GOVINDRAO WANJARI COLLEGE OF ENGINEERING & TECHNOLOGY
148,149 SALAI GODHANI, HUDKESWAR ROAD, NAGPUR

DEPARTMENT OF INFORMATION TECHNOLOGY

Sub Code: BTITC702 **Subject: Artificial Intelligence** **Sem: VII Sem**

Unit No. 4

Knowledge & reasoning: Knowledge representation issues, Representation & mapping, Approaches to knowledge representation, Issues in knowledge representation. Using predicate logic: Representing simple fact in logic, Representing instant & ISA relationship, Computable functions & predicates, Resolution, Natural deduction. Representing knowledge using rules: Procedural versus declarative knowledge, Logic programming, Forward versus backward reasoning, Matching, Control knowledge, Probabilistic reasoning: Representing knowledge in an uncertain domain, The semantics of Bayesian networks, Dempster-Shafer theory.

Knowledge Representation

• Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning

Describe Knowledge representation as following:

• Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.

• It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.

• It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Following are the kind of knowledge which needs to be represented in AI systems:

1. Object

• All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.

2. Events

• Events are the actions which occur in our world.

3. Performance

• It describe behavior which involves knowledge about how to do things.

4. Meta-knowledge

• It is knowledge above knowledge is known as Meta -knowledge.

5. Facts

• Facts are the truths about the real world and what we represent.

6. Knowledge-Base

• The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

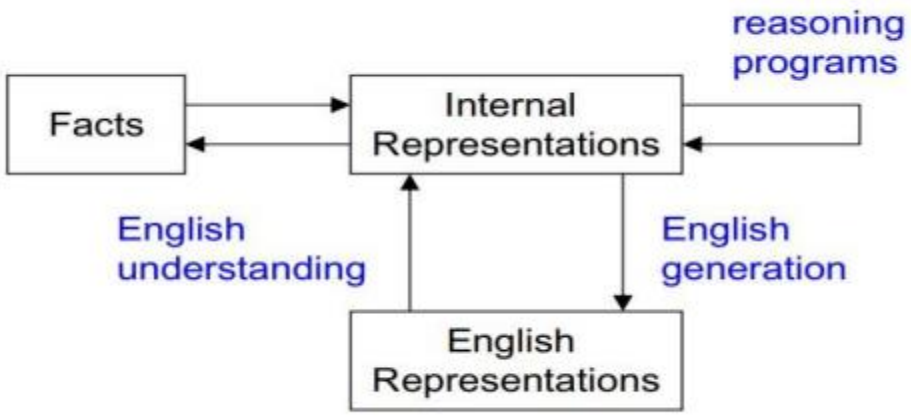
Representation and Mapping

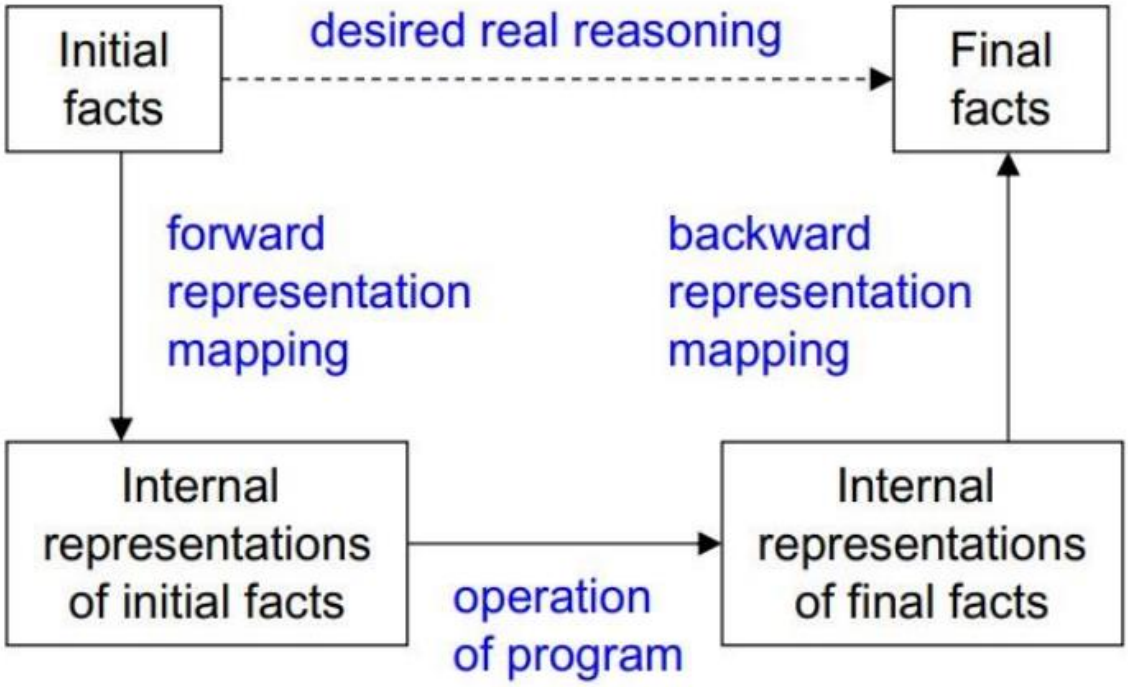
• Knowledge: Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Representation and Mapping

• Facts: things we want to represent.

• Representations of facts: things we can manipulate.


reasoning
programs
Facts
Internal
Representations
English
understanding
English
generation
English
Representations


Initial
facts
desired real reasoning
Final
facts
forward
representation
mapping
backward
representation
mapping
Internal
representations
of initial facts
operation
of program
Internal
representations
of final facts

Approaches to knowledge representation

1. Simple relational knowledge

• It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.

• This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.

• ExPlayer Weight Age

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable knowledge

• In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.

• In this approach, we apply inheritance property.

• Elements inherit values from other members of a class.

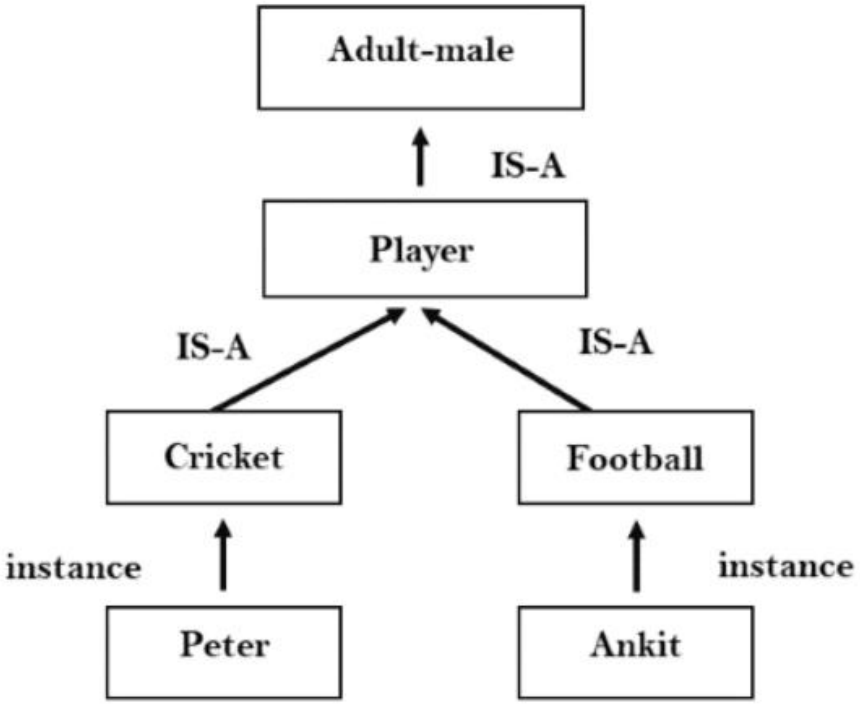
• This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.

• Every individual frame can represent the collection of attributes and its value.

• In this approach, objects and values are represented in Boxed nodes.

• We use Arrows which point from objects to their values.

Example:



3. Inferential knowledge

• Inferential knowledge approach represents knowledge in the form of formal logics.

• This approach can be used to derive more facts.

• It guaranteed correctness.

• Example: Let's suppose there are two statements:

1.Marcus is a man

2. All men are mortal

Then it can represent as;

man(Marcus)

∀x = man (x) ----------> mortal (x)s

4. Procedural knowledge

• Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.

• In this approach, one important rule is used which is If-Then rule.

• In this knowledge, we can use various coding languages such as LISP language and Prolog language.

• We can easily represent heuristic or domain-specific knowledge using this approach.

• But it is not necessary that we can represent all cases in this approach.

Requirements for knowledge Representation system

1. Representational Accuracy

• KR system should have the ability to represent all kind of required knowledge.

2. Inferential Adequacy

• KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.

3. Inferential Efficiency

• The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.

4. Acquisitional efficiency

• The ability to acquire the new knowledge easily using automatic methods.

Logic

Logic is concerned with the truth of statements about the world. Generally each statement is either *TRUE* or *FALSE*.

Logic includes :*Syntax* , *Semantics and Inference Procedure*.

◊ **Syntax :**

Specifies the *symbols* in the language about how they can be co mbined to form sentences. The **facts about the world are repre sented** as sentences in logic.

◊ **Semantic :**

Specifies how to assign a truth value to a sentence based on its *meaning* in the world. It Specifies what facts a sentence refers to. A fact is a claim about the world, and it may be *TRUE* or *FALSE*.

◊ **Inference Procedure :**

Specifies *methods* for computing new sentences from the existin g sentences.

Representing Simple Facts in Logic

Q Using propositional logic

–Real-world facts are represented as logical propositions wr itten as well-formed formulas (wff's)

–Example 1:

It is raining	RAINING
It is sunny	SUNNY
It is Windy	WINDY
If it is raining, then it is not sunny	RAINING → ¬SUNNY

Representing Simple Facts in Logic

– Example 2:

Socrates is a man	SOCRATESMAN
Plato is a man	PLATOMAN
All men are mortal	MORTALMAN

=> Since the assertions are separate, it is not possible to dra w any conclusion about similarities between Socrates and Pl ato.

Representing Simple Facts in Logic

It would be much better to represent these facts as

Socrates is a man	man(socrates)
Plato is a man	man(plato)
All men are mortal	mortalman

•This fails to capture the relationship between any individual being a man and that individual being a mortal.

•Therefore it is necessary to move to first order predicate logic as a way of representing knowledge because it permits representation o f things that cannot reasonably be represented in prepositional logi c.

•In predicate logic, real world facts are represented as statements written as wff's.

Representing Simple Facts in Logic

Q Propositional logic vs. predicate logic

–Using propositional logic

- Theorem proving is decidable
- Cannot represent objects and quantification

–Using predicate logic

- Can represent objects and quantification
- Theorem proving is semi-decidable

Representing Simple Facts in Logi

Consider the following set of sentences.

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar

Representing Simple Facts in Logic

1. Marcus was a man. man(Marcus)

2. Marcus was a Pompeian. Pompeian(Marcus)

3. All Pompeians were Romans.
□x: Pompeian(x) □ Roman(x)

4. Caesar was a ruler. ruler(Caesar)

Representing Simple Facts in Logic

5. All Romans were either loyal to Caesar or hated him.
In English the word 'or' means the logical inclusive-or and sometimes means the logical exclusive-or(XOR)

inclusive-or
□x: Roman(x) □ loyalto(x, Caesar) □ hate(x, Caesar)

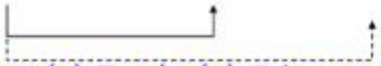
exclusive-or (XOR)
□x: Roman(x) □ (loyalto(x, Caesar) □ □ hate(x, Caesar))
□ (□ loyalto(x, Caesar) □ hate(x, Caesar))

Representing Simple Facts in Logic

6. Every one is loyal to someone.

□x →y: loyalto(x, y)

7. People only try to assassinate rulers they are not loyal to.



□x: □y: person(x) □ ruler(y) □ tryassassinate(x, y)

□ □loyalto(x, y)

8. Marcus tried to assassinate Caesar. tryassassinate(Marcus, Caesar)

Representing Simple Facts in Logic

To answer the question

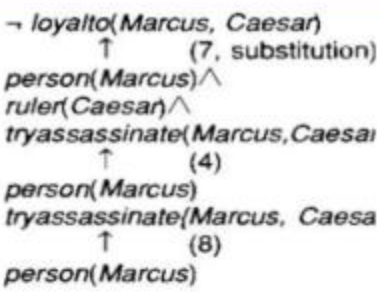
Was Marcus loyal to Caesar?

To produce a formal proof , reasoning backward from th e desired goal

□loyalto(Marcus, Caesar)

To prove the goal, rules of inference are to be used to trans form into another goal that in turn be transformed and s o on, until there are no insatisfied goals remaining.

Representing Simple Facts in Logi



•This attempt fails , since there is no way to satisfy the goal perso n(Marcus)with the statements available.
•The problem is although its known that Marcus was a man there is no way to conclude it.
•Therefore another representation is added namely

Representing Simple Facts in Logic

9.All men are people
□x:man(x)→person(x)
This satisfies the last goal and produce a proof that Marcus w as not loyal to ceasar.

Three important issues to be addressed in this process of converting English sentences to logical statements and the n using these statements to deduce new ones.

• Many English sentences are ambiguous. Choosing the corre ct interpretation may be difficult.

• There is often a choice of how to represent knowledge

• Obvious information may be necessary for reasoning

• We may not know in advance which statements to

•deduce (P or □P).

Representing Instance & Isa Relationships

•Attributes " IsA " and " Instance " support property inheritance and play important role in knowledge representation.The ways these two attributes "instance" and "isa", are logically expresse d are shown in the example below :

Example : A simple sentence like "Joe is a musician"

Here "is a" (called IsA) is a way of expressing what logically is called a class-instance relationship between the su bjects represented by the terms "Joe" and "musician".

◊ "Joe" is an instance of the class of things called "musician". " Joe" plays the role of instance,
"musician" plays the role of class in that sentence.

◊ Note : In such a sentence, while for a human there is no con fusion, but for computers each relationship have to be defined explicitly.

This is specified as: [Joe] IsA [Musician]
i.e., [Instance] IsA [Class]

Representing Instance & Isa Relationships

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $\forall x : Pompeian(x) \rightarrow Roman(x)$
4. $ruler(Caesar)$
5. $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$

1. $instance(Marcus, man)$
2. $instance(Marcus, Pompeian)$
3. $\forall x : instance(x, Pompeian) \rightarrow instance(x, Roman)$
4. $instance(Caesar, ruler)$
5. $\forall x : instance(x, Roman) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$

1. $instance(Marcus, man)$
2. $instance(Marcus, Pompeian)$
3. $isa(Pompeian, Roman)$
4. $instance(Caesar, ruler)$
5. $\forall x : instance(x, Roman) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. $\forall x : \forall y : \forall z : instance(x, y) \wedge isa(y, z) \rightarrow instance(x, z)$

Representing Instance & Isa Relationships

•The first part of the figure contains the representations, in which the class membership is represented with unary predicates, each corresponding to a class. Asserting that p(x) is true is equivalent to asserting that X is an instance of p.

•The second part uses the instance predicate explicitly. It is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs. The implication rule in statement 3 states that object is an instance of the subclass pompeian then it is an instance of the superclass Roman.

•The third part contains representations that use both the instance and isa predicates explicitly. Use of isa simplifies the representation of sentence 3 but requires one additional axiom.It describes how an instance relation and an isa relation combined to derive a new instance relation.

Computable Functions and Predicate

•All the simple facts can be expressed as combination of individual predicates such as

tryassassinate(Marcus, Caesar)

•This is fine if the number of facts is not very large. But suppose if we want to express simple facts such as greater-than and less-than relationships:

Q Computable predicates

greater-than(1, 0)	less-than(0, 1)
greater-than(2, 1)	less-than(1, 2)
greater-than(3, 2)	less-than(2, 3)
....	....

Computable Functions and Predicates

- It is not possible to write out the representation of each of the se facts individually as they are infinitely many of them.
- But if only the finite number of them are to be represented, it would be extremely inefficient to store explicitly a large set of statements instead so easily compute each one as we need it.
- Thus it becomes useful to argument the representations by c omputable predicates.
- A procedure will be invoked which is specified in addition to th e regular rules, that will evaluate it and return true or false.
- It is often useful to have computable functions as well as com putable predicates.
- Eg:gt(2+3,1), the value of the plus function is computed given the arguments 2 and 3 , and then send the arguments 5 and 1 to gt

Computable Functions and Predicate

Consider the following set of facts, again involving Marcus:

1. **Marcus was a man.**

   man(Marcus)

   Again we ignore the issue of tense.
2. Marcus was a Pompeian.

   Pompeian(Marcus)
3. Marcus was born in 40 A.D.

   born(Marcus, 40)
4. All men are mortal.

   $\forall x$: *man*$(x) \rightarrow$ *mortal*(x)
5. All Pompeians died when the volcano erupted in 79 A.D.

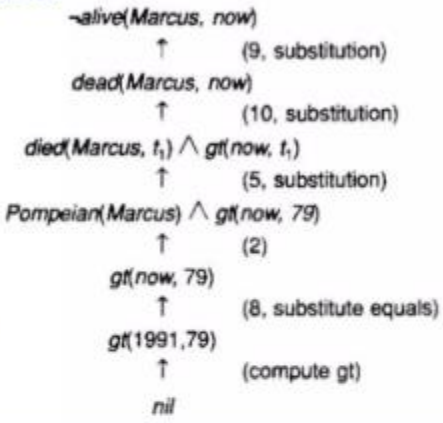
   erupted$(volcano, 79) \wedge \forall x$: [*Pompeian*$(x) \rightarrow$ *died*$(x, 79)$]

Computable Functions and Predicates

The question is “Is Marcus alive?” This question can be answered by proving

¬alive(Marcus,now)

The term nil at the end of each proof indicates that the lsit of conditions remaining to be proved is empty and so the proof has succeeded.



Resolution

From looking at the proofs, two things are clear.

•Even very simple conclusions can require many steps to p .

•A variety of processes such as matching, substitution are invol ved in the production of a proof.

Resolution

•This reduces some of the complexity because it operates on st atements that have first been converted to a single canonical fo rm.

•Resolution produces proofs by refutation.

•In other words to prove a statement(i.e show that it is valid) res olution attempts to show that the negation of the statement prod uces a contradiction with the known statements.

•This approach contrasts with the technique that generate proof s by chaining backward from the theorem to be proved to the ax iom.

Algorithm: Convert to Clause Form

1. Eliminate $\rightarrow$, using: $a \rightarrow b = \neg a \vee b$.
2. Reduce the scope of each $\neg$ to a single term, using:
   - $\neg (\neg p) = p$
   - deMorgan's laws: $\neg(a \wedge b) = \neg a \vee \neg b$
     $\neg(a \vee b) = \neg a \wedge \neg b$
   - $\neg \forall x\, P(x) = \exists x \neg P(x)$
   - $\neg \exists x\, P(x) = \forall x \neg P(x)$
3. Standardize variables.
4. Move all quantifiers to the left of the formula without changing their r elative order.
5. Eliminate existential quantifiers by inserting Skolem functions.
6. Drop the prefix.
7. Convert the expression into a conjunction of disjuncts, using associa tivity and distributivity.
8. Create a separate clause for each conjunct.
9. Standardize apart the variables in the set of clauses generated in st ep 8, using the fact that: $(\forall x: P(x) \wedge Q(x)) = \forall x: P(x) \wedge \forall x: Q(x)$

Resolution

Algorithm: Convert to Clause Form

1. Eliminate $\rightarrow$, using the fact that $a \rightarrow b$ is equivalent to $\neg a \vee b$. Performing this transformation on the wff given above yields

   $\forall x : \neg[Roman(x) \wedge know(x, Marcus)] \vee$
   $[hate(x, Caesar) \vee (\forall y : \neg(\exists z : hate(y, z)) \vee thinkcrazy(x, y))]$

2. Reduce the scope of each $\neg$ to a single term, using the fact that $\neg(\neg p) = p$, deMorgan's laws [which say that $\neg(a \wedge b) = \neg a \vee \neg b$ and $\neg(a \vee b) = \neg a \wedge \neg b$], and the standard correspondences between quantifiers [$\neg\forall x : P(x) = \exists x : \neg P(x)$ and $\neg\exists x : P(x) = \forall x : \neg P(x)$]. Performing this transformation on the wff from step 1 yields

   $\forall x : [\neg Roman(x) \vee \neg know(x, Marcus)] \vee$
   $[hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee thinkcrazy(x, y))]$

3. Standardize variables so that each quantifier binds a unique variable. Since variables are just dummy names, this process cannot affect the truth value of the wff. For example, the formula

   $\forall x : P(x) \vee \forall x : Q(x)$

   would be converted to

   $\forall x : P(x) \vee \forall y : Q(y)$

   This step is in preparation for the next.

Resolution

4. Move all quantifiers to the left of the formula without changing their relative order. This is possible since there is no conflict among variable names. Performing this operation on the formula of step 2, we get

$$\forall x : \forall y : \forall z : [\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkcrazy(x, y))]$$

   At this point, the formula is in what is known as *prenex normal form*. It consists of a *prefix* of quantifiers followed by a *matrix*, which is quantifier-free.
5. Eliminate existential quantifiers. A formula that contains an existentially quantified variable asserts that there is a value that can be substituted for the variable that makes the formula true. We can eliminate the quantifier by substituting for the variable a reference to a function that produces the desired value. Since we do not necessarily know how to produce the value, we must create a new function name for every such replacement. We make no assertions about these functions except that they must exist. So, for example, the formula

$$\exists y : President(y)$$

   can be transformed into the formula

$$President(S1)$$

Resolution

where S1 is a function with no arguments that somehow produces a value that satisfies President. If existential quantifiers occur within the scope of universal quantifiers, then the value that satisfies the predicate may depend on the values of the universally quantified variables. For example, in the formula

$$\forall x : \exists y : father\text{-}of(y, x)$$

the value of y that satisfies *father-of* depends on the particular value of x. Thus we must generate functions with the same number of arguments as the number of universal quantifiers in whose scope the expression occurs. So this example would be transformed into

Resolution

where SI is a function with no arguments that somehow produces a value that satisfies President. If existential quantifiers occur within the scope of universal quantifiers, then the value that satisfies the predicate may depend on the values of the universally quantified variables. For example, in the formula

$\forall x : \exists y : father\text{-}of(x,x)$

the value of y that satisfies *father-of* depends on the particular value of x. Thus we must generate functions with the same number of arguments as the number of universal quantifiers in whose scope the expression occurs. So this example would be transformed into

$\forall x : father\text{-}of(S2(x),x))$

These generated functions are called *Skolem functions*. Sometimes ones with no arguments are called *Skolem constants*.

Resolution

where SI is a function with no arguments that somehow produces a value that satisfies President. If existential quantifiers occur within the scope of universal quantifiers, then the value that satisfies the predicate may depend on the values of the universally quantified variables. For example, in the formula

$\forall x : \exists y : father\text{-}of(x,x)$

the value of y that satisfies *father-of* depends on the particular value of x. Thus we must generate functions with the same number of arguments as the number of universal quantifiers in whose scope the expression occurs. So this example would be transformed into

$\forall x : father\text{-}of(S2(x),x))$

These generated functions are called *Skolem functions*. Sometimes ones with no arguments are called *Skolem constants*.

Resolution

6. Drop the prefix. At this point, all remaining variables are universally quantified, so the prefix can just be dropped and any proof procedure we use can simply assume that any variable it sees is universally quantified. Now the formula produced in step 4 appears as

$$[\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkcrazy(x, y))]$$

7. Convert the matrix into a conjunction of disjuncts. In the case of our example, since there are no *and's*, it is only necessary to exploit the associative property of *or* [i.e., $(a \wedge b) \vee c = (a \vee c) \wedge (b \wedge c)$] and simply remove the parentheses, giving

Resolution

8. Create a separate clause corresponding to each conjunct. In order for a wff to be true, all the clauses that are generated from it must be true. If we are going to be working with several wff's, all the clauses generated by each of them can now be combined to represent the same set of facts as were represented by the original wff's.
9. Standardize apart the variables in the set of clauses generated in step 8. By this we mean rename the variables so that no two clauses make reference to the same variable. In making this transformation, we rely on the fact that

$$(\forall x : P(x) \wedge Q(x)) = \forall x : P(x) \wedge \forall x : Q(x)$$

Thus since each clause is a separate conjunct and since all the variables are universally quantified, there need be no relationship between the variables of two clauses, even if they were generated from the same wff.

Resolution

- After applying this entire procedure to a set of wff's, a set of clause s will be obtained each of which is a disjunction of literals.
- These clauses can now be exploited by the resolution procedure t o generate proofs.

Natural Deduction

Q Problems with resolution

–The heuristic information contained in the original stat ements can be lost in the transformation

□x: judge(x) □ □ crooked(x) □ educated(x)

□ judge(x) v crooked(x) v educated(x)

–People do not think in resolution

Q Natural deduction: A way of doing machine theorem proving that corresponds more closely to processes us ed in human theorem proving

Procedural and Declarative Knowledge

1. Procedural Knowledge

- The **Procedural knowledge** is a type of knowledge where the essential control information that is required to use the information is integrated in the knowledge itself.
- It also used with an interpreter to employ the knowledge which follows the instructions given in the knowledge.

- Ex - It can include a group of logical assertions merged with a resolution theorem prove to provide an absolute program for solving problems. Here, the implied income tax of an employee salary can be thought of as a procedural knowledge as it would require a process to calculate it as given below.

- So, this is how the tax of an employee is calculated by following a lengthy process instead of just collecting facts.

= GTI (Gross Taxable Income) = Annual Salary of an employee - (Standard deduction + deduction under section 80C)
= Tax computed on GTI (according to slab rate) = A,
= Rebate under section 87A = B;
= Total tax = A - Less B + add: health and education Cess @ 4% on (A-B)

2. Declarative Knowledge

- A **Declarative knowledge** is where only knowledge is described but not the use to which the knowledge is employed is not provided.
- So, in order to use this declarative knowledge, we need to add it with a program that indicates what is to be done to the

knowledge and how it is to be done.

Difference the Procedural and Declarative Knowledge

PROCEDURAL KNOWLEDGE	DECLARATIVE KNOWLEDGE
It is also known as Interpretive knowledge.	It is also known as Descriptive knowledge.
Procedural Knowledge means how a particular thing can be accomplished	While Declarative Knowledge means basic knowledge about something.
Procedural Knowledge is generally not used means it is not more popular.	Declarative Knowledge is more popular.
Procedural Knowledge can't be easily communicate.	Declarative Knowledge can be easily communicate
Procedural Knowledge is generally process oriented in nature	Declarative Knowledge is data oriented in nature.
In Procedural Knowledge debugging and validation is not easy.	In Declarative Knowledge debugging and validation is easy.

Forward Reasoning

- The solution of a problem generally includes the initial data and facts in order to arrive at the solution. These unknown facts and information is used to deduce the result

- For example, while diagnosing a patient the doctor first check the symptoms and medical condition of the body such as temperature, blood pressure, pulse, eye colour, blood, etcetera. After that, the patient symptoms are analysed and compared against the predetermined symptoms. Then the doctor is able to provide the medicines according to the symptoms of the patient. So, when a solution employs this manner of reasoning, it is known as **forward reasoning**.

Steps that are followed in the forward reasoning

1. In the first step, the system is given one or more than one constraints.
2. Then the rules are searched in the knowledge base for each constraint. The rules that fulfill the condition are selected(i.e., IF part).
3. Now each rule is able to produce new conditions from the conclusion of the invoked one. As a result, THEN part is again included in the existing one.
4. The added conditions are processed again by repeating step 2. The process will end if there is no new conditions exist.

Backward Reasoning

- The **backward reasoning** is inverse of forward reasoning in which goal is analysed in order to deduce the rules, initial facts and data.

- We can understand the concept by the similar example given in the above definition, where the doctor is trying to diagnose the patient with the help of the inceptive data such as symptoms. However, in this case, the patient is experiencing a problem in his body, on the basis of which the doctor is going to prove the symptoms. This kind of reasoning comes under backward reasoning.

Steps that are followed in the backward reasoning

1. Firstly, the goal state and the rules are selected where the goal state reside in the THEN part as the conclusion.
2. From the IF part of the selected rule the sub goals are made to be satisfied for the goal state to be true.
3. Set initial conditions important to satisfy all the sub goals.
4. Verify whether the provided initial state matches with the established states. If it fulfills the condition then the goal is the solution otherwise other goal state is selected.

Difference between backward chaining and forward chaining

Forward Chaining	Backward Chaining
It starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	It starts from the goal and works backward through inference rules to find the required facts that support the goal.
It is a bottom-up approach	It is a top-down approach
It is known as data-driven inference technique as we reach to the goal using the available data.	It is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
It applies a breadth-first search strategy.	It applies a depth-first search strategy.
It tests for all the available rules	It tests only for few required rules.
It is suitable for the planning, monitoring, control, and interpretation application.	It is suitable for diagnostic, prescription, and debugging application.
It can generate an infinite number of possible conclusions.	It generates a finite number of possible conclusions.
It operates in the forward direction.	It operates in the backward direction.

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**

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. $0 \le P(A) \le 1$, 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(¬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.

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:

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

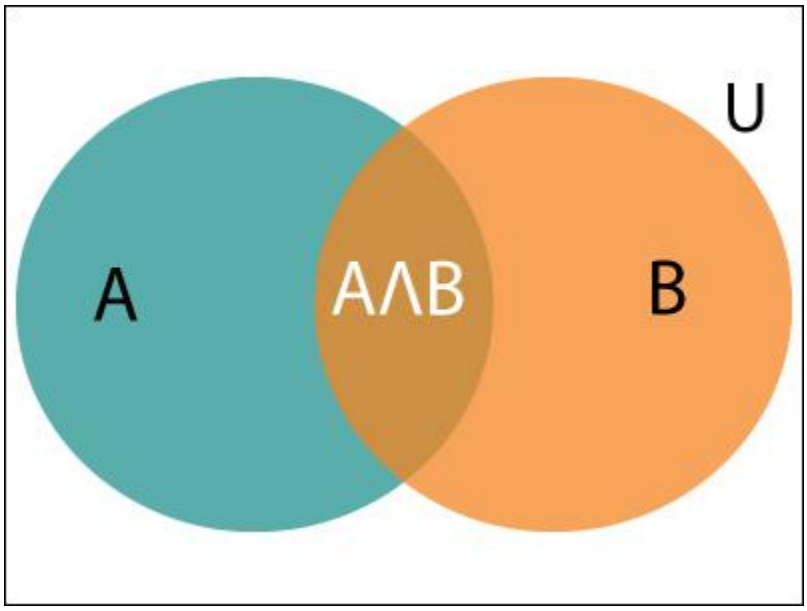
Where $P(A \wedge B)$= Joint probability of a and B

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.

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.

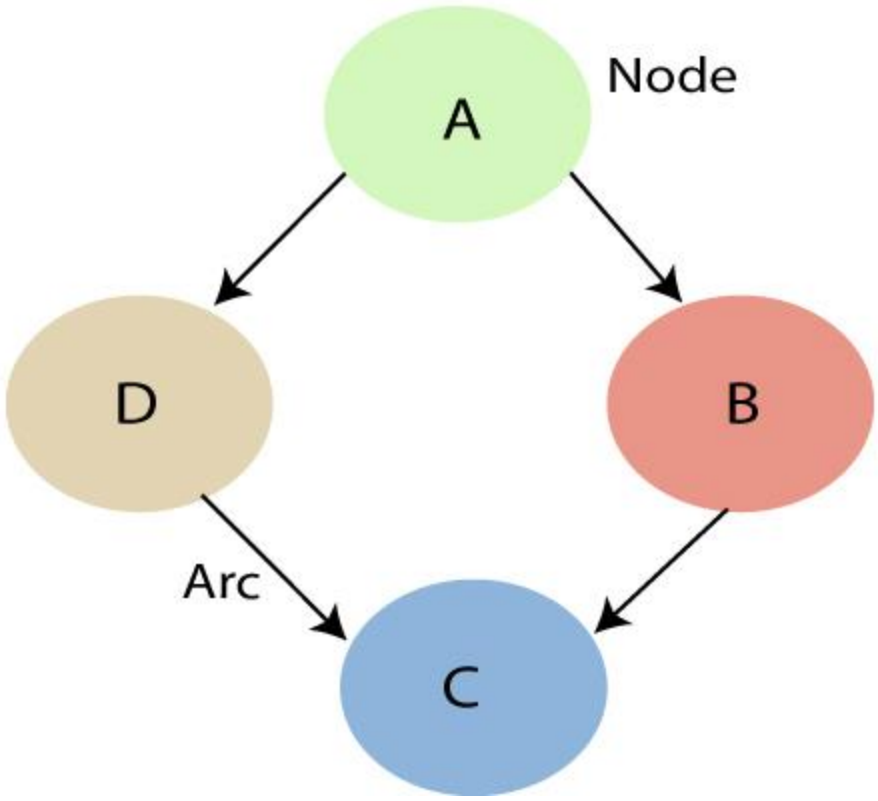
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision 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 **$P(X_i \mid Parent(X_i))$**, 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:

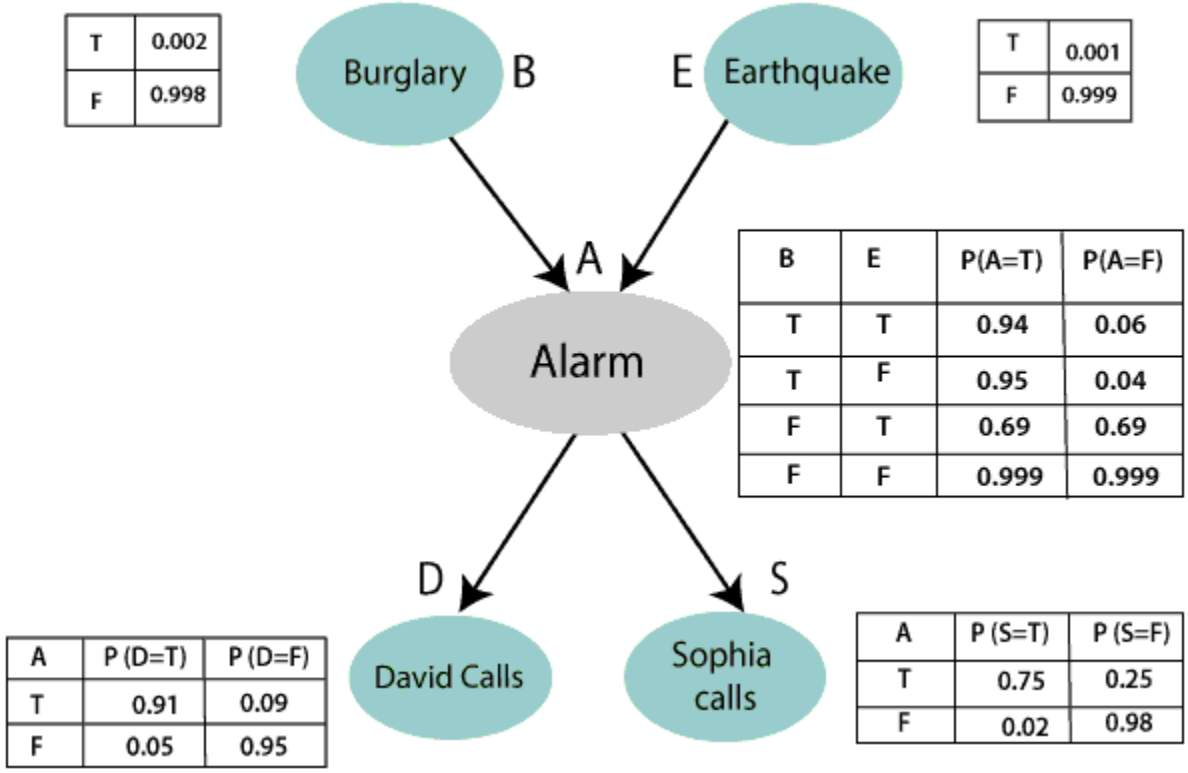
P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

=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]



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

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:

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.

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."

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.

Dempster Shafer Theory

-

Dempster-Shafer Theory was given by Arthur P. Dempster in 1967 and was later developed by his student Glenn Shafer in 1976. This theory was released because of the following reason:-

- Bayesian theory is only concerned about single evidence.
- Bayesian probability cannot describe ignorance.

Dempster Shafer Theory

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.

To solve these there are the **following possibilities**:

- 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.

Using the above example we can say:

Set of possible conclusion (P): {p1, p2….pn}

where P is a set of possible conclusions and cannot be exhaustive, i.e. at least one (p) I must be true.

(p)I must be mutually exclusive.

Power Set will contain 2^n elements where n is the number of elements in the possible set.

For e.g.:-

If P = { a, b, c}, then Power set is given as
{o, {a}, {b}, {c}, {a, d}, {d ,c}, {a, c}, {a, c ,d }}= 2_3 elements.

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
Lets 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. i.e.; Pl(K) = m(a) + m(d) + m(c) + m(a, d) + m(d, c) + m(a, c) + m(a, d, c)

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, Dempster'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