UNIT-3

INFERENCE TECHNIQUES

3.1. Propositional Logic

Propositional Logic: The simplest kind of logic is propositional logic (PL), in which all statements are made up of propositions. The term "Proposition" refers to a declarative statement that can be true or false. It's a method of expressing knowledge in logical and mathematical terms.

Example:

It is Sunday.

The Sun rises from West (False proposition)

3 + 3 = 7 (False proposition)

5 is a prime number.

Following are some basic facts about propositional logic:

- Because it operates with 0 and 1, propositional logic is also known as Boolean logic.
- In propositional logic, symbolic variables are used to express the logic, and any symbol can be used to represent a proposition, such as A, B, C, P, Q, R, and so on.
- Propositions can be true or untrue, but not both at the same time.
- An object, relations or functions, and logical connectives make up propositional logic.
- Logical operators are another name for these connectives.
- The essential parts of propositional logic are propositions and connectives.
- Connectives are logical operators that link two sentences together.
- Tautology, commonly known as a legitimate sentence, is a proposition formula that is always true.
- Contradiction is a proposition formula that is always false.
- Statements that are inquiries, demands, or opinions are not propositions, such as "Where is Raj","How are you", and "What is your name" are not propositions.

Syntax of propositional logic:

The allowed sentences for knowledge representation are defined by the syntax of propositional logic. Propositions are divided into two categories:

1) Atomic Propositions.
2) Compound propositions.

- **Atomic propositions:** Simple assertions are referred to as atomic propositions. It is made up of only one proposition sign. These are the sentences that must be true or untrue in order to pass.

  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:** Simpler or atomic statements are combined with parenthesis and logical connectives to form compound propositions.

  Example:

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

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

- **Logical Connectives:** Logical connectives are used to link two simpler ideas or to logically represent a statement. With the use of logical connectives, we can form compound assertions. There are five primary connectives, which are listed below:

  1) **Negation:** A statement like ¬P is referred to as a negation of P. There are two types of literals: positive and negative literals.

  Example: Rohan is intelligent and hardworking. It can be written as,

  P = Rohan is intelligent,

  Q = Rohan is hardworking. →P ∧ Q.

  2) **Conjunction:** A conjunction is a sentence that contains ∧ connective such as, P ∧ Q.

  Example: "Ritika is a doctor or Engineer",

  Here P = Ritika is Doctor. Q = Ritika is Doctor, so we can write it as P ∨ Q.

  3) Disjunction: A disjunction is a sentence with a connective ∨ , such as P ∨ Q, where P and Q are the propositions.

  4) **Implication:** An implication is a statement such as P → Q. If-then rules are another name for implications. It can be expressed as follows: If it rains, the street is flooded.

  Because P denotes rain and Q denotes a wet street, the situation is written as P and Q

  5) **Biconditional:** A sentence like P Q, for example, is a biconditional sentence. I am alive if I am breathing.

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

• Following is the summarized table for Propositional Logic Connectives:

- **Truth Table:**

Connective Symbol	Technical Term	Word	Example
∧	Conjunction	AND	**P ∧ Q**
V	Disjunction	OR	**P V Q**
→	Implication	Implies	**P → Q**
⇔	Biconditional	If and only If	**P ⇔ Q**
¬ or ~	Negation	Not	**¬P or ¬Q**

We need to know the truth values of propositions in all feasible contexts in propositional logic. With logical connectives, we can combine all possible combinations, and the representation of these combinations in a tabular manner is known as a truth table. The truth table for all logical connectives is as follows:

For Negation:

P	¬P
true	false
false	true

For Conjunction:

P	Q	P V Q
true	true	true
true	false	true
false	true	true
false	false	false

For Disjunction:

P	Q	P V Q
true	true	true
true	false	true
false	true	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:

You can build a proposition composing three propositions P, Q, and R. The truth table is made up of 8Xn Tuples as we have taken three proposition symbols.

P	Q	R	¬R	P V Q	P V Q → ¬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	true	true
false	false	false	true	true	true

Precedence of connectives:

Propositional connectors or logical operators, like arithmetic operators, have a precedence order. When evaluating a propositional problem, this order should be followed. The following is a list of the operator precedence order:

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

Logical equivalence:

One of the characteristics of propositional logic is logical equivalence. If and only if the truth table's columns are equal, two assertions are said to be logically comparable. Let's take two

propositions P and Q, so for logical equivalence, we can write it as P⇔Q. In below truth table we can see that column for ¬P ∨ Q and P→Q, are identical hence P is Equivalent to P.

P	Q	¬P	¬P ∨ Q	P → Q
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 ∧ Q= Q ∧ P, or

P ∨ Q = Q ∨ P.

Associativity:

(P ∧ Q) ∧ R = P ∧ (Q ∧ R),

(P ∨ Q) ∨ R= P ∨ (Q ∨ R).

Identity element:

P ∧ True = P,

P ∨ True= True.

Distributive:

P ∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).

P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).

DE Morgan's Law:

¬(P ∧ Q) = (¬P) ∨ (¬Q),

¬(P ∨ Q) = (¬ P) ∧ (¬Q).

Double-negation elimination:

¬(¬P) = P.

Limitations of Propositional logic:

- This is not possible to represent relations like ALL, some, or none with propositional logic.
  Example:
  - All the girls are intelligent.
  - Some apples are sweet.
- The expressive power of propositional logic is restricted.
- We can't explain propositions in propositional logic in terms of their qualities or logical relationships.

3.2. 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	∀, ∃

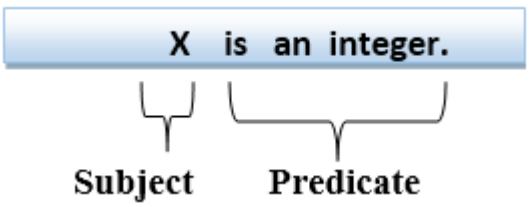
- **Atomic sentences:**
  - Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
  - We can represent atomic sentences as Predicate (term1, term2, ..., term n).
  - Example: Ravi and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



- **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.

If x is a variable, then ∀x is read as:

For all x

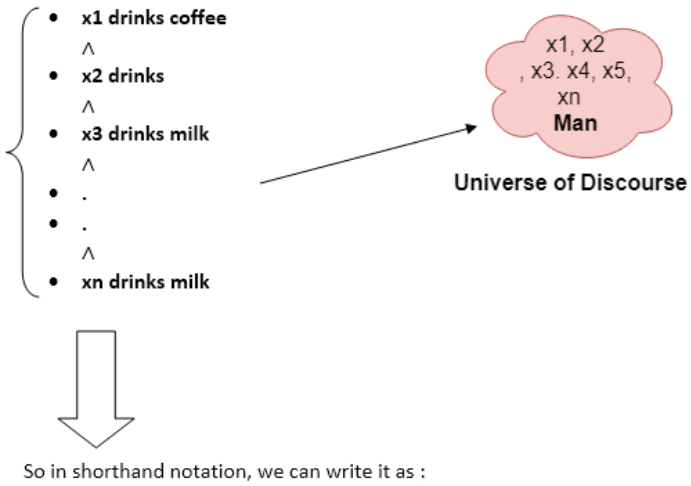
For each x

For every x.

Example:

All man drink coffee.

Let a variable x which refers to a cat 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.

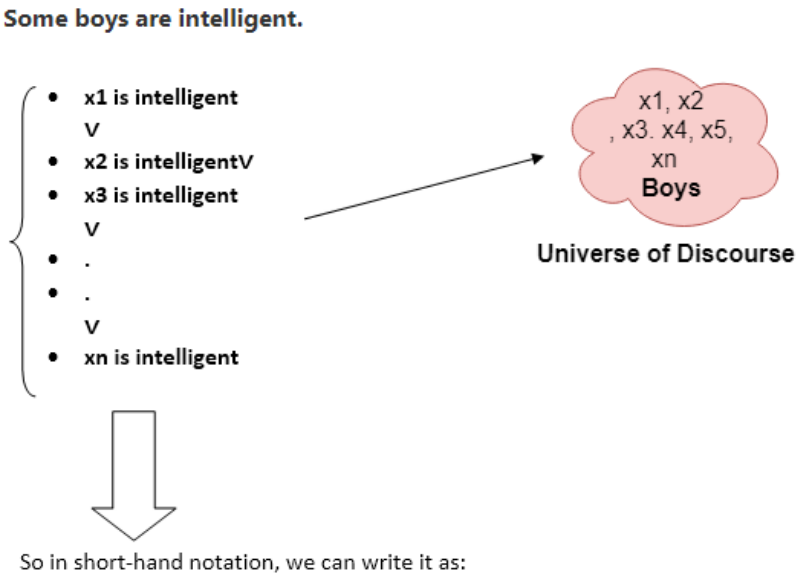
If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

There exists a 'x.'

For some 'x.'

For at least one 'x.'

Example:



∃x: boys(x) ∧ intelligent(x)

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

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)].

5. Only one student failed in Mathematics.

In this question, the predicate is "failed(x, y)," where x= student, and y= subject.

Since there is only one student who failed in Mathematics, so we will use following representation for this:

∃(x) [student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

3.3. Representing knowledge using rules

- One way to represent knowledge is by using rules that express what must happen or what does happen when certain conditions are met. Rules are usually expressed in the form of IF . . . THEN . . . statements, such as: IF A THEN B This can be considered to have a similar logical meaning as the following: A→B
- A is called the antecedent and B is the consequent in this statement. In expressing rules, the consequent usually takes the form of an action or a conclusion. In other words, the purpose of a rule

is usually to tell a system (such as an expert system) what to do in certain circumstances, or what conclusions to draw from a set of inputs about the current situation.

- In general, a rule can have more than one antecedent, usually combined either by AND or by OR (logically the same as the operators ∧ and ∨).
- Similarly, a rule may have more than one consequent, which usually suggests that there are multiple actions to be taken. In general, the antecedent of a rule compares an object with a possible value, using an operator.

  For example, suitable antecedents in a rule might be

  IF x > 3

  IF name is “Bob”

  IF weather is cold

- Here, the objects being considered are x, name, and weather; the operators are “>” and “is”, and the values are 3, “Bob,” and cold.
- Note that an object is not necessarily an object in the real-world sense—the weather is not a real world object, but rather a state or condition of the world.
- An object in this sense is simply a variable that represents some physical object or state in the real world.

  An example of a rule might be

  IF name is “Bob”

  AND weather is cold

  THEN tell Bob ‘Wear a coat’

- This is an example of a recommendation rule, which takes a set of inputs and gives advice as a result.
- The conclusion of the rule is actually an action, and the action takes the form of a recommendation to Bob that he should wear a coat.
- In some cases, the rules provide more definite actions such as “move left” or “close door,” in which case the rules are being used to represent directives.

  Rules can also be used to represent relations such as:

  IF temperature is below 0

  THEN weather is cold

3.4. Procedure versus Declarative knowledge

- We can express the knowledge in various forms to the inference engine in the computer system to solve the problems. There are two important representations of knowledge namely, procedural knowledge and declarative knowledge. The basic difference between procedural and declarative knowledge is that procedural knowledge gives the control information along with the knowledge, whereas declarative knowledge just provides the knowledge but not the control information to implement the knowledge.
- Read through this article to find out more about procedural knowledge and declarative knowledge and how they are different from each other.
- **What is Procedural Knowledge?**
  Procedural or imperative knowledge clarifies how to perform a certain task. It lays down the steps to perform. Thus, the procedural knowledge provides the essential control information required to implement the knowledge.
- **What is Declarative Knowledge?**
  Declarative or functional knowledge clarifies what to do to perform a certain task. It lays down the function to perform. Thus, in the declarative knowledge, only the knowledge is provided but not the control information to implement the knowledge. Thus, in order to use the declarative knowledge, we have to add the declarative knowledge with a program which provides the control information.

Key	Procedural Knowledge	Declarative Knowledge
Meaning	Procedural knowledge provides the knowledge of how a particular task can be accomplished.	Declarative knowledge provides the basic knowledge about something.
Alternate name	Procedural knowledge is also termed as imperative knowledge.	Declarative knowledge is also termed as functional knowledge.
Basis	Procedural knowledge revolves around the "How" of the concept.	Declarative knowledge revolves around the "What" of the concept.
Communication	Procedural knowledge is difficult to communicate.	Declarative knowledge is easily communicable.
Orientation	Procedural knowledge is process-oriented.	Declarative knowledge is data-oriented.
Validation	Validation is not very easy in procedural knowledge.	Validation is quite easy in declarative knowledge.
Debugging	Debugging is not very easy in procedural knowledge.	Debugging is quite easy in declarative knowledge.

Use	Procedural knowledge is less commonly used.	Declarative knowledge is more general.
Representation	Procedural knowledge is represented by a set of rules.	Declarative knowledge is represented by production systems.
Source	Procedural knowledge is obtained from actions, experiences, subjective insights, etc.	Declarative knowledge is obtained from principles, procedures, concepts, processes, etc.

3.5. Forward versus Backward Reasoning

What is Forward Reasoning?

- Forward reasoning is a process in artificial intelligence that finds all the possible solutions of a problem based on the initial data and facts. Thus, the forward reasoning is a data-driven task as it begins with new data. The main objective of the forward reasoning in AI is to find a conclusion that would follow. It uses an opportunistic type of approach.
- Forward reasoning flows from incipient to the consequence. The inference engine searches the knowledge base with the given information depending on the constraints. The precedence of these constraints have to match the current state.
- In forward reasoning, the first step is that the system is given one or more constraints. The rules are then searched for in the knowledge base for every constraint. The rule that fulfils the condition is selected. Also, every rule can generate a new condition from the conclusion which is obtained from the invoked one. This new conditions can be added and are processed again.
- The step ends if no new conditions exist. Hence, we can conclude that forward reasoning follows the top-down approach.

What is Backward Reasoning?

- Backward reasoning is the reverse process of the forward reasoning in which a goal or hypothesis is selected and it is analyzed to find the initial data, facts, and rules. Therefore, the backward reasoning is a goal driven task as it begins with conclusions or goals that are uncertain. The main objective of the backward reasoning is to find the facts that support the conclusions.
- Backward reasoning uses a conservative type of approach and flows from consequence to the incipient. The system helps to choose a goal state and reasons in a backward direction. The first step in the backward reasoning is that the goal state and rules are selected. Then, sub-goals are made from the selected rule, which need to be satisfied for the goal state to be true.

- The initial conditions are set such that they satisfy all the sub-goals. Also, the established states are matched to the initial state provided. If the condition is fulfilled, the goal is the solution, otherwise the goal is rejected. Therefore, backward reasoning follows bottom-up technique.
- Backward reasoning is also known as a decision-driven or goal-driven inference technique because the system selects a goal state and reasons in the backward direction.

Difference between Forward and Backward Reasoning in AI

Sr. 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.