MODULE 4

KNOWLEDGE REPRESENTATION & REASONING

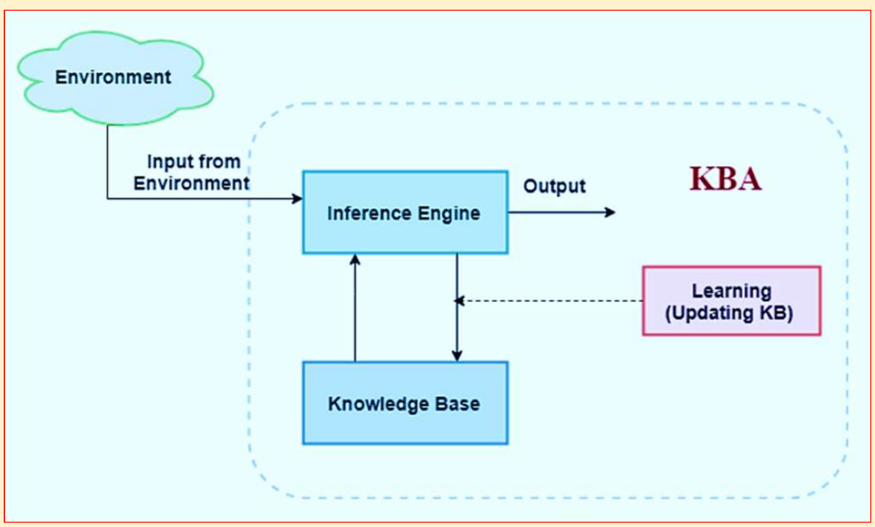
KNOWLEDGE-BASED AGENT

- An intelligent agent needs **knowledge** about the real world for taking decisions and **reasoning** to act efficiently.
- Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.
- Knowledge-based agents are composed of two main parts:
  - **Knowledge-base**
  - **Inference system.**

❖A knowledge-based agent must able to do the following:

- An agent should be able to represent states, actions, etc.
- An agent Should be able to incorporate new percepts.
- An agent can update the internal representation of the world.
- An agent can deduce the internal representation of the world.
- An agent can deduce appropriate actions.

❖The architecture of knowledge-based agent (KBA):



- Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base.
- A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information
- The knowledge-based agent (KBA) take input from the environment by perceiving the environment.
- The input is taken by the inference engine of the agent and which also communicate with KB to decide as per the knowledge store in KB.
- The learning element of KBA regularly updates the KB by learning new knowledge.

❖Knowledge base:

- Knowledge-base is a central component of a knowledge-based agent, it is also known as KB.
- It is a collection of sentences (here 'sentence' is a technical term and it is not identical to sentence in English).
- These sentences are expressed in a language which is called a knowledge representation language.
- The Knowledge-base of KBA stores fact about the world.
- Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

❖AI knowledge cycle

- An Artificial intelligence system has the following components for displaying intelligent behavior:

➢ Perception

➢ Learning

➢ Knowledge Representation and Reasoning

➢ Planning

➢ Execution

- An inference system works mainly in two rules which are given as:
- **Forward chaining**
- **Backward chaining**

❖LOGIC

- Logic is the key behind any knowledge. It allows a person to filter the necessary information from the bulk and draw a conclusion. In artificial intelligence, the representation of knowledge is done via logics.

➢ There are three main components of logic, which are as follows:

- **Syntax:** It is the sequence of a specific language which should be followed in order to form a sentence. Syntax is the representation of a language. Every language has its own syntax.

○ **For example**, $\boldsymbol{ax^2}$**+bx+c** is a well-formed syntax of a quadratic equation.

- **Semantics:** The sentence or the syntax which a logic follows should be meaningful. Semantics defines the sense of the sentence which relates to the real world.
  - **For example,** Indian people celebrate Diwali every year. This sentence represents the true fact about the country and its people who are Indians. Therefore, the sentence is syntactically as well as semantically correct.
- **Logical Inference:** Inference means to infer or draw some conclusions about some fact or a problem. Logical inference is thinking all the possible reasons which could lead to a proper result. Inference algorithms are used to perform logical inference.

APPROACHES TO KNOWLEDGE REPRESENTATION

1. Simple relational knowledge

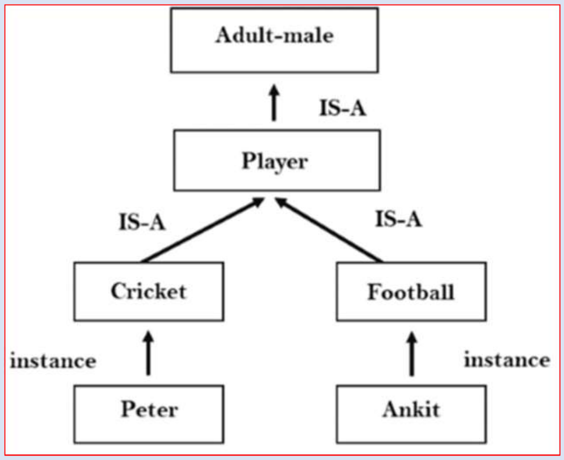
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This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

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.
- Use Arrows which point from objects to their values.



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;

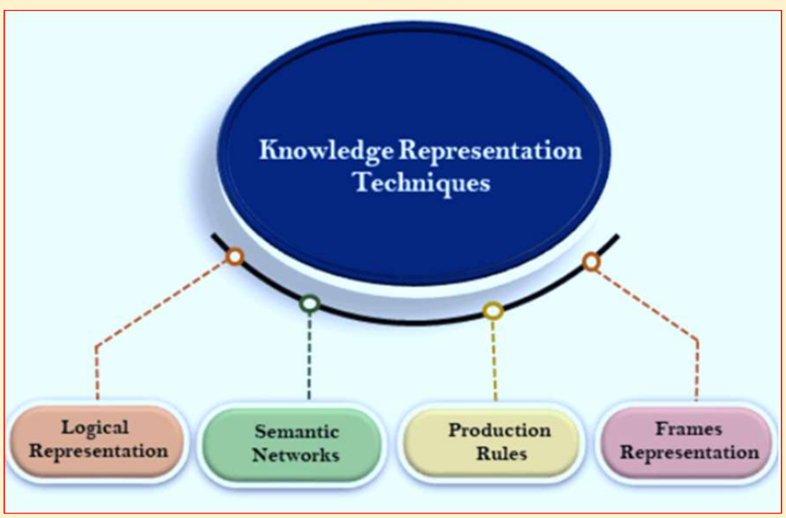
man(Marcus)

∀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**.
- In this knowledge, we can use various coding languages such as **LISP 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.

Knowledge representation Techniques



❖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.
- 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**

❖ **Advantages of logical representation**

- Logical representation enables us to do logical reasoning.
- Logical representation is the basis for the programming languages.

PROPOSITIONAL LOGIC

- 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:**

It is Sunday.

The Sun rises from West (False proposition)

3+3= 7(False proposition)

5 is a prime number.

❖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**.
- 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.

➢There are two types of Propositions:

- **Atomic Propositions**
- **Compound propositions**

❖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:

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

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

❖Compound proposition

- Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

"It is raining today and street is wet."

"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.
- Create compound propositions with the help of logical connectives.
- There are mainly five connectives, which are given as follows:
- **Negation**
- **Conjunction**
- **Disjunction**
- **Implication**
- **Biconditional**

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

❖Truth Table

- In propositional logic, we need to know the truth values of propositions in all possible scenarios.
- Combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called Truth table

❖The truth table for all logical connectives

For Negation:

P	¬ P
True	**False**
False	**True**

For Conjunction:

P	Q	P∧Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

For disjunction:

P	Q	P V Q.
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

For Implication:

P	Q	P→ Q
True	True	**True**
True	False	**False**
False	True	**True**
False	False	**True**

For Biconditional:

P	Q	P⇔ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**True**

❖Truth table with three propositions

- A proposition composing three propositions P, Q, and R.
- This truth table is made-up of 8 Tuples as we have taken three proposition symbols.

P	Q	R	¬R	Pv Q	PvQ→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

❖Precedence of connectives

- There is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem.

Precedence	Operators
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction(AND)
Fourth Precedence	Disjunction(OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

❖Logical equivalence

- Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.
- Let's take two propositions A and B, so for logical equivalence, we can write it as A⇔B. In below truth table we can see that column for ¬AV B and A→B, are identical hence A is Equivalent to B

A	B	¬A	¬AV B	A→B
T	T	F	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	F	T	**T**	**T**

❖Properties of Operators

- **Commutativity:**
  - $P \wedge Q = Q \wedge P$, or
  - $P \vee Q = Q \vee P$.
- **Associativity:**
  - $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$,
  - $(P \vee Q) \vee R = P \vee (Q \vee R)$
- **Identity element:**
  - $P \wedge True = P$,
  - $P \vee True = True$.
- **Distributive:**
  - $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$.
  - $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$.
- **DE Morgan's Law:**
  - $\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$
  - $\neg (P \vee Q) = (\neg P) \wedge (\neg Q)$.
- **Double-negation elimination:**
  - $\neg (\neg P) = P$.

❖Limitations of Propositional logic

- Cannot represent relations like ALL, some, or none with propositional logic.

Example:

All the girls are intelligent.

Some apples are sweet.

- Propositional logic has limited expressive power.
- In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

FIRST ORDER LOGIC

- In propositional logic (PL), we can only represent the facts, which are either true or false. It is not sufficient to represent the complex sentences or natural language statements.
- The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

"Some humans are intelligent"

"Sachin likes cricket"

- To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

❖First-Order logic (FOL)

- 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-ary 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**
- 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:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
Equality	==
Quantifier	$\forall, \exists$

❖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).

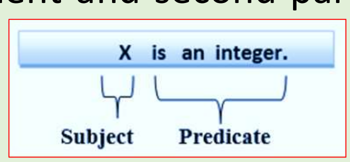
Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).
Chinky is a cat: => cat (Chinky).

❖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.



❖Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- 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).**

➢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.
- In universal quantifier we use implication "→"
- If x is a variable, then **∀x** is read as:

  For all x

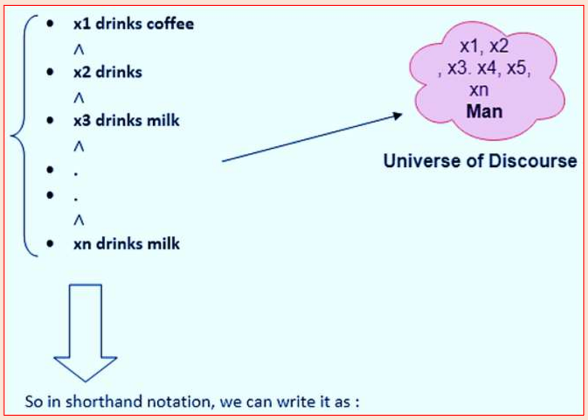
  For each x

  For every x

Example:

All man drink coffee.

- Let a variable x which refers to a man so all x can be represented in UOD as below:



∀x man(x) → drink (x, 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.
- In Existential quantifier we always use AND or Conjunction symbol (∧).

- 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.'

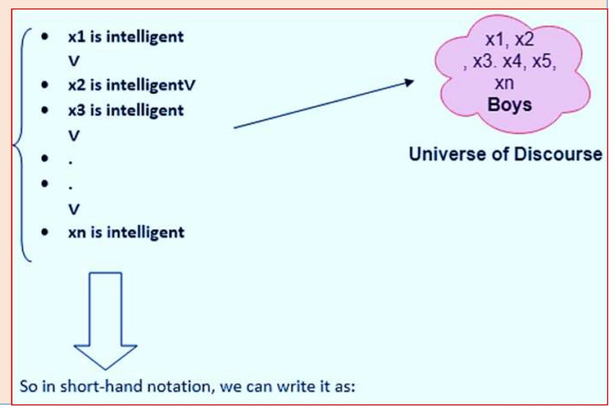
  For at least one 'x.'

Example:

Some boys are intelligent.

∃x: boys(x) ∧ intelligent(x)

- It will be read as: There are some x where x is a boy who is intelligent.



- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

❖**Properties of Quantifiers:**

- In universal quantifier, ∀x∀y is similar to ∀y∀x.
- In Existential quantifier, ∃x∃y is similar to ∃y∃x.
- ∃x∀y is not similar to ∀y∃x.

❖**Some Examples of FOL using quantifier:**

1. **All birds fly.**

In this question the predicate is "**fly(bird)**."
And since there are all birds who fly so it will be represented as follows.

∀x bird(x) →fly(x).

2. **Every man respects his parent.**

In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.
Since there is every man so will use ∀, and it will be represented as follows:

∀x man(x) → respects (x, parent).

3. **Some boys play cricket.**

In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use **∃,** and it will be represented as:

∃x boys(x) → play(x, cricket).

4. **Not all students like both Mathematics and Science.**

In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.
Since there are not all students, so we will use ∀ **with negation,** so following representation for this:

¬∀ (x) [student(x) → like(x, Mathematics) ∧ like(x, Science)].