Unit-4

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:

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)
4. d) 5 is a prime number.

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
- 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:

a. **Atomic Propositions**

b. **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:

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

1. a) "It is raining today, and street is wet."
2. 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**.
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 Doctor, 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

Truth Table:

In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called **Truth table.** Following are 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 ∨ 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:

We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n 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:

Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

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

Note: For better understanding use parenthesis to make sure of the correct interpretations. Such as ¬R∨ Q, It can be interpreted as (¬R) ∨ Q.

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 $A \Leftrightarrow B$. In below truth table we can see that column for $\neg A \vee B$ and $A \rightarrow B$, are identical hence A is Equivalent to B

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

Limitations of Propositional logic:

- We cannot represent relations like ALL, some, or none with propositional logic. Example:
    a. **All the girls are intelligent.**
    b. **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 Artificial intelligence

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL 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", or**
- **"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:

- 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:
    a. **Syntax**
    b. **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:

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 ∀, which resembles an inverted A.

> Note: In universal quantifier we use implication "→".

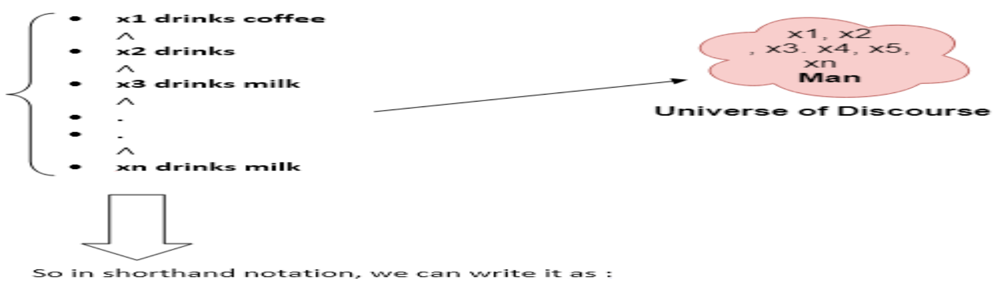
If x is a variable, then ∀x is read as:

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

Example:

All man drink coffee.

Let a variable x which refers to a 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.

Note: 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.

Points to remember:

- 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)].

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)].

Free and Bound Variables:

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

Free Variable: A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

Bound Variable: A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: ∀x [A (x) B(y)], here x and y are the bound variables.

Knowledge Engineering in First-order logic

What is knowledge-engineering?

The process of constructing a knowledge-base in first-order logic is called as knowledge-engineering. In **knowledge-engineering**, someone who investigates a particular domain, learns important concept of that domain, and generates a formal representation of the objects, is known as **knowledge engineer**.

In this topic, we will understand the Knowledge engineering process in an electronic circuit domain, which is already familiar. This approach is mainly suitable for creating **special-purpose knowledge base**.

The knowledge-engineering process:

Following are some main steps of the knowledge-engineering process. Using these steps, we will develop a knowledge base which will allow us to reason about digital circuit (**One-bit full adder**) which is given below

1. Identify the task:

The first step of the process is to identify the task, and for the digital circuit, there are various reasoning tasks.

At the first level or highest level, we will examine the functionality of the circuit:

- **Does the circuit add properly?**
- **What will be the output of gate A2, if all the inputs are high?**

At the second level, we will examine the circuit structure details such as:

- **Which gate is connected to the first input terminal?**
- **Does the circuit have feedback loops?**

2. Assemble the relevant knowledge:

In the second step, we will assemble the relevant knowledge which is required for digital circuits. So for digital circuits, we have the following required knowledge:

- Logic circuits are made up of wires and gates.

- Signal flows through wires to the input terminal of the gate, and each gate produces the corresponding output which flows further.
- In this logic circuit, there are four types of gates used: **AND, OR, XOR, and NOT**.
- All these gates have one output terminal and two input terminals (except NOT gate, it has one input terminal).

3. Decide on vocabulary:

The next step of the process is to select functions, predicate, and constants to represent the circuits, terminals, signals, and gates. Firstly we will distinguish the gates from each other and from other objects. Each gate is represented as an object which is named by a constant, such as, **Gate(X1)**. The functionality of each gate is determined by its type, which is taken as constants such as **AND, OR, XOR, or NOT**. Circuits will be identified by a predicate: **Circuit (C1)**.

For the terminal, we will use predicate: **Terminal(x)**.

For gate input, we will use the function **In(1, X1)** for denoting the first input terminal of the gate, and for output terminal we will use **Out (1, X1)**.

The function **Arity(c, i, j)** is used to denote that circuit c has i input, j output.

The connectivity between gates can be represented by predicate **Connect(Out(1, X1), In(1, X1))**.

We use a unary predicate **On (t)**, which is true if the signal at a terminal is on.

4. Encode general knowledge about the domain:

To encode the general knowledge about the logic circuit, we need some following rules:

- If two terminals are connected then they have the same input signal, it can be represented as:

1. ∀ t1, t2 Terminal (t1) ∧ Terminal (t2) ∧ Connect (t1, t2) → Signal (t1) = Signal (2).

- Signal at every terminal will have either value 0 or 1, it will be represented as:

1. ∀ t Terminal (t) →Signal (t) = 1 ∨Signal (t) = 0.

- Connect predicates are commutative:

1. ∀ t1, t2 Connect(t1, t2) → Connect (t2, t1).

- Representation of types of gates:

1. ∀ g Gate(g) ∧ r = Type(g) → r = OR ∨r = AND ∨r = XOR ∨r = NOT.

- Output of AND gate will be zero if and only if any of its input is zero.

1. ∀ g Gate(g) ∧ Type(g) = AND →Signal (Out(1, g))= 0 ⇔ ∃n Signal (In(n, g))= 0.

- Output of OR gate is 1 if and only if any of its input is 1:

1. ∀ g Gate(g) ∧ Type(g) = OR → Signal (Out(1, g))= 1 ⇔ ∃n Signal (In(n, g))= 1

- Output of XOR gate is 1 if and only if its inputs are different:

1. ∀ g Gate(g) ∧ Type(g) = XOR → Signal (Out(1, g)) = 1 ⇔ Signal (In(1, g)) ≠ Signal (In(2, g)).

- Output of NOT gate is invert of its input:

1. ∀ g Gate(g) ∧ Type(g) = NOT → Signal (In(1, g)) ≠ Signal (Out(1, g)).

- All the gates in the above circuit have two inputs and one output (except NOT gate).

1. ∀ g Gate(g) ∧ Type(g) = NOT → Arity(g, 1, 1)
2. ∀ g Gate(g) ∧ r =Type(g) ∧ (r= AND ∨r= OR ∨r= XOR) → Arity (g, 2, 1).

- All gates are logic circuits:

1. ∀ g Gate(g) → Circuit (g).

5. Encode a description of the problem instance:

Now we encode problem of circuit C1, firstly we categorize the circuit and its gate components. This step is easy if ontology about the problem is already thought. This step involves the writing simple atomics sentences of instances of concepts, which is known as ontology.

For the given circuit C1, we can encode the problem instance in atomic sentences as below:

Since in the circuit there are two XOR, two AND, and one OR gate so atomic sentences for these gates will be:

1. For XOR gate: Type(x1)= XOR, Type(X2) = XOR
2. For AND gate: Type(A1) = AND, Type(A2)= AND
3. For OR gate: Type (O1) = OR.

And then represent the connections between all the gates.

> Note: Ontology defines a particular theory of the nature of existence.

6. Pose queries to the inference procedure and get answers:

In this step, we will find all the possible set of values of all the terminal for the adder circuit. The first query will be:

What should be the combination of input which would generate the first output of circuit C1, as 0 and a second output to be 1?

1. ∃ i1, i2, i3 Signal (In(1, C1))=i1 ∧ Signal (In(2, C1))=i2 ∧ Signal (In(3, C1))= i3
2. ∧ Signal (Out(1, C1)) =0 ∧ Signal (Out(2, C1))=1

7. Debug the knowledge base:

Now we will debug the knowledge base, and this is the last step of the complete process. In this step, we will try to debug the issues of knowledge base.

In the knowledge base, we may have omitted assertions like 1 ≠ 0.

Inference in First-Order Logic

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.

Substitution:

Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write **F[a/x]**, so it refers to substitute a constant "**a**" in place of variable "**x**".

> Note: First-order logic is capable of expressing facts about some or all objects in the universe.

Equality:

First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL. For this, we can use **equality symbols** which specify that the two terms refer to the same object.

Example: Brother (John) = Smith.

As in the above example, the object referred by the **Brother (John)** is similar to the object referred by **Smith**. The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example: ¬(x=y) which is equivalent to x ≠y.

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 ∀ x P(x).
- It can be represented as: $\frac{P(c)}{\forall x\, P(x)}$.
- This rule can be used if we want to show that every element has a similar property.
- In this rule, x must not appear as a free variable.

Example: Let's represent, P(c): "**A byte contains 8 bits**", so for ∀ **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.
- The new KB is logically equivalent to the previous KB.
- As per UI, **we can infer any sentence obtained by substituting a ground term for the variable**.
- The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ **x P(x) for any object in the universe of discourse**.
- It can be represented as: $\frac{\forall x\, P(x)}{P(c)}$.

Example:1.

IF "Every person like ice-cream"=> ∀x P(x) so we can infer that
"John likes ice-cream" => P(c)

Example: 2.

Let's take a famous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

∀x king(x) ∧ greedy (x) → Evil (x),

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

- **King(John) ∧ Greedy (John) → Evil (John),**
- **King(Richard) ∧ Greedy (Richard) → Evil (Richard),**
- **King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),**

3. Existential Instantiation:

- Existential instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic.
- It can be applied only once to replace the existential sentence.
- The new KB is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable.
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.
- The restriction with this rule is that c used in the rule must be a new term for which P(c) is true.
- It can be represented as: $\frac{\exists x\, P(x)}{P(c)}$

Example:

From the given sentence: **∃x Crown(x) ∧ OnHead(x, John),**

So we can infer: **Crown(K) ∧ OnHead(K, John),** as long as K does not appear in the knowledge base.

- The above used K is a constant symbol, which is called **Skolem constant**.
- The Existential instantiation is a special case of **Skolemization process**.

4. Existential introduction

- An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as: $\frac{P(c)}{\exists x P(x)}$
- **Example: Let's say that,**
  "Priyanka got good marks in English."
  "Therefore, someone got good marks in English."

Generalized Modus Ponens Rule:

For the inference process in FOL, we have a single inference rule which is called Generalized Modus Ponens. It is lifted version of Modus ponens.

Generalized Modus Ponens can be summarized as, " P implies Q and P is asserted to be true, therefore Q must be True."

According to Modus Ponens, for atomic sentences **pi, pi', q**. Where there is a substitution θ such that SUBST **(θ, pi',) = SUBST(θ, pi)**, it can be represented as:

$$\frac{p1', p2', \ldots, pn', (p1 \land p2 \land \ldots \land pn \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

Example:

We will use this rule for Kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.

1. Here let say, p1' is king(John)   p1 is king(x)
2. p2' is Greedy(y)   p2 is Greedy(x)
3. θ is {x/John, y/John}   q is evil(x)
4. SUBST(θ,q).

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.

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:
f. **John likes peanuts.**

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

a. $\forall x: food(x) \rightarrow likes(John, x)$
b. $food(Apple) \land food(vegetables)$
c. $\forall x \forall y: eats(x, y) \land \neg killed(x) \rightarrow food(y)$
d. $eats(Anil, Peanuts) \land alive(Anil)$.
e. $\forall x : eats(Anil, x) \rightarrow eats(Harry, x)$
f. $\forall x: \neg killed(x) \rightarrow alive(x)$ **added predicates.**
g. $\forall x: alive(x) \rightarrow \neg killed(x)$
h. $likes(John, Peanuts)$

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

- **Eliminate all implication (→) and rewrite**
  a. $\forall x \neg food(x) \lor likes(John, x)$
  b. $food(Apple) \land food(vegetables)$
  c. $\forall x \forall y \neg [eats(x, y) \land \neg killed(x)] \lor food(y)$
  d. $eats(Anil, Peanuts) \land alive(Anil)$

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).

- **Move negation (¬)inwards and rewrite**

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

- **Rename variables or standardize variables**

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

- **Eliminate existential instantiation quantifier by elimination.**
  In this step, we will eliminate existential quantifier ∃, and this process is known as **Skolemization**. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.
- **Drop Universal quantifiers.**
  In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.

. ¬ food(x) V likes(John, x)
a. food(Apple)
b. food(vegetables)
c. ¬ eats(y, z) V killed(y) V food(z)
d. eats (Anil, Peanuts)
e. alive(Anil)

f. ¬ eats(Anil, w) V eats(Harry, w)

g. killed(g) V alive(g)

h. ¬ alive(k) V ¬ killed(k)

i. likes(John, Peanuts).

Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

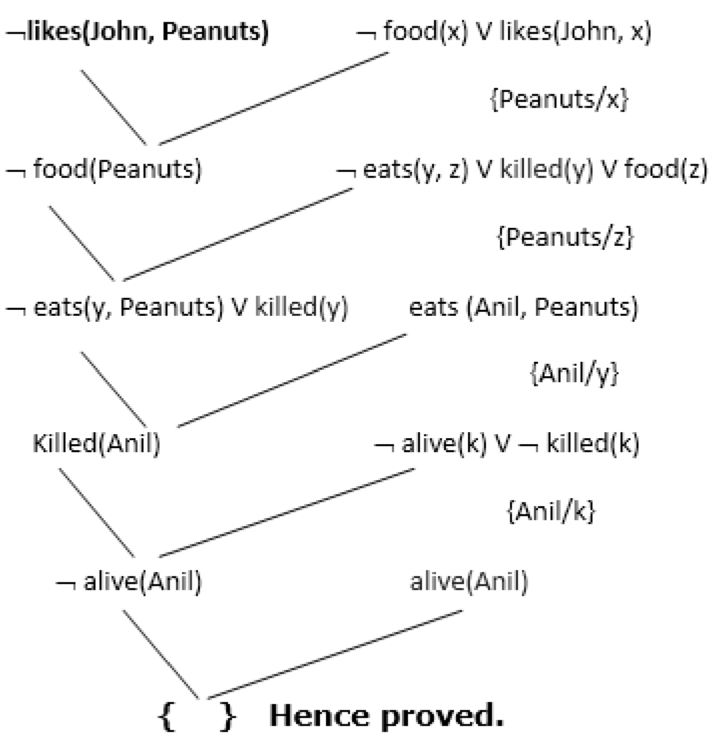
- **Distribute conjunction ∧ over disjunction ¬.**
  This step will not make any change in this problem.

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:

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



Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution graph:

- In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**
- In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .
- In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .
- In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .
- In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.

Reasoning in Artificial intelligence

In previous topics, we have learned various ways of knowledge representation in artificial intelligence. Now we will learn the various ways to reason on this knowledge using different logical schemes.

Reasoning:

The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or we can say, "**Reasoning is a way to infer facts from existing data**." It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

Types of Reasoning

In artificial intelligence, reasoning can be divided into the following categories:

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning
- Common Sense Reasoning

- Monotonic Reasoning
- Non-monotonic Reasoning

Note: Inductive and deductive reasoning are the forms of propositional logic.

1. Deductive reasoning:

Deductive reasoning is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.

Deductive reasoning is a type of propositional logic in AI, and it requires various rules and facts. It is sometimes referred to as top-down reasoning, and contradictory to inductive reasoning.

In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.

Deductive reasoning mostly starts from the general premises to the specific conclusion, which can be explained as below example.

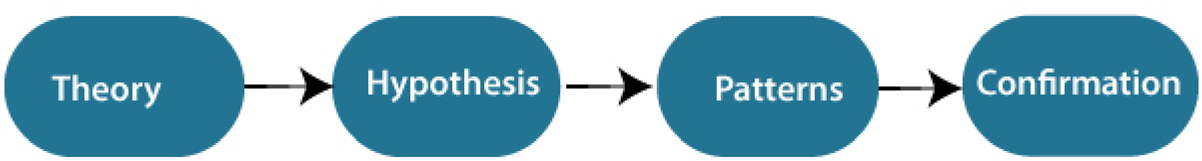
Example:

Premise-1: All the human eats veggies

Premise-2: Suresh is human.

Conclusion: Suresh eats veggies.

The general process of deductive reasoning is given below:



2. Inductive Reasoning:

Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with the series of specific facts or data and reaches to a general statement or conclusion.

Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.

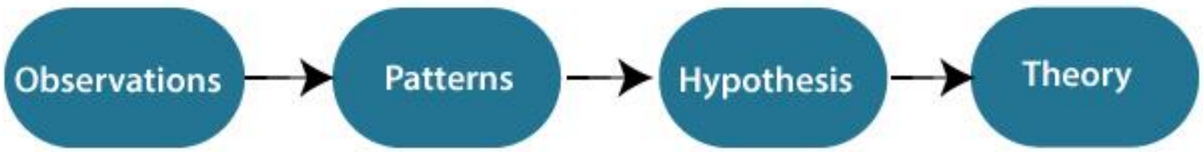
In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.

In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.

Example:

Premise: All of the pigeons we have seen in the zoo are white.

Conclusion: Therefore, we can expect all the pigeons to be white.



3. Abductive reasoning:

Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.

Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, the premises do not guarantee the conclusion.

Example:

Implication: Cricket ground is wet if it is raining

Axiom: Cricket ground is wet.

Conclusion It is raining.

4. Common Sense Reasoning

Common sense reasoning is an informal form of reasoning, which can be gained through experiences.

Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day.

It relies on good judgment rather than exact logic and operates on **heuristic knowledge** and **heuristic rules**.

Example:

1. **One person can be at one place at a time.**
2. **If I put my hand in a fire, then it will burn.**

The above two statements are the examples of common sense reasoning which a human mind can easily understand and assume.

5. Monotonic Reasoning:

In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Any theorem proving is an example of monotonic reasoning.

Example:

- **Earth revolves around the Sun.**

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

Advantages of Monotonic Reasoning:

- In monotonic reasoning, each old proof will always remain valid.
- If we deduce some facts from available facts, then it will remain valid for always.

Disadvantages of Monotonic Reasoning:

- We cannot represent the real world scenarios using Monotonic reasoning.
- Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
- Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

6. Non-monotonic Reasoning

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example: Let suppose the knowledge base contains the following knowledge:

- **Birds can fly**
- **Penguins cannot fly**
- **Pitty is a bird**

So from the above sentences, we can conclude that **Pitty can fly**.

However, if we add one another sentence into knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

Advantages of Non-monotonic reasoning:

- For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
- In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

Disadvantages of Non-monotonic Reasoning:

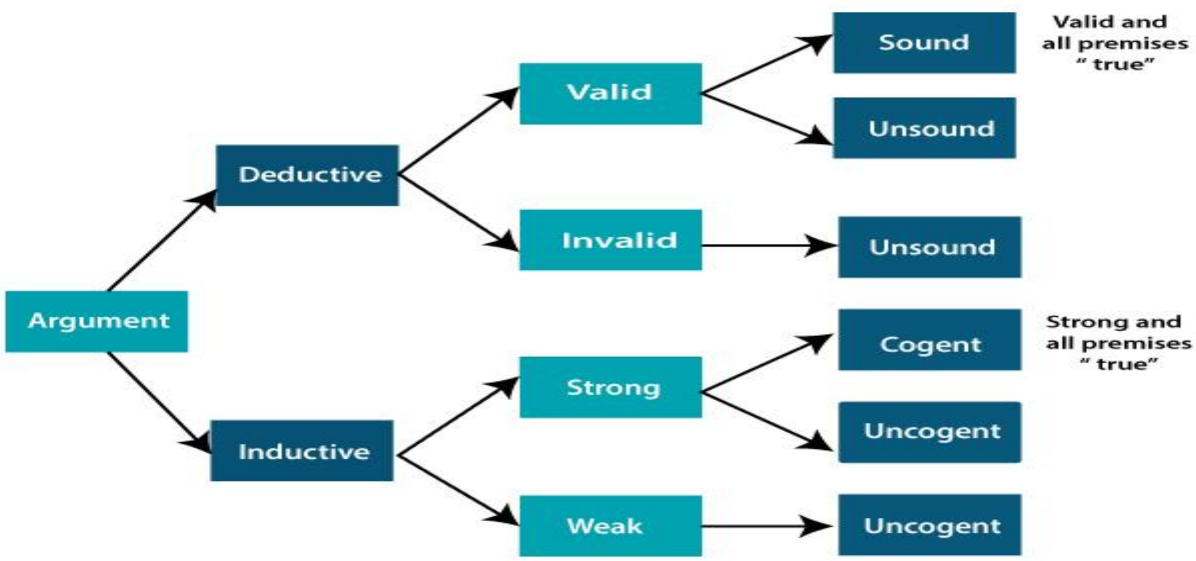
- In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
- It cannot be used for theorem **proving**.

Difference between Inductive and Deductive reasoning

Reasoning in artificial intelligence has two important forms, Inductive reasoning, and Deductive reasoning. Both reasoning forms have premises and conclusions, but both reasoning are contradictory to each other. Following is a list for comparison between inductive and deductive reasoning:

- Deductive reasoning uses available facts, information, or knowledge to deduce a valid conclusion, whereas inductive reasoning involves making a generalization from specific facts, and observations.
- Deductive reasoning uses a top-down approach, whereas inductive reasoning uses a bottom-up approach.
- Deductive reasoning moves from generalized statement to a valid conclusion, whereas Inductive reasoning moves from specific observation to a generalization.
- In deductive reasoning, the conclusions are certain, whereas, in Inductive reasoning, the conclusions are probabilistic.
- Deductive arguments can be valid or invalid, which means if premises are true, the conclusion must be true, whereas inductive argument can be strong or weak, which means conclusion may be false even if premises are true.

The differences between inductive and deductive can be explained using the below diagram on the basis of arguments:



Comparison Chart:

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	Inductive reasoning arrives at a conclusion by the process of generalization using specific facts or data.

	information.	
Approach	Deductive reasoning follows a top-down approach.	Inductive reasoning follows a bottom-up approach.
Starts from	Deductive reasoning starts from Premises.	Inductive reasoning starts from the Conclusion.
Validity	In deductive reasoning conclusion must be true if the premises are true.	In inductive reasoning, the truth of premises does not guarantee the truth of conclusions.
Usage	Use of deductive reasoning is difficult, as we need facts which must be true.	Use of inductive reasoning is fast and easy, as we need evidence instead of true facts. We often use it in our daily life.
Process	Theory→ hypothesis→ patterns→confirmation.	Observations-→patterns→hypothesis→Theory.
Argument	In deductive reasoning, arguments may be valid or invalid.	In inductive reasoning, arguments may be weak or strong.
Structure	Deductive reasoning reaches from general facts to specific facts.	Inductive reasoning reaches from specific facts to general facts.

Probabilistic reasoning in Artificial intelligence

Uncertainty:

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors
3. Equipment fault
4. Temperature variation
5. Climate change.

Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- **Bayes' rule**
- **Bayesian Statistics**

Note: We will learn the above two rules in later chapters.

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

1. $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
1. $P(A) = 0$, indicates total uncertainty in an event A.
1. $P(A) = 1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional probability:

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

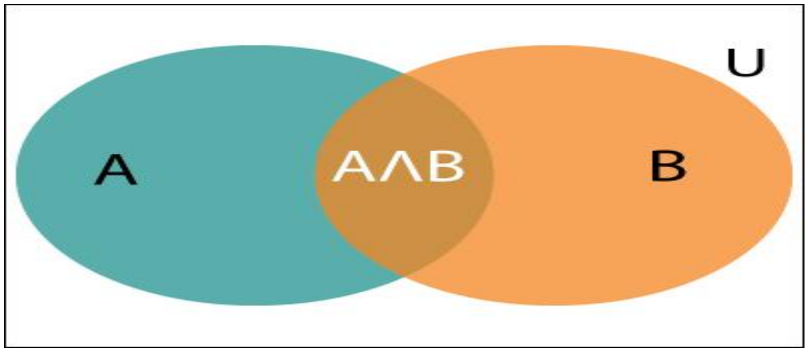
Where $P(A \wedge B)$= Joint probability of a and B

P(B)= Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A∧*B*) by P(B)**.



Example:

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

Solution:

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

Bayes' theorem in Artificial intelligence

Bayes' theorem:

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:

As from product rule we can write:

1. $P(A \wedge B)= P(A|B) P(B)$ or

Similarly, the probability of event B with known event A:

1. $P(A \wedge B)= P(B|A) P(A)$

Equating right hand side of both the equations, we will get:

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \quad \text{....(a)}$$

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.

It shows the simple relationship between joint and conditional probabilities. Here,

P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.

P(A) is called the **prior probability**, probability of hypothesis before considering the evidence

P(B) is called **marginal probability**, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i) * P(B|A_i)}{\sum_{i=1}^{k} P(A_i) * P(B|A_i)}$$

Where A_1, A_2, A_3,........, A_n is a set of mutually exclusive and exhaustive events.

Applying Bayes' rule:

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(*B*), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$\mathbf{P(cause|effect)} = \frac{P(effect|cause)\, P(cause)}{P(effect)}$$

Example-1:

Question: what is the probability that a patient has diseases meningitis with a stiff neck?

Given Data:

A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

- The Known probability that a patient has meningitis disease is 1/30,000.
- The Known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as:

P(a|b) = 0.8

P(b) = 1/30000

P(a)= .02

$$\mathbf{P(b|a)} = \frac{P(a|b)P(b)}{P(a)} = \frac{0.8*(\frac{1}{30000})}{0.02} = 0.001333333.$$

Hence, we can assume that 1 patient out of 750 patients has meningitis disease with a stiff neck.

Example-2:

Question: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.

Solution:

$$P(king|face) = \frac{P(Face|king) * P(King)}{P(Face)} \quad ....(i)$$

P(king): probability that the card is King= 4/52= 1/13

P(face): probability that a card is a face card= 3/13

P(Face|King): probability of face card when we assume it is a king = 1

Putting all values in equation (i) we will get:

$$P(king|face) = \frac{1 * (\frac{1}{13})}{(\frac{3}{13})} = 1/3,$$ **it is a probability that a face card is a king card.**

Application of Bayes' theorem in Artificial intelligence:

Following are some applications of Bayes' theorem:

- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem.

Bayesian Belief Network in artificial intelligence

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.

Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

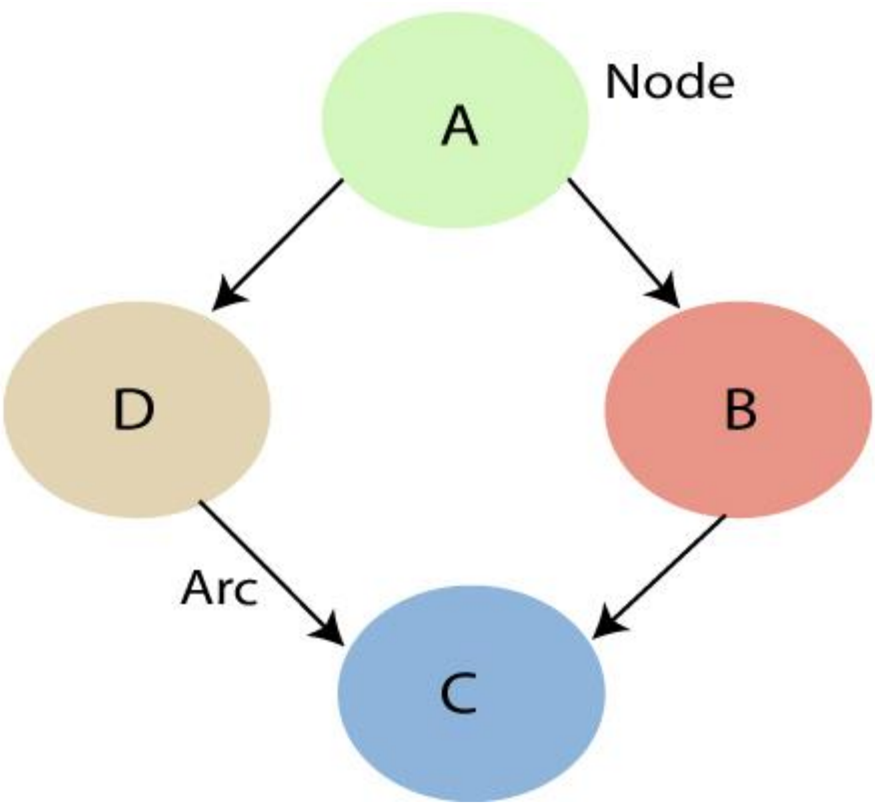
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other

- **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
- **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
- **Node C is independent of node A.**

Note: The Bayesian network graph does not contain any cyclic graph. Hence, it is known as a **directed acyclic graph or DAG**.

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution **$P(X_i | Parent(X_i))$**, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint probability distribution:

If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

$P[x_1, x_2, x_3,....., x_n]$, it can be written as the following way in terms of the joint probability distribution.

$= P[x_1| x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$

$= P[x_1| x_2, x_3,....., x_n]P[x_2|x_3,....., x_n]....P[x_{n-1}|x_n]P[x_n]$.

In general for each variable Xi, we can write the equation as:

$$P(X_i|X_{i-1},........., X_1) = P(X_i |Parents(X_i))$$

Explanation of Bayesian network:

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem:

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

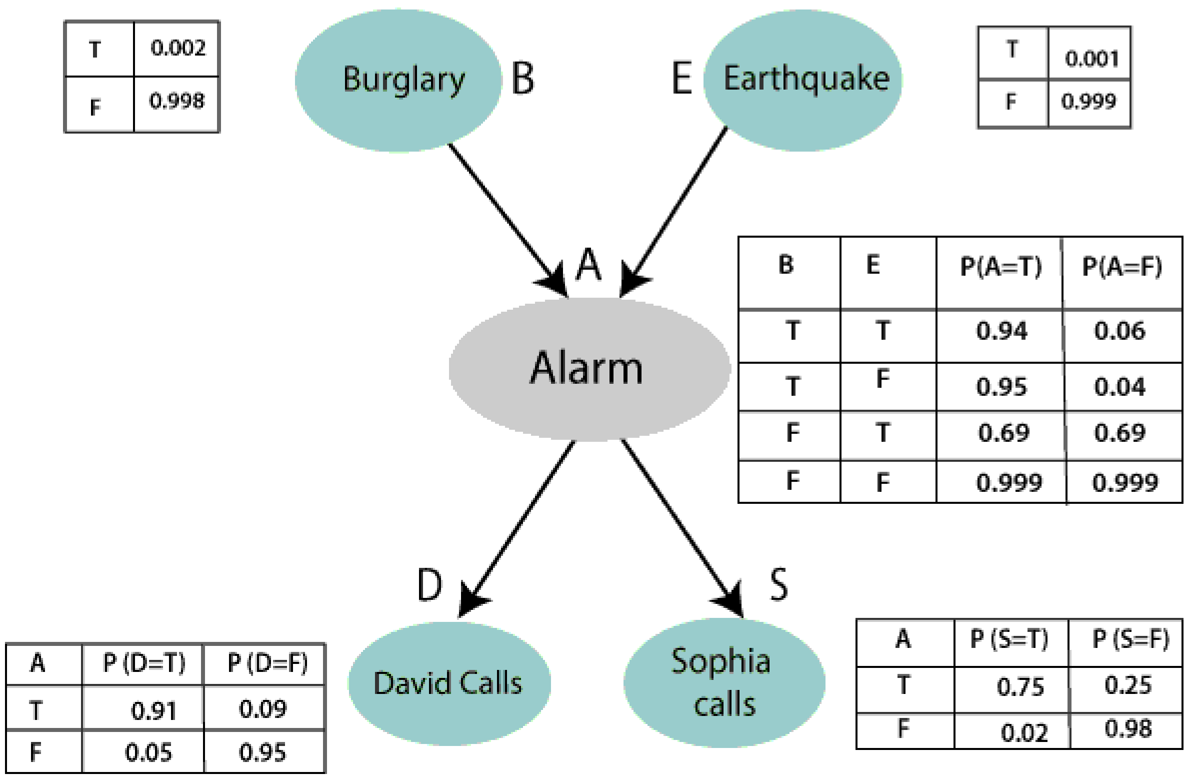
P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]



Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).

= 0.75* 0.91* 0.001* 0.998*0.999

= 0.00068045.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

The semantics of Bayesian Network:

There are two ways to understand the semantics of the Bayesian network, which is given below:

1. To understand the network as the representation of the Joint probability distribution.

It is helpful to understand how to construct the network.

2. To understand the network as an encoding of a collection of conditional independence statements.

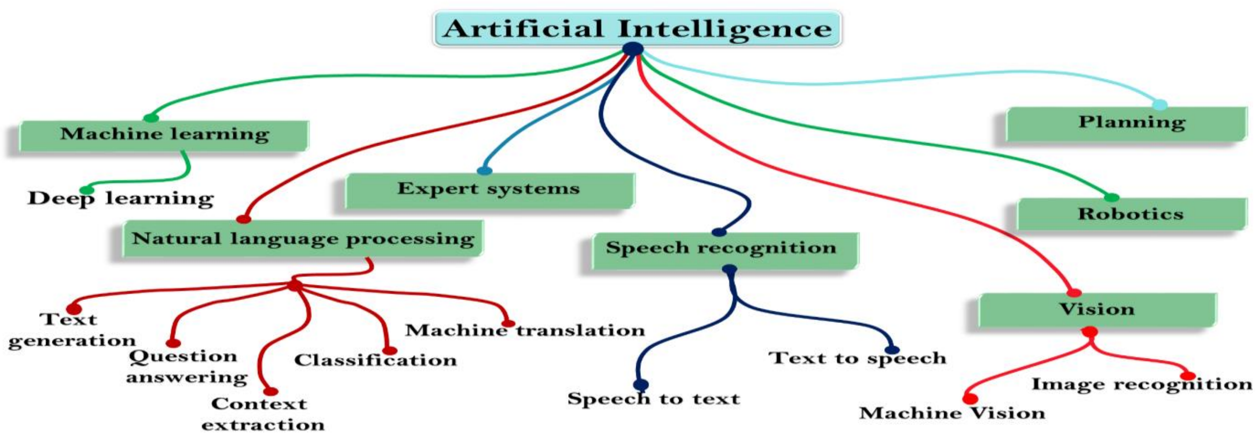
It is helpful in designing inference procedure.

Subsets of Artificial Intelligence

Till now, we have learned about what is AI, and now we will learn in this topic about various subsets of AI. Following are the most common subsets of AI:

- **Machine Learning**
- **Deep Learning**
- **Natural Language processing**
- **Expert System**
- **Robotics**
- **Machine Vision**
- **Speech Recognition**

> Note: Among all of the above, Machine learning plays a crucial role in AI. Machine learning and deep learning are the ways of achieving AI in real life.

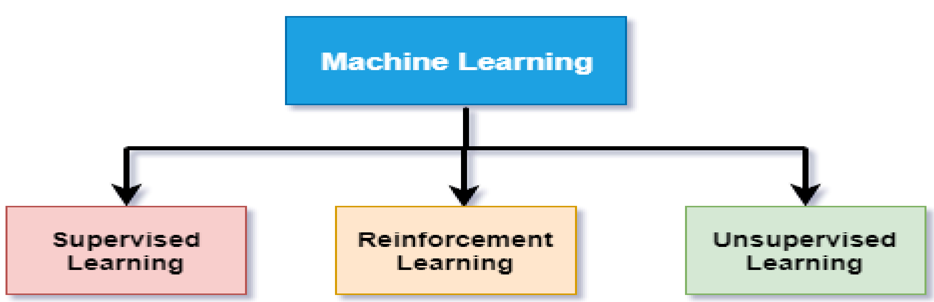


Machine Learning

Machine learning is a part of AI which provides intelligence to machines with the ability to automatically learn with experiences without being explicitly programmed.

- It is primarily concerned with the design and development of algorithms that allow the system to learn from historical data.
- Machine Learning is based on the idea that machines can learn from past data, identify patterns, and make decisions using algorithms.
- Machine learning algorithms are designed in such a way that they can learn and improve their performance automatically.
- Machine learning helps in discovering patterns in data.

Types of Machine Learning



Machine learning can be subdivided intothe main three types:

- **Supervised learning:**
  Supervised learning is a type of machine learning in which machine learn from known datasets (set of training examples), and then predict the output. A supervised

learning agent needs to find out the function that matches a given sample set.
Supervised learning further can be classified into two categories of algorithms:

a. **Classifications**
b. **Regression**

- **Reinforcement learning:**
  Reinforcement learning is a type of learning in which an AI agent is trained by giving some commands, and on each action, an agent gets a reward as a feedback.Using these feedbacks, agent improves its performance.
  Reward feedback can be positive or negative which means on each good action, agent receives a positive reward while for wrong action, it gets a negative reward.
  Reinforcement learning is of two types:

. **Positive Reinforcement learning**

a. **Negative Reinforcement learning**

- **Unsupervised learning:**
  Unsupervised learning is associated with learning without supervision or training. In unsupervised learning, the algorithms are trained with data which is neither labeled nor classified. In unsupervised learning, the agent needs to learn from patterns without corresponding output values.
  Unsupervised learning can be classified into two categories of algorithms:

. **Clustering**

a. **Association**

Natural Language processing

Natural language processing is a subfield of computer science and artificial intelligence. NLP enables a computer system to understand and process human language such as English.

NLP plays an important role in AI as without NLP, AI agent cannot work on human instructions, but with the help of NLP, we can instruct an AI system on our language. Today we are all around AI, and as well as NLP, we can easily ask Siri, Google or Cortana to help us in our language.

Natural language processing application enables a user to communicate with the system in their own words directly.

The Input and output of NLP applications can be in two forms:

- **Speech**
- **Text**

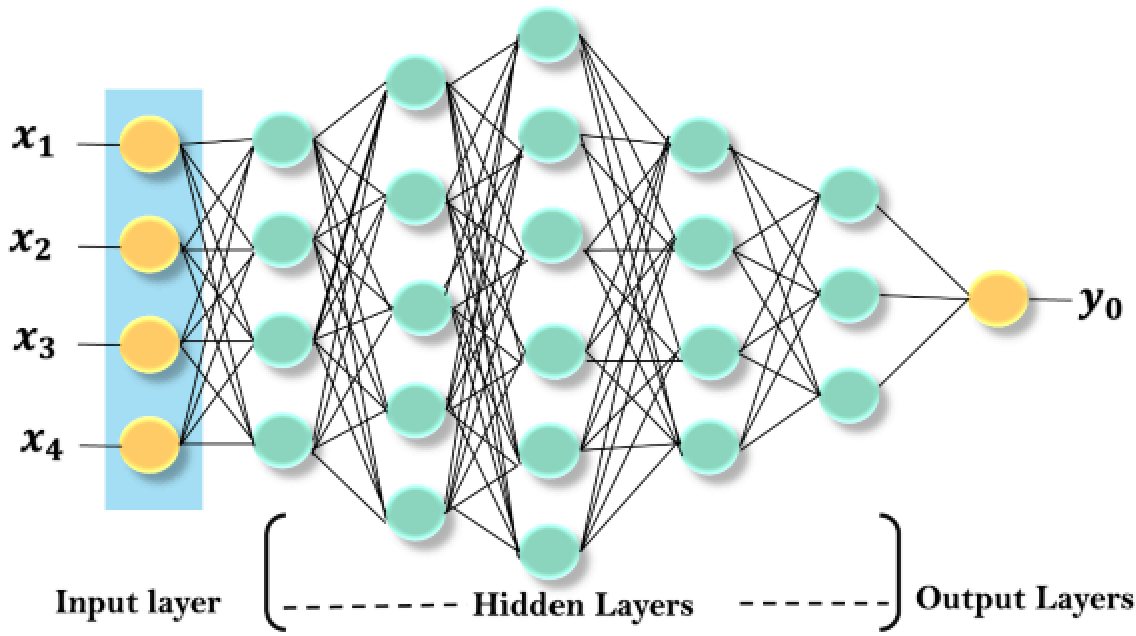
Deep Learning

Deep learning is a subset of machine learning which provides the ability to machine to perform human-like tasks without human involvement. It provides the ability to an AI agent to mimic the human brain. Deep learning can use both supervised and unsupervised learning to train an AI agent.

- Deep learning is implemented through neural networks architecture hence also called a **deep neural network**.
- Deep learning is the primary technology behind self-driving cars, speech recognition, image recognition, automatic machine translation, etc.
- The main challenge for deep learning is that it requires lots of data with lots of computational power.

How deep learning works:

- **Deep Learning Algorithms work on deep neural networks, so it is called deep learning. These deep neural networks are made of multiple layers.**
- **The first layer is called an Input layer, the last layer is called an output layer, and all layers between these two layers are called hidden layers.**
- **In the deep neural network, there are multiple hidden layers, and each layer is composed of neurons. These neurons are connected in each layer.**
- **The input layer receives input data, and the neurons propagate the input signal to its above layers.**
- **The hidden layers perform mathematical operations on inputs, and the performed data forwarded to the output layer.**
- **The output layer returns the output to the user.**



Expert Systems

- **An expert system is an application of artificial intelligence. In artificial intelligence, expert systems are the computer programs that rely on obtaining the knowledge of human experts and programming that knowledge into a system.**
- **Expert systems emulate the decision-making ability of human experts. These systems are designed to solve the complex problem through bodies of knowledge rather than conventional procedural code.**
- **One of the examples of an expert system is a Suggestion for the spelling error while typing in the Google search box.**
- **Following are some characteristics of expert systems:**
    - **High performance**
    - **Reliable**
    - **Highly responsive**
    - **Understandable**

Robotics

- Robotics is a branch of artificial intelligence and engineering which is used for designing and manufacturing of robots.

- Robots are the programmed machines which can perform a series of actions automatically or semi-automatically.
- AI can be applied to robots to make intelligent robots which can perform the task with their intelligence. AI algorithms are necessary to allow a robot to perform more complex tasks.
- Nowadays, AI and machine learning are being applied on robots to manufacture intelligent robots which can also interact socially like humans. One of the best examples of AI in robotics is Sophia robot.

Machine Vision

- Machine vision is an application of computer vision which enables a machine to recognize the object.
- Machine vision captures and analyses visual information using one or more video cameras, analog-to-digital conversations, and digital signal processing.
- Machine vision systems are programmed to perform narrowly defined tasks such as counting objects, reading the serial number, etc.
- Computer systems do not see in the same way as human eyes can see, but it is also not bounded by human limitations such as to see through the wall.
- With the help of machine learning and machine vision, an AI agent can be able to see through walls.

Speech Recognition:

Speech recognition is a technology which enables a machine to understand the spoken language and translate into a machine-readable format. It can also be said as automatic Speech recognition and computer speech recognition. **It is a way to talk with a computer, and on the basis of that command, a computer can perform a specific task**.

There is some speech recognition software which has a limited vocabulary of words and phrase. This software requires unambiguous spoken language to understand and perform specific task. Today's there are various software or devices which contains speech recognition technology such as Cortana, Google virtual assistant, Apple Siri, etc.

We need to train our speech recognition system to understand our language. In previous days, these systems were only designed to convert the speech to text, but now there are various devices which can directly convert speech into commands.

Speech recognition systems can be used in the following areas:

- **System control or navigation system**

- **Industrial application**
- **Voice dialing system**

There are two types of speech recognition

1. **Speaker Dependent**
2. **Speaker Independent**

Note: You will study the above topics in detail in later chapters.

What is an Expert System?

An Expert System is a computer program that is designed to solve complex problems with the help of database of expert knowledge. The system helps in decision making for complex problems using both facts and heuristics like a human expert.

It is called as an Expert System as it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain.

The expert system is a part of AI that solves the most complex issue as an expert by extracting the knowledge stored in its database. It performs this by extracting knowledge from its knowledge base and by applying inference rules according to the user queries. Expert systems have specific knowledge of a particular domain such as medicine, science, etc. The performance of the expert system is based on the expert's knowledge stored in the database. The more knowledge stored in the database, the more system improve its performance.

One of the common examples of an Expert System is a suggestion of spelling errors wile typing in the Google search box.

The first expert system was developed in the year **1970** and it was the first successful approach of artificial intelligence.

Examples of Expert Systems

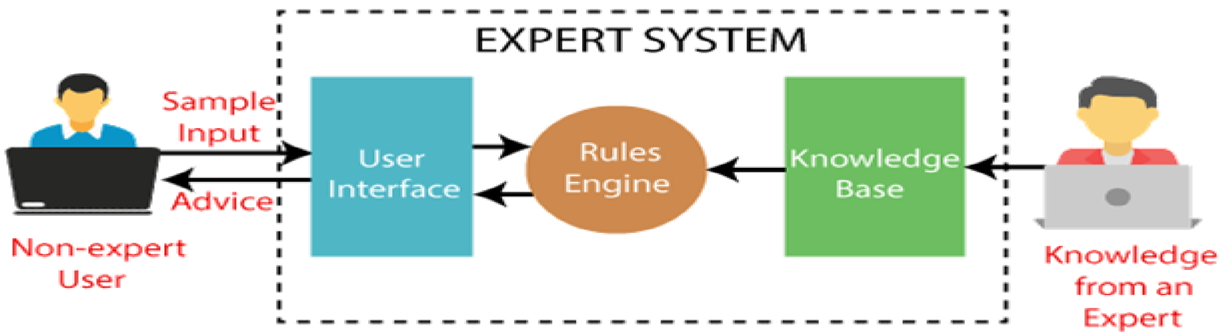
- **MYCIN:** It was backward chaining expert system that was designed to identify bacteria causing severe infections like **bacteraemia** and **meningitis**. It was also used for the recommendation of antibiotics and diagnosis of blood clotting diseases.
- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.

- **PXDES:** It is an expert system that is used to determine the type and degree of lung cancer. To determine the disease it takes picture of from the upper body which looks like the shadow. This shadow identifies the type and degree of harm.
- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

Building Blocks of Expert Systems

An Expert System mainly consists of three components:

- **User Interface**
- **Inference Engine**
- **Knowledge Base**



An expert system consists of the following components:

1. User Interface

The user interface is an important component of an Expert System. It interacts with the user, takes queries as an input in readable format, and passes it to the inference engine. After getting the response from the inference engine it displays the output to the user.

In other words, **it is an interface that helps a non-expert user to communicate with the expert system to find a solution**.

2. Inference Engine(Rules of Engine)

Inference engine is known as the brain of the expert system as it is the main processing unit of the Expert System. It applies logical rules or inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.

With the help of inference engine, Expert System extracts the knowledge from the knowledge base.

There are two types of inference engine:

- **Deterministic Inference engine:** The conclusion drawn from this type of inference engine is assured to be true. It is based on the facts and **rules**.
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in **conclusions**.

3. Knowledge Base

- The knowledge base is type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as a big container of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of Objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base:

- **Factual Knowledge:** The knowledge which is based on facts and accepted by knowledge engineers comes under the factual knowledge.
- **Heuristic Knowledge:** This knowledge is based on practice, ability to guess, evaluation,

Characteristics of Expert System

The characteristics of Expert Systems are given below:

- **High Performance:** The expert systems provide the high performance for solving any type of complex problem of a specific domain with high efficiency, and accuracy.
- **Understandable:** It provides response in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output as when user
- **Highly responsive:** ES provides result for any complex query within a very short period of time

Capabilities of Expert System

Below are some capabilities of an Expert System:

- **Advising:** It is capable to advise the human being for query of any domain from the particular ES.
- **Provide decision making capabilities:** It provide the capability of decision making in any domain such as for making any financial decision
- **Demonstrate a device:** To demonstrate a device
- **Problem solving:**
- **Explaining a problem:**
- **Predicting results:**

Development of an Expert System

Below are the main steps to develop an Expert System:

- **Identify the problem domain**
- **Design the System**
- **Prototype development**
- **Testing the and improving the prototype**
- **Complete the development of ES**
- **Maintaining the system**

Advantages of Expert System

Below are the some advantages of an Expert System:

- These systems are highly reproducible.
- They can be used for the risky places where the human presence is not safe.
- Error possibilities are less, if the KB contains correct knowledge.
- The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- They provide a very high speed to response for a particular query.

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being it cannot produce a creative output for different scenarios.
- Its maintenance and development cost is very high.
- Knowledge acquisition for designing is much difficult.
- For each domain we require a specific ES, which is one of the big limitations.
- It cannot learn from itself, and hence requires manual updates.

Applications of Expert System

- **In designing and manufacturing domain**
  ES can be broadly used for designing and manufacturing physical devices such as camera lenses and automobile.
- **In knowledge domain:**
  These systems are primarily used for the publishing the relevant knowledge to the users. The two popular ES used for this domain are Advisor and a tax advisor.
- **In finance domain:**
  In finance industries, it is used to detect any type of possible fraud, suspicious activity, and advice bankers to determine if they should provide loans for business or not.
- **In diagnosis and troubleshooting of devices**
  In medical diagnosis the ES system is used and it was the first area where these systems are used.
- **Planning and Scheduling**
  These systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.