UNIT - III

Representation of Knowledge: Knowledge representation issues, predicate logic- logic programming, semantic nets- frames and inheritance, constraint propagation, representing Knowledge using rules, rules based deduction systems. Reasoning under uncertainty, review of probability, Bayes' probabilistic interferences and dempstershafer theory

Representation of Knowledge

Procedural Knowledge:

Procedural Knowledge also known as interpretive knowledge, is the type of knowledge in which it clarifies how a particular thing can be accomplished. It is not so popular because it is generally not used.
Ex:

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

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

Declarative Knowledge:
Declarative Knowledge also known as Descriptive knowledge, is the type of knowledge which tells the basic knowledge about something and it is more popular than Procedural Knowledge.
It emphasize what to do something to solve a given problem.
Let's see it with an example:

```
var a=[1, 2, 3, 4, 5];
var b=a.map(function(number)
{
return number*1});
console.log(b);
```

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

Knowledge representation issues:

Knowledge representation issues encompass a range of challenges in the fields of artificial intelligence, cognitive science, and computer science.

Inference and Reasoning: Effective knowledge representation must support various forms of reasoning, such as deductive, inductive, and abductive reasoning, which can be computationally intensive.

Scalability: As the amount of knowledge grows, maintaining efficient storage and retrieval systems becomes crucial. Representations must be scalable without losing performance.

Dynamic Knowledge: Knowledge is not static; it evolves over time. Developing systems that can adapt to new information while retaining old knowledge is a complex issue.

Ethical and Social Implications: The way knowledge is represented can have ethical implications, particularly in areas like bias in AI systems, decision-making, and privacy.

1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarative sentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta-knowledge:

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

4. Heuristic knowledge:

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

5. Structural knowledge:

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

Sr. No.	Key	Procedural Knowledge	Declarative Knowledge
1	Name	Procedural knowledge is also termed as imperative knowledge.	Declarative knowledge is also termed as functional knowledge
2	Basis	Procedural knowledge revolves around **How to** the concept.	Declarative knowledge revolves around **What to** the concept.

Sr. No.	Key	Procedural Knowledge	Declarative Knowledge
3	Communication	Procedural knowledge is difficult to communicate.	Declarative knowledge is easily communicable.
4	Orientation	Procedural knowledge is process-oriented.	Declarative knowledge is data-oriented.
5	Validation	Validation is not very easy in procedural knowledge.	Validation is quite easy in declarative knowledge.
6	Debugging	Debugging is not very easy in procedural knowledge.	Debugging is quite easy in declarative knowledge.

Predicate logic- logic programming,

Predicate logic in artificial intelligence, also known as first-order logic or first order predicate logic in AI, is a formal system used in logic and mathematics to represent and reason about complex relationships and structures. It plays a crucial role in knowledge representation, which is a field within artificial intelligence and philosophy concerned with representing knowledge in a way that machines or humans can use for reasoning and problem-solving.

$\wedge$	*and* [conjunction]
$\vee$	*or* [disjunction]
$\Rightarrow$	*implies* [implication]
$\neg$	*not* [negation]
$\forall$	*For all*
$\exists$	*There exists*

Basic Components of Predicate Logic

1. **Predicates:** Predicates are statements or propositions that can be either true or false depending on the values of their arguments. They represent properties, relations, or characteristics of objects. For example, "IsHungry(x)" can be a predicate, where "x" is a variable representing an object, and the predicate evaluates to true if that object is hungry.

2. **Variables:** Variables are symbols that can take on different values. In predicate logic, variables are used to represent objects or entities in the domain of discourse. For example, "x" in "Is Hungry(x)" can represent any object in the domain, such as a person, animal, or thing.

3. **Constants:** Constants are specific values that do not change. They represent particular objects in the domain. For instance, in a knowledge base about people, "Alice" and "Bob" might be constants representing specific individuals.

4. **Quantifiers:** Quantifiers are used to specify the scope of variables in logical expressions. There are two main quantifiers in predicate logic

- Existential Quantifier (∃): Denoted as ∃, it indicates that there exists at least one object for which the statement within the quantifier is true. For example, "∃x IsHungry(x)" asserts that there is at least one object that is hungry.
- Universal Quantifier (∀): Denoted as ∀, it indicates that the statement within the quantifier is true for all objects in the domain. For example, "∀x IsHuman(x) → IsMortal(x)" asserts that all humans are mortal.

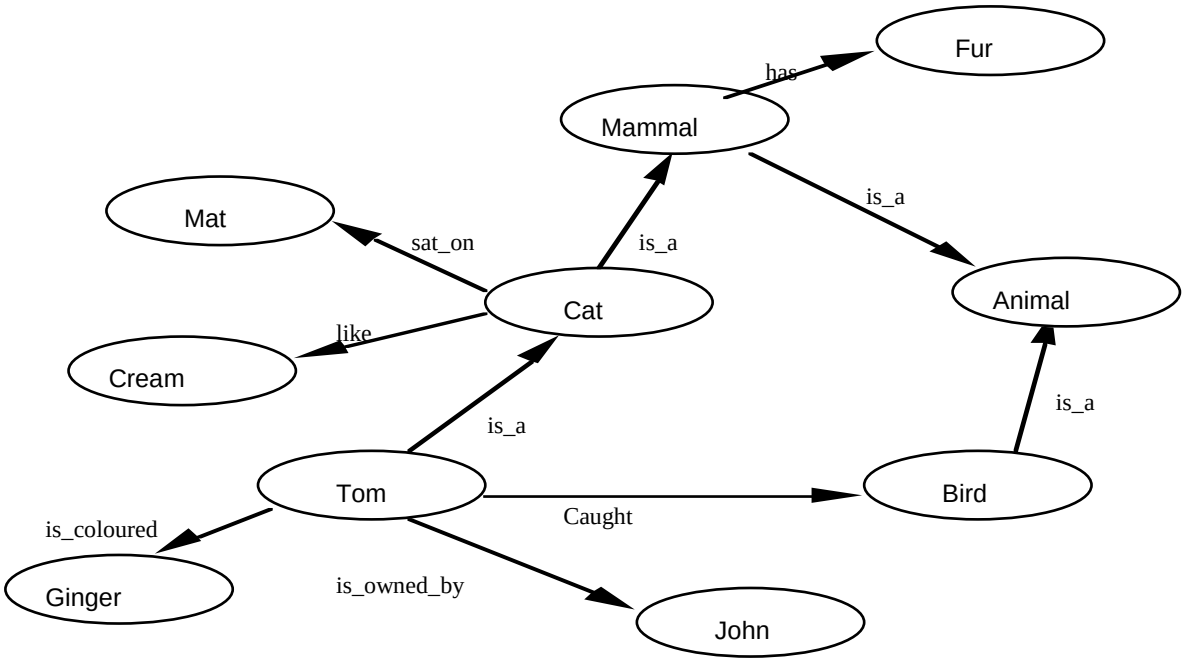
Logic programming

	Rule	First-order logic
R1	IF animal has hair THEN species is mammal	$\forall x \text{Has\_hair}(x) \Rightarrow \text{Mammal}(x)$
R2	IF animal gives milk THEN species is mammal	$\forall x\ \text{Gives\_milk}(x) \Rightarrow \text{Mammal}(x)$
R3	IF animal eats meat THEN species type is carnivore	$\forall x\ \text{Eats\_meat}(x) \Rightarrow \text{Carnivore}(x)$
R4	IF animal has pointed teeth AND animal has claws AND animal has forward eyes THEN species type is carnivore	$\forall x \text{Has\_pointed\_teeth}(x) \wedge \text{Has\_claws}(x) \wedge \text{Has\_forward\_eyes}(x) \Rightarrow \text{Carnivore}(x)$
R5	IF animal is mammal AND animal has hooves THEN mammal group is ungulate	$\forall x\ \text{Mammal}(x) \wedge \text{Has\_hooves}(x) \Rightarrow \text{Ungulate}(x)$
R6	IF species is mammal AND animal chews cud THEN mammal group is ungulate	$\forall x\ \text{Mammal}(x) \wedge \text{Chews\_cud}(x) \Rightarrow \text{Ungulate}(x)$
R7	IF species is mammal AND species type is carnivore AND animal has tawny colour AND animal has dark spots THEN animal is cheetah	$\forall x\ \text{Mammal}(x) \wedge \text{Carnivore}(x) \wedge \text{Has\_tawny\_colour}(x) \wedge \text{Has\_dark\_spots}(x) \Rightarrow \text{Cheetah}(x)$
R8	IF species is mammal AND species type is carnivore AND animal has tawny colour AND animal has black stripes THEN animal is tiger	$\forall x\ \text{Mammal}(x) \wedge \text{Carnivore}(x) \wedge \text{Has\_tawny\_colour}(x) \wedge \text{Has\_black\_stripes}(x) \Rightarrow \text{Tiger}(x)$
R9	IF species is ungulate AND animal has dark spots AND animal has long neck THEN animal is giraffe	$\forall x \text{Ungulate}(x) \wedge \text{Has\_dark\_spots}(x) \wedge \text{Has\_long\_neck}(x) \Rightarrow \text{Giraffe}(x)$
R10	IF species is ungulate AND animal has black stripes THEN animal is zebra	$\forall x \text{Ungulate}(x) \wedge \text{Has\_black\_stripes}(x) \Rightarrow \text{Zebra}(x)$

Semantic nets- frames and inheritance:

Semantic networks are an alternative to predicate logic as a form of knowledge representation. The idea is that we can store our knowledge in the form of a graph, with nodes representing objects in the world, and arcs representing relationships between those objects.

For example



Is intended to represent the data:

Tom is a cat.

Tom caught a bird.

Tom is owned by John.

Tom is ginger in colour.

Cats like cream.

The cat sat on the mat.

A cat is a mammal.

A bird is an animal.

All mammals are animals. Mammals have fur.

It is argued that this form of representation is closer to the way humans structure knowledge by building mental links between things than the predicate logic we considered earlier. Note in particular how all the information about a particular object is concentrated on the node representing that object, rather than scattered around several clauses in logic

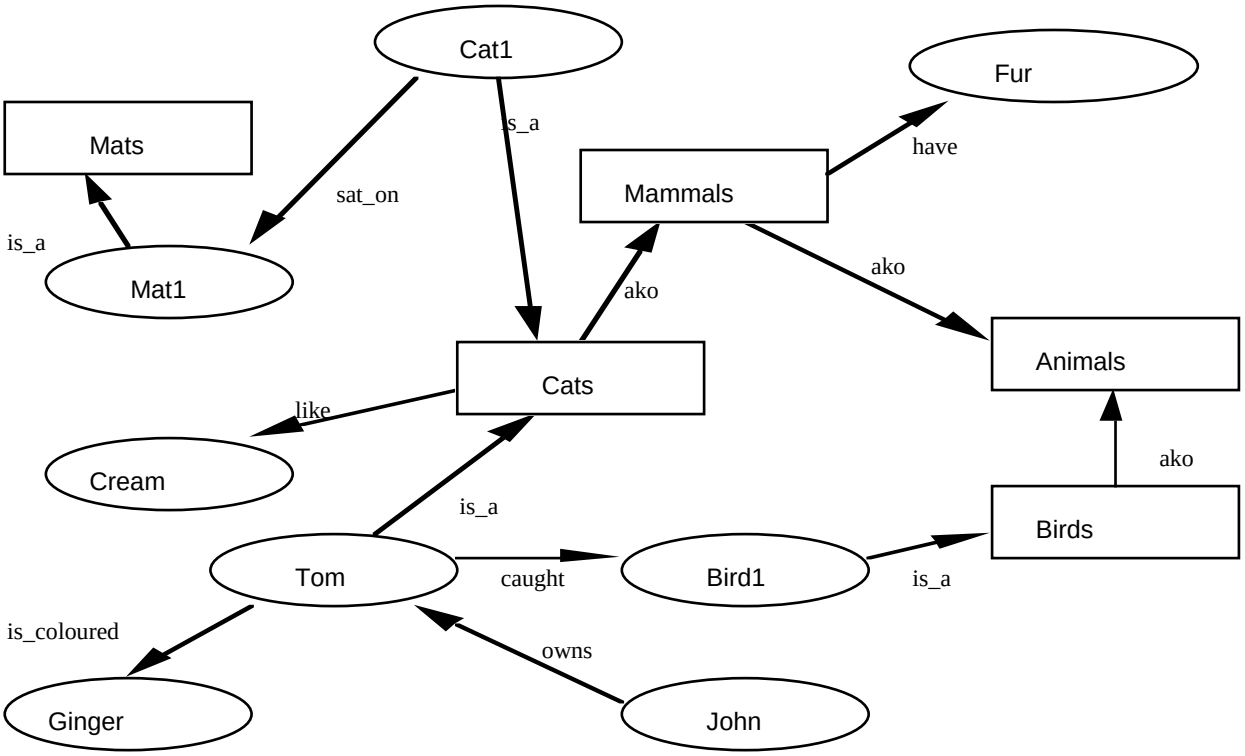
Tom is a cat is represented by Cat(Tom)

The cat sat on the mat is represented by $\exists x \exists y(Cat(x) \wedge Mat(y) \wedge SatOn(x,y))$

A cat is a mammal is represented by $\forall_{x(Cat(X) \rightarrow Mammal(x))}$

also the modification which causes the link labelled is_owned_by to be reversed in direction. This is in order to avoid links representing passive relationships. In general a passive sentence can be replaced by an active one, so “Tom is owned by John” becomes “John owns Tom”. In general the rule which converts passive to active in English converts sentences of the form “X is Yed by Z” to “Z Ys X”. This is just an example (though often used for illustration) of the much more general principle of looking beyond the immediate surface structure of a sentence to find its deep structure.

The revised semantic net is



that where we had an unnamed member of some class, we have had to introduce a node with an invented name to represent a particular member of the class. This is a process similar to the Skolemisation we considered previously as a way of dealing with existential quantifiers. For example, “Tom caught a bird” would be represented in logic by ∃x(bird(x)∧caught(Tom,x))

A direct Prolog representation can be used, with classes represented by predicates, thus:

cat (tom). cat(cat1).
mat(mat1).
sat_on(cat1,mat1).
bird(bird1). caught(tom,bird1).

like(X,cream) :– cat(X). mammal(X) :–
cat(X). has(X,fur) mammal(X). animal(X)
:mammal(X). animal(X) :
```
bird(X).
owns(john,tom). is_coloured(tom,ginger)
```

Constraint propagation:

Constraint propagation is a technique used in artificial intelligence and computer science to reduce the search space in constraint satisfaction problems (CSPs). It involves enforcing constraints throughout a network of variables, which helps to deduce the values that variables can take, thereby narrowing down possible solutions

Basics of Constraint Propagation

1. **Variables and Domains**: In a constraint satisfaction problem (CSP), you have a set of variables, each with a domain (possible values).
2. **Constraints**: These are rules that restrict the values the variables can take. For example, in a scheduling problem, two tasks might not be able to occur at the same time.
3. **Propagation**: When a variable's value is assigned or eliminated, the domains of related variables are updated. This can eliminate impossible values from their domains.

Techniques

- **Forward Checking**: When a variable is assigned a value, forward checking immediately removes inconsistent values from the domains of unassigned variables.
- **Arc Consistency**: This ensures that for every value in the domain of one variable, there exists a compatible value in the domain of another variable. If a value is found to have no compatible pair, it's removed.
- **Path Consistency**: This checks for consistency among triples of variables. If any pair of variables has a value that leads to inconsistencies with a third variable, those values are eliminated.

Example

Consider a simple CSP with two variables, X and Y, each with domains {1, 2, 3}, and a constraint $X \neq Y$. Constraint propagation will iteratively reduce the domains as follows:

- If X is assigned 1, then Y cannot be 1, so Y's domain becomes {2, 3}.

- If Y is then assigned 2, X cannot be 2, so X's domain is reduced to {1, 3}.
- This process continues until a stable state is reached.

Applications

- **Scheduling Problems**: Ensuring that tasks do not overlap.
- **Puzzle Solving**: Such as Sudoku, where constraints dictate valid placements
- **Resource Allocation**: Distributing limited resources among competing needs while satisfying constraints.

Representing Knowledge using rules

1. Knowledge about relationships in the world and
2. Knowledge about how to solve the problem using the content of the rules.

Procedural Knowledge

A representation in which the control information that is necessary to use the knowledge is embedded in the knowledge itself for e.g. computer programs, directions, and recipes

These indicate specific use or implementation

The real difference between declarative and procedural views of knowledge lies in where control information reside.

For example, consider the following

Man (Marcus)
Man (Caesar)
Person (Cleopatra) ∀x: Man(x) → Person(x)
Now, try to answer the question. ?Person(y)
The knowledge base justifies any of the following answers.
Y=Marcus
Y=Caesar
Y=Cleopatra

Declarative Knowledge

A statement in which knowledge specified, but the use to which that knowledge is to be put is not given

For example, laws, people's name; these are the facts which can stand alone, not dependent on other knowledge

So to use declarative representation, we must have a program that explains what is to do with the knowledge and how.

So logical assertions can view as a procedural representation of knowledge.

For example, a set of logical assertions can combine with a resolution theorem prover to give a complete program for solving problems but in some cases, the logical assertions can view as a program rather than data to a program.

2. Logic programming
Logic Programming – Representing Knowledge Using Rules
Logic programming is a programming paradigm in which logical assertions viewed as programs
These are several logic programming systems, PROLOG is one of them

But when we apply this transformation, any variables that occurred in negative literals and so now occur in the antecedent become existentially quantified, while the variables in the consequent are still universally quantified.

For example the
PROLOG clause P(x): – Q(x, y)
is equal to logical expression $\forall x: \exists y: Q(x, y) \rightarrow P(x)$.

Consider the following example:

1. Logical representation
$\forall x$: pet(x) $\wedge$ small (x) $\rightarrow$ apartmentpet(x)
$\forall x$: cat(x) $\wedge$ dog(x) $\rightarrow$ pet(x)
$\forall x$: poodle (x) $\rightarrow$ dog (x) small (x)
poodle (fluffy)

2. Prolog representation
apartmentpet (x) : pet(x), small (x)
pet (x): cat (x)
pet (x): dog(x)
dog(x): poodle (x)
small (x): poodle(x)
poodle (fluffy)

1. Reason forward from the initial state:

Step 1. Begin building a tree of move sequences by starting with the initial configuration at the root of the tree.

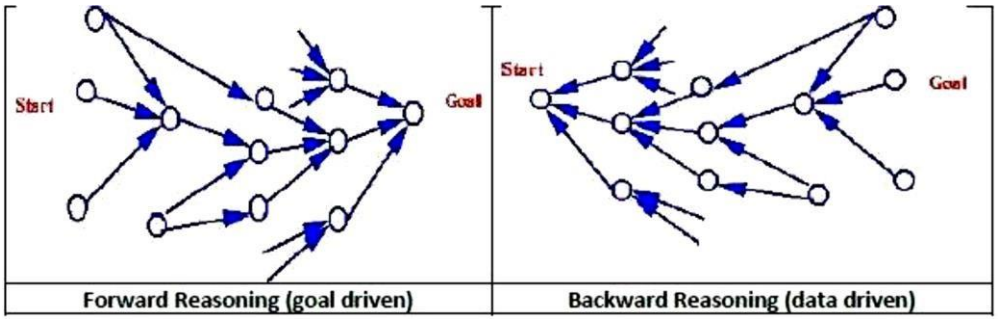
Step 2. Generate the next level of the tree by finding all rules whose left-hand side matches against the root node. The right-hand side is used to create new configurations.

2. Reasoning backward from the goal states:

Step 1. Begin building a tree of move sequences by starting with the goal node configuration at the root of the tree.

Step 2. Generate the next level of the tree by finding all rules whose right-hand side matches against the root node. The left-hand side used to create new configurations

Combining Forward and Backward Reasoning



Rules-based deduction systems

Rules-based deduction systems are a form of artificial intelligence that use logical rules to infer new information from existing facts. They are commonly used in expert systems, decision-making applications, and knowledge representation. Here's an overview of how they work and their key components

Key Components

1. Knowledge Base: This contains facts and rules about a specific domain. Facts are the data points (e.g., "The sky is blue"), while rules define relationships and logic (e.g., "If it is daytime, then the sky is blue").

2. Inference Engine: This is the core component that applies logical rules to the knowledge base to draw conclusions or make decisions. It processes the facts and rules to derive new facts.

3. Rules: Typically expressed in the form of "IF-THEN" statements:

IF (condition) **THEN** (consequence)

For example: "IF it is raining THEN the ground is wet."

4. Forward Chaining: A data-driven approach where the system starts with known facts and applies rules to infer new facts until no more inferences can be made.

5. **Backward Chaining**: A goal-driven approach where the system starts with a hypothesis and works backward to see if it can be supported by existing facts and rules.

Applications

1. **Expert Systems**: Such as medical diagnosis tools that suggest possible conditions based on symptoms

2. **Automated Reasoning**: In legal reasoning or financial analysis where rules govern decision-making.

3. Game AI: Where rules dictate character behaviour or game state management.

Advantages

1. Transparency: The rules are explicit, making the reasoning process easy to understand and validate.

2. Modularity: Rules can be added or modified without overhauling the entire system.

3. **Flexibility**: Can adapt to changes in knowledge without requiring extensive programming.

Limitations

1. Scalability: As the number of rules grows, the complexity of inference can increase significantly.

2. **Maintenance**: Keeping the knowledge base updated can be challenging, especially in rapidly changing domains.

3. **Performance**: Depending on the inference strategy, some systems may face performance bottlenecks.

Rules-based deduction systems leverage logical rules to derive conclusions, making them powerful tools in various fields

Reasoning under uncertainty

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.

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.

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.

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

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.

$0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
$P(A) = 0$, indicates total uncertainty in an event A.
$P(A) = 1$, 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(¬A) = probability of a not happening event.
- P(¬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.

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

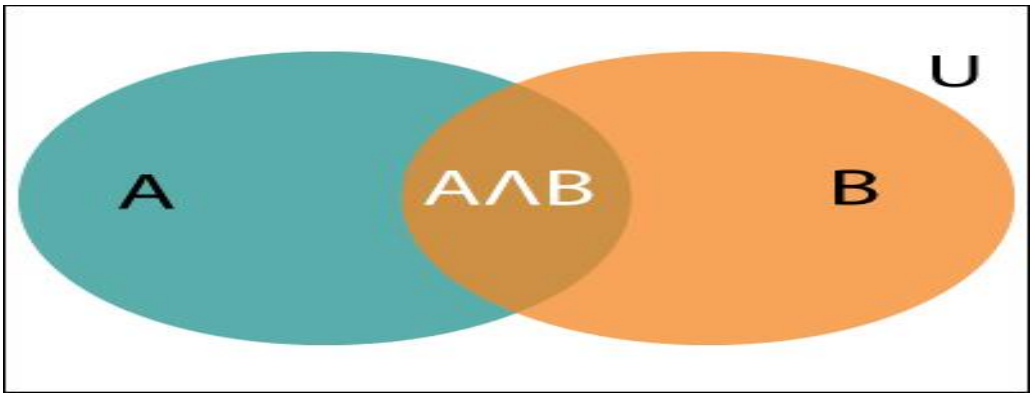
Where P ($A \wedge B$) = Joint probability of a and B

P (B) = Marginal probability of B.

f 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.

Review of probability

Basic Terms

• Experiment: Any action or process that yields a set of outcomes, like flipping a coin or rolling a die.

• Outcome: A possible result of an experiment. For example, getting heads in a coin flip.

• Sample **Space (S)**: The set of all possible outcomes of an experiment. For example, for a coin toss, S={heads, tails}

• Event **(E)**: A subset of the sample space. For instance, getting an even number when rolling a die (E={2,4,6}E

Types of Probability

Types of Probability

Classical Probability: Assumes that all outcomes in the sample space are equally likely. For example, the probability of rolling a 3 on a fair die is 1/6

Empirical (or Experimental) Probability: Based on observations or experiments. For example, if a coin lands on heads 70 times out of 100 flips, the probability of heads is 70/100=0.7

Calculating Probability

Basic Formula:

$$P(E) = \frac{\text{Number of favourable outcomes}}{\text{Total number of outcomes}}$$

• Complement **Rule**: The probability of an event not occurring is 1−P (E)1 - P(E)1−P(E).

• **Addition Rule**: For mutually exclusive events A and B, the probability of either A or B occurring is P(A∪B)=P(A)+P(B)P(A \cup B) = P(A) + P(B)P(A∪B)=P(A)+P(B).

• **Multiplication Rule**: For independent events and BBB, the probability of both occurring is P(A∩B)=P(A)·P(B)P(A \cap B) = P(A) \cdot P(B)P(A∩B)=P(A)·P(B).

Conditional Probability

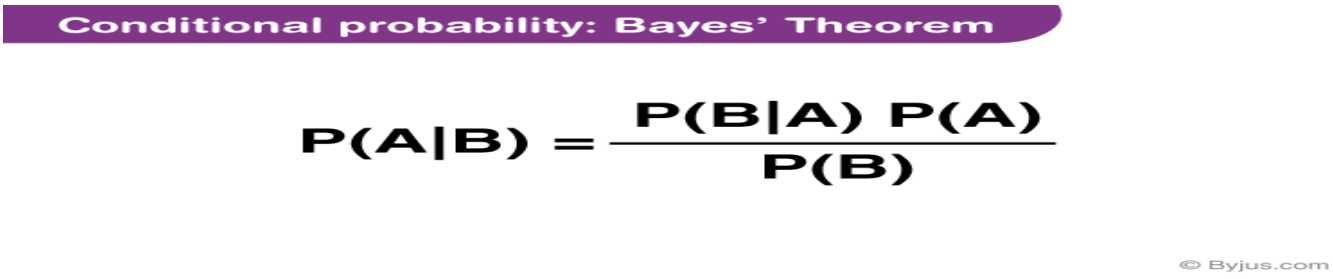
Definition: The probability of an event AAA occurring given that BBB has occurred, denoted P (A|B)P(A | B)P(A|B).

Formula:

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

Bayes' Theorem

A formula that allows us to update our probability estimates based on new information:

- 

- P(A|B) – the probability of event A occurring, given event B has occurred
- P(B|A) – the probability of event B occurring, given event A has occurred
- P(A) – the probability of event A
- P(B) – the probability of event B

Applications of Probability

• Statistics: Probability forms the foundation of statistical inference, enabling estimations and predictions based on data.
Risk **Assessment**: Used in fields like insurance, finance, and project management to assess the likelihood of adverse events.

Machine **Learning**: Probability helps model uncertainty in predictions and improve decision-making algorithms.

Bayes' probabilistic interferences and dempstershafer theory.

Bayes' Probabilistic Inference

Bayesian inference and Dumpsters-Shafer Theory (DST) are both frameworks for reasoning under uncertainty, but they differ in how they handle probabilities and interpret evidence.

Dempster Shafer Theory (DST) is an evidence theory, it combines all possible outcomes of the problem. Hence it is used to solve problems where there may be a chance that a piece of different evidence will lead to some different result.

The uncertainty in this model is given by:-

1. Consider all possible outcomes.
2. Belief will lead to belief in some possibility by bringing out some evidence. (What is this supposed to mean?)
3. Plausibility will make evidence compatible with possible outcomes.

Example:

Let us consider a room where four people are present, A, B, C, and D. Suddenly the lights go out and when the lights come back, B has been stabbed in the back by a knife, leading to his death. No one came into the room and no one left the room. We know that B has not committed suicide. Now we have to find out who the murderer is.

- Either {A} or {C} or {D} has killed him.
- Either {A, C} or {C, D} or {A, D} have killed him.
- Or the three of them have killed him i.e.; {A, C, D}
- None of them have killed him {o} (let's say).

There will be possible evidence by which we can find the murderer by the measure of plausibility.

Mass function m (K): It is an interpretation of m({K or B}) i.e; it means there is evidence for {K or B} which cannot be divided among more specific beliefs for K and B.

Belief in K: The belief in element K of Power Set is the sum of masses of the element which are subsets of K. This can be explained through an example Let's say K = {a, d, c}

Bel (K) = m(a) + m(d) + m(c) + m(a, d) + m(a, c) + m(d, c) + m(a, d, c)

Plausibility in K: It is the sum of masses of the set that intersects with K

Pl (K) = m(a) + m(d) + m(c) + m(a, d) + m(d, c) + m(a, c) + m(a, d, c)

Bayesian Inference

- **Concept**: Bayesian inference relies on Bayes' theorem to update the probability of a hypothesis as new evidence becomes available.

- **Bayes' Theorem**:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

$$= \frac{P(E \mid H)P(H)}{P(E \mid H)P(H) + P(E \mid \neg H)P(\neg H)}$$

$$= \frac{1}{1 + \left(\frac{1}{P(H)} - 1\right)\frac{P(E|\neg H)}{P(E|H)}}$$

- P (H|E): Posterior probability, the probability of hypothesis H given evidence E.

 P (E|H): Likelihood, the probability of observing E if H is true.

 P (H): Prior probability, the initial probability of H before considering E.

 P (E): Marginal likelihood, the total probability of E under all hypotheses.

Applications of Bayes' Theorem:

Medical **Diagnosis**: Updating the probability of a disease given symptoms or test results.

Machine **Learning**: In Bayesian learning, models update their predictions as they receive new data.

Natural **Language Processing**: Used in spam filtering, where the probability of an email being spam is updated based on words it contains.

Strengths and Limitations:

Strengths: Provides a structured way to update beliefs, allowing for the incorporation of prior knowledge and continuous refinement.

Limitations: Requires a well-defined prior probability, which can be subjective and hard to determine in the absence of strong initial information.

Characteristics of Dempster Shafer Theory:

- **Uncertainty Representation**: The DST is designed to handle situations where there is uncertainty of information and it provides a way to represent and reason incomplete evidence.

- **Conflict of Evidence**: The DST allows for the combination of multiple sources of evidence. It provides a rule, Dumpster's rule of combination, to combine belief functions from different sources.
- **Decision-Making** Ability: By deriving measures such as belief, probability and plausibility from the combined belief function it helps in decision making.

Advantages of Dempster Shafer Theory:

- As we add more information, the uncertainty interval reduces.
- DST has a much lower level of ignorance.
- Diagnose hierarchies can be represented using this.
- Person dealing with such problems is free to think about evidence

Disadvantages of Dempster Shafer Theory:

- In this, computation effort is high, as we have to deal with 2^n sets