UNIT NO:4 KNOWLEDGE AND REASONING

DEFINITION OF KNOWLEDGE

- Knowledge in AI refers to the **comprehensive dataset** and **cognitive frameworks** that AI systems utilize **to process information, learn from experiences, and make informed decisions.**
- This Knowledge can be derived from various sources, such as databases, human expertise, and real-world interactions.
- Knowledge in AI refers to any **data or information that an AI system can store, process, and apply to perform intelligent actions.**
- This isn't just raw data; it's data that has been encoded, organized, and structured in a machine-readable format, making it usable for reasoning and problem-solving.

DEFINITION OF KNOWLEDGE

- **Critical Components of AI Knowledge**
- **1. Data and Information:** Raw data are collected from various sources and then processed to extract useful information.
- **2. Concepts and Relationships:** Fundamental ideas and their connections that help AI understand complex scenarios.
- **3. Rules and Inferences:** Logical rules and the ability to infer new information from existing knowledge.

IMPORTANCE OF KNOWLEDGE IN AI

- The primary goal is to equip AI systems with the capability to mimic human understanding and reasoning to a certain extent.
- **1. Enhanced Decision-Making:** AI systems with robust Knowledge bases can make more accurate and reliable decisions. For instance, AI can assist doctors in healthcare by providing evidence-based recommendations.
- **2. Improved Learning Capabilities:** The more knowledge an AI system has, the better it can learn and adapt. This is crucial for developing advanced AI applications that require continuous learning and improvement.

IMPORTANCE OF KNOWLEDGE IN AI

- **3. Efficiency and Automation:** Knowledge enables AI to automate complex tasks efficiently. For example, AI can handle routine customer service inquiries, freeing human agents to tackle more complex issues.
- **4. Personalization:** AI systems use knowledge to provide personalized experiences. In e-commerce, AI can recommend products based on user preferences and past behavior.
- **5. Innovation:** Knowledge fuels innovation in AI by enabling the development of new applications and solutions across various industries, from finance to entertainment.

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.*

TYPES OF KNOWLEDGE IN AI

- **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*

TYPES OF KNOWLEDGE IN AI

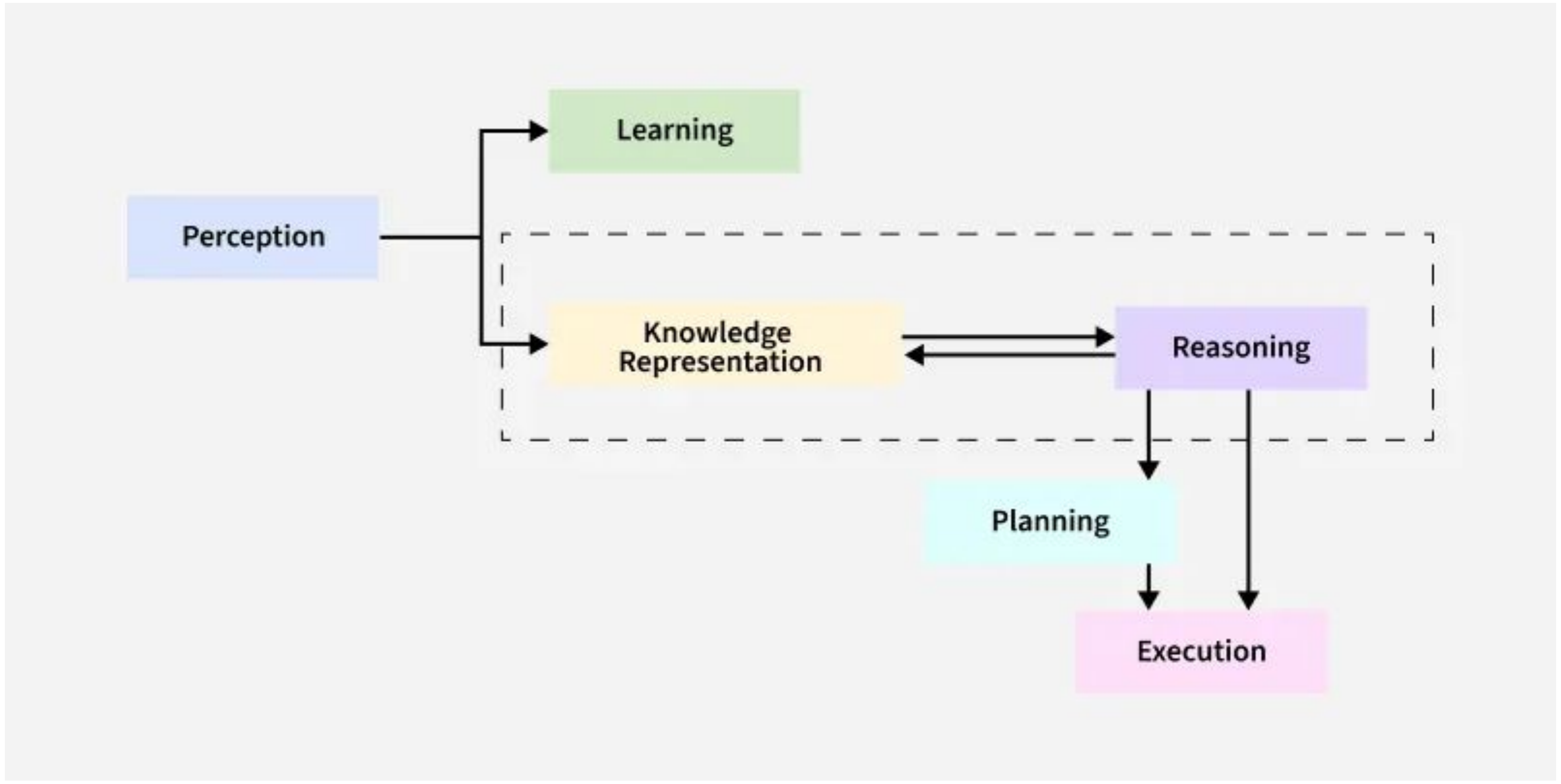
- **4. Heuristic Knowledge (Experience-Based Knowledge)**
- Heuristic knowledge is derived from experience, intuition, and trial-and-error methods. It allows AI systems to make educated guesses or approximate solutions when exact answers are difficult to compute.
- *For example, a navigation system suggesting 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**."

TYPES OF KNOWLEDGE IN AI

- **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.

KNOWLEDGE REPRESENTATION IN AI

- knowledge representation (KR) in AI refers to **encoding information about the world into formats that AI systems can utilize to solve complex tasks.** This process enables machines to reason, learn, and make decisions by structuring data in a way that mirrors human understanding.



KNOWLEDGE REPRESENTATION IN AI

- **Perception:** AI takes in information from its surroundings, like listening, seeing, or reading. This helps it understand the world. For example, it listens to spoken words, sees images, and reads text to gather knowledge about its environment.
- **Learning:** AI uses deep learning algorithms to study and remember what it perceives. It’s like taking notes to get better at something. Through learning, AI becomes skilled at recognizing patterns and making predictions based on its experiences.
- **Knowledge and Reasoning:** These parts are like AI’s brain. They help it understand and think smartly. They find important information for AI to learn. AI’s knowledge and reasoning components sift through its data to identify valuable insights, allowing it to make informed decisions.
- **Planning and Doing:** AI uses what it learned to make plans and take action. It’s like using knowledge to make good decisions. With its plans in place, AI carries out tasks efficiently and adapts to changes in its environment, demonstrating intelligent behavior.

KNOWLEDGE REPRESENTATION IN AI

- **1. Logic-Based Systems**
- Logic-based methods use formal rules to model knowledge. These systems prioritize precision and are ideal for deterministic environments.
- Formal logic in AI is usually expressed as **Propositional Logic** or **First-Order Logic (FOL)**.
- **Propositional Logic**
- It Deals with statements that are either **true (T)** or **false (F)**.
- Represents knowledge as declarative statements (propositions) linked by logical operators like AND, OR, and NOT. For example, "If it rains (A) AND the ground is wet (B), THEN the road is slippery (C)."
- **First-Order Logic (FOL)**
  Extends propositional logic by introducing variables, quantifiers, and predicates.

KNOWLEDGE REPRESENTATION IN AI

- **2. Structured Representations**
- These are ways to organize and represent knowledge so machines can reason more like humans.
- **Semantic Networks**
  Represent knowledge as nodes (concepts) and edges (relationships). For example, "Dog" links to "Animal" via an "Is-A" connection. They simplify inheritance reasoning but lack formal semantics.
- **Frames**
  Group related attributes into structured "frames." A "Vehicle" frame may include slots like wheels, engine type, and fuel. Frames excel in default reasoning but struggle with exceptions.
- **Ontologies**
  Define concepts, hierarchies, and relationships within a domain using standards like OWL (Web Ontology Language). Ontologies power semantic search engines and healthcare diagnostics by standardizing terminology.
- E-commerce platforms use ontologies to classify products and enhance search accuracy.

KNOWLEDGE REPRESENTATION IN AI

3. Probabilistic Models

- These systems handle uncertainty by assigning probabilities to outcomes.
- **Bayesian Networks**
  - **What it is:** A **directed graph** where nodes are variables and edges show causal dependencies.
  - **Example:** Predicting equipment failure.
    - Node: *Maintenance History*
    - Node: *Usage Level*
    - Node: *Failure Probability*
  - **Strength:** Allows reasoning under uncertainty (e.g., "Given poor maintenance, the chance of failure is 80%").
- **Markov Decision Processes(MDPS)**
  Model sequential decision-making in dynamic environments. MDPs help robotics systems navigate obstacles by evaluating potential actions and rewards.
- Weather prediction systems combine historical data and sensor inputs using probabilistic models to forecast storms

KNOWLEDGE REPRESENTATION IN AI

3. Probabilistic Models

- **Markov Decision Processes(MDPS)**
  - **What it is:** A mathematical framework for **sequential decision-making**.
  - **Example:** A robot navigating obstacles.
    - The robot considers states (its position), actions (move forward, turn), and rewards (safe path vs. collision).
  - **Strength:** Helps choose optimal actions in **dynamic environments**.

- Real-World Example
  - **Weather Prediction:**
    - Uses probabilistic models to combine historical data + sensor inputs.
    - Forecasts like “70% chance of storm tomorrow” are based on Bayesian-like reasoning.

KNOWLEDGE REPRESENTATION IN AI

4.Distributed Representations

- Modern AI leverages neural networks to encode knowledge as numerical vectors, capturing latent patterns in data.
- **Embeddings**
- **What it is:** Convert words, images, or entities into **dense numerical vectors**.
  Capture hidden patterns and similarities
- **Example:**
- *Word2Vec / GloVe* → "king" - "man" + "woman" ≈ "queen".
- Similar words (synonyms) are placed close together in vector space.
- **Strength:** Captures **semantic meaning** (e.g., “dog” and “puppy” are close in embedding space).
- **Use Cases:** Search engines, chatbots, recommendation systems, image recognition.

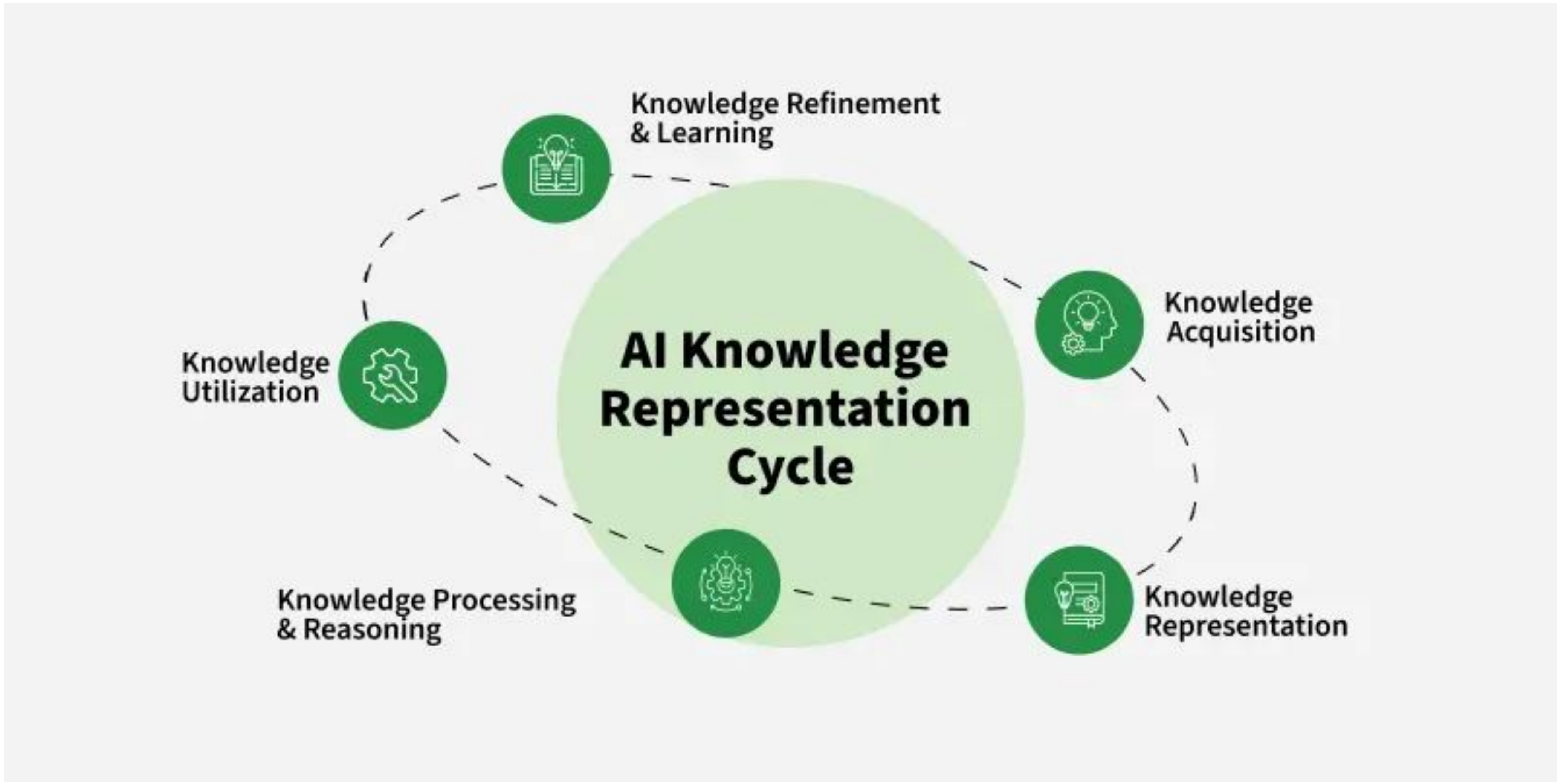
KNOWLEDGE REPRESENTATION IN AI

4.Distributed Representations

- Knowledge Graphs
  - **What it is:** A graph structure + embeddings to represent **entities and relationships**.
  - Organize facts + relationships, powered by embeddings.
  - **Example:**
    - *Google Knowledge Graph* links “Barack Obama → Born In → Hawaii”.
    - Entities (people, places, things) are **nodes**, relationships are **edges**.
  - **Strength:** Makes AI more context-aware.
  - **Use Cases:** Google Search results, Siri, Alexa, healthcare diagnostics.
- Together, they allow AI to **understand meaning**, not just raw data.

AI KNOWLEDGE CYCLE

- The AI Knowledge Cycle represents the continuous process through which AI systems acquire, process, utilize, and refine knowledge.


Knowledge Refinement & Learning
Knowledge Acquisition
AI Knowledge Representation Cycle
Knowledge Utilization
Knowledge Processing & Reasoning
Knowledge Representation

AI KNOWLEDGE CYCLE

- This cycle ensures that AI remains adaptive and improves over time.
- **1. Knowledge Acquisition**: AI gathers data from various sources, including structured databases, unstructured text, images, and real-world interactions. Techniques such as machine learning, natural language processing (NLP), and computer vision enable this acquisition.
- **2. Knowledge Representation** : Once acquired, knowledge must be structured for efficient storage and retrieval. Represented through methods explained above:
- **3. Knowledge Processing & Reasoning**: AI applies logical inference, probabilistic models, and deep learning to process knowledge. This step allows AI to:
  - Draw conclusions (deductive and inductive reasoning)
  - Solve problems using heuristic search and optimization
  - Adapt through reinforcement learning and experience

AI KNOWLEDGE CYCLE

- **4. Knowledge Utilization**: AI applies knowledge to real-world tasks, including decision-making, predictions, and automation. Examples include:
  - Virtual assistants understanding user queries
  - AI-powered recommendation systems suggesting content
  - Self-driving cars making real-time navigation decisions
- **5. Knowledge Refinement & Learning**: AI continuously updates its knowledge base through feedback loops. Techniques like reinforcement learning, supervised fine-tuning, and active learning help improve accuracy and adaptability. This ensures AI evolves based on new data and experiences.

CHALLENGES IN KNOWLEDGE REPRESENTATION

- While knowledge representation is fundamental to AI, it comes with several challenges:
- **Complexity**: Representing all possible knowledge about a domain can be highly complex, requiring sophisticated methods to manage and process this information efficiently.
- **Ambiguity and Vagueness**: Human language and concepts are often ambiguous or vague, making it difficult to create precise representations.
- **Scalability**: As the amount of knowledge grows, AI systems must scale accordingly, which can be challenging both in terms of storage and processing power.
- **Knowledge Acquisition**: Gathering and encoding knowledge into a machine-readable format is a significant hurdle, particularly in dynamic or specialized domains.
- **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.

PROPERTIES OF KNOWLEDGE REPRESENTATION SYSTEMS

- A good knowledge representation (KR) system possesses four key properties:
- **Representational Adequacy** (the ability to represent all required knowledge),
- **Inferential Adequacy** (the ability to derive new knowledge from existing information),
- **Inferential Efficiency** (the speed and efficiency of deriving conclusions),
- **Acquisitional Efficiency** (the ease of adding new knowledge to the system)

PROPERTIES OF KNOWLEDGE REPRESENTATION SYSTEMS

1. **Representational Adequacy**
   - The system must be **able to represent all relevant knowledge about the domain effectively.** This includes facts, relationships, and rules that are necessary for reasoning and decision-making.
   - **Example**: In a medical diagnosis AI, the system must represent knowledge about symptoms, diseases, and treatments in a way that allows for accurate diagnosis.
   - **Challenge**: Ensuring that the system can accommodate the vast and diverse knowledge of a domain without becoming too complex.

2. **Inferential Adequacy**
   - **Description**: The system should be **able to generate new knowledge by applying inference mechanisms, such as deduction, induction, or abduction, to the represented knowledge.**
   - **Example**: An AI system might use inferential reasoning to deduce that a person with a high fever and sore throat likely has the flu based on existing rules and facts.
   - **Challenge**: Developing efficient algorithms that can handle large datasets while providing fast and accurate inferences.

PROPERTIES OF KNOWLEDGE REPRESENTATION SYSTEMS

- **3. Inferential Efficiency**
- **Description**: The system must be capable of making inferences quickly and efficiently. It should use resources such as memory and processing power in an optimal manner.
- **Example**: A financial AI needs to infer stock market trends in real time to make investment decisions, requiring both speed and accuracy in its inferences.
- **Challenge**: Balancing the complexity of inferences with the need for rapid decision-making, especially in time-sensitive applications.
- **4. Acquisitional Efficiency**
- **Description**: The system must be able to easily acquire and integrate new knowledge as it becomes available. This includes updating existing knowledge structures and incorporating new data without disrupting the system.
- **Example**: An AI system designed for customer service should be able to incorporate new FAQs and policies dynamically as the business evolves.
- **Challenge**: Ensuring that the system remains scalable and adaptive as new knowledge is continuously added over time.

PROPERTIES OF KNOWLEDGE REPRESENTATION SYSTEMS

Consistency

- **Description**: The knowledge representation system must maintain consistency across all facts, rules, and relationships. Inconsistent or contradictory information can lead to incorrect reasoning and unreliable results.
- **Example**: In an AI system for legal reasoning, if the system contains contradictory laws or regulations, it may struggle to provide accurate legal advice.
- **Challenge**: Ensuring that

PROPOSITIONAL LOGIC (PL)

(1)Translate the following Propositional Logic to English sentences.

- Let: • E=Aditya is eating • H= Aditya is hungry
  - (a) $E \Rightarrow \neg H$
  - (b) $E \wedge \neg H$

(2) Translate the following English sentences to Propositional Logic.

Propositions: (R)aining, Sara is (S)ick, Sara is (H)ungry, Sara is (HA)appy, Sara owns a (C)at, Liron owns a (D)og

(a) It is raining if and only if Sara is sick

(b) If Sara is sick then it is raining, and vice versa

(c) It is raining is equivalent to Sara is sick

(d) Sara is hungry but happy

(e) Sara either owns a cat or a dog

PREDICATE LOGIC IN AI

- In Artificial Intelligence **(AI)**, reasoning plays a crucial role in building systems that can **make decisions** and **infer knowledge** based on facts and conditions.
- However, **propositional logic** is limited in its ability to represent **complex relationships** or **detailed information**.
- **Predicate logic**, also known as **first-order logic (FOL)**, extends propositional logic by allowing AI systems to **represent relationships between objects** and their **properties**.
- This makes predicate logic a powerful tool for **knowledge representation and reasoning**.
- It enables AI systems to understand relationships like "John is the father of Mary" or "All humans are mortal."

PREDICATE LOGIC IN AI

- Unlike propositional logic, which deals with simple true/false statements, predicate logic introduces **predicates, variables, constants, and quantifiers**.
- These elements help in modeling real-world problems that involve **multiple objects** and their **interactions**.
- **Role of Predicate Logic in AI**
- **Knowledge Representation:** It provides a structure for representing complex facts about objects and their relationships in a system.
- **Reasoning:** AI systems use predicate logic to **infer new information** from existing facts, making it suitable for **decision-making tasks**.
- **Example:**
- "John is the father of Mary" can be represented as: Father(John,Mary)

COMPONENTS OF PREDICATE LOGIC

- Predicate logic involves several key components that allow it to **represent relationships** and **properties** of objects in a structured way.

1. Predicates:

- A **predicate** is a function that returns either **true** or **false** based on the relationship between its arguments.
- Example: IsHungry(John)IsHungry(John)IsHungry(John) This predicate represents whether John is hungry, returning true if he is and false if not.

COMPONENTS OF PREDICATE LOGIC

2. Variables:

- **Variables** are placeholders for objects within a domain. They allow us to represent general statements that apply to multiple objects.
- Example: In **IsHungry(x)**, the variable **x** can represent any person.

3. Constants:

- **Constants** represent specific objects or entities in the domain.
- Example: **John** is a constant in the predicate **IsHungry(John)**.

STRUCTURE OF PREDICATES

- Predicate consists of two key elements: the **predicate symbol** and **arguments**. Predicates are enhanced with **quantifiers** to specify the **scope** of variables involved.

1. Predicate Symbol

- The **predicate symbol** defines the **property** or **relationship** being described.
- Example:
  - **IsHungry(x)** represents whether a person (x) is hungry.
  - **Married(x, y)** denotes that person **x** is married to person **y**.
- Predicates are named based on the relationship or property they represent. The symbol is followed by **arguments** enclosed in parentheses.

STRUCTURE OF PREDICATES

2. Arguments and Arity

- **Arguments** refer to the specific **objects** that the predicate is applied to.
- The **arity** of a predicate refers to the **number of arguments** it takes.
- Examples:
- **IsHungry(x)**: A predicate with **1 argument** (arity = 1).
- **Married(x, y)**: A predicate with **2 arguments** (arity = 2).
- **X(a, b, c)**: A predicate with **3 arguments** (arity = 3), representing something like "a + b + c = 0."

STRUCTURE OF PREDICATES

- **3. Quantifiers in Predicate Logic**
- Quantifiers allow us to **specify the scope** of variables. There are two main types:
- **Existential Quantifier (∃)**
  - **Meaning:** There exists at least **one object** that satisfies the given condition.
  - **Example:** ∃x IsHungry(x)
    This statement means that **at least one person** is hungry.
  - **Negation:** The negation of the existential quantifier means that no such object exists. ¬∃x IsHungry(x)
    This means that **no one is hungry**.
- **Universal Quantifier (∀)**
  - **Meaning:** The given condition holds **for all objects** in the domain.
  - **Example:** ∀x (IsHuman(x)→IsMortal(x))
    This means that **all humans are mortal**.
  - **Negation:** The negation of the universal quantifier means there is **at least one exception**.
    ¬∀x Is Human(x)→IsMortal(x)
    This implies that **at least one human is not mortal**.

EXAMPLES OF PREDICATE LOGIC

1.**Simple Predicate**

- **Predicate: IsHungry(John)**
  - **Meaning:** This predicate represents the **state** of whether John is hungry.
  - It takes one argument (John) and returns **true** if John is hungry, otherwise false.
- **Application in AI:**
  - In **NLP-based chatbots**, predicates like this could help infer the user’s intent.
  - For example, if a chatbot detects that the user is hungry, it could suggest nearby restaurants.

EXAMPLES OF PREDICATE LOGIC

2. Equality Predicate Example:

- **Predicate: E(x,y)≡(x=y)**
  - **Meaning:** This predicate denotes that **x is equal to y**.
  - It returns **true** if the two objects are identical.
- **Application in AI:**
  - **AI-based reasoning systems** use equality predicates to match objects.
  - For example, in a **robot warehouse**, a robot may use this predicate to determine if an object picked matches the one requested (e.g., E(Package1, RequestedItem)).

EXAMPLES OF PREDICATE LOGIC

3. Mathematical Predicate Example:

- **Predicate: $X(a,b,c) \equiv (a+b+c=0)$**
  - **Meaning:** This predicate checks whether the sum of **a, b,** and **c** equals zero.
  - It returns **true** if the equation holds, otherwise **false**.
- **Application in AI:**
  - In **optimization problems**, AI models might use mathematical predicates to check if constraints are satisfied.
  - For example, in **scheduling systems**, such predicates can validate if certain conditions are met (e.g., X(shiftA, shiftB, totalTime) checks if the total shift hours are balanced).

EXAMPLES OF PREDICATE LOGIC

4. Relationship Predicate Example:

- **Predicate: F(x,y)≡x is Father of y**
  - **Meaning:** This predicate expresses a **relationship** between two objects, indicating that **x** is father of **y**.
- **Application in AI:**
  - In **family tree AI systems**, relationship predicates are used to **infer relationships** among family members.
  - For instance, if F**(John, Mary)** is true, the system can infer that John is Mary's Father.

EXAMPLES OF PREDICATE LOGIC

5. Universal Quantification Example:

- **Expression: $\forall x\ (IsHuman(x) \rightarrow IsMortal(x))$**
  - **Meaning:** This statement reads as "**For all x, if x is human, then x is mortal.**"
  - It applies to every object in the domain of humans.
  - If an object is found to be human, it must also be mortal for the statement to hold true.
- **Application in AI:**
  - **Knowledge-based systems** use such rules to infer properties about objects.
  - For example, in **medical diagnosis systems**, rules like "All viruses can spread infections"
  - ($\forall x\ IsVirus(x) \rightarrow CanSpreadInfection(x)$) help the system reason about diseases.

EXAMPLES OF PREDICATE LOGIC

6. Existential Quantification Example:

- **Expression: ∃x IsHungry(x)**
  - **Meaning:** This reads as "**There exists at least one x such that x is hungry.**"
  - It indicates that **at least one object** in the domain satisfies the condition of being hungry.
- **Application in AI:**
  - In **robot planning**, a robot could use existential quantifiers to plan actions.
  - For example, "There exists a task that requires charging"
  - (∃x TaskRequires(x, Charging)) might guide the robot to prioritize charging tasks.

EXAMPLES OF PREDICATE LOGIC

7. Compound Example with Multiple Quantifiers:

- **Expression: $\forall x \exists y$ (Parent(x,y))**
  - **Meaning:** This statement means "**For every person x, there exists a person y such that x is the parent of y.**" It shows how multiple quantifiers can be used together to represent complex relationships.
- **Application in AI:**
  - This logic is often used in **social network AI models** to analyze relationships.
  - In a **family tree system**, the model could use such logic to infer relationships between family members.

PROPOSITIONS WITH MULTIPLE QUANTIFIERS

- In predicate logic, **multiple quantifiers** can be used within a single proposition to express more complex ideas.
- The **order of quantifiers** is crucial, as it can **change the meaning** of the statement.
- **Example 1: Order of Quantifiers Matters**
- **$\forall x \exists y$ (Parent(x,y))**
- **Meaning:** For every person **x**, there exists at least one person **y** such that **x** is the parent of **y**.
- **Example in AI:** In a **family tree system**, this could represent the rule that every parent must have at least one child.

PROPOSITIONS WITH MULTIPLE QUANTIFIERS

- Now, let's reverse the quantifiers:
- **$\exists y \forall x$ (Parent(x,y))**
- **Meaning:** There exists a person **y** such that every person **x** is the parent of **y**.
- **Interpretation:** This is logically impossible under normal circumstances, as a single person cannot have all people as parents.

PROPOSITIONS WITH MULTIPLE QUANTIFIERS

- **Example 2: Nested Quantifiers in AI**
- **$\forall x \exists y$ (RobotCanPerform(x,y))**
- **Meaning:** For every task **x**, there exists a robot **y** that can perform the task.
- **AI Application:** This could represent a rule in a **robot planning system**, where every task must have at least one robot capable of completing it.
- Now, consider the reversed version:
- **$\exists y \forall x$ (RobotCanPerform(x,y))**
- **Meaning:** There exists a robot **y** that can perform every task **x**.
- **AI Application:** This would imply that a single robot can perform all tasks, which may not always be practical.

PROPERTIES OF QUANTIFIERS

- Quantifiers in First Order Logic in AI follow some properties as stated below.
- In the universal quantifier, $\forall x \forall y$ is equivalent to $\forall y \forall x$.
- In the existential quantifier, $\exists x \exists y$ is equivalent to $\exists y \exists x$.
- $\exists x \forall y$ is not equivalent to $\forall y \exists x$.
- **Quantifier Duality:**
  Each quantifier can be expressed using the other.
  E.g., ∀ x Predicate(x) is same as ¬ ∃ x ¬ Predicate(x).

POINTS TO REMEMBER ABOUT QUANTIFIERS

- While using quantifiers in writing expressions for First Order Logic in Artificial Intelligence, we need to keep the following points in mind.
- The main connective for the universal quantifier ∀ is the **implication (⇒)**.
- The main connective for existential quantifier ∃ is **and (∧)**.
- **Free and Bound Variables**
- There are two types of variables based upon their interaction with the quantifiers in a First Order Logic in AI, namely free and bound variables.
- **Free Variables:**
  Free variables are those variables that do not come under the scope of the quantifier. For instance, in an expression $\forall x \exists y P(x, y, z)$,z is a free variable because it doesn't come under the scope of any quantifier.
- **Bound Variables:**
  Bound variables are those variables that occur inside the scope of the quantifier. For instance, in an expression $\forall x \exists y P(x, y, z)$,x *and* y are bound variables because they occur inside the scope of the quantifiers.

POINTS TO REMEMBER ABOUT QUANTIFIERS

- **Use one variable**
- When the statement is about **a property of a single object**.
- **All humans are mortal**
  $\forall x\, (Human(x) \rightarrow Mortal(x))$
- **Use two variables**
- When the statement expresses a **relation between two different objects**.
- **Every student loves some book**
  $\forall x\, (Student(x) \rightarrow \exists y\, (Book(y) \land Loves(x,y)))$
- **Use three or more variables**
- When the relation naturally involves **3+ objects**.
- **"For every doctor, there is a patient and a medicine they prescribe"**
  $\forall d\, (Doctor(d) \rightarrow \exists p\, \exists m\, (Patient(p) \land Medicine(m) \land Prescribes(d,p,m)))$

EXAMPLES TO SOLVE

- **All Birds can fly.**
- **Every student loves some book.**
- **Some dogs are friendly.**
- **If it rains, then the ground is wet.**
- **Everyone who studies hard passes the exam.**
- **There exists a teacher who teaches every student.**
- **Not all birds can fly.**
- **Every parent loves their child.**
- **There exists someone who likes ice cream and pizza.**
- **If a person is hungry, they eat something.**

EXAMPLES TO SOLVE

- Everyone Like Everyone
- All graduates are unemployed
- Every dolphin is Mammal
- No purple mushroom is poisonous.
- Every gardener loves sun.
- You can fool someone all the time.
- All Romans were either loyal to ceaser or hated him
- Every student Smiles
- No one talks
- At least one student failed in History
- Every person who buys an insurance policy is smart
- No Person buys an expensive policy

INFERENCE RULES

- **What is Inference?**
- Inference is the process of logically deriving conclusions from a given set of facts or premises.
- In artificial intelligence (AI), **inference mechanisms enable machines to reason, make decisions, and generate new knowledge based on available data**.
- It is a fundamental aspect of **knowledge representation and automated reasoning**, **allowing AI systems to function intelligently**.
- Rules of inference are standard logical patterns that allow to derive a conclusion from one or more given premises in a logically valid way.

INFERENCE RULES

- There are two primary types of inference:
- **Deductive Inference**
  - This form of reasoning moves from **general rules to specific conclusions**.
  - If the premises are true, the conclusion **must also be true**.
  - **Example:**
    - Premise 1: All humans are happy.
    - Premise 2: John is a human.
    - Conclusion: John is happy.
- **Inductive Inference** –
  - This method involves **drawing general conclusions from specific observations**.
  - Unlike deduction, inductive inference does not guarantee certainty but is useful for AI models that rely on pattern recognition.
  - **Example:** If an AI observes that all previous birds it encountered could fly, it may conclude that all birds can fly, even though exceptions exist.

TYPES OF INFERENCE RULES IN AI

- Inference rules are fundamental to logical reasoning in AI, enabling systems to derive valid conclusions from given premises.
- These rules are widely used in expert systems, knowledge-based AI, and automated decision-making.
- Below are the key types of inference rules used in AI:

1. Modus Ponens(Implication-elimination)

- Modus Ponens is a fundamental rule of inference that follows the **if-then** logic.
- It states:
- If **P** → **Q** (if P implies Q) is true,
- And **P** is true,
- Then **Q must also be true**.

TYPES OF INFERENCE RULES IN AI

2. Modus Tollens

- Modus Tollens is the contrapositive of Modus Ponens and is structured as follows:
- If **P → Q** (if P implies Q) is true, And **Q is false**,
- Then **P must also be false**.
- ***Form:*** *If p → q and ¬q, then ¬p.*
- Example:
- *Premise: If it rains, the ground will be wet.*
- *Premise: The ground is not wet.*
- *Conclusion: It is not raining.*

TYPES OF INFERENCE RULES IN AI

3. Hypothetical Syllogism

- Hypothetical Syllogism, also known as transitive reasoning, follows:
- If **P** → **Q** and **Q** → **R**,Then **P** → **R**.
- ***Form:*** *If p → q and q → r, then p → r.*
- In robotic navigation, AI can apply this logic:
- **Premise 1:** If the robot detects an obstacle, it will stop.
- **Premise 2:** If the robot stops, it will recalculate the path.
- **Conclusion:** If the robot detects an obstacle, it will recalculate the path.
- This rule enables AI planning systems to handle chained decision-making processes.

TYPES OF INFERENCE RULES IN AI

4. Disjunctive Syllogism

- Disjunctive Syllogism allows reasoning through elimination and is represented as:
- If **P** ⋁ **Q** (either P or Q is true),
- And **P is false**, Then **Q must be true**.
- ***Form:*** *If p ⋁ q and ¬p, then q.*
- **Example Use Case in AI**
- In a speech recognition system:
- **Premise:** The AI must identify whether a spoken word is "Hello" or "Help" (Hello ⋁ Help).
- **Observation:** The AI determines it is not "Hello".
- **Conclusion:** The word must be "Help".

TYPES OF INFERENCE RULES IN AI

5. Addition

- The Addition rule allows AI to introduce new possibilities into reasoning:
- If **P is true**,Then **P ∨ Q is also true** (even if Q is unknown).
- **Form:** If p, then p ∨ q
- **Example Scenario in AI Problem-Solving**
- In a smart assistant:
- **Premise:** “The user likes coffee.” (Likes_Coffee)
- **Inference:** The system suggests: “The user likes coffee or tea.” (Likes_Coffee ∨ Likes_Tea)

TYPES OF INFERENCE RULES IN AI

6. Conjunction

- If two statements are true, then their conjunction (an "and" statement) is also true.
- ***Form:*** *If p and q, then p* $\wedge$ *q.*
- **Example:**
- *Premise: It is raining.*
- *Premise: It is windy.*
- *Conclusion: It is raining and windy.*

TYPES OF INFERENCE RULES IN AI

7. Simplification

- Simplification enables AI to extract relevant facts from compound statements:
- If **P ∧ Q** (both P and Q are true),Then **P is true** and **Q is true** separately.
- ***Form:*** *If p ∧ q, then p*
- **Use Case in AI-Driven Decision-Making**
- In a recommendation system:
- **Premise:** "User prefers action and thriller movies" (Action ∧ Thriller).
- **Inference:** The AI can recommend either action or thriller movies individually.
- This rule helps AI break down complex user preferences for more accurate recommendations.

TYPES OF INFERENCE RULES IN AI

8. Resolution

- Resolution is a fundamental inference rule in propositional and first-order logic, often used in automated theorem proving and AI knowledge bases. It states:
- If **P ∨ Q** and **¬Q ∨ R** are true, Then we can conclude **P ∨ R**.
- ***orm:*** *If P ∨Q and ¬P ∨R, then Q ∨R.*
- Resolution is widely used in:
- **Prolog-based AI systems**, where logical clauses are resolved to derive conclusions.
- Example:
- **Premise 1:** “The patient has a fever or a cold” (Fever ∨ Cold).
- **Premise 2:** “The patient does not have a cold or has a flu” (¬Cold ∨ Flu).
- **Inference:** The AI concludes “The patient has a fever or flu” (Fever ∨ Flu).
- This method allows AI to logically resolve uncertainties and derive meaningful insights from incomplete information.

TYPES OF INFERENCE RULES IN AI

9. Absorption(Abs)

- If a conditional statement (an "if-then" statement) is true, then the antecedent implies a conjunction of itself and the consequent.
- ***Form:*** *If $P \rightarrow Q$, then $P \rightarrow (P \wedge Q)$*
- **Example**:
- *Premise: If it is raining, then the ground is wet.*
- *Conclusion: If it is raining, then it is raining **and** the ground is wet.*

Rule of Inference	Form	Tautology	Description
Modus Ponens (MP)	If $p \to q$ and p, then q.	$p \land (p \to q)) \to \neg q$	If P implies Q, and P is true, then Q is true.
Modus Tollens (MT)	If $p \to q$ and $\neg q$, then $\neg p$.	$(\neg q \land (p \to q)) \to \neg p$	If P implies Q, and Q is false, then P is false.
Hypothetical Syllogism (HS)	If $p \to q$ and $q \to r$, then $p \to r$.	$((p \to q) \land (q \to r)) \to (p \to r)$	If P implies Q and Q implies R, then P implies R.
Disjunctive Syllogism (DS)	If $p \lor q$, and $\neg p$, then q.	$(\neg p \land (p \lor q)) \to q$	If P or Q is true, and P is false, then Q is true.
Conjunction (Conj)	If p and q, then $p \land q$.	$(p \land q) \to (p \land q)$ or $p \to (q \to (p \land q))$	If P and Q are true, then P and Q are true.
Simplification (Simp)	If $p \land q$, then p	$(p \land q) \to p$	If P and Q are true, then P is true
Addition (Add)	If p, then $p \lor q$	$p \to (p \lor q)$	If P is true, then P or Q is true.
Absorption(Abs)	If $p \to q$, then $p \to (p \land q)$	$(p \to q) \to (p \to (p \land q))$	If P implies Q, then P implies P or Q is true.
Resolution	If $p \lor q$, and $\neg p \lor r$, then $q \lor r$.	$p \lor q, \neg p \lor r \Rightarrow q \lor r$	If P or Q is true, and not P or R is true, then Q or R is true.

Rule of Inference	Form	Meaning
Universal instantiation	$\forall x P(x) \Rightarrow P(c)$	If something is true for all x, it’s true for a particular case c.
Universal generalization	$P(c) \Rightarrow \forall x P(x)$	If something is true for any arbitrary element, it’s true for all.
Existential instantiation	$\exists x P(x) \Rightarrow P(c)$	If something exists, we can give it a name (c).
Existential generalization	$P(c) \Rightarrow \exists x P(x)$	If something is true for a particular c, it’s true for “some x”.

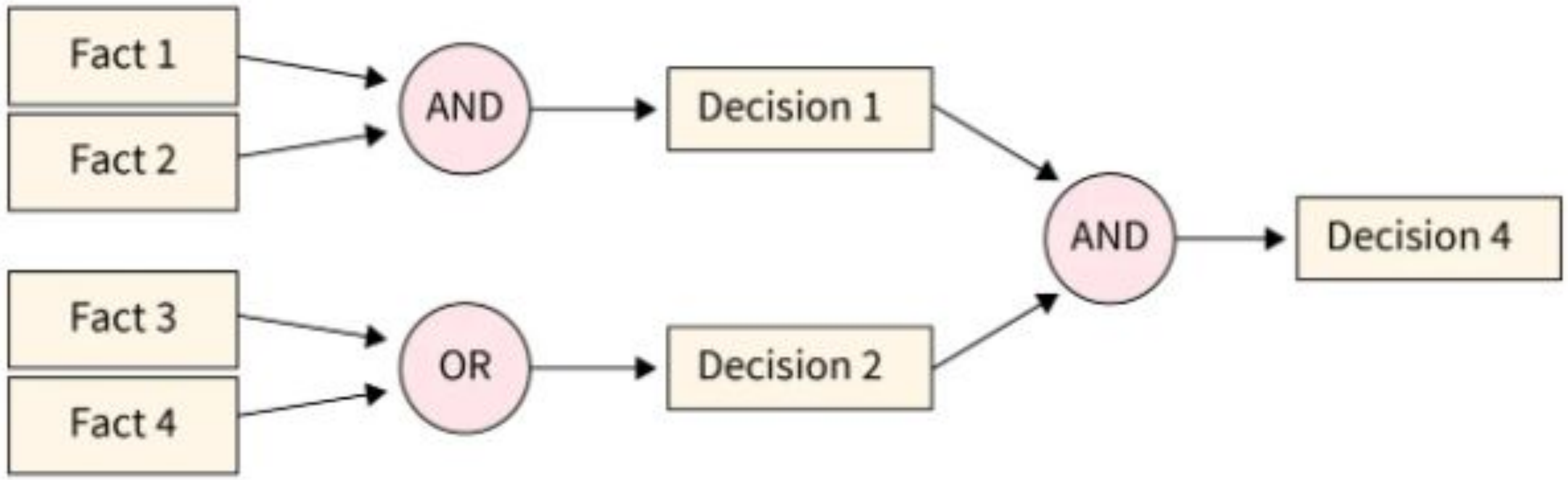
FORWARD CHAINING

- In developing intelligent systems, reasoning plays a crucial role in drawing conclusions from the existing knowledge.
- Two primary reasoning methods employed in AI are **Forward and Backward Chaining.**
- These techniques allow machines to engage in logical reasoning and tackle complex problems efficiently.
- An **inference engine** is the core component of expert systems and rule-based AI models.
- It applies logical rules to a knowledge base to derive conclusions or make decisions.
- The engine uses reasoning strategies—such as **forward chaining** and **backward chaining**—to search through data or rules and provide answers.

FORWARD CHAINING

- Forward chaining is a **data-driven** reasoning strategy in AI.
- It starts with known facts and applies rules to generate new facts or reach a conclusion.
- The process continues until no more new facts can be inferred or a goal is achieved.
- This approach is often used in expert systems for tasks such as troubleshooting and diagnostics.

FORWARD CHAINING



FORWARD CHAINING

- **How Forward Chaining Works**

Step 1: Begins with a set of initial facts in the knowledge base.

Step 2: The inference engine looks for rules whose conditions match the known facts.

Step 3: When rules are triggered, their conclusions (new facts) are added to the knowledge base.

Step 4: The process repeats with all applicable rules until a goal is met or no further rules apply

FORWARD CHAINING

- Example
- Suppose in a medical diagnosis system:
- Fact 1: The patient has a fever.
- Fact 2: The patient has a sore throat.
- Rule: If the patient has a fever and a sore throat, then the patient might have the flu.
- Given these facts and rule, forward chaining will infer **"the patient might have the flu"** by systematically checking which rules can be applied as new facts are added

FORWARD CHAINING

- Key Properties
- **Data-Driven:** The reasoning starts from available data (facts) and works toward a goal.
- **Bottom-Up Approach:** It builds knowledge from facts, gradually moving towards conclusions.
- **Breadth-First Search Strategy:** The inference engine explores multiple rules simultaneously, applying them step by step.
- **Possibility of Irrelevant Rules:** Forward chaining may explore rules that do not contribute to the final solution, making it less efficient in some cases.
- Forward chaining is particularly useful when the AI system has extensive information and needs to explore all possible logical outcomes from the data provided.

FORWARD CHAINING

- Example: Animal Identification

Initial Facts

- Fact 1: **Animal has feathers.**
- Fact 2: **Animal can fly.**

Rules

- Rule 1: **If an animal has feathers, then it is a bird.**
- Rule 2: **If an animal is a bird and can fly, then it is likely a sparrow.**
- Rule 3: **If an animal is a bird and cannot fly, then it is likely a penguin.**

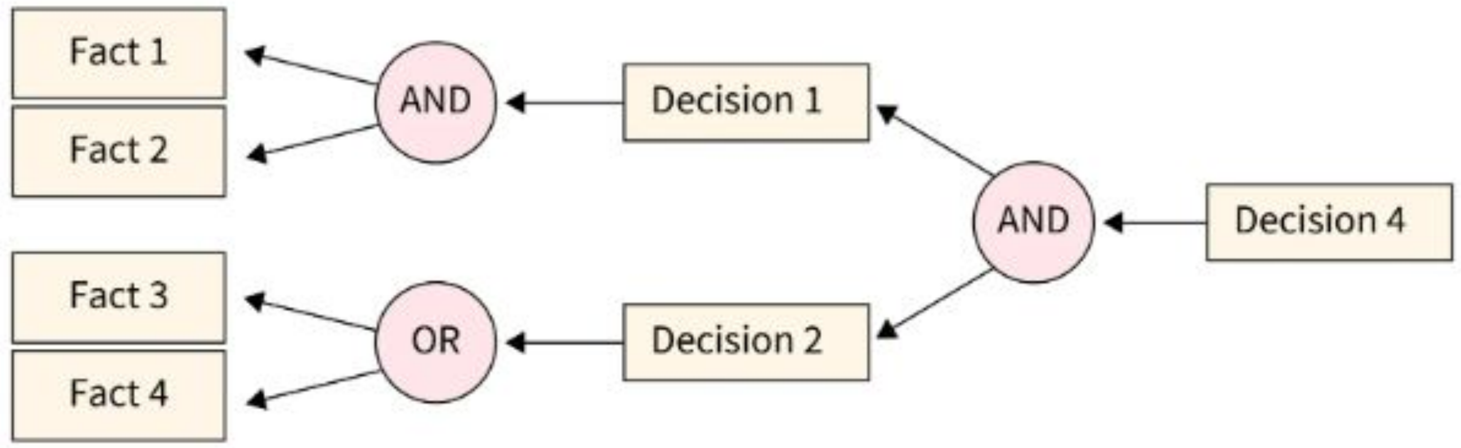
FORWARD CHAINING

Step-by-Step Chaining

- Step 1: Start with facts — "**Animal has feathers**" and "**Animal can fly**".
- Step 2: Apply Rule 1 (If animal has feathers, it is a bird). **Since the animal has feathers,** deduce: "**Animal is a bird**".
- Step 3: Update knowledge base — Now, facts are: **Animal has feathers**, **Animal can fly**, **Animal is a bird**.
- Step 4: Apply Rule 2 (If animal is a bird and can fly, it is likely a sparrow). Both conditions are now satisfied, so infer: "**Animal is likely a sparrow**".
- Step 5: **No more rules** can be triggered with the available facts, and the goal is reached.
- This process shows how forward chaining incrementally uses facts and rules to deduce new information, reaching a logical conclusion in a stepwise manner.

BACKWARD CHAINING

- Backward chaining is a goal-driven reasoning technique in artificial intelligence that starts with a desired conclusion or hypothesis and works backward to determine if the facts support that goal.
- It is often used for **diagnostics, troubleshooting, or when the endpoint is clearly defined**.



BACKWARD CHAINING

- **Properties of Backward Chaining:**
- **Goal-Driven:** Reasoning begins with a desired goal and searches for evidence to support it.
- **Top-Down Approach:** The system starts from the goal and works back to find relevant facts.
- **Depth-First Search Strategy:** The inference engine follows a path deeply before exploring other possibilities, prioritizing each goal or sub-goal in sequence.
- **Possibility of Infinite Loops:** If not handled properly, backward chaining may get stuck in loops while looking for evidence to support the goal.

BACKWARD CHAINING

- How Backward Chaining Works
- Start with the goal: The inference engine begins with the goal or hypothesis it aims to prove.
- Identify supporting rules: It looks for rules that could lead to the desired goal.
- Check rule conditions: For each rule, it checks if the conditions are satisfied, which may require verifying additional sub-goals or facts.
- Recursive process: This process is repeated, working backward until all facts required for the goal are proven or the goal is deemed unattainable.

BACKWARD CHAINING

- Example
- Suppose a medical expert system wants to determine, "Does the patient have the flu?"
- Goal: **Patient has the flu.**
- Rule: **If the patient has a fever and a sore throat, then the patient might have the flu.**
- Sub-goals: The system checks if the patient has a fever and if the patient has a sore throat.
- Process: It verifies each sub-goal by seeking supporting facts, working backward from the goal until both symptoms are confirmed, allowing it to conclude the diagnosis.

BACKWARD CHAINING

- Problem
- **Goal: Determine if Fritz (a pet) is green.**

- Rules and Facts
- Rule 1: If X croaks and X eats flies, then X is a frog.
- Rule 2: If X is a frog, then X is green.

- Fact 1: Fritz croaks.
- Fact 2: Fritz eats flies.

BACKWARD CHAINING

- Backward Chaining Steps
- Start with Goal: "Is Fritz green?"
- Look for rules to prove Goal: Rule 2 states to prove “Fritz is green,” prove “Fritz is a frog.”
- New Goal: Prove “Fritz is a frog.”
- Look for rules to prove new Goal: Rule 1 states to prove “Fritz is a frog,” prove "Fritz croaks" and "Fritz eats flies."
- Check Facts: Both "Fritz croaks" and "Fritz eats flies" are given as facts.
- Since all conditions for Rule 1 are met, conclude: “Fritz is a frog.”
- With “Fritz is a frog” true,
- conclude: “Fritz is green” (by Rule 2).

RESOLUTION IN FOPL

- **A resolution is a rule in first-order logic (FOL) that allows us to derive new conclusions from previously established data.**
- Resolution in First-Order Predicate Logic (FOPL) is a powerful, rule-based **method for automated theorem proving.**
- **It works by refutation**: **assuming the negation of the statement to be proved and then deriving a contradiction**.
- According to the concept of **proof by contradiction**, if assuming something is incorrect results in a contradiction, the assumption must be true.
- The resolution approach is widely employed in logic-based problem solving and automated reasoning because it produces systematic, thorough, and efficient proofs of logical statements.

RESOLUTION IN FOPL

- Resolution in First-Order Logic (FOL) depends on several key components that enhance the effectiveness of the inference process.
- **Clauses:** Disjunctions of literals
  - A clause is a dis-junction that serves as a fundamental unit in resolution-based proofs.
  - P(x)¬Q(x) → This expression consists of two literals linked by OR ().
  - ¬ABC → This expression includes three literals, meaning at least one of them must be true.
- **Literals:** A literal is either the atomic proposition (fact) or its negation. Literals are the building blocks of clauses.
  - P(x) is a positive literal that is either true or false.
  - ¬Q(y) is a negative literal which is the negation of a fact.

RESOLUTION IN FOPL

- **Unification:**
  - The process of finding substitutions for variables that make two predicates identical.
  - Let us consider two predicates −
  - Predicate 1: Likes (x, Mary).
  - Predicate 2: Likes (John,y).
  - To unify these two predicates, we have to find the substitutes for x and y such that they become identical. Substituting x=John and y=Mary gives us Likes(John, Mary). So, after unification, both predicates are the same.

- **Substitution:**
  - The actual replacement of variables with terms to achieve unification.
  - Consider this predicate with a function:
  - Teaches(Prof, Subject). Substituting = {Prof/Dr. Smith, Subject/Mathematics} yields the results in Teaches(Dr. Smith, Mathematics).

RESOLUTION IN FOPL

- **Skolemization:**
  - Removing existential quantifiers by introducing Skolem functions/constants, making formulas quantifier-free.
  - Let us consider a statement, x y Likes(x, y), which means "for every x, there exists some y such that x Likes y."
  - To remove y, we introduce a Skolem function f(x), resulting in x Likes(x, f(x)), where f(x) represents a specific person that x Likes , making the statement fully quantifier-free.

- **Conjunctive Normal Form (CNF):**
  - The form required for resolution, where a formula is a conjunction of disjunctions of literals.

- **Step 1: Convert all sentences to Conjunctive Normal Form – CNF**
- Every statement in FOL must be expressed in **CNF**, which is a conjunction of disjunctions of literals.
- **Example transformation:**
- Eliminate implications ($\rightarrow$)
- Move negations inward (using De Morgan's laws)
- Standardize variables (rename to avoid confusion)
- Move all quantifiers to the front (prenex form)
- **Skolemize** (eliminate existential quantifiers by introducing constants or functions)
- Drop universal quantifiers (they're implicit)
- Distribute $\vee$ over $\wedge$ to get CNF

- Step 2: **Convert all CNF statements into a set of clauses**
- Step 3: **Negate the statement to be proved**
- **Step 4: Apply the Resolution Rule repeatedly**
  - The **Resolution Rule** says:
  - If two clauses contain **complementary literals**, you can infer a new clause (the **resolvent**) that contains all remaining literals from both clauses.
  - In FOL, **unification** is used to make literals match
- **Step 5: Continue resolving until**
- Derive the **empty clause (□)** → contradiction found → theorem proven
- Or no new clauses can be produced → no proof found

RESOLUTION IN FOPL

- Steps to Convert FOPL to CNF

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

- Replace $\alpha \leftrightarrow \beta$ with $(\alpha \to \beta) \wedge (\beta \to \alpha)$.
- Replace $\alpha \to \beta$ with $\neg\alpha \vee \beta$.

Step 2: Move Negations (¬) Inward:

- Use De Morgan's laws and quantifier negation rules:
  - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
  - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  - $\neg\forall x\, P(x) \equiv \exists x\, \neg P(x)$
  - $\neg\exists x P(x) \equiv \forall x \neg P(x)$
  - Eliminate double negations: $\neg(\neg\alpha) \equiv \alpha$.

RESOLUTION IN FOPL

Step 3: Standardize Variables Apart:

- Rename variables so that different quantifiers use different variables.

Step 4: Skolemization (Remove Existential Quantifiers ∃):

- Replace existential variables with Skolem constants

(if no universal quantifiers in scope) or Skolem functions (if universal quantifiers in scope).

- This removes existential quantifiers and introduces Skolem terms.

RESOLUTION IN FOPL

Step 5: Drop Universal Quantifiers (∀):

- After Skolemization, all remaining variables are universally quantified and can be implicitly assumed, so quantifiers are dropped.

Step 6: Distribute Disjunction (∨) Over Conjunction (∧):

- Use distributive laws to bring the formula to a conjunction of disjunctions of literals (clauses):
  - $\alpha \lor (\beta \land \gamma) \equiv (\alpha \lor \beta) \land (\alpha \lor \gamma)$

Step 7: Rewrite as a Set of Clauses:

- Express the CNF formula as a set of disjunctions of literals, each clause representing one disjunction of atomic or negated atomic formulas.

RESOLUTION IN FOPL

- Example:
- "Everyone who is a student likes ice cream. Alice is a student. Prove **Alice likes ice cream.**"

Step 1: Define predicates

- Student(x) → x is a student
- Likes(x,y) → x likes y
- Constants: Alice, IceCream

Step 2: Write statements in predicate logic

- **Everyone who is a student likes ice cream:**
- $\forall x \, (Student(x) \rightarrow Likes(x, IceCream))$
- **Alice is a student:**
- $Student(Alice)$
- **Goal:** Prove Likes(Alice, IceCream)

RESOLUTION IN FOPL

Step 3: Convert to CNF (clauses)

- $\forall x\big(Student(x) \rightarrow Likes(x, IceCream)\big)$ →**remove** →:$\forall x\big(\neg Student(x) \lor Likes(x, IceCream)\big)$
- →Clause: **C1 = {¬Student(x), Likes(x,IceCream)}**
- $Student(Alice)$→
- Clause: **C2 = {Student(Alice)}**
- Negate the goal for refutation:
- ¬Likes(Alice, IceCream) → Clause: **C3 = {¬Likes(Alice, IceCream)}**

RESOLUTION IN FOPL

Step 4: Resolution

- **Step 1:** Resolve C1 and C3
- C1: {¬Student(x), Likes(x,IceCream)}
- C3: {¬Likes(Alice, IceCream)}
- **Unify:** x := Alice → {¬Student(Alice), Likes(Alice,IceCream)} and ¬Likes(Alice,IceCream) → Resolvent: **R1 = {¬Student(Alice)}**
- **Step 2:** Resolve R1 and C2
- R1: {¬Student(Alice)}
- C2: {Student(Alice)}
- Cancel Student(Alice) → Empty clause ⊥

Step 5: Conclusion

- We derived a **contradiction**, so the negation of the goal is false.
  Therefore: Likes(Alice, IceCream) is **true**.

RESOLUTION IN FOPL

- Every employee of Google is knowledgeable.
- Jayesh is an employee of Google.
- **Prove that Jayesh is knowledgeable**.

RESOLUTION IN FOPL

- Problem Statement:
- 1. Ravi likes all kind of food.
- 2. Apples and Pizza are food
- 3. Anything anyone eats and is not killed is food
- 4. Ajay eats peanuts and is still alive
- 5. Rita eats everything that Ajay eats
- Prove by resolution that **Ravi likes peanuts using resolution.**