# Introduction to Artificial Intelligence

## MODULE 02

## Knowledge Representation

Department of Computer Science, YIASCM

# Index:

- Introduction
- Techniques of Knowledge Representation
- Knowledge representation issues
- Resolution in FOL
- Difference between procedural and declarative knowledge
- Forward reasoning vs backward reasoning

# What is knowledge representation?

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning.

# Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

# What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

- **Knowledge:** Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

## Types of knowledge

Following are the various types of knowledge:

1. **Declarative Knowledge:**
   - Declarative knowledge is to know about something.
   - It includes concepts, facts, and objects.
   - It is also called descriptive knowledge and expressed in declarative sentences.
   - It is simpler than procedural language.
2. **Procedural Knowledge**
   - It is also known as imperative knowledge.
   - Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
   - It can be directly applied to any task.
   - It includes rules, strategies, procedures, agendas, etc.
   - Procedural knowledge depends on the task on which it can be applied.

## 3. Meta-knowledge:

- Knowledge about the other types of knowledge is called Meta-knowledge.

## 4. Heuristic knowledge:

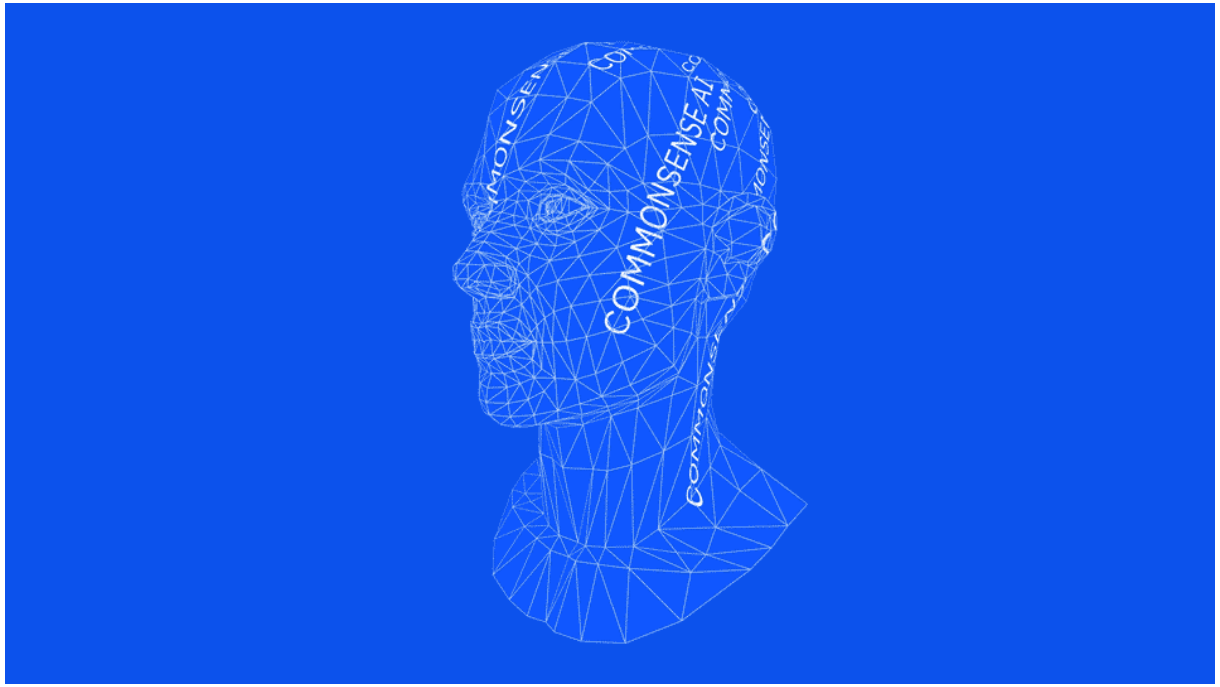
- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

## 5. Structural knowledge:

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

# Activity 06

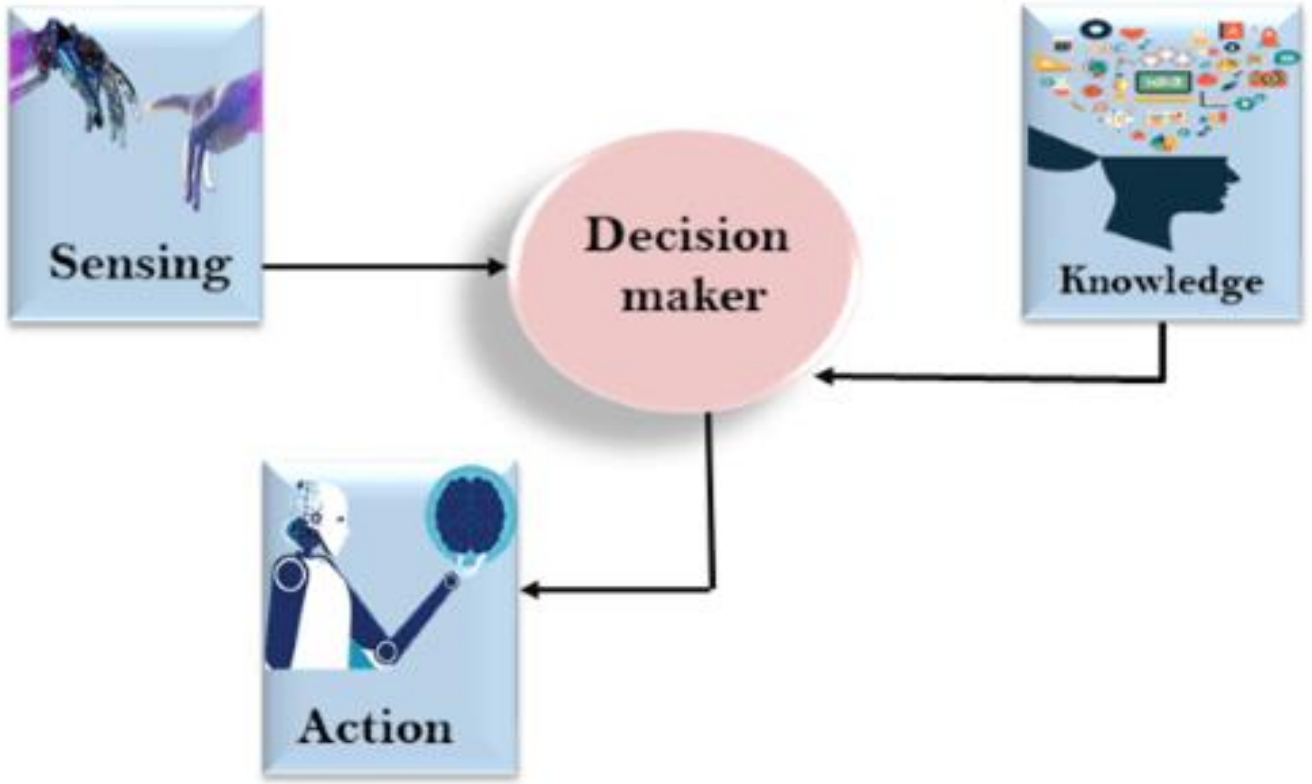
- Provide real-world applications of Knowledge Representation in AI.
- Explain how different types of knowledge and KR techniques are applied in these examples.
- Highlight any KR issues encountered and how they were addressed.



# The relation between knowledge and intelligence:

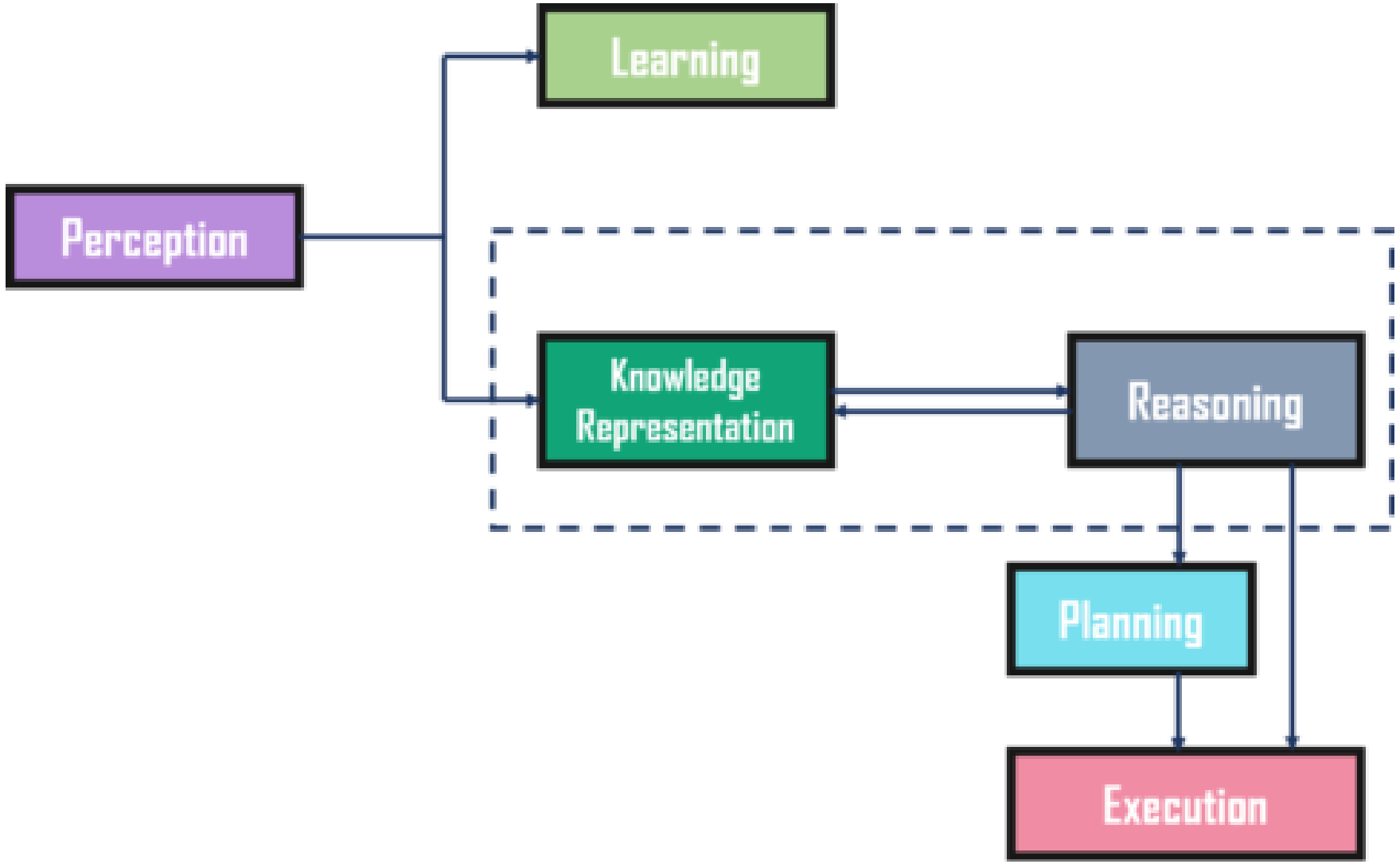
- Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence and also demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input.
- Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.

- As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.



# AI knowledge cycle:

- An Artificial intelligence system has the following components for displaying intelligent behavior:
  - Perception
  - Learning
  - Knowledge Representation and Reasoning
  - Planning
  - Execution


Learning
Perception
Knowledge Representation
Reasoning
Planning
Execution

- The diagram shows how an AI system interacts with the real world.
- AI systems have a perception component to retrieve information from the environment.
- The perception component can capture visual, audio, or other sensory inputs.
- The learning component is responsible for learning from data captured by the perception component.
- The main components in the cycle are knowledge representation and reasoning.
- Knowledge representation and reasoning show intelligence in machines like humans.
- These components are independent but also coupled together.
- Planning and execution depend on the analysis of knowledge representation and reasoning.

# Approaches to knowledge representation:

There are mainly four approaches to knowledge representation, which are given below:

1. Simple relational knowledge
2. Inheritable knowledge
3. Inferential knowledge
4. Procedural knowledge

## 1. Simple relational knowledge:

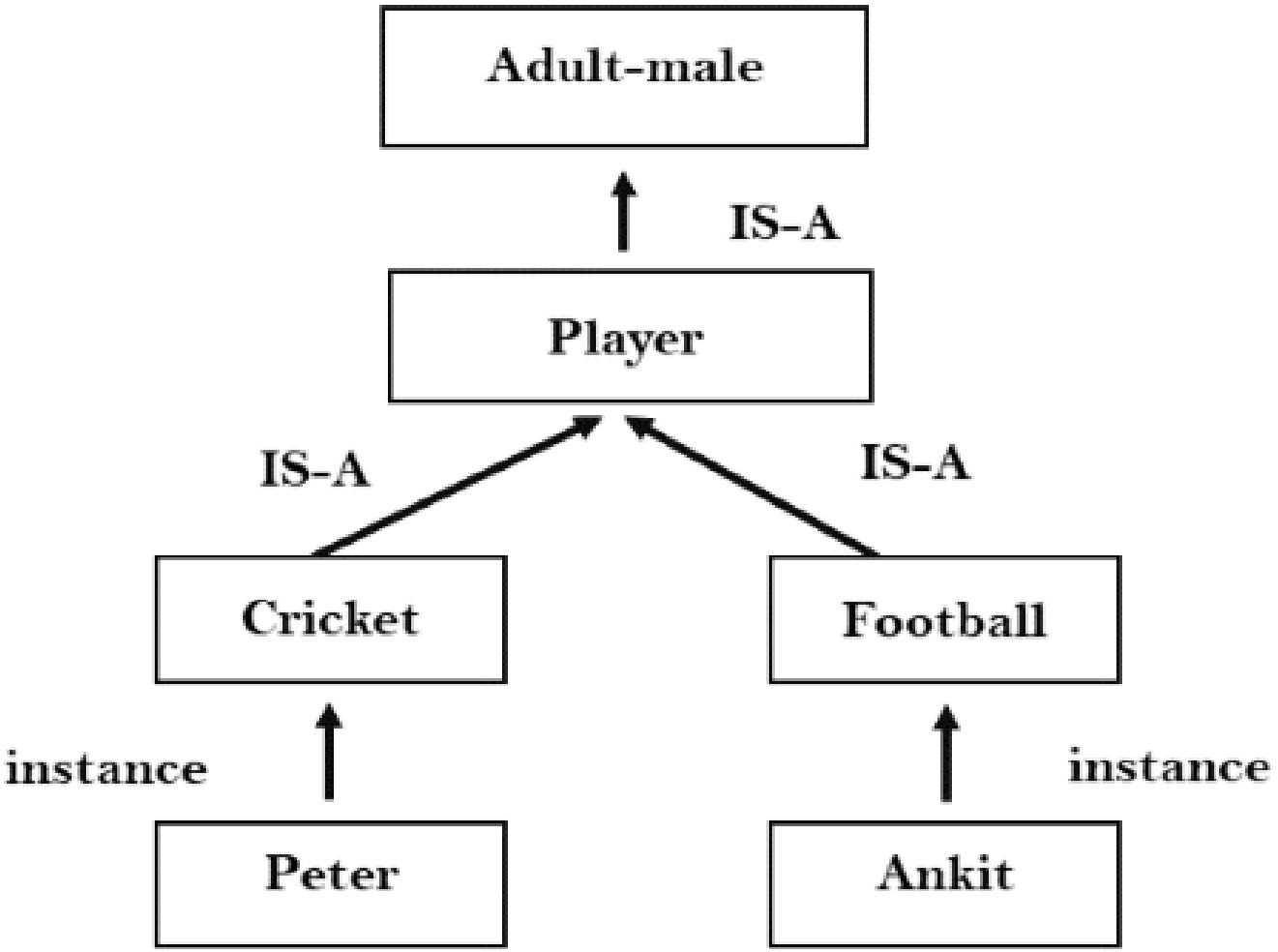
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.
- **Example: The following is the simple relational knowledge representation.**

| Player | Weight | Age |
|---|---|---|
| Player1 | 65 | 23 |
| Player2 | 58 | 18 |
| Player3 | 75 | 24 |

# 2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.

- Example


Adult-male
IS-A
Player
IS-A
IS-A
Cricket
Football
instance
instance
Peter
Ankit

# 3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
  - Marcus is a man
  - All men are mortal
    Then it can represent as;

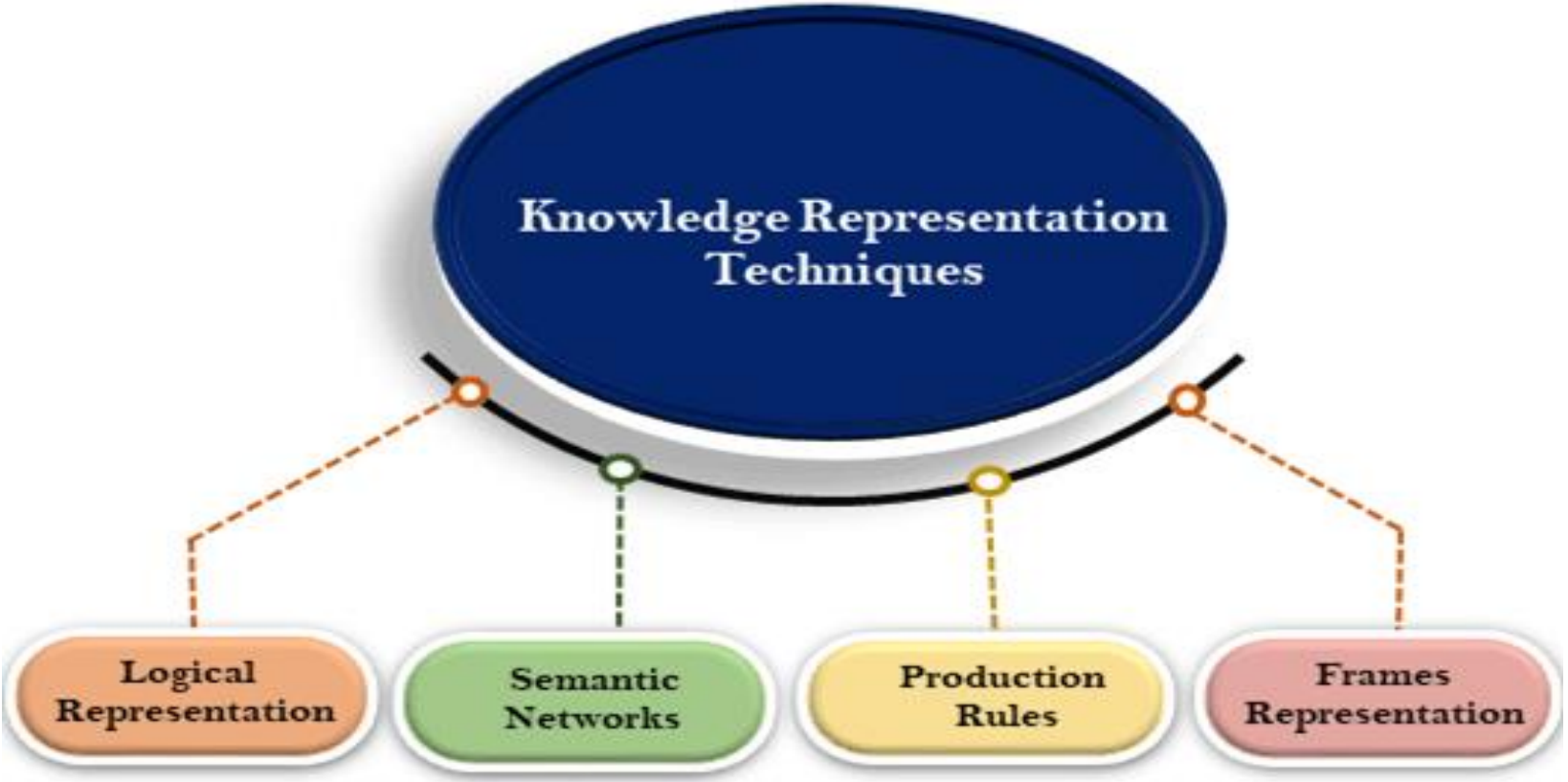
    **man(Marcus)**
    **∀x = man (x) ----------> mortal (x)s**

## 4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is **If-Then rule**.
  - **Example:** If it is raining, then take an umbrella
- In this knowledge, we can use various coding languages such as **LISP** (LISt Processing) **language** and **Prolog language**.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

# Techniques of knowledge representation

- There are mainly four ways of knowledge representation which are given as follows:



# 1. Logical Representation

- Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

## Syntax:

- Syntaxes are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

# Semantics:

- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.
- Logical representation can be categorized into mainly two logics:
  1. Propositional Logics
  2. Predicate logics

For example:

❑P: "The sky is blue"

❑Q: "It is raining"

We can represent the statement

"If the sky is blue, then it's not raining" as:

P → ¬Q (If P, then ¬Q)

- Advantages of logical representation:
  1. Logical representation enables us to do logical reasoning.
  2. Logical representation is the basis for the programming languages.

- Disadvantages of logical Representation:
  1. Logical representations have some restrictions and are challenging to work with.
  2. Logical representation technique may not be very natural, and inference may not be so efficient.

# 2. Semantic Network Representation

- Semantic networks are alternative of predicate logic for knowledge representation.
- Predicate knowledge, often referred to as predicate logic or first-order logic, a powerful tool for knowledge representation that allows for the representation of complex relationships, properties, and quantified statements.

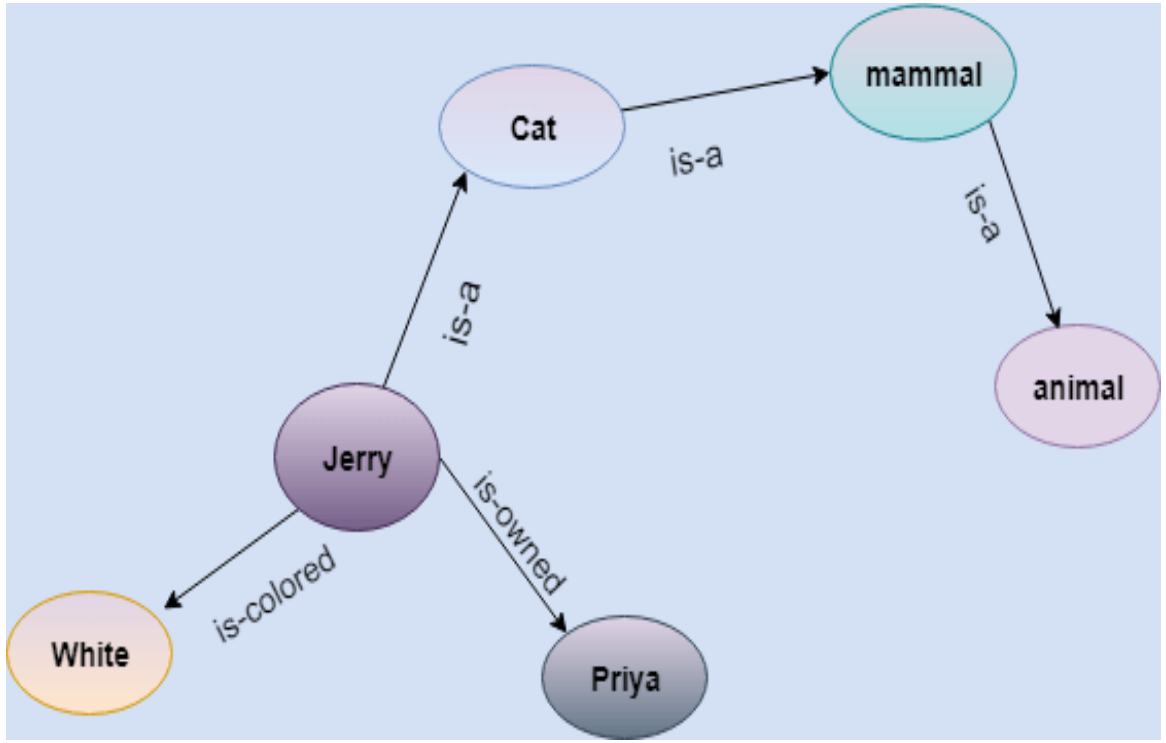
# 2. Semantic Network Representation

- In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- This representation consist of mainly two types of relations:
  1. IS-A relation (Inheritance)
  2. Kind-of-relation

**Example:** Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.
4. Jerry is brown colored.
5. All Mammals are animal.




In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

## Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.

## Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

# 3. Frame Representation

- A frame is a data structure used to represent stereotyped situations, and each frame consists of slots that store specific pieces of information or data. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.
- **Facets:** The various aspects or characteristics of a slot within a frame is known as **Facets**. Facets are features of frames which enable us to put constraints on the frames.
- Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

- Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful.
- Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base.
- The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

# Example: 1

## Let's take an example of a frame for a book

| Slots | Filters |
|---|---|
| **Title** | Artificial Intelligence |
| **Genre** | Computer Science |
| **Author** | Peter Norvig |
| **Edition** | Third Edition |
| **Year** | 1996 |
| **Page** | 1152 |

# Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. Frame representation is easy to understand and visualize.

## Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

# 4. Production Rules

Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

- The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.
- If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

- Example:
- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

## Advantages of Production rule:

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

## Disadvantages of Production rule:

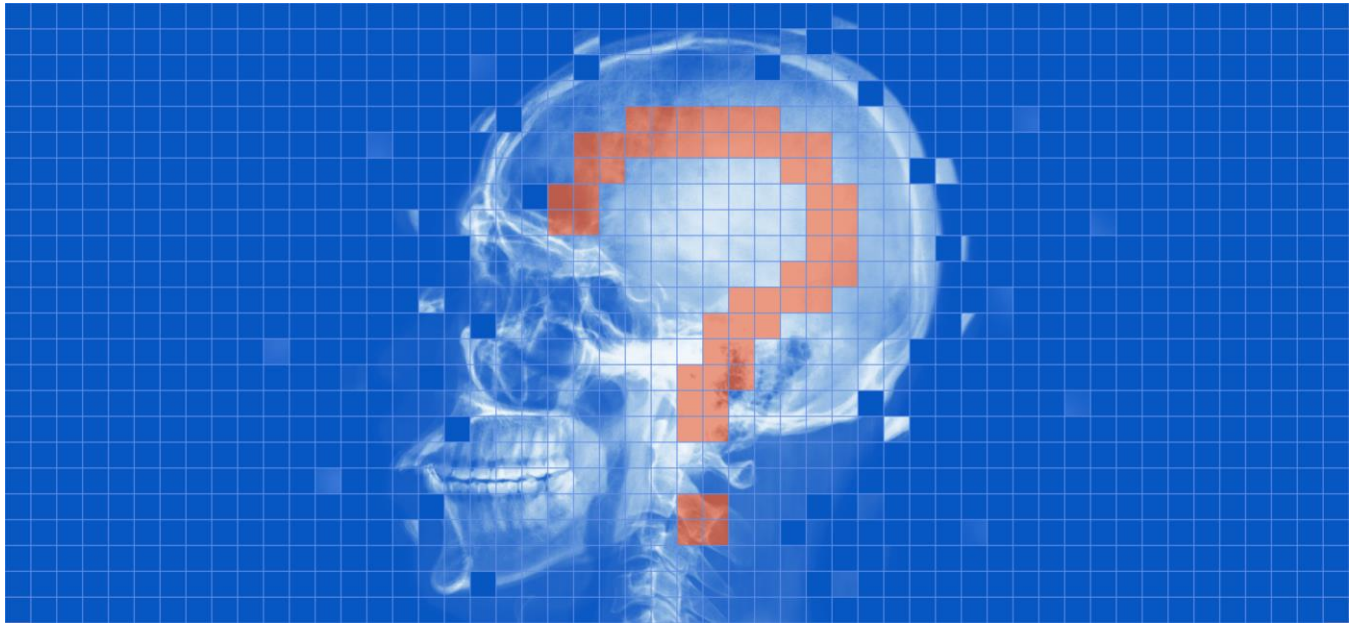
1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. They are typically rigid and require predefined rules, which makes adapting to new or changing environments challenging.

# Propositional logic in Artificial intelligence

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.
- Example:
  a) Normal human body temperature is 98.6F.
  b) The Sun rises from West (False proposition)
  c) 3+3= 7(False proposition)
  d) 5 is a prime number.

# Activity 07

- Write and interpret logical expressions using predicate logic and quantifiers.
- Solve sample problems involving the application of quantifiers.



# Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and **logical connectives**.
- These connectives are also called logical operators.

- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.
- A proposition formula which has both true and false values is called **contingency**.
- Statements which are questions, commands, or opinions are not propositions such as "**Where is Rohini**", "**How are you**", "**What is your name**", are not propositions.

# Syntax of propositional logic:

- The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

**1.Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

**Example:**

a) 2+2 = 4, it is an atomic proposition as it is a **true** fact.

b) "The Sun is cold" is also a proposition as it is a **false** fact.

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

**Example:**

a) "It is raining today, and street is wet."

b) "Ankit is a doctor, and his clinic is in Mumbai."

# Logical Connectives:

- Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.

2. **Conjunction:** A sentence which has ∧ connective such as, **P ∧ Q** is called a conjunction.
   **Example:** Rohan is intelligent and hardworking. It can be written as,
   **P = Rohan is intelligent**,
   **Q = Rohan is hardworking. → P∧ Q**.

## Logical Connectives:

3. **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
   **Example: "Ritika is a doctor or Engineer"**,
   Here P= Ritika is Doctor.

   Q= Ritika is Engineer, so we can write it as **P ∨ Q**.

**4.Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as

**If** it is raining, then the street is wet.

Let P= It is raining, and Q= Street is wet, so it is represented as P → Q

**5.Biconditional:** A sentence such as **P⇔ Q is a Biconditional sentence, example:**

**If I am breathing, then I am alive**

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

Following is the summarized table for Propositional Logic Connectives

| Connective symbols | Word | Technical term | Example |
|---|---|---|---|
| ∧ | AND | Conjunction | A ∧ B |
| ∨ | OR | Disjunction | A ∨ B |
| → | Implies | Implication | A → B |
| ⇔ | If and only if | Biconditional | A⇔ B |
| ¬ or ~ | Not | Negation | ¬ A or ¬ B |

# Limitations of Propositional logic:

- We cannot represent relations like ALL, some, or none with propositional logic. Example:
  - **Not all girls are smart.**
  - **Some apples are sweet..**
- In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

# Knowledge representation issues:

Knowledge representation in AI involves how information is structured and stored so that AI systems can utilize it effectively. Some common issues and challenges in knowledge representation include:

1. **Expressiveness**: Ensuring that the representation language or framework can express all necessary concepts and relationships in the domain accurately. Some domains may require complex relationships or uncertain information, which can be challenging to represent.
2. **Efficiency**: Representations should be efficient in terms of storage, retrieval, and inference. Large-scale knowledge bases can become computationally expensive to manage and query.

3. **Scalability**: As knowledge bases grow, ensuring that the representation scales effectively without sacrificing performance is crucial. This involves managing both the size and complexity of the representation.

4. **Uncertainty and Vagueness**: Many real-world domains involve uncertain or vague information. Representing and reasoning with uncertainty (probabilities, fuzzy logic) is a significant challenge in knowledge representation.

5. **Integration of Multiple Sources**: AI systems often need to integrate knowledge from various sources, which may use different representation languages or frameworks. Ensuring interoperability and consistency across these sources can be difficult.

6. **Dynamic Knowledge**: Some domains involve knowledge that changes over time. Representing and updating dynamic knowledge efficiently poses a challenge.

7. **Context Sensitivity**: Knowledge can be highly context-dependent. Representing context and ensuring that knowledge is interpreted correctly in different contexts is crucial for accurate reasoning.

8. **Inferential Adequacy**: The ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.

9. **Domain Specificity**: Representations often need to be tailored to specific domains or applications. General-purpose representations may not adequately capture domain-specific knowledge or nuances.

10. **Human Understandability**: In some applications, it's important for humans to understand and interact with the knowledge representation. Representations should be designed to be intuitive and interpretable where necessary.

# First-Order logic:

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.

- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
  - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, .....
  - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
  - **Function:** Father of, best friend, third inning of, end of, .....
- As a natural language, first-order logic also has two main parts:
  - **Syntax**
  - **Semantics**

## Syntax of First-Order logic:

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

## Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

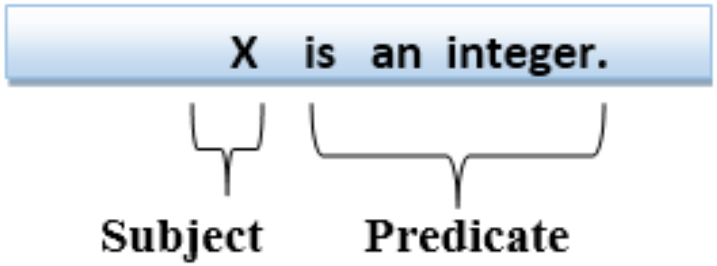
| | |
|---|---|
| **Constant** | 1, 2, A, John, Mumbai, cat,.... |
| **Variables** | x, y, z, a, b,.... |
| **Predicates** | Brother, Father, >,.... |
| **Function** | sqrt, LeftLegOf, .... |
| **Connectives** | ∧, ∨, ¬, ⇒, ⇔ |
| **Equality** | == |
| **Quantifier** | ∀, ∃ |

## Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ....., term n)**.
- **Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay). Chinky is a cat: => cat (Chinky)**.

# Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.
- **First-order logic statements can be divided into two parts:**
- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



# Complex Sentences:

**Natural Language Sentence:**

"If every student in the class studies hard, then there exists at least one student who will get an A grade."

First-Order Logic Representation:

$$\forall x\,(Student(x) \land InClass(x) \rightarrow StudiesHard(x)) \rightarrow \exists y\,(Student(y) \land InClass(y) \land GetsAGrade(y))$$

**Explanation:**

- $\forall x$: For all $x$ (where $x$ represents a student),
- $Student(x)$: $x$ is a student,
- $InClass(x)$: $x$ is in the class,
- $StudiesHard(x)$: $x$ studies hard,
- $\exists y$: There exists some $y$ (where $y$ is a student),
- $GetsAGrade(y)$: $y$ gets an A grade.

## Quantifiers in First-order logic:

- A quantifier is a part of language that expresses how many things in a certain group have a specific property. Quantification tells us the amount or number of items in the group that meet a condition.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
  - **Universal Quantifier, (for all, everyone, everything)**
  - **Existential quantifier, (for some, at least one).**

# 1. Universal Quantifier:

- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
- The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.
- If x is a variable, then ∀x is read as:

- **For all x**
- **For each x**
- **For every x.**

**Example:**

All men drink coffee.

$\forall x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$.

It will be read as: There are all x where x is a man who drink coffee

# • Existential Quantifier:

- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.
- If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:
- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

- Example

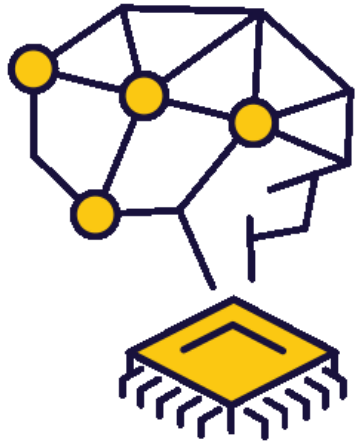
Some people like Football.

So, in logical notation, it can be written as:

$\exists x$: people(x) $\wedge$ likes Football(x)

It can be interpreted as: There are some x where x is people who like football.

# Activity 08



Convert the following predicate calculus statements to Horn Clauses and use the Resolution to solve them.

1. Predicate Calculus Statement: "If someone is a human, then they are mortal." Convert this to Horn clauses and use resolution to prove that Socrates is mortal given that Socrates is a human.
2. Predicate Calculus Statement: "If it is raining, then the ground is wet." Convert this to Horn clauses and use resolution to prove that the ground is wet given that it is raining.
3. Predicate Calculus Statement: "All birds can fly, except penguins." Convert this to Horn clauses and use resolution to determine if Tweety, a bird but not a penguin, can fly.
4. Predicate Calculus Statement: "If someone is a parent, then they have a child." Convert this to Horn clauses and use resolution to show that John has a child given that John is a parent.
5. Predicate Calculus Statement: "A person who studies passes the exam." Convert this to Horn clauses and use resolution to prove that Alice passes the exam given that Alice studies.

# Resolution in FOL

- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.
- Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.
- **Clause**: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.
- **Conjunctive Normal Form**: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

# Steps for Resolution:

1.Conversion of facts into first-order logic.

2.Convert FOL statements into CNF

3.Negate the statement which needs to prove (proof by contradiction)

4.Draw resolution graph (unification).

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

# RESOLUTION

# EXAMPLE

1. It is sunny & warm day you will enjoy
2. It is raining you will get wet
3. it is warm day
4. it is raining
5. it is sunny

Goal: You will enjoy

# 1. CONVERT FACTS INTO FOL

- It is sunny & warm day you will enjoy

Fol:- sunny ∧ Warm →enjoy

- If it is raining you will get wet

Fol:- Raining →Wet

- it is warm day

  Warm

- it is raining

  raining

- it is sunny

  sunny

# 2. CONVERT FOL INTO CNF

- Delete implies and apply formula a →b= ¬a ∨ b
- The implication A→B means "if A is true, then B must also be true." However, this can also be interpreted in another way: the only time A→B is false is when A is true and B is false. In all other cases, A→B is true.

| A (T/F) | B (T/F) | $A \rightarrow B$ | $\neg A \vee B$ |
|---|---|---|---|
| T | T | T | T |
| T | F | F | F |
| F | T | T | T |
| F | F | T | T |

It is sunny & warm day you will enjoy

1. sunny ∧ Warm →enjoy

- ¬ (sunny ∧ Warm) ∨ enjoy : Either it is not sunny and warm, or you will enjoy it.

2. :- Rainng →Wet

- ¬ Rainng ∨ wet

# 3. Negate the Statement to be Proven (Proof by Contradiction)

- To perform resolution, we need to prove a statement by contradiction. This involves negating the statement that needs to be proven.
- Suppose we want to prove that "you will enjoy it." For proof by contradiction, we negate this statement:

  ¬enjoy

- Now, we combine this negated statement with the CNF obtained from the previous step:

  ¬(sunny ∧ warm) ∨ enjoy

- Combining the two, we get:

  ¬enjoy ∧ (¬(sunny ∧ warm) ∨ enjoy)

# 3. Negate the Statement to be Proven (Proof by Contradiction)

**Initial Expression:**

$\neg enjoy \wedge (\neg(sunny \wedge warm) \vee enjoy$

**Step 1: Apply De Morgan's Law to the inner expression.**

The expression $\neg(sunny \wedge warm)$ can be simplified using De Morgan's Law:

$\neg(sunny \wedge warm) \equiv \neg sunny \vee \neg warm$

So, the expression becomes:

$\neg enjoy \wedge ((\neg sunny \vee \neg warm) \vee enjoy)$

**Step 2: Simplify the expression inside the parentheses.**

Notice that inside the parentheses, we have a disjunction (OR) $(\neg sunny \vee \neg warm) \vee enjoy$. The associative property of OR allows us to group terms without changing the result:

$(\neg sunny \vee \neg warm) \vee enjoy \equiv \neg sunny \vee \neg warm \vee enjoy$

# 3. Negate the Statement to be Proven (Proof by Contradiction)

**Step 3: Apply distribution (AND over OR).**

Now, combine this with the outer conjunction (AND):

$$\neg enjoy \wedge (\neg sunny \vee \neg warm \vee enjoy)$$

Here, ¬enjoy is ANDed with the entire expression $\neg sunny \vee \neg warm \vee enjoy$

**Step 4: Use the distributive property and simplify.**

Let's distribute ¬enjoy over the OR expression. However, notice that distributing will lead to expressions that can be simplified by logic laws (e.g., $\neg enjoy \wedge enjoy = False$, which simplifies parts of the expression):

The result is:

$$\neg sunny \vee \neg warm \vee enjoy$$

## 4. Draw the Resolution Graph (Unification)

From the clause

¬sunny ∨ ¬warm ∨ enjoy

and the negated statement ¬enjoy, we can resolve by eliminating enjoy:

This results in the clause:

¬sunny ∨ ¬warm

# STEP-1: CONVERSION OF FACTS INTO FOL

- EXAMPLE

a) John likes all kind of food.

b) Apple and vegetable are food

c) Anything anyone eats and not killed is food.

d) Anil eats peanuts and still alive

e) Harry eats everything that Anil eats.

Prove by resolution that:

John likes peanuts

a. $\forall x$: food(x) → likes(John, x)

b. food(Apple) ∧ food(vegetables)

c. $\forall x \forall y$: eats(x, y) ∧ ¬ killed(x) → food(y)

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

e. $\forall x$ : eats(Anil, x) → eats(Harry, x)

likes(John, Peanuts)

- STEP2: CONVERSION OF FOL INTO CNF(a →b= ¬a ∨ b)

Eliminate all implication (→) and rewrite

a. ∀x ¬ food(x) V likes(John, x)
b. food(Apple) Λ food(vegetables)
c. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
d. eats (Anil, Peanuts) Λ alive(Anil)
e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

likes(John, Peanuts).

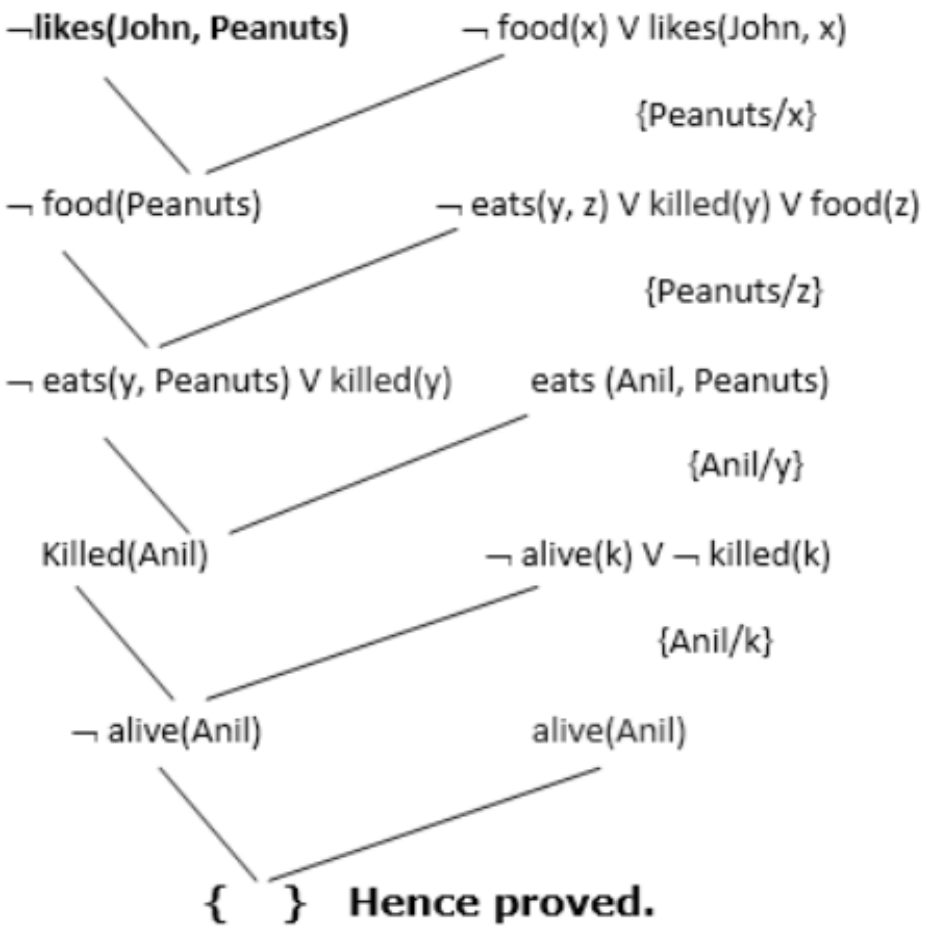
- MOVE NEGATION (¬)INWARDS AND REWRITE

a. ∀x ¬ food(x) V likes(John, x)
b. food(Apple) Λ food(vegetables)
c. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
d. eats (Anil, Peanuts) Λ alive(Anil)
e. ∀x ¬ eats(Anil, x) V eats(Harry, x)
f. ∀x ¬killed(x) ] V alive(x)
g. ∀x ¬ alive(x) V ¬ killed(x)
h. likes(John, Peanuts).

- RENAME VARIABLES OR STANDARDIZE VARIABLES

a. ∀x ¬ food(x) V likes(John, x)
b. food(Apple) Λ food(vegetables)
c. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
d. eats (Anil, Peanuts) Λ alive(Anil)
e. ∀w¬ eats(Anil, w) V eats(Harry, w)
f. ∀g ¬killed(g) ] V alive(g)
g. ∀k ¬ alive(k) V ¬ killed(k)
h. likes(John, Peanuts).

- Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



# FRAMES:

A frame is a data structure that represents a "snapshot" of the world at a particular moment in time.

It contains all of the information that an AI system needs to know about the world in order to make decision.

# WHAT ARE THE BENEFITS OF USING FRAMES IN AI?

- Frames provide a structure for representing knowledge that can be used by AI systems to reason about the world.
- They can also be used to store and retrieve information from memory, and to make inferences about new situations.
- Frames can also be used to represent plans and goals, and to generate new actions.

# WHAT ARE SOME COMMON FRAME TYPES IN AI?

- **The rule-based system:** This type of AI uses a set of rules to determine how to act in a given situation.
- **The decision tree:** This type of AI uses a tree-like structure to make decisions.
- **The neural network:** This type of AI uses a network of inter connected nodes to make decisions.
- **The genetic algorithm:** This type of AI uses a process of evolution to find solutions to problems.
- **The fuzzy logic system:** This type of AI uses a set of rules that are not precise(exact) to make decisions

# PARTITIONED NET :

A partitioned semantic net is a semantic net which can be divided into one or more networks for the description of an individual network.

# PROCEDURAL AND DECLARATIVE KNOWLEDGE

- **Procedural Knowledge:** Procedural Knowledge also known as Interpretive knowledge, is the type of knowledge in which it clarifies how a particular thing can be accomplished.
- **Interpretive knowledge:** Interpretive knowledge is the understanding of how to interpret or make sense of information. It involves knowing how to apply knowledge in specific situations, often requiring analysis, judgment, and the ability to draw conclusions from given data or scenarios.

# PROCEDURAL AND DECLARATIVE KNOWLEDGE

**EXAMPLE:**

```
var a=[1, 2, 3, 4, 5];
var b=[];
for(var i=0;i<a.length;i++)
{
b.push(a[i]);
}
console.log(b);
```

Output is: [1, 2, 3, 4, 5]

# Activity 09

Create a knowledge representation model for a library system using Semantic Nets, Frames, and Partitioned Nets. Include the following:

- Semantic Net: Build a semantic net showing relationships between books, authors, genres, and patrons.
- Frames: Develop a frame for a book, including attributes like title, author, genre, and availability.
- Partitioned Net: Design a partitioned net to organize the library sections (e.g., Fiction, Non-Fiction) and explain the connections.

# Declarative Knowledge:

- Declarative Knowledge also known as Descriptive knowledge, is the type of knowledge which tells the basic knowledge about something and it is more popular than Procedural Knowledge.
- It emphasize **what to do** something to solve a given problem.

## Let’s see it with an example:

```
# Declarative knowledge example: Facts about fruits
fruits = {
    "apple": "A sweet, edible fruit produced by an apple tree.",
    "banana": "A long, curved fruit that grows in clusters and has soft pulpy flesh.",
    "orange": "A citrus fruit that is known for being rich in vitamin C."
}


# Accessing declarative knowledge
print(fruits["apple"])   # Output: A sweet, edible fruit produced by an apple tree.
print(fruits[" orange "])  # Output: A citrus fruit that is known for being rich in vitamin C.
```

| Sl.no | Procedural Knowledge | Declarative Knowledge |
|---|---|---|
| 1 | It is also known as Interpretive knowledge. | It is also known as Descriptive knowledge. |
| 2 | Procedural Knowledge means how a particular thing can be accomplished. | While Declarative Knowledge means basic knowledge about something. |
| 3 | Procedural Knowledge is generally not used means it is not more popular. | Declarative Knowledge is more popular. |
| 4 | Procedural Knowledge can't be easily communicate. | Declarative Knowledge can be easily communicate. |
| 5 | Procedural Knowledge is generally process oriented in nature. | Declarative Knowledge is data oriented in nature. |
| 6 | In Procedural Knowledge debugging and validation is not easy. | In Declarative Knowledge debugging and validation is easy. |
| 7 | Procedural Knowledge is less effective in competitive programming. | Declarative Knowledge is more effective in competitive programming. |

# Difference between Forward and Backward Reasoning in AI:

## Forward Reasoning (Forward Chaining)

Forward reasoning, also known as forward chaining, is a process that starts with the available data and applies inference rules to extract more data until a goal or conclusion is reached.

**Example**:

**If** it is raining, **then** the ground will be wet.

**If** the ground is wet, **then** it might be slippery.

Starting with the fact "it is raining," forward reasoning will conclude that "the ground is wet" and possibly "the ground is slippery."

# Difference between Forward and Backward Reasoning in AI:

## Backward Reasoning (Backward Chaining)

Backward reasoning, also known as backward chaining, is a process that starts with a goal or conclusion and works backward to determine what data or conditions must be true for that goal to be achieved.

**Example**:

**Goal**: The ground is slippery.

**To prove**: The ground is wet.

**To prove further**: It has been raining.

In this case, backward reasoning starts with the goal "the ground is slippery" and works backward to find that "the ground must be wet," which further requires that "it has been raining."

# Difference between Forward and Backward Reasoning in AI:

| S.No. | Forward Reasoning | Backward Reasoning |
|---|---|---|
| 1. | It is a data-driven task. | It is a goal driven task. |
| 2. | It begins with new data. | It begins with conclusions that are uncertain. |
| 3. | The objective is to find a conclusion that would follow. | The objective is to find the facts that support the conclusions. |
| 4. | It uses an opportunistic type of approach. | It uses a conservative type of approach. |
| 5. | It flows from incipient to the consequence. | It flows from consequence to the incipient. |
| 6. | Forward reasoning begins with the initial facts. | Backward reasoning begins with some goal (hypothesis). |
| 7. | Forward reasoning tests all the rules. | Backward reasons tests some rules. |

| | | |
|---|---|---|
| 8. | Forward reasoning is a bottom-up approach. | Backward reasoning is a top-down approach. |
| 9. | Forward reasoning can produce an infinite number of conclusion. | Backward reasoning produces a finite number of conclusions. |
| 10. | In the forward reasoning, all the data is available. | In the backward reasoning, the data is acquired on demand. |
| 11. | Forward reasoning has a small number of initial states but a large number of conclusions. | Backward reasoning has a smaller number of goals and a larger number of rules. |
| 12. | In forward reasoning, the goal formation is difficult. | In backward reasoning, it is easy to form a goal. |
| 13. | Forward reasoning works in forward direction to find all the possible conclusions from facts. | Backward reasoning work in backward direction to find the facts that justify the goal. |
| 14. | Forward reason is suitable to answer the problems such as planning, control, monitoring, etc. | Backward reasoning is suitable for diagnosis like problems. |

# References:


- E. Rich and K. Knight, “Artificial intelligence”, TMH, 2nd ed., 1992.
- Nilsson, N. J. (1986). Principles of artificial intelligence. Morgan Kaufmann.
- Craig, J. J. (2009). Introduction to robotics: mechanics and control, 3/E. Pearson Education India.
- Klafter, R. D., Chmielewski, T. A., &Negin, M. (1989). Robotic engineering : an integrated approach. Prentice-Hall.
- Yoshikawa, T. (1990). Foundations of robotics: analysis and control. MIT press.