

✓ The ground is wet.

Why Resolution Is Important in AI

- Used heavily in **expert systems**, **logic-based agents**, and **theorem provers**
- Works with **FOPL**, making it more powerful than simple propositional logic
- Forms the basis of **logic programming (Prolog)**
- Provides a **unified framework** for automated reasoning
- Enables **efficient contradiction-based proof** mechanisms