

Mod-3

A Knowledge Based Agent

Knowledge-Based Agent in Artificial intelligence

- 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 and**
 - **Inference system.**

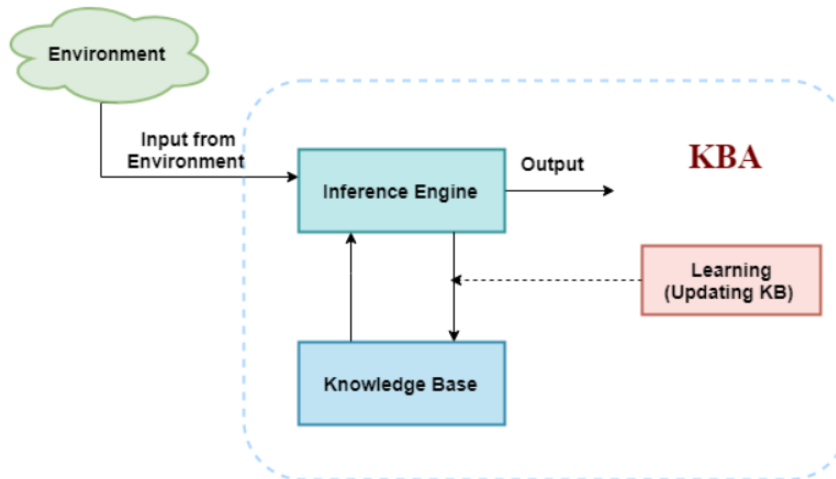
Approaches to designing a knowledge-based agent:

There are mainly two approaches to build a knowledge-based agent:

1. **1. Declarative approach:** We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with. This approach is called Declarative approach.
2. **2. Procedural approach:** In the procedural approach, we directly encode desired behavior as a program code. Which means we just need to write a program that already encodes the desired behavior or agent.

However, in the real world, a successful agent can be built by combining both declarative and procedural approaches, and declarative knowledge can often be compiled into more efficient procedural code.

The architecture of knowledge-based agent:



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.

Inference system

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.

Operations Performed by KBA

Following are three operations which are performed by KBA in order to show the intelligent behavior:

1. **TELL:** This operation tells the knowledge base what it perceives from the environment.
2. **ASK:** This operation asks the knowledge base what action it should perform.
3. **Perform:** It performs the selected action.

Approaches to designing a knowledge-based agent:

There are mainly two approaches to build a knowledge-based agent:

1. **1. Declarative approach:** We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with. This approach is called Declarative approach.
2. **2. Procedural approach:** In the procedural approach, we directly encode desired behavior as a program code. Which means we just need to write a program that already encodes the desired behavior or agent.

However, in the real world, a successful agent can be built by combining both declarative and procedural approaches, and declarative knowledge can often be compiled into more efficient procedural code.

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.

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

Approaches to knowledge representation:

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

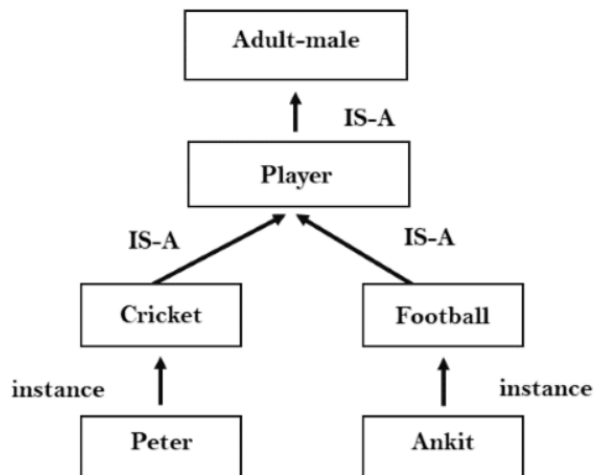
1. Simple relational knowledge:

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.
- **Example:**



3. Inferential knowledge:

- **Example:** Let's suppose there are two statements:

a. Marcus is a man

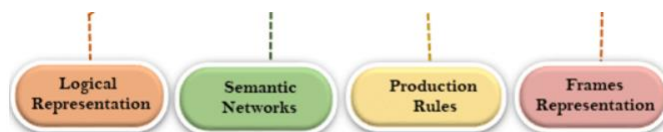
b. All men are mortal

Then it can represent as;

man(Marcus)

$\forall x = \text{man}(x) \text{ -----} \rightarrow \text{mortal}(x)$

Techniques of knowledge representation



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

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.

AD

Logical representation can be categorised into mainly two logics:

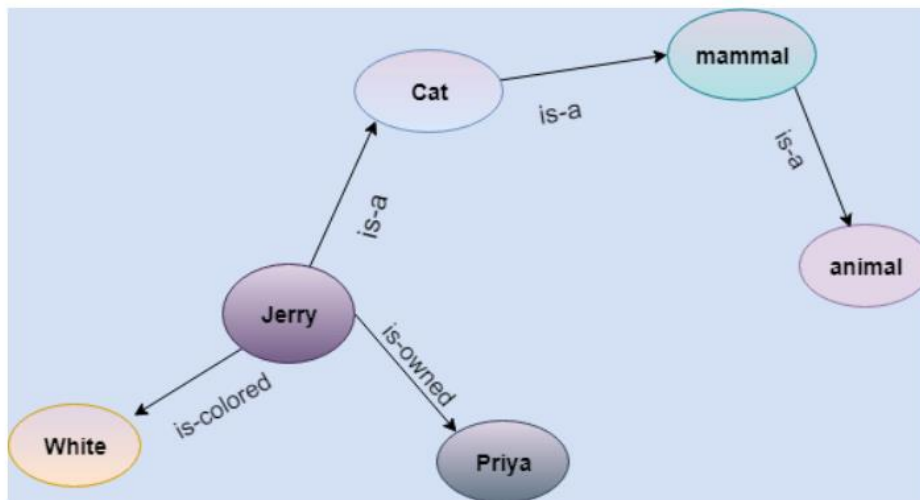
- a. Propositional Logics
- b. Predicate logics

2. Semantic Network Representation

Semantic networks are alternative of predicate logic for knowledge 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.

Statements:

- a. Jerry is a cat.
- b. Jerry is a mammal
- c. Jerry is owned by Priya.
- d. Jerry is brown colored.
- e. All Mammals are animal.



3. Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot

Example 2:

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

4. Production Rules

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

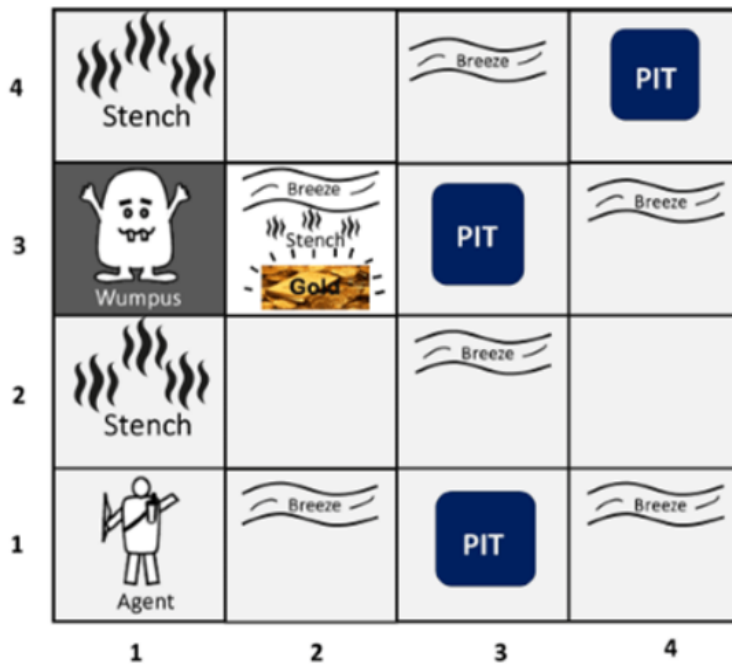
- The set of production rules
- Working Memory
- The recognize-act-cycle

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).**

Wumpus world Environment

Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.



PEAS description of Wumpus world:

To explain the Wumpus world we have given PEAS description as below:

Performance measure:

- +1000 reward points if the agent comes out of the cave with the gold.
- -1000 points penalty for being eaten by the Wumpus or falling into the pit.
- -1 for each action, and -10 for using an arrow.
- The game ends if either agent dies or came out of the cave.

Environment:

- A 4*4 grid of rooms.
- The agent initially in room square [1, 1], facing toward the right.
- Location of Wumpus and gold are chosen randomly except the first square [1,1].
- Each square of the cave can be a pit with probability 0.2 except the first square.

Actuators:

- Left turn,
- Right turn
- Move forward
- Grab
- Release
- Shoot.

Sensors:

- The agent will perceive the **stench** if he is in the room adjacent to the Wumpus. (Not diagonally).
- The agent will perceive **breeze** if he is in the room directly adjacent to the Pit.
- The agent will perceive the **glitter** in the room where the gold is present.
- The agent will perceive the **bump** if he walks into a wall.
- When the Wumpus is shot, it emits a horrible **scream** which can be perceived anywhere in the cave.
- These percepts can be represented as five element list, in which we will have different indicators for each sensor.
- Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as:
[Stench, Breeze, None, None, None].

The Wumpus world Properties:

- **Partially observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
- **Sequential:** The order is important, so it is sequential.
- **Static:** It is static as Wumpus and Pits are not moving.
- **Discrete:** The environment is discrete.
- **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

Logic

LOGICAL SYSTEM

- **Intelligent agents** need knowledge about the world to choose good actions/decisions.
 - A **logical system** is a system that knows about its **partially observable environment** and can reason about possible actions by **inferring from the hidden information**. **Reasoning** is also known as **inferencing**. An agent that acts upon logical system is known as **Knowledge based Agent**.
 - Issues in construction of a logical system:
 1. **Knowledge representation**: how do we represent information? Knowledge representation should be somewhat natural, expressive and efficient.
 2. **Knowledge reasoning**: how do we use information to reach decisions and/or derive new facts?
 - **Knowledge** in the form of a set of **facts** about our **environment** are **stored in a knowledge base (KB)**.
 - **Facts** are claims about the environment which are either **true** or **false**. **Facts** are represented by sentences
 - A **sentence** is an assertion about the world. **Sentences** are expressed in a representation language.
-

LOGIC

- The objective of **knowledge representation** is to express knowledge in a **computer-tractable form**, so that agents can perform well.
 - ***Logics are formal languages for representing information such that conclusions can be drawn.***
 - A **formal knowledge representation language** is defined by:
 - its **syntax**, which **defines all possible sequences of symbols that can be put together to constitute sentences of the language**.
 - its **semantics**, which **determines the facts in the world to which the sentences refer**. It defines the "meaning" of sentences.
 - Each **sentence** makes a claim about the world. An agent is said to believe a sentence about the world.
 - E.g., the language of arithmetic
 - $x+2 \geq y$ is a sentence; $x2y +> \{\}$ is not a sentence
 - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
 - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
 - $x+2 \geq y$ is false in a world where $x = 0, y = 6$
-

Propositional Logic

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.

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.

- **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.

a) $2+2$ is 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.

- **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."

Following is the summarized table for Propositional Logic Connectives:

Connective symbols	Word	Technical term	Example
\wedge	AND	Conjunction	$A \wedge B$
\vee	OR	Disjunction	$A \vee B$
\rightarrow	Implies	Implication	$A \rightarrow B$
\Leftrightarrow	If and only if	Biconditional	$A \Leftrightarrow B$
\neg or \sim	Not	Negation	$\neg A$ or $\neg B$

For disjunction:

P	Q	$P \vee Q$
True	True	True
False	True	True
True	False	True
False	False	False

For Implication:

P	Q	$P \rightarrow Q$
True	True	True
True	False	False
False	True	True
False	False	True

For Biconditional:

P	Q	$P \leftrightarrow Q$
True	True	True
True	False	False
False	True	False
False	False	True

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

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:**
 - $P \wedge \text{True} = P$,
 - $P \vee \text{True} = \text{True}$.
- **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:**
 - $\neg (\neg P) = P$.

Inference rules:

Inference rules are the templates for generating valid arguments. Inference rules are applied to derive proofs in artificial intelligence, and the proof is a sequence of the conclusion that leads to the desired goal.

In inference rules, the implication among all the connectives plays an important role. Following are some terminologies related to inference rules:

- **Implication:** It is one of the logical connectives which can be represented as $P \rightarrow Q$. It is a Boolean expression.
- **Converse:** The converse of implication, which means the right-hand side proposition goes to the left-hand side and vice-versa. It can be written as $Q \rightarrow P$.
- **Contrapositive:** The negation of converse is termed as contrapositive, and it can be represented as $\neg Q \rightarrow \neg P$.
- **Inverse:** The negation of implication is called inverse. It can be represented as $\neg P \rightarrow \neg Q$.

From the above term some of the compound statements are equivalent to each other, which we can prove using truth table:

P	Q	$P \rightarrow Q$	$Q \rightarrow P$	$\neg Q \rightarrow \neg P$	$\neg P \rightarrow \neg Q$
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

Types of Inference rules:

1. Modus Ponens:

The Modus Ponens rule is one of the most important rules of inference, and it states that if P and $P \rightarrow Q$ is true, then we can infer that Q will be true. It can be represented as:

$$\text{Notation for Modus ponens: } \frac{P \rightarrow Q, P}{\therefore Q}$$

Example:

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$

Statement-2: "I am sleepy" $\Rightarrow P$

Conclusion: "I go to bed." $\Rightarrow Q$.

Hence, we can say that, if $P \rightarrow Q$ is true and P is true then Q will be true.

2. Modus Tollens:

The Modus Tollens rule state that if $P \rightarrow Q$ is true and $\neg Q$ is true, then $\neg P$ will also true. It can be represented as:

$$\text{Notation for Modus Tollens: } \frac{P \rightarrow Q, \neg Q}{\neg P}$$

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$

Statement-2: "I do not go to the bed." $\Rightarrow \neg Q$

Statement-3: Which infers that "I am not sleepy" $\Rightarrow \neg P$

3. Hypothetical Syllogism:

The Hypothetical Syllogism rule state that if $P \rightarrow R$ is true whenever $P \rightarrow Q$ is true, and $Q \rightarrow R$ is true. It can be represented as the following notation:

Example:

Statement-1: If you have my home key then you can unlock my home. $P \rightarrow Q$

Statement-2: If you can unlock my home then you can take my money. $Q \rightarrow R$

Conclusion: If you have my home key then you can take my money. $P \rightarrow R$

4. Disjunctive Syllogism:

The Disjunctive syllogism rule state that if $P \vee Q$ is true, and $\neg P$ is true, then Q will be true. It can be represented as:

AD

$$\text{Notation of Disjunctive syllogism: } \frac{P \vee Q, \neg P}{Q}$$

Example:

Statement-1: Today is Sunday or Monday. $\Rightarrow P \vee Q$

Statement-2: Today is not Sunday. $\Rightarrow \neg P$

Conclusion: Today is Monday. $\Rightarrow Q$

5. Addition:

The Addition rule is one the common inference rule, and it states that If P is true, then $P \vee Q$ will be true.

$$\text{Notation of Addition: } \frac{P}{P \vee Q}$$

Example:

AD

Statement: I have a vanilla ice-cream. $\Rightarrow P$

Statement-2: I have Chocolate ice-cream.

Conclusion: I have vanilla or chocolate ice-cream. $\Rightarrow (P \vee Q)$

6. Simplification:

The simplification rule state that if $P \wedge Q$ is true, then **Q or P** will also be true. It can be represented as:

$$\text{Notation of Simplification rule: } \frac{P \wedge Q}{Q} \text{ Or } \frac{P \wedge Q}{P}$$

Proof by Truth-Table:

P	Q	$P \wedge Q$
0	0	0
1	0	0
0	1	0
1	1	1

7. Resolution:

The Resolution rule state that if $P \vee Q$ and $\neg P \wedge R$ is true, then $Q \vee R$ will also be true. **It can be represented as**

$$\text{Notation of Resolution} \frac{P \vee Q, \neg P \wedge R}{Q \vee R}$$

Propositional theorem proving

Syntax and semantics of first-order logic

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.

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	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
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: \Rightarrow Brothers(Ravi, Ajay).

Chinky is a cat: \Rightarrow 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.

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:
 - a. **Universal Quantifier, (for all, everyone, everything)**
 - b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

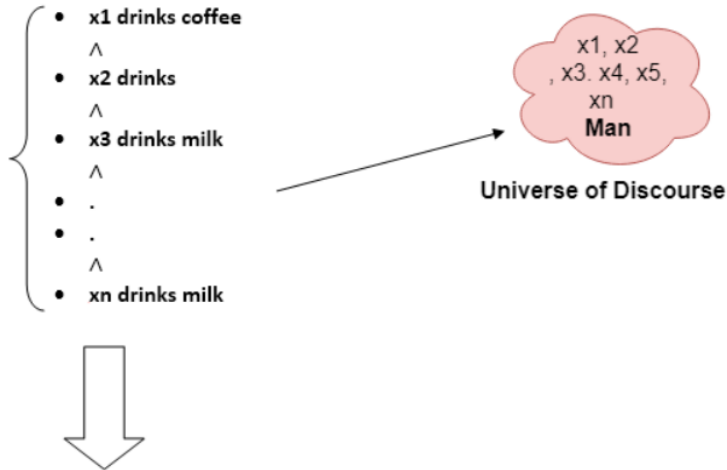
Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol \forall , which resembles an inverted A.

Example:

All man drink coffee.

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



So in shorthand notation, we can write it as :

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

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 \exists , which resembles as inverted E. When it is used with a predicate variable then it is called as an existential 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.

$$\forall x \text{ bird}(x) \rightarrow \text{fly}(x).$$

AD

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 \forall , and it will be represented as follows:

$$\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent}).$$

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 \exists , and it will be represented as:

$$\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket}).$$

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 \forall with negation, so following representation for this:

$$\neg \forall (x) [\text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \wedge \text{like}(x, \text{Science})].$$

Inference in First-Order Logic



Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences.

FOL inference rules for quantifier:

As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

- **Universal Generalization**
- **Universal Instantiation**
- **Existential Instantiation**
- **Existential introduction**

1. Universal Generalization:

- Universal generalization is a valid inference rule which states that if premise $P(c)$ is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as $\forall x P(x)$.

- It can be represented as: $\frac{P(c)}{\forall x P(x)}$.

Example: Let's represent, $P(c)$: "A byte contains 8 bits", so for $\forall x P(x)$ "All bytes contain 8 bits.", it will also be true.

2. Universal Instantiation:

- Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

- It can be represented as:
$$\frac{\forall x P(x)}{P(c)} .$$

Example:1.

IF "Every person like ice-cream" $\Rightarrow \forall x P(x)$ so we can infer that
"John likes ice-cream" $\Rightarrow P(c)$

$\forall x \text{king}(x) \wedge \text{greedy}(x) \rightarrow \text{Evil}(x),$

So from this information, we can infer any of the following statements using Universal Instantiation:

- **King(John) \wedge Greedy (John) \rightarrow Evil (John),**
- **King(Richard) \wedge Greedy (Richard) \rightarrow Evil (Richard),**
- **King(Father(John)) \wedge Greedy (Father(John)) \rightarrow Evil (Father(John)),**

propositional vs. First-order inference

Unification and Lifting

What is Unification?

[< Prev](#)[Next >](#)

- Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.
- It takes two literals as input and makes them identical using substitution.

Forward and Backward Chaining

S. No.	Forward Chaining	Backward Chaining
1.	Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the required facts that support the goal.
2.	It is a bottom-up approach	It is a top-down approach
3.	Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
4.	Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
5.	Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
6.	Forward chaining is suitable for the planning, monitoring, control, and interpretation application.	Backward chaining is suitable for diagnostic, prescription, and debugging application.
7.	Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number of possible conclusions.

A. Forward Chaining

Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

Prove that **"Robert is criminal."**

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts Conversion into FOL:

- It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)
American (p) \wedge weapon(q) \wedge sells (p, q, r) \wedge hostile(r) \rightarrow Criminal(p) ... (1)
- Country A has some missiles. **?p Owns(A, p) \wedge Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.
Owns(A, T1) (2)
Missile(T1) (3)
- All of the missiles were sold to country A by Robert.
?p Missiles(p) \wedge Owns (A, p) \rightarrow Sells (Robert, p, A) (4)
- Missiles are weapons.
Missile(p) \rightarrow Weapons (p) (5)
- Enemy of America is known as hostile.
Enemy(p, America) \rightarrow Hostile(p) (6)
- Country A is an enemy of America.
Enemy (A, America) (7)
- Robert is American
American(Robert). (8)

Forward chaining proof:

Step-1:

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1)**. All these facts will be represented as below.

AD

American (Robert)	Missile (T1)	Owns (A, T1)	Enemy (A, America)
-------------------	--------------	--------------	--------------------

Step-2:

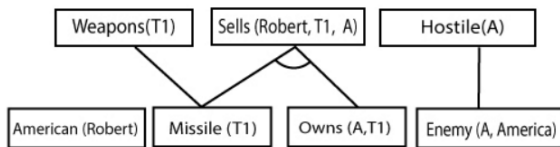
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

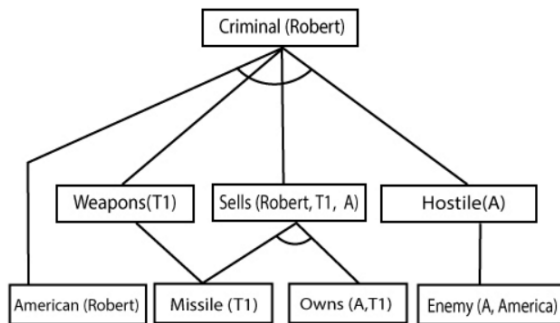
Rule-(4) satisfy with the substitution {p/T1}, so **Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



Step-3:

At step-3, as we can check Rule-(1) is satisfied with the substitution $\{p/\text{Robert}, q/T1, r/A\}$, so we can add **Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is Criminal using forward chaining approach.

Example:

In backward-chaining, we will use the same above example, and will rewrite all the rules.

- **American (p) \wedge weapon(q) \wedge sells (p, q, r) \wedge hostile(r) \rightarrow Criminal(p) ... (1)**
- **Owns(A, T1) (2)**
- **Missile(T1)**
- **?p Missiles(p) \wedge Owns (A, p) \rightarrow Sells (Robert, p, A) (4)**
- **Missile(p) \rightarrow Weapons (p) (5)**
- **Enemy(p, America) \rightarrow Hostile(p) (6)**
- **Enemy (A, America) (7)**
- **American(Robert). (8)**

Backward-Chaining proof:

In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.

Resolution

Basis for comparison	Deductive Reasoning	Inductive Reasoning
Definition	Deductive reasoning is the form of valid reasoning, to deduce new information or conclusion from known related facts and information.	Inductive reasoning arrives at a conclusion by the process of generalization using specific facts or data.
Approach	Deductive reasoning follows a top-down approach.	Inductive reasoning follows a bottom-up approach.
Process	Theory→ hypothesis→ patterns→confirmation.	Observations-→patterns→hypothesis→Theory.