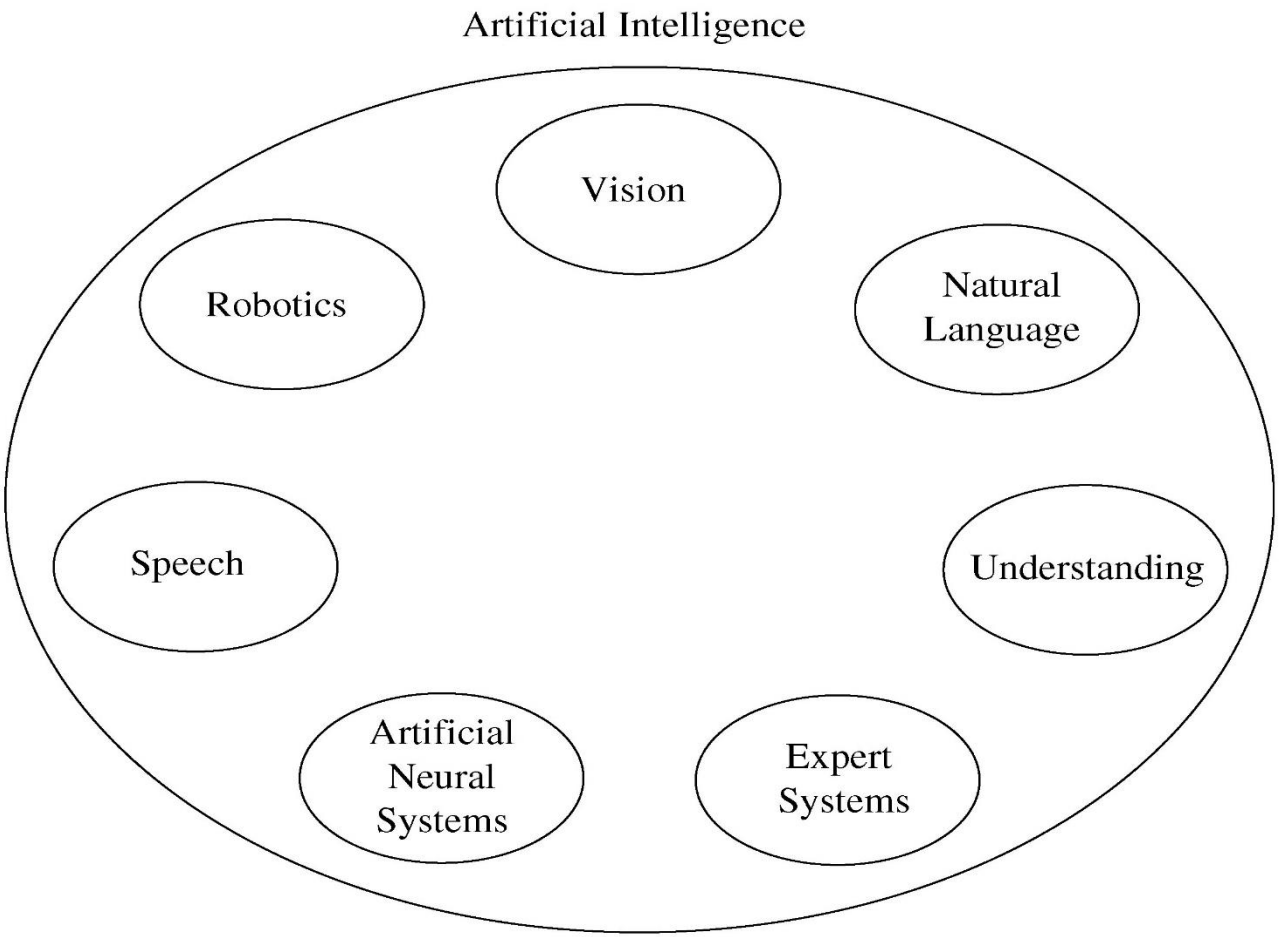
Expert Systems

- An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert.

(Or)

- The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.
- It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.
- The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence.

Areas of Artificial Intelligence



Expert Systems

- Expert Systems solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using **both facts and heuristics like a human expert**.
- It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science,** etc.
- The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance.
- One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Characteristics of Expert Systems

- **High performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds 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.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Capabilities of Expert Systems

The expert systems are capable of −

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem

In capabilities of Expert Systems

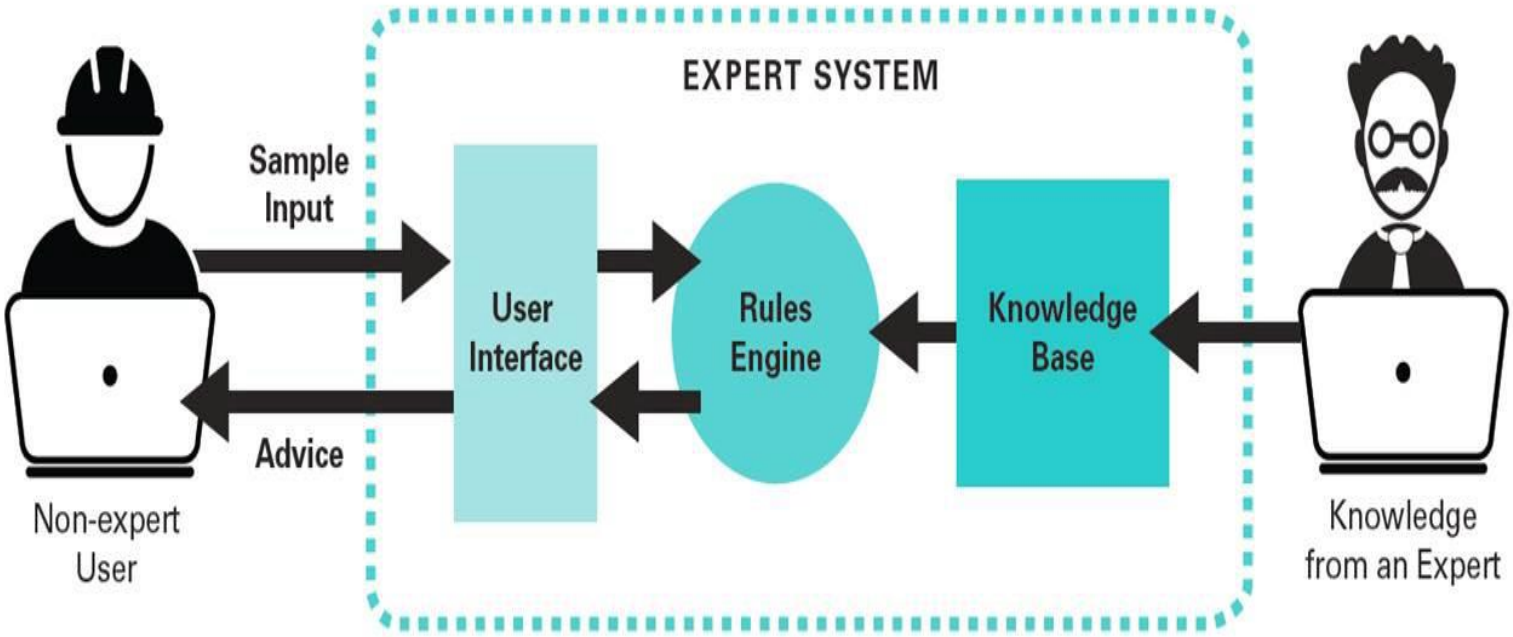
They are incapable of −

- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Components of Expert Systems

The components of ES include −

- Knowledge Base
- Inference Engine
- User Interface



The user inputs information about: the equpment,
the component
and the oil being tested . . . The Expert System does the rest.

Components of Expert Systems

Knowledge Base:

- It contains domain-specific and high-quality knowledge.
- Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

What is Knowledge?

- The data is collection of facts. The information is organized as data and facts about the task domain. **Data, information,** and **past experience** combined together are termed as knowledge.

Components of Expert Systems

Components of Knowledge Base:

The knowledge base of an ES is a store of both, factual and heuristic knowledge.

- **Factual Knowledge** − It is the information widely accepted by the Knowledge Engineers and scholars in the task domain.
- **Heuristic Knowledge** − It is about practice, accurate judgement, one's ability of evaluation, and guessing.

Components of Expert Systems

Inference Engine(Rules of Engine)

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies 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 an inference engine, the system extracts the knowledge from the knowledge base.

There are two types of inference engine:

- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.

Components of Expert Systems

- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

Components of Expert Systems

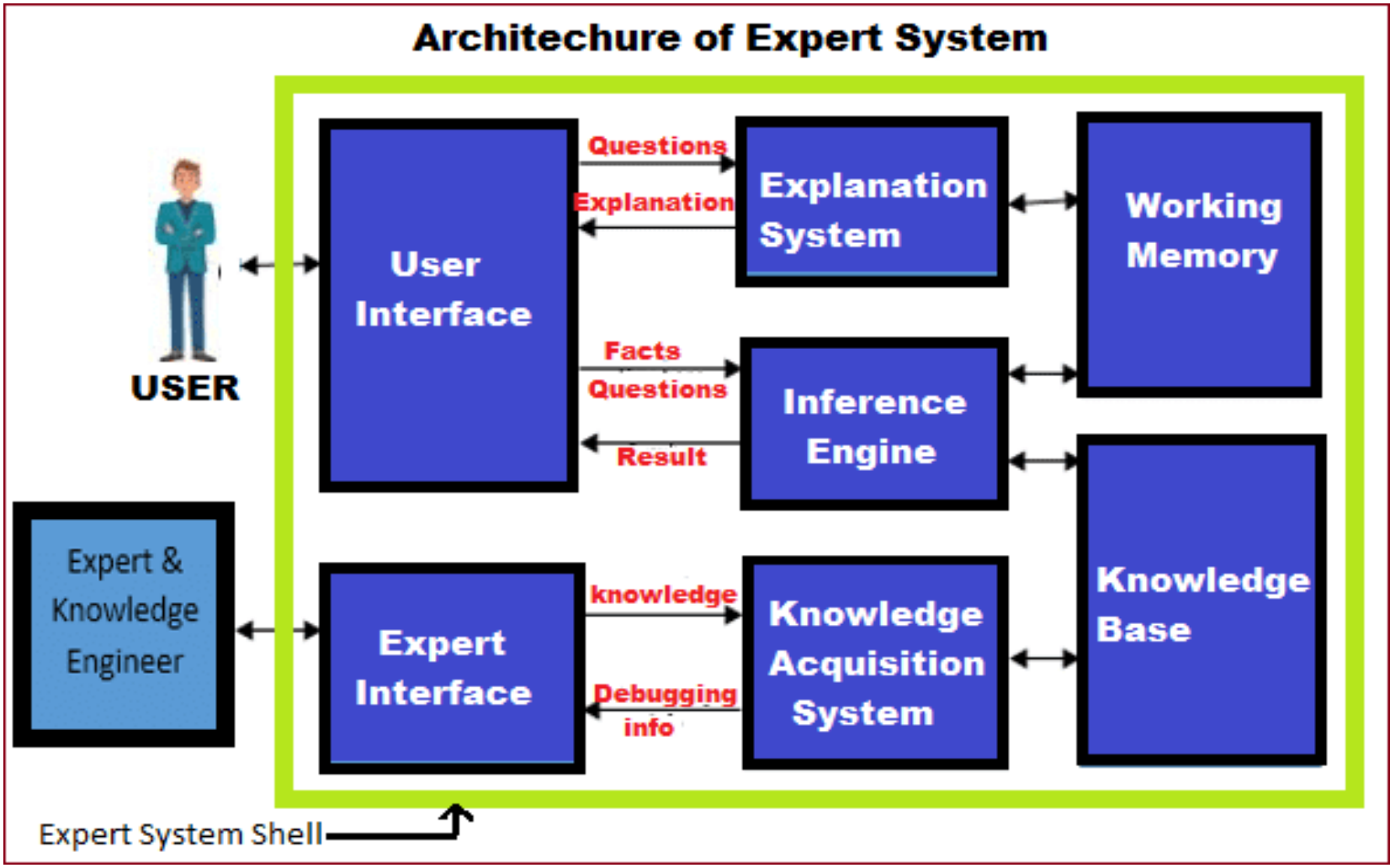
User Interface:

- With the help of a user interface, the expert system interacts with the user, takes queries as an input in a 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**.

Popular Examples of Expert System

- **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.
- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture 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.

Architecture of Expert System



Architecture of Expert System

- **Knowledge Base:** It is warehouse of special heuristics or rules, which are used directly by knowledge, facts (productions). It has knowledge that is needed for understanding, formulating, & problem solving.
- **Working Memory:** It helps to describe the current running problem and record intermediate output.

  Records Intermediate Hypothesis & Decisions: 1. Plan, 2. Agenda, 3. Solution
- **Inference Engine:** It is important part of expert system which helps to manage entire structure of expert system, and it delivers to different methodology for reasoning.
- **Explanation System:** It helps to trace responsibility and justify the behavior of expert system by firing questions and answers, such as Why, How, What, Where, When, Who.

Architecture of Expert System

- **User Interface:** It allows users to insert their queries with using own Natural Language Processing otherwise menus & graphics.
- **Knowledge Engineer:** Main objective of this engineer is to design system for specific problem domain with using of expert system shell.
- **Users:** They are non expert person who want to seek direct advice.
- **Expert system shell:** It contains the special software development environment, and it has basic components of expert system such as – Knowledge-based management system, Workplace, Explanation facility, Reasoning capacity, Inference engine, user interface.
- This shell is linked along with pre-defined method for designing different applications through configuring of those components.

Phases in building Expert System

The following points highlight the five main phases to develop an expert system.

The phases are:

1. Identification
2. Conceptualization
3. Formalization (Designing)
4. Implementation
5. Testing (Validation, Verification and Maintenance).

Phases in building Expert System

- A knowledge engineer is an AI specialist, perhaps a computer scientist or programmer, who is skilled in the 'Art' of developing expert systems.
- You don't need a degree in "knowledge engineering" to call yourself a knowledge engineer; in fact, nearly everyone who has ever contributed to the technical side of the expert system development process could be considered a knowledge engineer.
- A domain expert is an individual who has significant expertise in the domain of the expert system being developed.
- It is not critical that the domain expert understand AI or expert systems; that is one of the functions of the knowledge engineer.

Phases in building Expert System

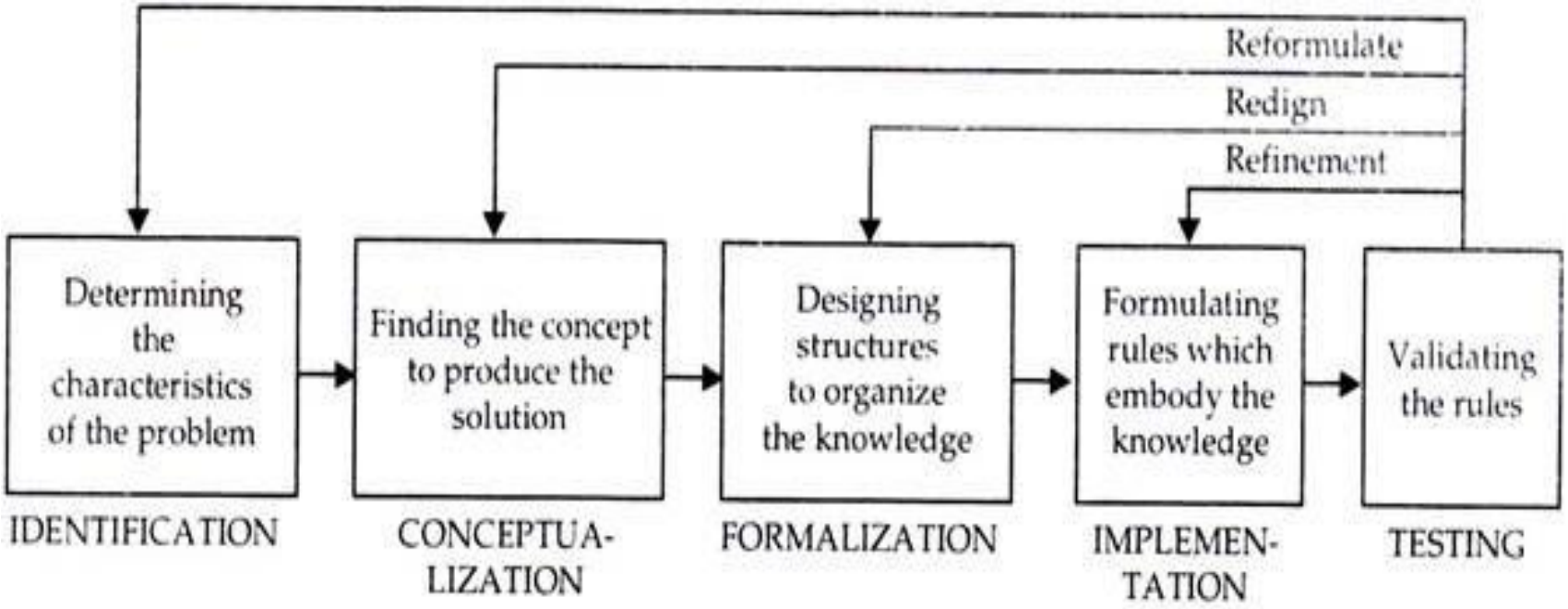


Fig. 12.8. *The five stages of expert system development.*

The knowledge engineer and the domain expert usually work very closely together for long periods of time throughout the several stages of the development process.

1.Identification Phase

- To begin, the knowledge engineer, who may be unfamiliar with this particular domain, consults manuals and training guides to gain some familiarity with the subject. Then the domain expert describes several typical problem states.
- The knowledge engineer attempts to extract fundamental concepts from the similar cases in order to develop a more general idea of the purpose of the expert system.
- After the domain expert describes several cases, the knowledge engineer develops a ‘first-pass’ problem description.
- Typically, the domain expert may feel that the description does not entirely represent the problem.
- The domain expert then suggests changes to the description and provides the knowledge engineer with additional examples to illustrate further the problem’s fine points.

1.Identification Phase

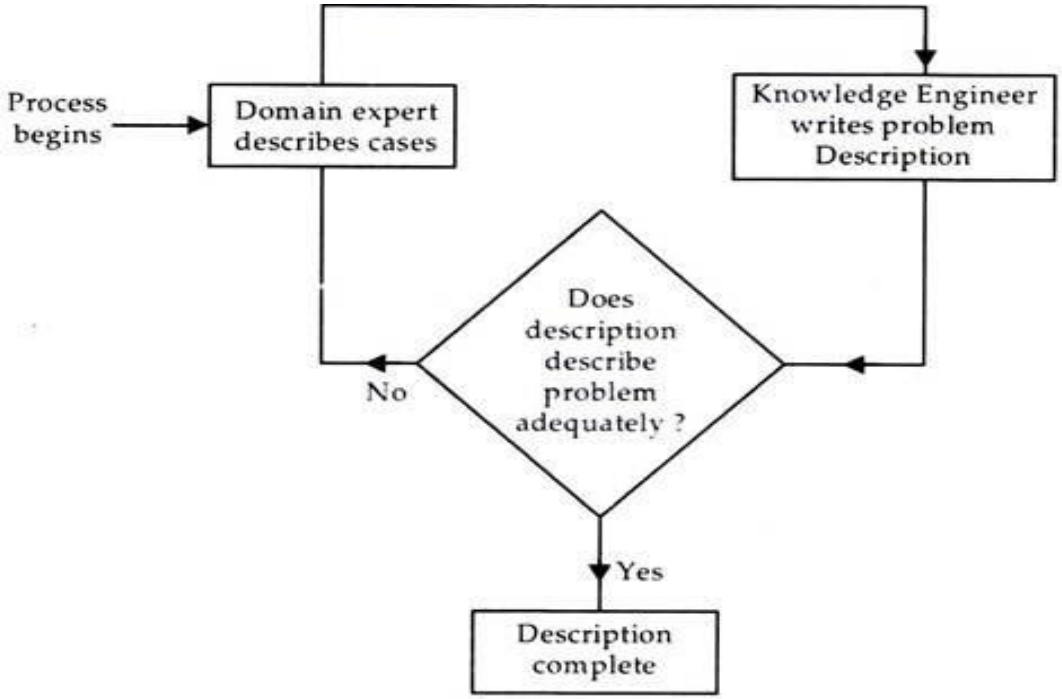


Fig. 12.9. *The iterative process of identifying the problem which the expert system is to solve.*

Next, the knowledge engineer revises the description, and the domain expert suggests further changes. This process is repeated until the domain expert is satisfied that the knowledge engineer understands the problems and until both are satisfied that the description adequately portrays the problem which the expert system is expected to solve.

2.Conceptualisation Phase

- In the conceptualisation stage, the knowledge engineer frequently creates a diagram of the problem to depict graphically the relationships between the objects and processes in the problem domain.
- It is often helpful at this stage to divide the problem into a series of sub-problems and to diagram both the relationships among the pieces of each sub-problem and the relationships among the various sub-problems.
- As in the identification stage, the conceptualisation stage involves a circular procedure of iteration and reiteration between the knowledge engineer and the domain expert. When both agree that the key concepts-and the relationships among them-have been adequately conceptualised, this stage is complete.

2.Conceptualisation Phase

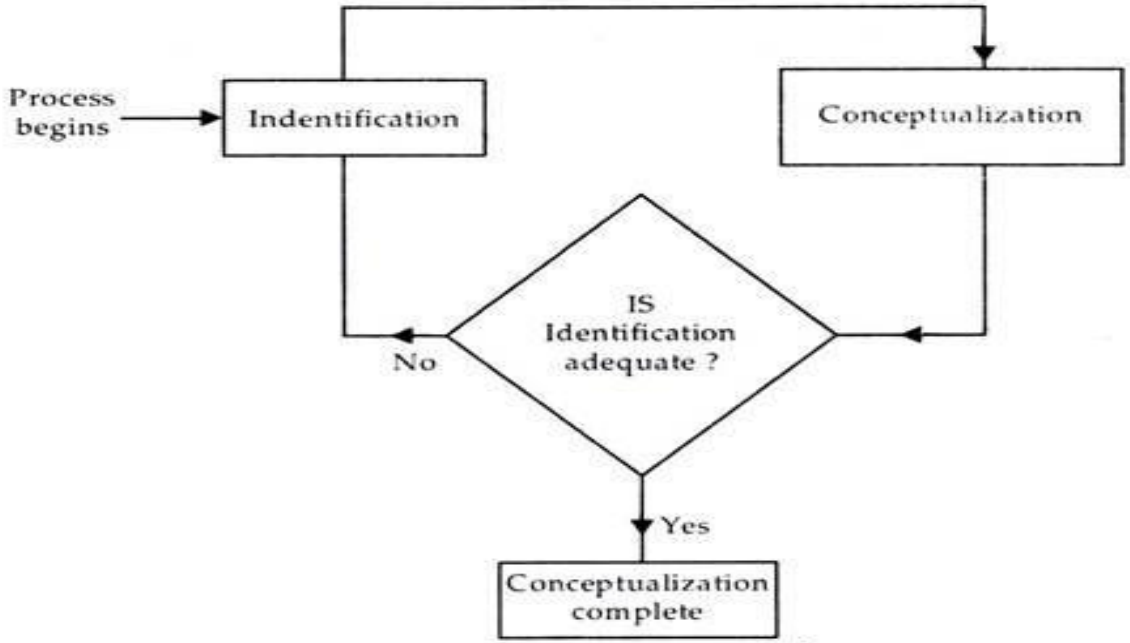


Fig. 12.10. *The iterative relationship between the indentification and conceptualisation stages of expert system development.*

Not only is each stage in the expert system development process circular, the relationships among the stages may be circular as well. Since each stage of the development process adds a level of detail to the previous stage, any stage may expose a weakness in a previous stage

3.Formalisation (Designing) Phase

- The formalisation process is often the most interactive stage of expert system development, as well as the most time consuming.
- The knowledge engineer must develop a set of rules and ask the domain expert if those rules adequately represent the expert's knowledge.
- The domain expert reviews the rules proposed by the knowledge engineer and suggests changes, which are then incorporated into the knowledge base by the knowledge engineer.
- As in the other development stages, this process also is iterative: the rule review is repeated and the rules are refined continually until the results are satisfactory. It is not unusual for the formalisation process of a complex expert system to last for several years.

4.Implementation Phase

- During the implementation stage the formalised concepts are programmed into the computer which has been chosen for system development, using the predetermined techniques and tools to implement a ‘first-pass’ (prototype) of the expert system.
- If the prototype works at all, the knowledge engineer may be able to determine if the techniques chosen to implement the expert system were the appropriate ones.
- On the other hand, the knowledge engineer may discover that the chosen techniques simply cannot be implemented. It may not be possible, for example, to integrate the knowledge representation techniques selected for different sub-problems.
- At that point, the concepts may have to be re-formalised, or it even may be necessary to create new development tools to implement the system efficiently.

4.Implementation Phase

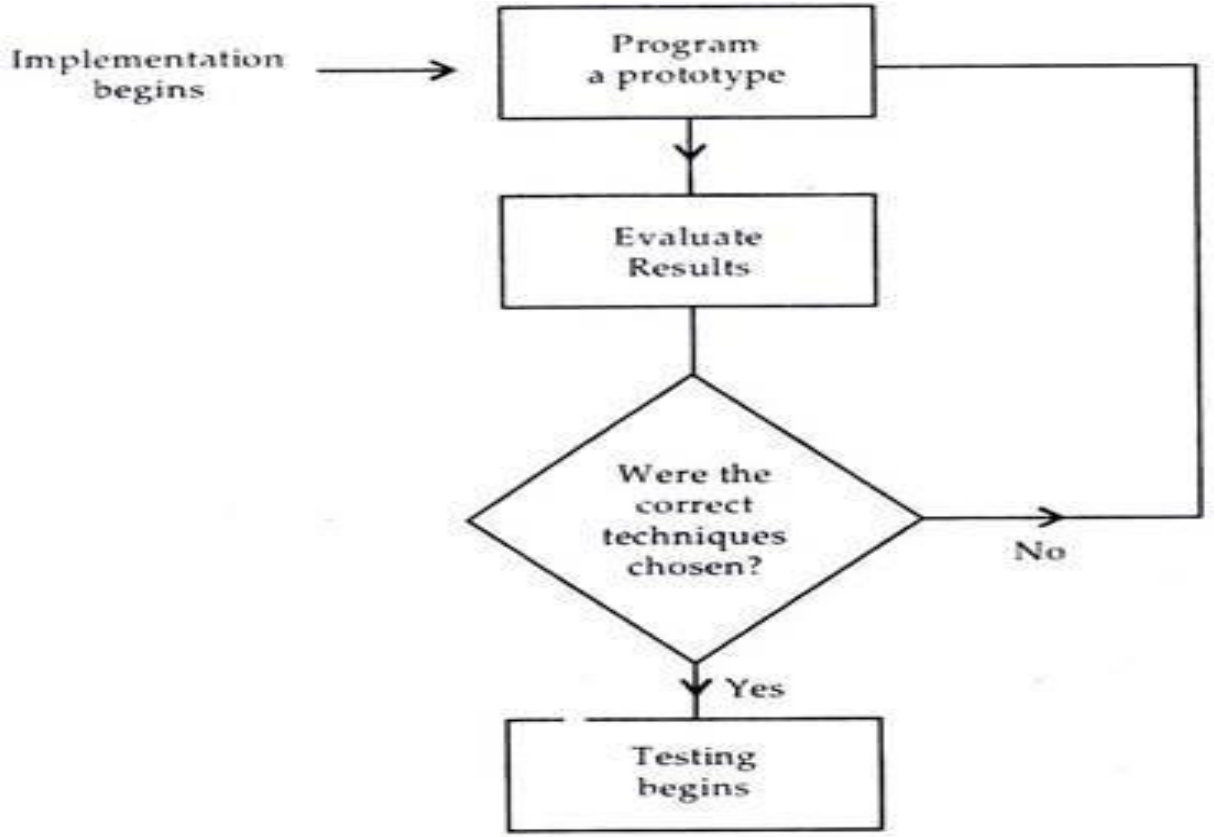


Fig. 12.11. *The implementation stage of expert system development.*

Once the prototype system has been refined sufficiently to allow it to be executed, the expert system is ready to be tested thoroughly to ensure that it expertise's correctly.

5.Testing (Validation, Verification and Maintenance) Phase

- Testing provides an opportunity to identify the weaknesses in the structure and implementation of the system and to make the appropriate corrections.
- Depending on the types of problems encountered, the testing procedure may indicate that the system was implemented incorrectly, or perhaps that the rules were implemented correctly but were poorly or incompletely formulated.
- Results from the tests are used as ‘feedback’ to return to a previous stage and adjust the performance of the system.

5.Testing (Validation, Verification and Maintenance) Phase

- Once the system has proven to be capable of correctly solving straight-forward problems, the domain expert suggests complex problems which typically would require a great deal of human expertise.
- These more demanding tests should uncover more serious flaws and provide ample opportunity to ‘fine tune’ the system even further.
- Ultimately, an expert system is judged to be entirely successful only when it operates at the level of a human expert.
- The testing process is not complete until it indicates that the solutions suggested by the expert system are consistently as valid as those provided by a human domain expert.

Applications of Expert System

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

Uncertainty

- In AI knowledge representation uses techniques such as 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.

- Information occurred from unreliable sources.
- Experimental Errors
- Equipment fault
- Temperature variation
- 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.

  $0 \leq P(A) \leq 1$, 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

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

Examples:

- In a drawer of ten socks where 8 of them are yellow, there is a 20% chance of choosing a sock that is not yellow.

- There are 9 red candies in a bag and 1 blue candy in the same bag. The chance of picking a blue candy is 10%.

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:

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

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

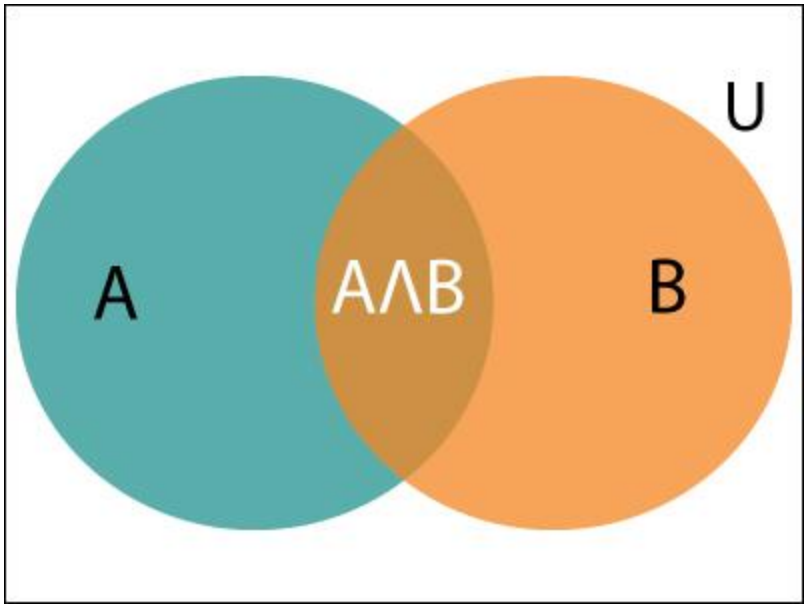
Conditional Probability

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

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

Conditional Probability

It can be explained by using 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 C Language and 40% of the students who likes C and Java, and then what is the percent of students those who like C Language also like Java?

Solution:

Let, A is an event that a student likes Java

B is an event that a student likes C Language.

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

Hence, 57% are the students who like C also like Java.

Prior Probability

Prior Probability- Degree of belief in an event, in the absence of any other information

Example:

- P(rain tomorrow)= 0.7
- P(no rain tomorrow)= 0.3

Conditional Probability

What is the probability of an event , given knowledge of another event.

Example:

- P(raining | sunny)
- P(raining | cloudy)
- P(raining | cloudy, cold)

Conditional Probability…

In some cases , given knowledge of one or more random variables, we can improve our prior belief of another random variable.

For Example:

- P(slept in stadium) = 0.5
- P(slept in stadium | liked match) = 0.33
- P(didn’t slept in stadium | liked match) = 0.67

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 …

- 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:

$P(A \wedge B) = P(A|B)P(B)$ and

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

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

Bayes Theorem …

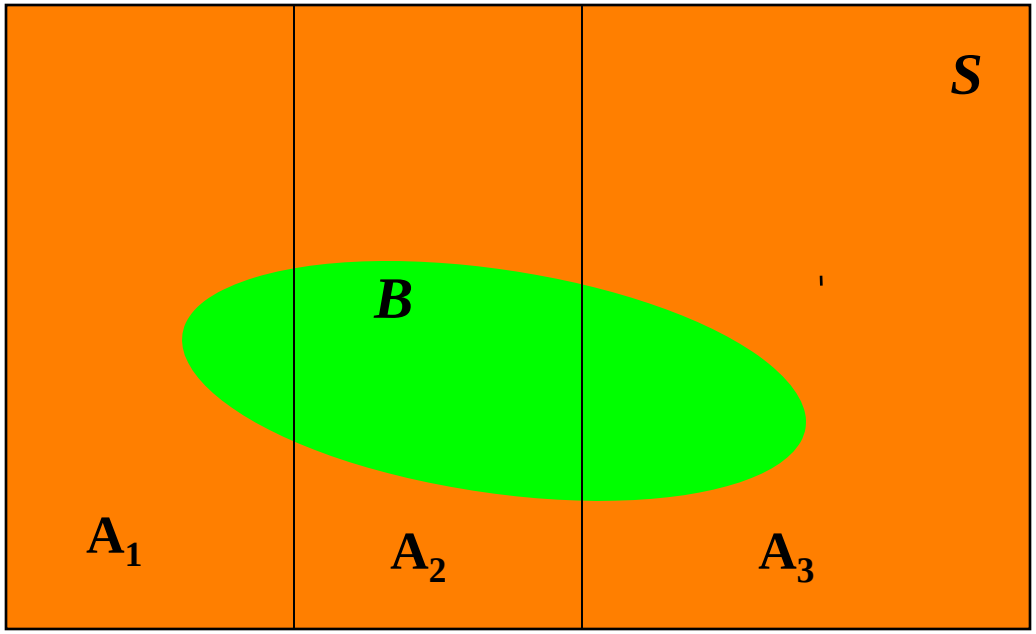
Equating right hand side of both the equations, we will get:

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

- 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.

Bayes Theorem …

Multiple Prior Probabilities



$$P(B) = P(A_1).P(B \mid A_1) + P(A_2).P(B \mid A_2) + P(A_3).P(B \mid A_3)$$

Bayes Theorem …

- 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) = $P(A_i)*P(B|A_i)$, hence the Bayes' rule can be written as:

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

Where $A_1, A_2, A_3, \ldots\ldots, A_n$ 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:

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

Example 1:

- Suppose a patient exhibits symptoms that make her physician concerned that she may have a particular disease. The disease is relatively rare in this population, with a prevalence of 0.2% (meaning it affects 2 out of every 1,000 persons). The physician recommends a screening test that costs Rs.10000 and requires a blood sample. Before agreeing to the screening test, the patient wants to know what will be learned from the test, specifically she wants to know the probability of disease, given a positive test result, i.e., P(Disease | Screen Positive).
- The physician reports that the screening test is widely used and has a reported sensitivity of 85%. In addition, the test comes back positive 8% of the time and negative 92% of the time.

Example 1:

- The information that is available is as follows:
- P(Disease)=0.002, i.e., prevalence = 0.002
- P(Screen Positive | Disease)=0.85, i.e., the probability of screening positive, given the presence of disease is 85% (the sensitivity of the test), and
- P(Screen Positive)=0.08, i.e., the probability of screening positive overall is 8% or 0.08. We can now substitute the values into the above equation to compute the desired probability,
- We know that P(Disease)=0.002, P(Screen Positive | Disease)=0.85 and P(Screen Positive)=0.08. We can now substitute the values into the above equation to compute the desired probability,
- P(Disease | Screen Positive) = (0.85)(0.002)/(0.08) = 0.021

Example 1:

- The information that is available is as follows:
- P(Disease)=0.002, i.e., prevalence = 0.002
- P(Screen Positive | Disease)=0.85, i.e., the probability of screening positive, given the presence of disease is 85% (the sensitivity of the test), and
- P(Screen Positive)=0.08, i.e., the probability of screening positive overall is 8% or 0.08. We can now substitute the values into the above equation to compute the desired probability,
- We know that P(Disease)=0.002, P(Screen Positive | Disease)=0.85 and P(Screen Positive)=0.08. We can now substitute the values into the above equation to compute the desired probability,
- P(Disease | Screen Positive) = (0.85)(0.002)/(0.08) = 0.021
- The patient undergoes the test and it comes back positive, there is a 2.1% chance that he has the disease.
- Also, note, however, that without the test, there is a 0.2% chance that she has the disease (the prevalence in the population).

Example 2:

- In a recent newspaper article, it was reported that light trucks, which include SUV's, pick-up trucks and minivans, accounted for 40% of all personal vehicles on the road in 2018. Assume the rest are cars. Of every 100,000 car accidents, 20 involve a fatality; of every 100,000 light truck accidents, 25 involve a fatality. If a fatal accident is chosen at random, what is the probability the accident involved a light truck?

Example 2:

Events

C- Cars

T –Light truck

F –Fatal Accident

N- Not a Fatal Accident

Given, $P(F|C) = 20/10000$ and $P(F|T) = 25/100000$

$P(T) = 0.4$

In addition we know C *and* T are complementary events

$P(C)=1-P(T)=0.6$

Our goal is to compute the conditional probability of a Light truck accident given that it is fatal $P(T|F)$.

Example 2:

Consider $P(T|F)$

Conditional probability of a Light truck accident given that it is fatal

How do we calculate?

Using conditional probability formula

$$P(T \mid F) = \frac{P(T \cap F)}{P(F)} = \frac{P(F \mid T) \cdot P(T)}{P(F \mid T) \cdot P(T) + P(F \mid C) \cdot P(C)}$$

$$= \frac{(0.00025)(0.4)}{(0.00025)(0.4) + (0.0002)(0.6)}$$

$$= 0.4545$$

Bayesian Networks

- 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**.

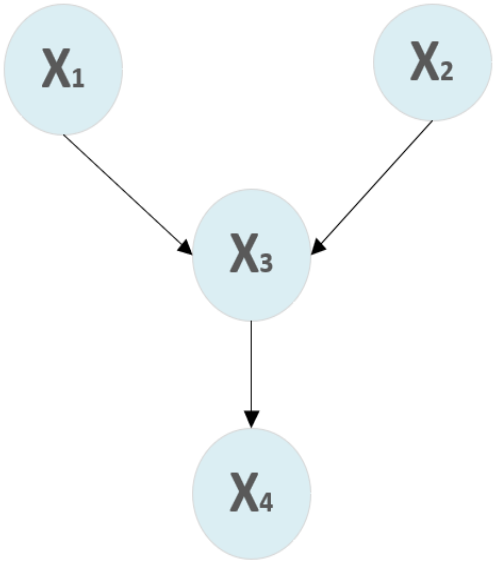
Bayesian Networks

- 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.

A **Bayesian network graph (Directed Acyclic 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

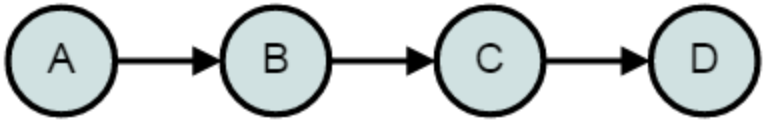
- In the above diagram X1, X2,X3 and X4 are random variables represented by the nodes of the network graph.
- If we are considering node X3, which is connected with node X1 by a directed arrow, then node X1 is called the parent of Node X3.
- Node X4 is independent of node X1.

Conditional Probability Tables- CPTs

- The conditional probability tables in the network give the probabilities for the value of the random variables depending on the combination of values for the parent nodes.
- Each row must be sum to 1.
- All variables are Boolean, and therefore, the probability of a true value is p, the probability of false value must be 1-p.
- A table for a boolean variable with k-parents contains 2^k independently specifiable probabilities.
- A variable with no parents has only one row, representing the prior probabilities of each possible values of the variable.

Joint Probability Distribution

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



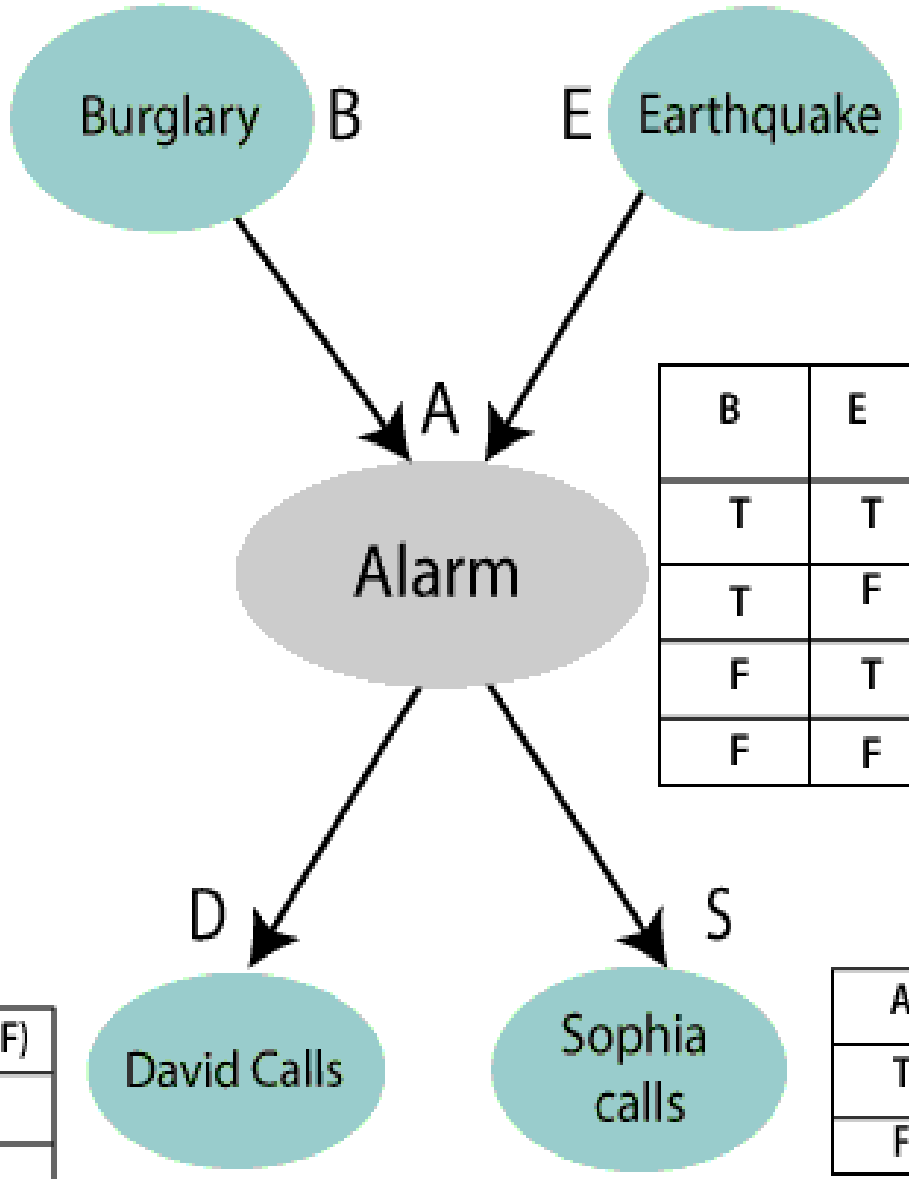
- If we have variables A,B,C,D then the probabilities of a different combination of their variables are known as Joint probability distribution.

- A directed graph G with four vertices A,B,C and D. If $P(x_A, x_B, x_C, x_D)$ factorizes with respect to G, then we must have

$$P(x_A, x_B, x_C, x_D) = P(x_A)P(x_B|x_A)P(x_C|x_B)P(x_D|x_C)$$

Bayesian Network- Burglar Alarm

- You have installed a new burglar-alarm at home.
- It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes.
- You have also have two neighbors, David and Sophia, who have promised to call you at work when they hear the alarm.
- David always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too.
- Sophia, on the other hand, likes loud music and misses the alarm altogether.
- Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.



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

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.

Problem:

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:

$$P[D, S, A, B, E] = P[D \mid A]. P[S \mid A]. P[A \mid B, E]. P[B \mid E]. P[E]$$

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

$$P(S, D, A, \neg B, \neg E) = P(S|A) * P(D|A) * P(A|\neg B \wedge \neg E) * P(\neg B) * P(\neg E).$$

$$= 0.75 * 0.91 * 0.001 * 0.998 * 0.999$$

$$= \mathbf{0.00068045.}$$

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

Inference in Bayesian Networks

Purpose:

- Probabilistic Inference System is to compute Posterior Probability Distribution for a set of query variables given some observed events.
- That is, some assignment of values to a set of evidence variables.

Inference in Bayesian Networks

Notations:

- **X -** Denotes the query variable.
- **E -** Set of Evidence variables
- **e -** Particular observed event
- **Y -** Non evidence, non query variable $\mathbf{Y_1,\ldots..,Y_n}$ (Hidden variables)
- The complete set of variables X=**{X} U E U Y**
- A typical query ask for the Posterier Probability Distribution **P{X|e}**

- In the burglary network, we might observe the event in which

  JohnCalls = true and MaryCalls =true

- We could then ask for, say, the probability that a burglary has occurred:

  P(Burglary | JohnCalls = true, MaryCalls = true) = ?

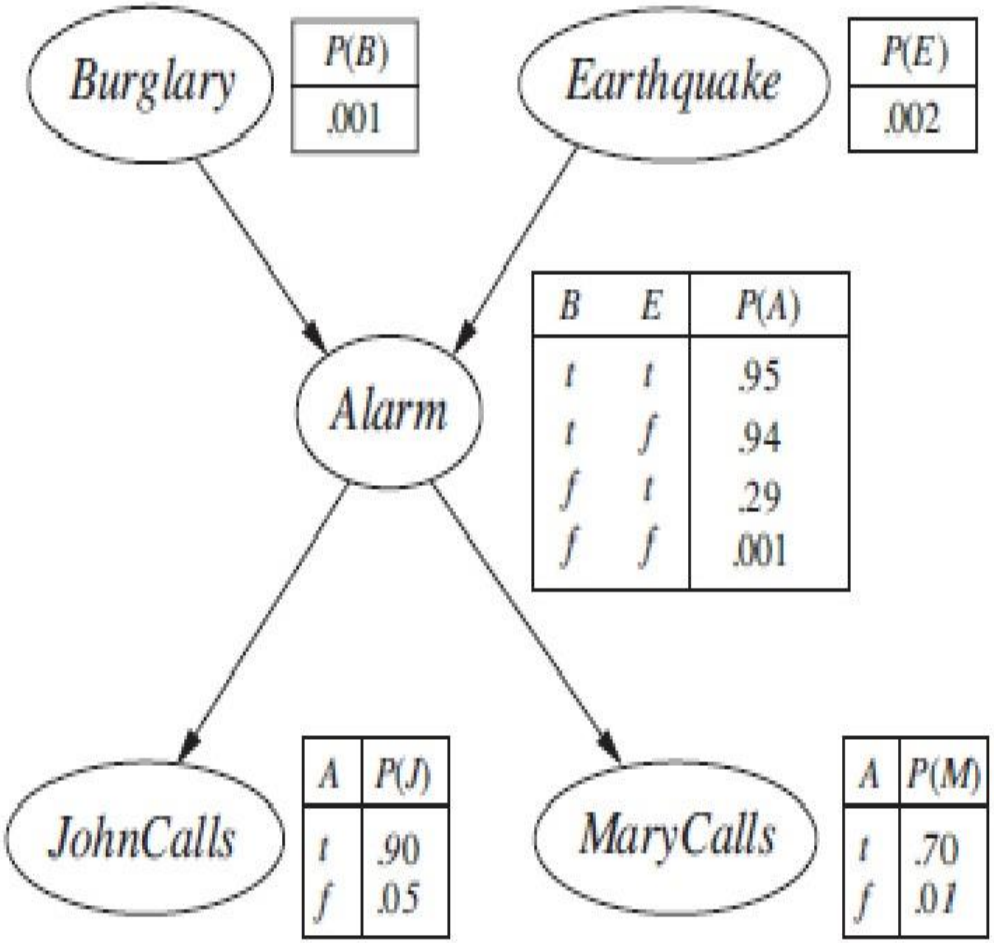
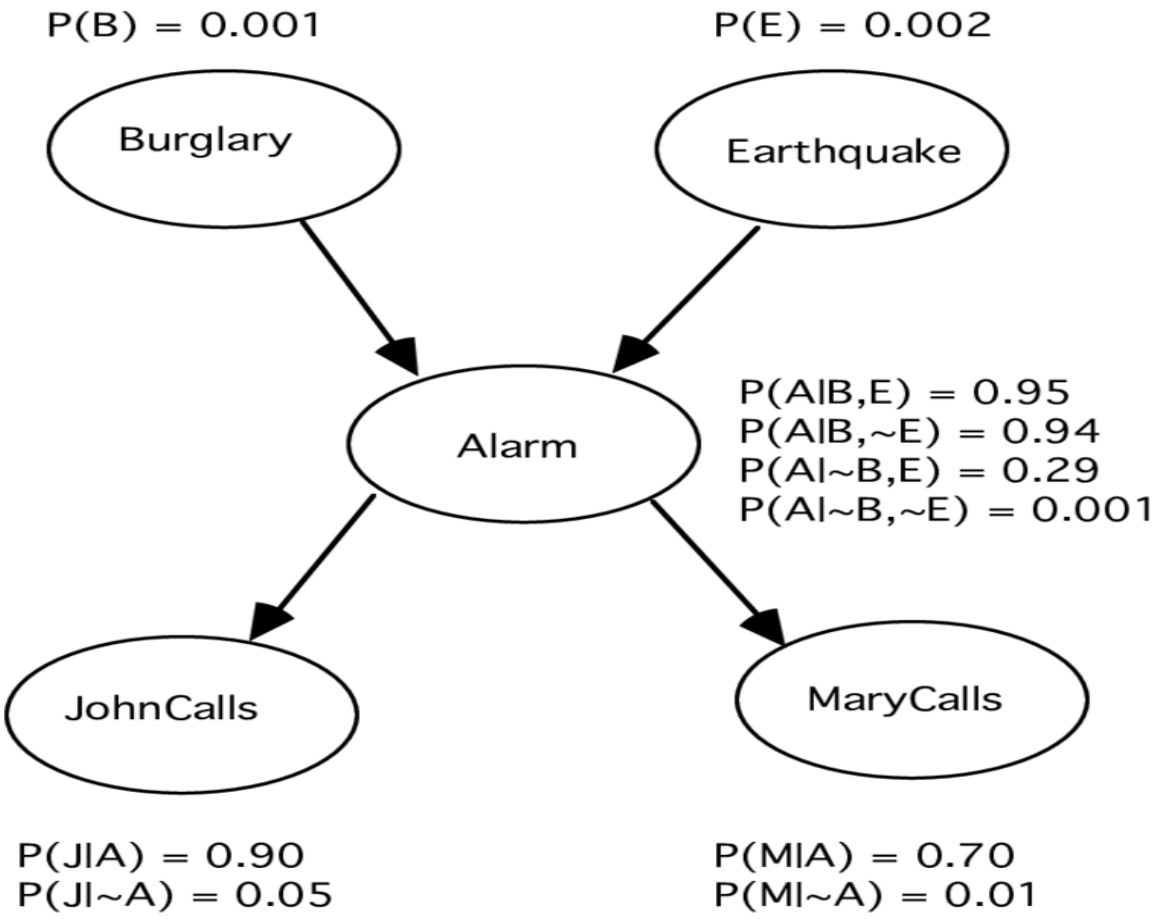


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

P(B | J=true, M=true)



Types of Inferences:

- Inference by Enumeration

  (inference by listing or recording all variables)

- Inference by Variable Elimination

  (inference by variable removal)

Inference by Enumeration

- Any conditional probability can be computed by summing terms from the full joint distribution.
- More specifically, a query P(X | e) can be answered using equation:

$$P(X \mid e) = \alpha P(X, e) = \alpha \Sigma_y P(X, e, y)$$

where α is normalized constant

X – Query variable

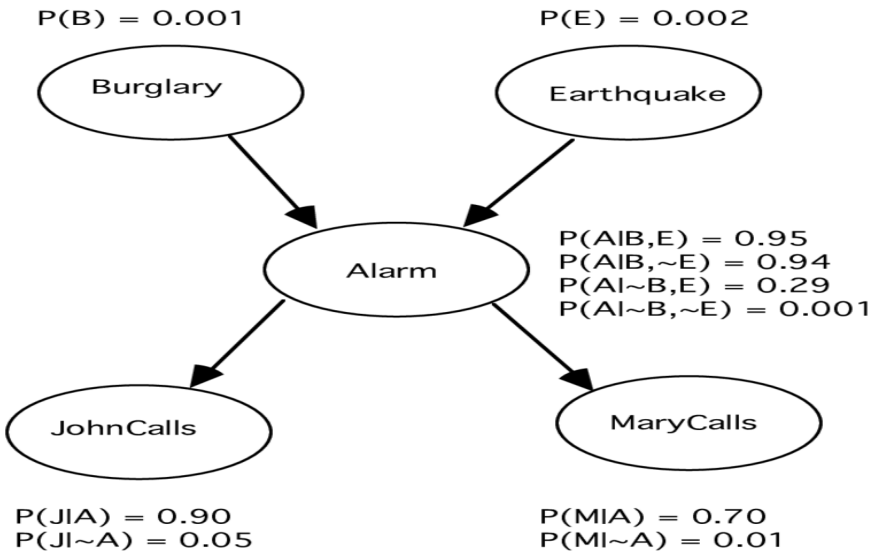
e - event

y – number of terms (hidden variables)

Inference by Enumeration…

- Consider P (Burglary | JohnCalls = true, MaryCalls = true)
- Burglary - query variable (X)
- JohnCalls - Evidence variable 1 (E1)
- MaryCalls - Evidence variable 2 (E2)
- The hidden variable of this query are earthquake and alarm.

Inference by Enumeration…



$$P(B \mid j, m) = P(B,j,m) / P(j,m)$$

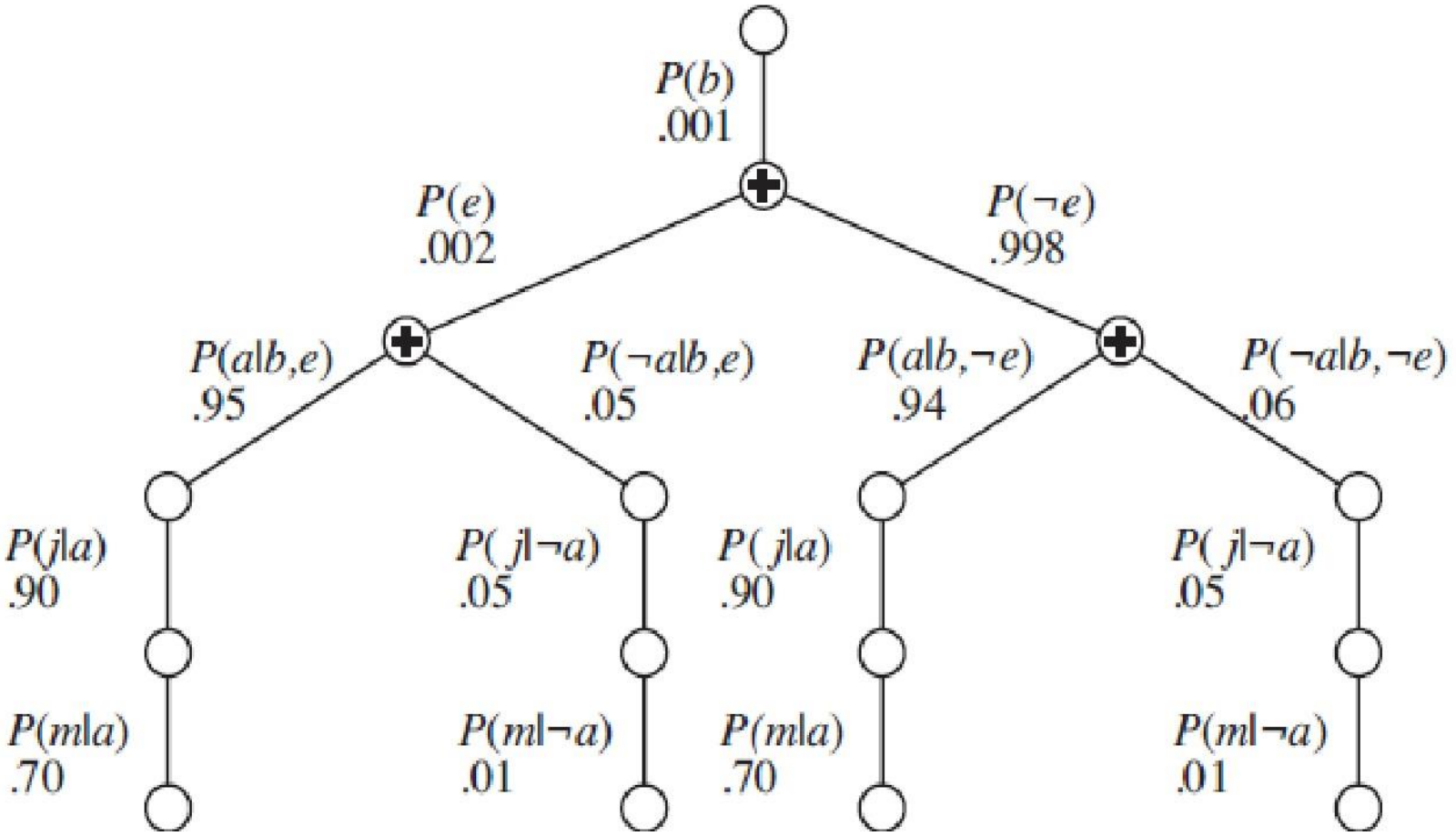
$$P(B \mid j, m) = \alpha\, P(B,j,m)$$

$$P(B \mid j, m) = \alpha \Sigma_{E,A}\, P(B,E,A,j,m)$$

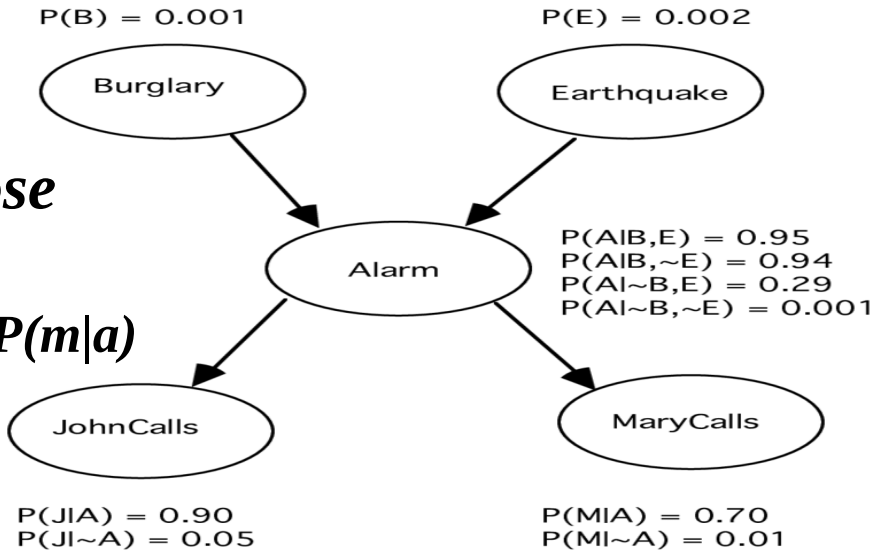
$$P(B \mid j, m) = \alpha \Sigma_{E,A}\, P(B)P(E)P(A|E,B)P(j|A)P(m|A)$$

$$P(B \mid j, m) = \alpha\, P(B) \Sigma_{E}\, P(E)\, \Sigma_{A} P(A|E,B)P(j|A)P(m|A)$$

$$P(B \mid j, m) = \alpha\, P(B) \Sigma_E\, P(E)\, \Sigma_A\, P(A|B,E) P(j|A) P(m|A)$$



Inference by Enumeration…



Let us consider for simplicity purpose burglary = true

$$P(b \mid j, m) = \alpha\, P(b) \Sigma_e\, P(e)\, \Sigma_a P(a|e,b) P(j|a) P(m|a)$$

This expression can be evaluated by looping through the variables in order, multiplying CPT entries as we go. For each summation, we also need to loop over the variable's possible values. Using the numbers from above figure, we obtain P(b | j , m) = α×0.00059224. The corresponding computation for ¬b yields α×0.0014919; hence,

P(B | j , m) = α (0.00059224, 0.0014919) ≈ 0.284, 0.716 .

That is, the chance of a burglary, given calls from both neighbors, is about 28%.

Inference by Variable Elimination

- The enumeration algorithm can be improved substantially by eliminating repeated calculations.
- The idea is simple: do the calculation once and save the results for later use. This is a form of dynamic programming.
- Variable elimination works by evaluating expressions, from the previous equation (derived in inference by enumeration)

$$\boldsymbol{P(b \mid j, m) = \alpha\, P(b)\Sigma_e\, P(e)\, \Sigma_a P(a|e,b)P(j|a)P(m|a)}$$

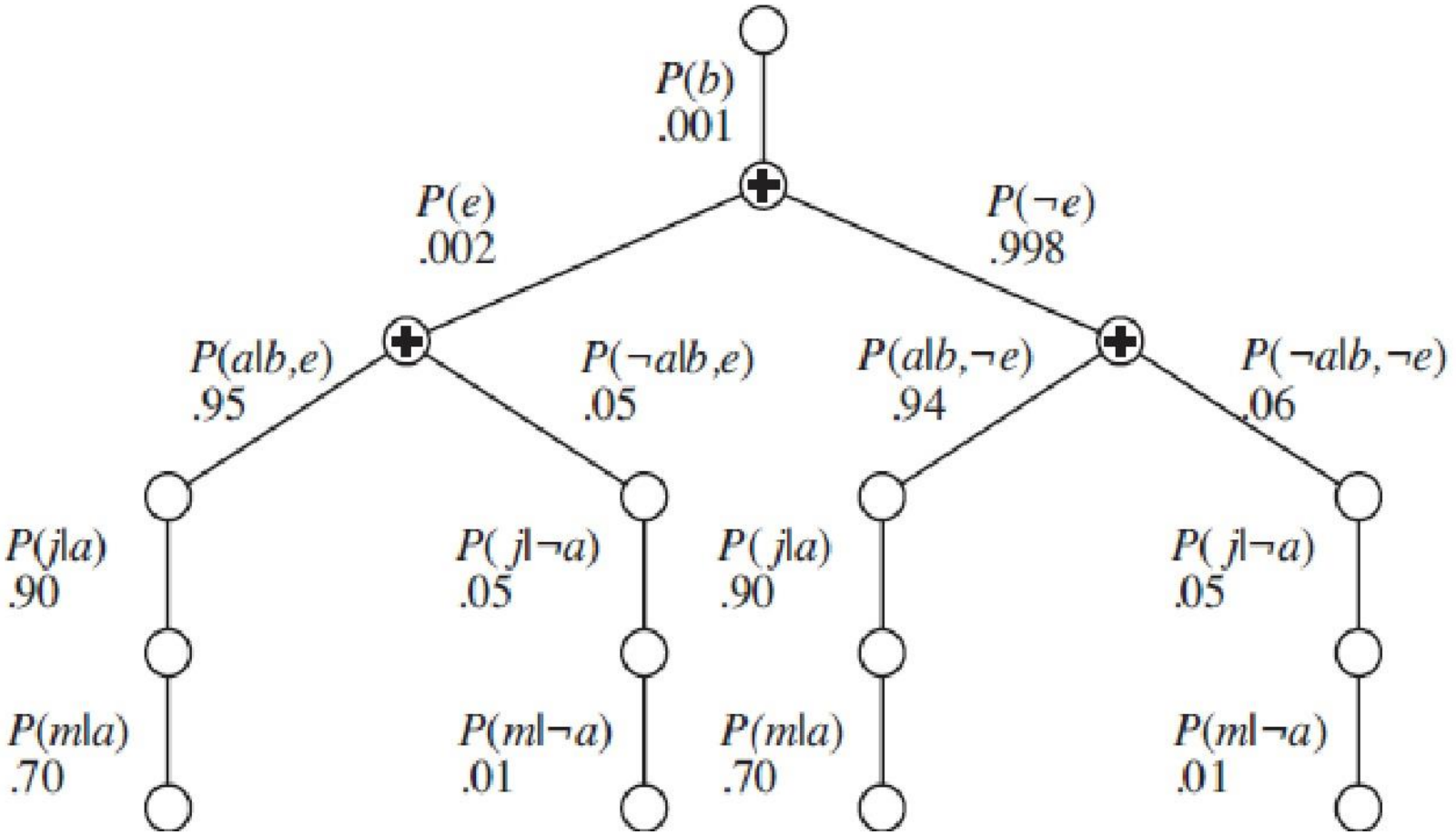
Inference by Variable Elimination…

- Intermediate results are stored, and summations over each variable are done only for those portions of the expression that depend on the variable.
- Let us illustrate this process for the burglary network.
- We evaluate the below expression in such a way that we annotated each part of the expression with the name of the associated variable; these parts are called factors

$$\boldsymbol{P(B \mid j, m) = \alpha\, P(B)\Sigma_E\, P(E)\, \Sigma_A\, P(A|E,B)P(j|A)P(m|A)}$$

Variable Elimination

$$P(B \mid j, m) = \alpha\, P(B) \Sigma_E\, P(E)\, \Sigma_A\, P(A|B,E) P(j|A) P(m|A)$$



Inference by Variable Elimination…

$$\alpha\, P(B)\Sigma_E\, P(E)\, \Sigma_A P(A|E,B)P(j|A)P(m|A)$$

P(J | A)	
A	0.90 (0.1)
-A	0.05 (0.95)

P(M | A)	
A	0.70 (0.30)
-A	0.01 (0.99)

P(j|A)P(m|A)	
A	0.9 * 0.7
-A	0.05 * 0.01

Inference by Variable Elimination…

$$\alpha\, P(B)\Sigma_E\, P(E)\, \Sigma_A P(A|E,B) f1(A)$$

P(J | A)	
A	0.90 (0.1)
-A	0.05 (0.95)

P(M | A)	
A	0.70 (0.30)
-A	0.01 (0.99)

f1 (A)	
A	0.63
-A	0.0005

Inference by Variable Elimination…

$$\alpha\, P(B)\Sigma_E\, P(E)\, \Sigma_A P(A|E,B) f1(A)$$

f1 (A)	
A	0.63
-A	0.0005

P(A | E,B)	
e , b	0.95 (0.05)
e , -b	0.29 (0.71)
-e , b	0.94(0.06)
-e , -b	0.001(0.999)

Σ_AP(A | E,B) f1(A)	
e , b	0.95 * 0.63 + 0.05 * 0.0005
e , -b	0.29 *0.63 + 0.71 * 0.0005
-e , b	0.94 * 0.63 + 0.06 * 0.0005
-e , -b	0.001 * 0.63 + 0.999 * 0.0005

Inference by Variable Elimination…

$\alpha\ P(B)\Sigma_E\ P(E)\ f2(E,B)$

f1 (A)	
A	0.63
-A	0.0005

P(A|E,B)	
e , b	0.95 (0.05)
e , -b	0.29 (0.71)
-e , b	0.94(0.06)
-e , -b	0.001(0.999)

f2(E,B)	
e , b	0.60
e , -b	0.18
-e , b	0.59
-e , -b	0.001

Inference by Variable Elimination…

$\alpha\ P(B)\Sigma_E\ P(E)\ f2(E,B)$

P(E=T)	P(E=F)
0.002	0.998

P(B=T)	P(B=F)
0.001	0.999

f2(E,B)	
e , b	0.60
e , -b	0.18
-e , b	0.59
-e , -b	0.001

$P(B)\Sigma_E\ P(E)\ f2(E,B)$	
b	0.60 * 0.002 * 0.001 + 0.59 * 0.998 * 0.001
-b	0.18 * 0.002 * 0.999 + 0.001 * 0.998 * 0.999

Inference by Variable Elimination…

α *f3(B)*

P(E=T)	P(E=F)
0.002	0.998

P(B=T)	P(B=F)
0.001	0.999

f2(E,B)	
e , b	0.60
e , -b	0.18
-e , b	0.59
-e , -b	0.001

f3(B)	
b	0.0006
-b	0.0013

Inference by Variable Elimination…

α ***f3(B)*** $\rightarrow$ ***P(B | j , m)***

f3(B)	
b	0.0006
-b	0.0013

N= 0.0006 + 0.0013 =0.0019

P(B | j , m)	
b	0.32
-b	0.68

That is, the chance of a burglary, given calls from both neighbors, is about 32%.

WHAT IS FUZZY LOGIC?

- Definition of fuzzy
  - Fuzzy – “not clear, distinct, or precise; blurred”
- Definition of fuzzy logic
  - A form of knowledge representation suitable for notions that cannot be defined precisely, but which depend upon their contexts.
  - So we cannot decide in real life that the given problem or statement is either true or false.
  - At that time, this concept provides many values between the true and false and gives the flexibility to find the best solution to that problem.

WHAT IS FUZZY LOGIC?

The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as −

The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

CERTAINLY YES
POSSIBLY YES
CANNOT SAY
POSSIBLY NO
CERTAINLY NO

Implementation

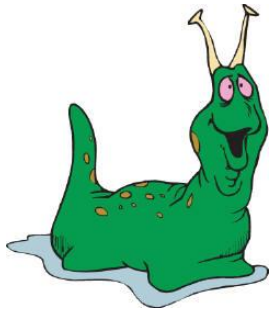
- It can be implemented in systems with various sizes and capabilities ranging from small micro-controllers to large, networked, workstation-based control systems.
- It can be implemented in hardware, software, or a combination of both.

Why Fuzzy Logic?

Fuzzy logic is useful for commercial and practical purposes.

- It can control machines and consumer products.
- It may not give accurate reasoning, but acceptable reasoning.
- Fuzzy logic helps to deal with the uncertainty in engineering.

TRADITIONAL REPRESENTATION OF LOGIC



Slow

Fast

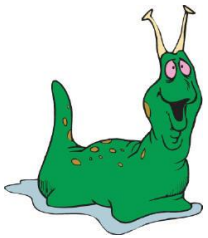
Speed = 0

Speed = 1

```
bool speed;
get the speed
if ( speed == 0) {
//  speed is slow
}
else {
//  speed is fast
}
```

FUZZY LOGIC REPRESENTATION

- For every problem must represent in terms of fuzzy sets.
- What are fuzzy sets?



Slowest

[0.0 – 0.25]

Slow

[0.25 – 0.50]

Fast

[0.50 – 0.75]

Fastest

[0.75 – 1.00]

FUZZY LOGIC REPRESENTATION CONT.

Slowest Slow Fast Fastest

```
float speed;
get the speed
if ((speed >= 0.0)&&(speed < 0.25)) {
//  speed is slowest
}
else if ((speed >= 0.25)&&(speed < 0.5))
{
//  speed is slow
}
else if ((speed >= 0.5)&&(speed < 0.75))
{
//  speed is fast
}
else // speed >= 0.75 && speed < 1.0
{
//  speed is fastest
}
```

Fuzzy Set

- The set theory of classical is the subset of Fuzzy set theory. Fuzzy logic is based on this theory, which is a generalization of the classical theory of set (i.e., crisp set) introduced by Zadeh in 1965.
- A fuzzy set is a collection of values which exist between 0 and 1.
- Fuzzy sets are denoted or represented by the tilde (~) character. The sets of Fuzzy theory were introduced in 1965 by Lofti A. Zadeh and Dieter Klaua.
- In the fuzzy set, the partial membership also exists. This theory released as an extension of classical set theory.

Fuzzy Set

This theory is denoted mathematically as A fuzzy set (Ã) is a pair of U and M, where U is the Universe of discourse and M is the membership function which takes on values in the interval [0, 1]. The universe of discourse (U) is also denoted by Ω or X.

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$$

Operations on Fuzzy Set

Given Ã and B are the two fuzzy sets, and X be the universe of discourse with the following respective member functions:

$\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$

The operations of Fuzzy set are as follows:

1. **Union Operation:** The union operation of a fuzzy set is defined by:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Example:

Let's suppose A is a set which contains following elements:

$A = \{(X_1, 0.6), (X_2, 0.2), (X_3, 1), (X_4, 0.4)\}$

And, B is a set which contains following elements:

$B = \{(X_1, 0.1), (X_2, 0.8), (X_3, 0), (X_4, 0.9)\}$

then,

$AUB = \{(X_1, 0.6), (X_2, 0.8), (X_3, 1), (X_4, 0.9)\}$

Operations on Fuzzy Set

2. Intersection Operation: The intersection operation of fuzzy set is defined by:

$$\mu_{A \cap B}(x) = \min (\mu_A(x), \mu_B(x))$$

Example:

Let's suppose A is a set which contains following elements:

$A = \{(X_1, 0.3), (X_2, 0.7), (X_3, 0.5), (X_4, 0.1)\}$

and, B is a set which contains following elements:

$B = \{(X_1, 0.8), (X_2, 0.2), (X_3, 0.4), (X_4, 0.9)\}$

then,

$A \cap B = \{(X_1, 0.3), (X_2, 0.2), (X_3, 0.4), (X_4, 0.1)\}$

Operations on Fuzzy Set

3. Complement Operation: The complement operation of fuzzy set is defined by:

$$\mu_{\bar{A}}(x) = 1-\mu_A(x),$$

Example:

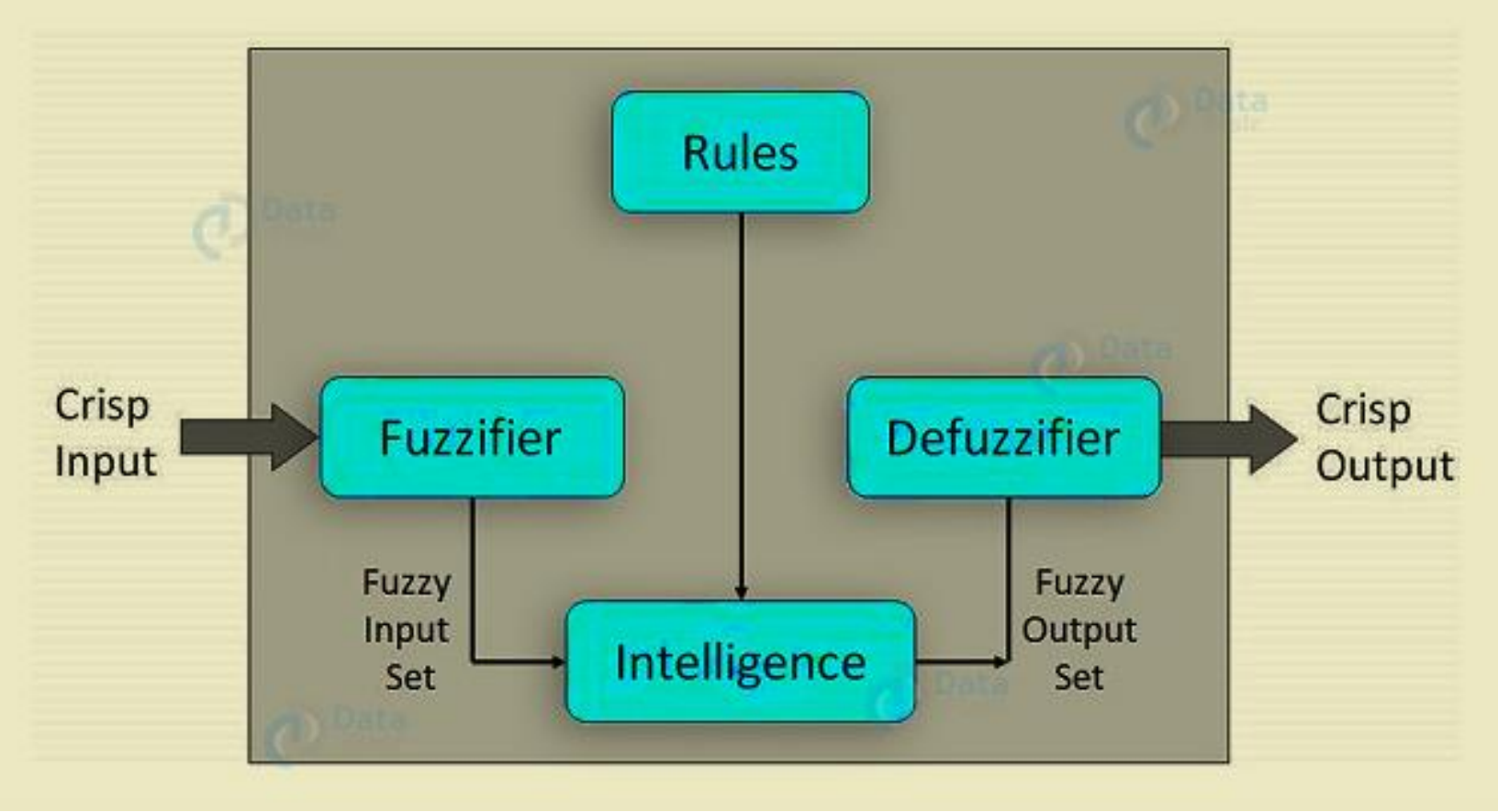
Let's suppose A is a set which contains following elements:

$$A = \{(X_1, 0.3), (X_2, 0.8), (X_3, 0.5), (X_4, 0.1)\}$$

then,

$$\bar{A}= \{(X_1, 0.7), (X_2, 0.2), (X_3, 0.5), (X_4, 0.9)\}$$

Fuzzy Logic Systems Architecture



Fuzzy Logic Systems Architecture

Fuzzy Logic architecture has four main parts as shown in the diagram:

Fuzzification:

- Fuzzification step helps to convert inputs. It allows you to convert, crisp numbers into fuzzy sets. Crisp inputs measured by sensors and passed into the control system for further processing. This component divides the input signals into following five states in any Fuzzy Logic system:
  - Large Positive (LP)
  - Medium Positive (MP)
  - Small (S)
  - Medium Negative (MN)
  - Large negative (LN)

Fuzzy Logic Systems Architecture

Rule Base:

- It contains all the rules and the if-then conditions offered by the experts to control the decision-making system. The recent update in fuzzy theory provides various methods for the design and tuning of fuzzy controllers. This updates significantly reduce the number of the fuzzy set of rules.

Inference Engine:

- It helps you to determines the degree of match between fuzzy input and the rules. Based on the % match, it determines which rules need implment according to the given input field. After this, the applied rules are combined to develop the control actions.

Fuzzy Logic Systems Architecture

Defuzzification:

- At last the Defuzzification process is performed to convert the fuzzy sets into a crisp value. There are many types of techniques available, so you need to select it which is best suited when it is used with an expert system.

Fuzzy logic algorithm

1) Initialization process:

- Define the linguistic variables.
- Construct the fuzzy logic membership functions that define the meaning or values of the input and output terms used in the rules.
- Construct the rule base (Break down the control problem into a series of IF X AND Y, THEN Z rules based on the fuzzy logic rules).

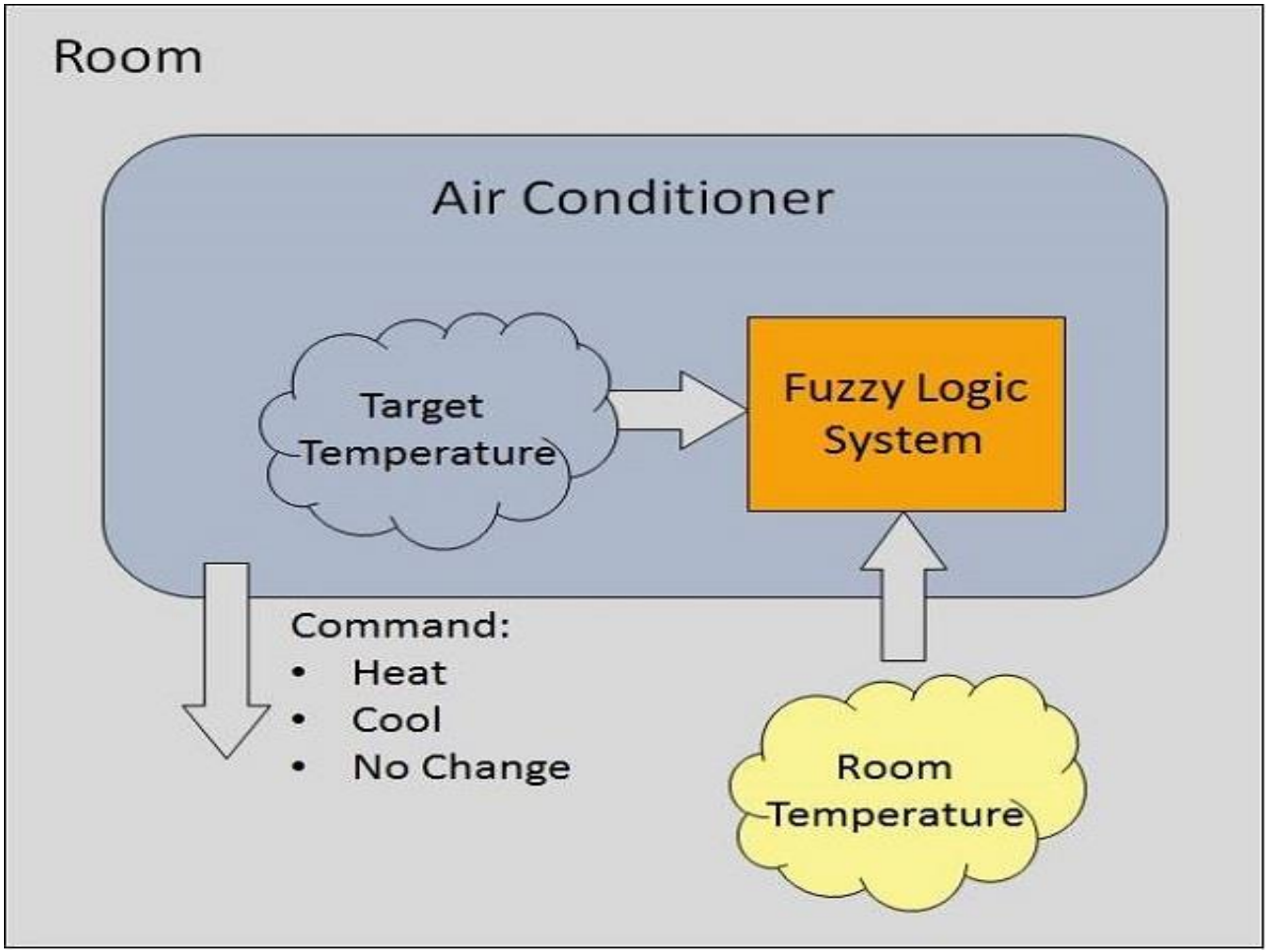
2)Convert crisp input data to fuzzy values using the membership functions (fuzzification).

3) Evaluate the rules in the rule base (inference).

4) Combine the results of each rule (inference).

5)Convert the output data to non-fuzzy values (defuzzification).

Example: Air conditioner system controlled by a FLS



Example: Air conditioner system controlled by a FLS

- The system adjusts the temperature of the room according to the current temperature of the room and the target value. The fuzzy engine periodically compares the room temperature and the target temperature, and produces a command to heat or cool the room.

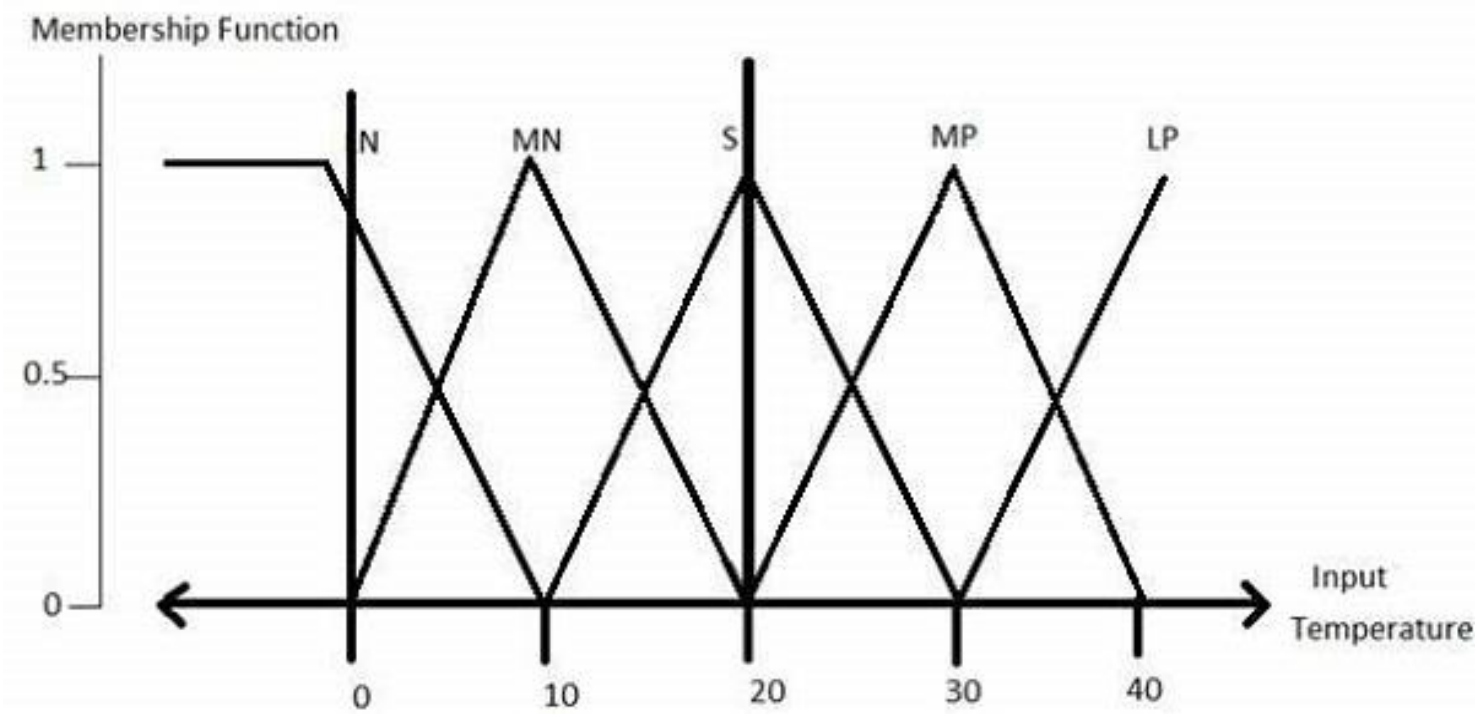
Define linguistic variables and terms

- Linguistic variables are input and output variables in the form of simple words or sentences. For room temperature, cold, warm, hot, etc., are linguistic terms.
- Temperature (t) = {too-cold, cold, warm, hot, too hot}
- Every member of this set is a linguistic term and it can cover some portion of overall temperature values.

Example: Air conditioner system controlled by a FLS

Construct membership functions for them

The membership functions of temperature variable are as shown −



Example: Air conditioner system controlled by a FLS

Construct knowledge base rules

Create a matrix of room temperature values versus target temperature values that an air conditioning system is expected to provide.

RoomTemp. /Target	Too Cold	Cold	Warm	Hot	Too Hot
Too Cold	No_Change	Heat	Heat	Heat	Heat
Cold	Cool	No_Change	Heat	Heat	Heat
Warm	Cool	Cool	No_Change	Heat	Heat
Hot	Cool	Cool	Cool	No_Change	Heat
Too Hot	Cool	Cool	Cool	Cool	No_Change

Example: Air conditioner system controlled by a FLS

Build a set of rules into the knowledge base in the form of IF-THEN-ELSE structures.

For Air Conditioner example, the following rules can be used:

1) IF (temperature is cold OR too-cold) AND (target is warm) THEN command is heat.

2) IF (temperature is hot OR too-hot) AND (target is warm) THEN command is cool.

3) IF (temperature is warm) AND (target is warm) THEN command is nochange.

Example: Air conditioner system controlled by a FLS

Defuzzification: The result is a fuzzy value and should be defuzzified to obtain a crisp output. This is the purpose of the defuzzifier component of a FLS. Defuzzification is performed according to the membership function of the output variable. o This defuzzification is not part of the 'mathematical fuzzy logic' and various strategies are possible.

The mostly-used algorithms for defuzzification are listed.

1) Finding the center of gravity.
2) Finding the center of gravity for singletons.
3) Finding the average mean.
4) Finding the left most maximum.
5) Finding the right most maximum.

Membership Function

- As this function allows you to quantify linguistic term. Also, represent a fuzzy set graphically. Although, MF for a fuzzy set A on the universe of discourse. That X is defined as $\mu A:X \rightarrow [0,1]$.
- In this function, between a value of 0 and 1, each element of X is mapped. We can define it as the degree of membership. Also, it quantifies the degree of membership of the element. That is in X to the fuzzy set A.

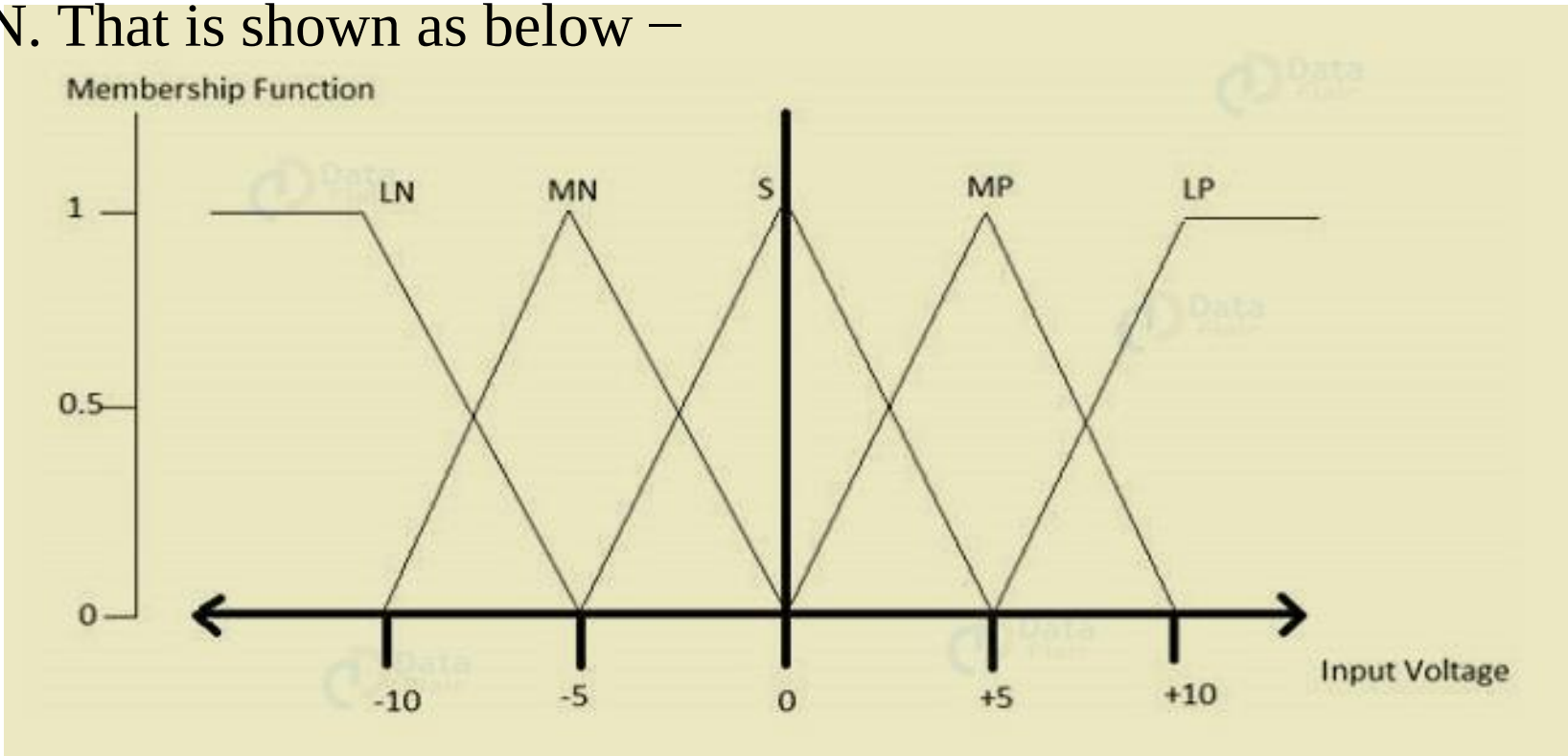
  x-axis– It represents the universe of discourse.

  y-axis – It represents the degrees of membership in the [0, 1] interval.

Membership Function

We can apply different membership functions to fuzzify a numerical value. Also, we use simple functions as complex. As they do not add more precision in the output.

We can define all membership functions for LP, MP, S, MN, and LN. That is shown as below −



Fuzzy Logic System – Membership Function

Triangular membership functions

$$\mu_F(x, a, b, c) = \begin{cases} 0, & \text{if } x < a \\ (x-a)/(b-a), & \text{if } a \le x \le b \\ (c-x)/(c-b), & \text{if } b \le x \le c \\ 0, & \text{if } c < x \end{cases}$$

Cont…

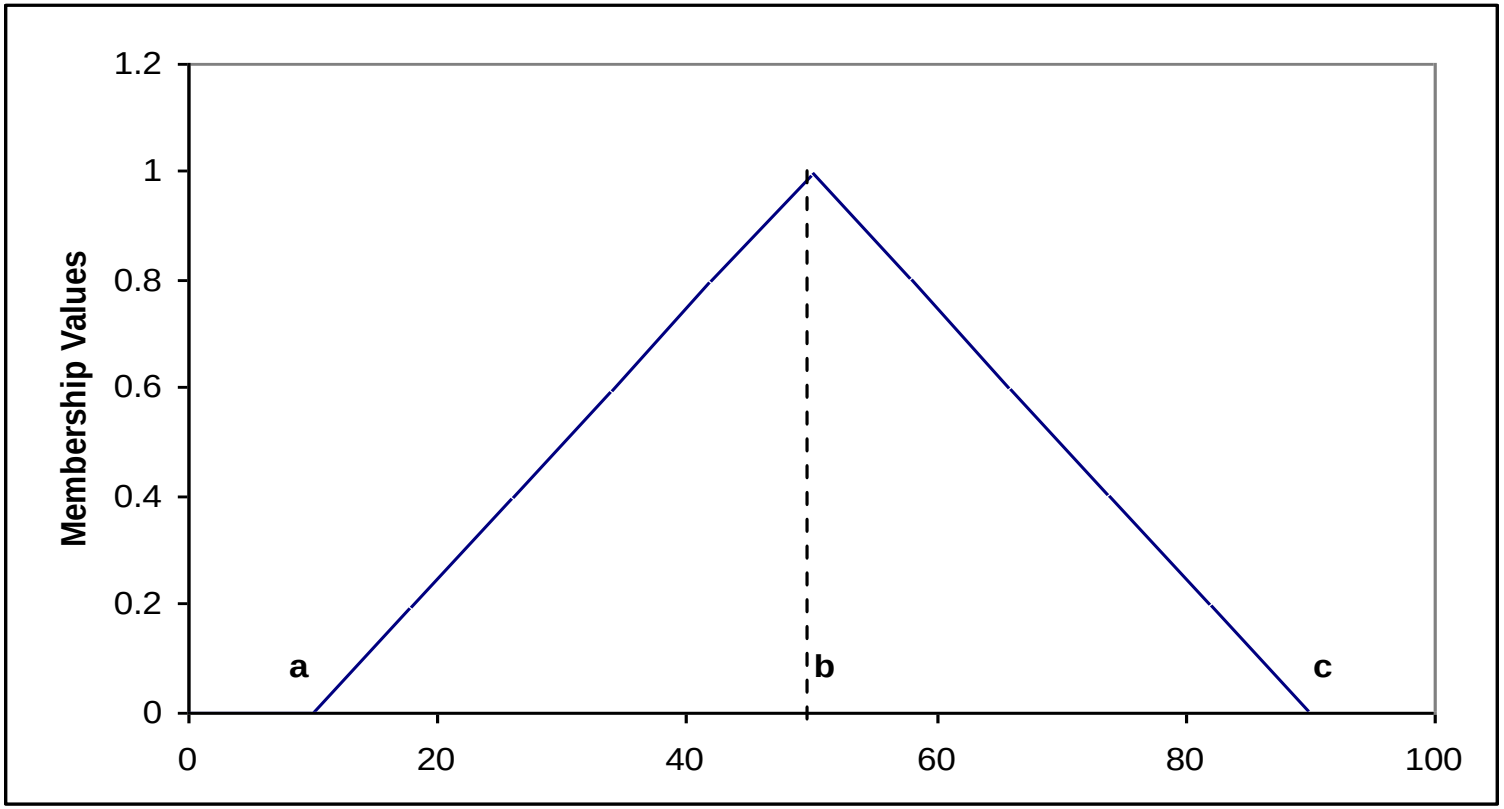


Figure Triangular Function

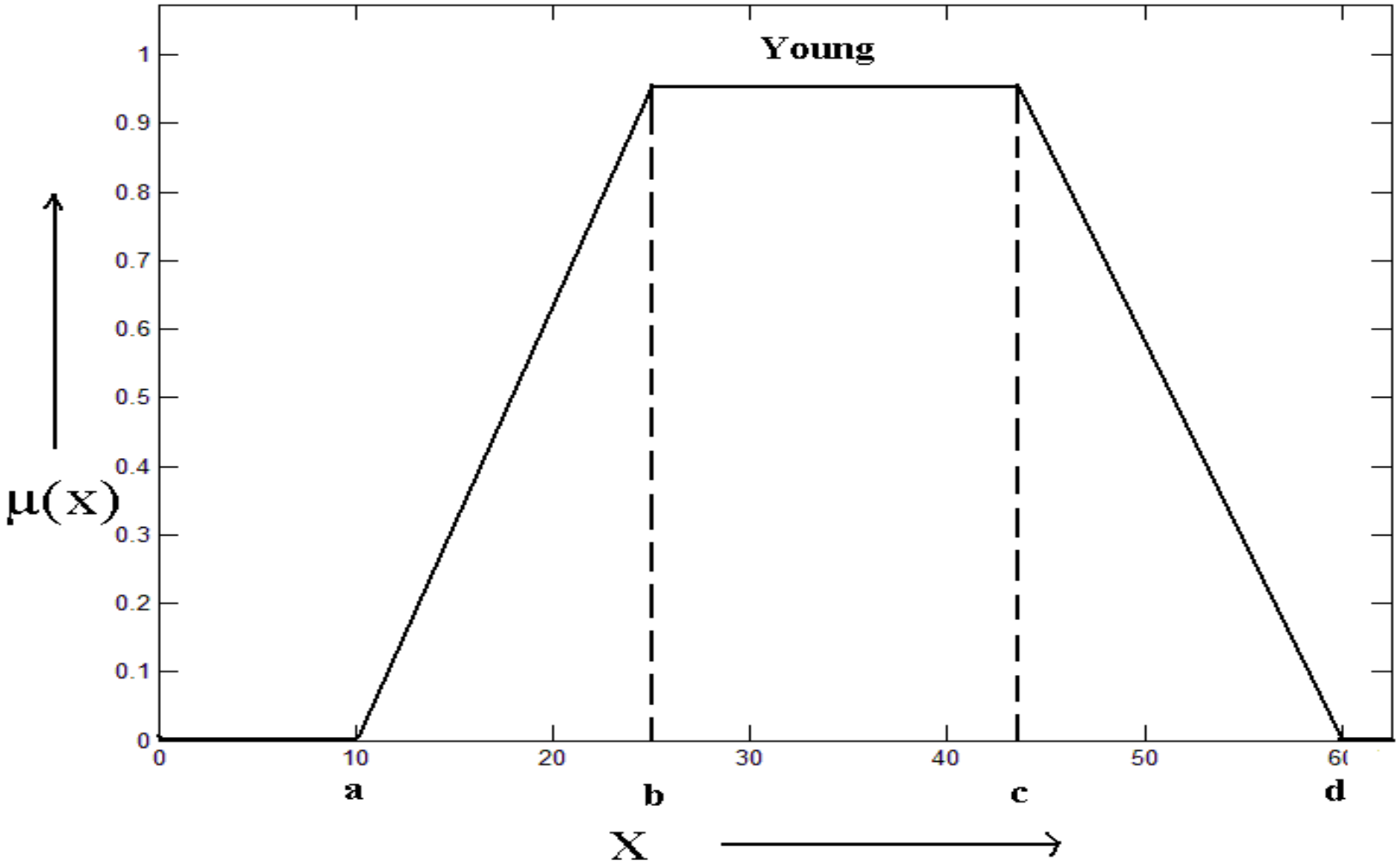
Trapezoidal membership function

$$\mu_F(x, a, b, c, d) = \begin{cases} 0, & \text{if } x < a \\ (x-a)/(b-a), & \text{if } a \le x \le b \\ 1, & \text{if } b < x < c \\ (d-x)/(d-c), & \text{if } c \le x \le d \\ 0, & \text{if } d < x \end{cases}$$

Trapezoidal membership function

$$\mu_F(x, a, b, c, d) = \begin{cases} 0, & \text{if } x < a \\ (x-a)/(b-a), & \text{if } a \le x \le b \\ 1, & \text{if } b < x < c \\ (d-x)/(d-c), & \text{if } c \le x \le d \\ 0, & \text{if } d < x \end{cases}$$

Cont…



Gaussian membership function

$$\mu(x,a,b) = e^{\frac{-(x-b)^2}{2a^2}}$$

The graph given in Fig. 10.6 is for parameters a = 0.22, b = 0.78

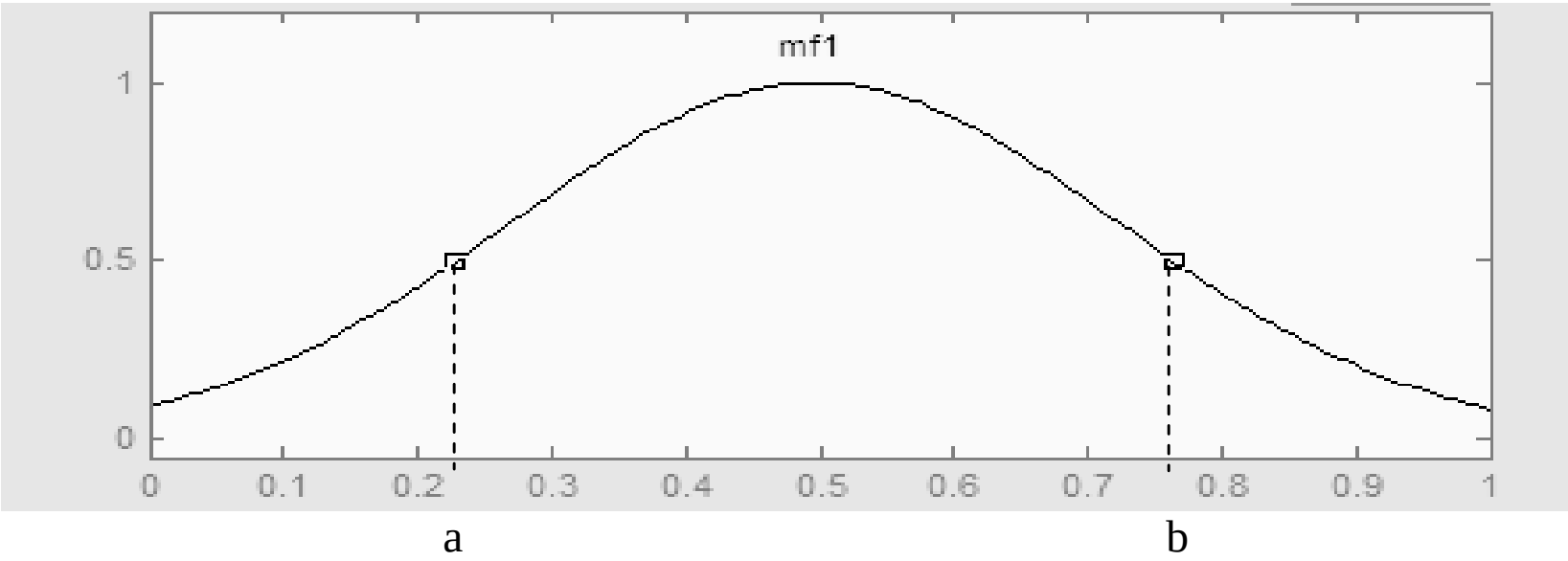


Figure Gaussian Membership Function

Applications of Fuzzy Logic

Following are the different application areas where the Fuzzy Logic concept is widely used:

- It is used in **Businesses** for decision-making support system.
- It is used in **Automative systems** for controlling the traffic and speed, and for improving the efficiency of automatic transmissions. **Automative systems** also use the shift scheduling method for automatic transmissions.
- This concept is also used in the **Defence** in various areas. Defence mainly uses the Fuzzy logic systems for underwater target recognition and the automatic target recognition of thermal infrared images.
- It is also widely used in the **Pattern Recognition and Classification** in the form of Fuzzy logic-based recognition and handwriting recognition. It is also used in the searching of fuzzy images.

Applications of Fuzzy Logic

- Fuzzy logic systems also used in **Securities**.
- It is also used in **microwave oven** for setting the lunes power and cooking strategy.
- This technique is also used in the area of **modern control systems** such as expert systems.
- **Finance** is also another application where this concept is used for predicting the stock market, and for managing the funds.
- It is also used for controlling the brakes.
- It is also used in the **industries of chemicals** for controlling the ph, and chemical distillation process.
- It is also used in the **industries of manufacturing** for the optimization of milk and cheese production.
- It is also used in the vacuum cleaners, and the timings of washing machines.
- It is also used in heaters, air conditioners, and humidifiers.

Utility Theory and utility functions

- Decision theory, in its simplest form, deals with choosing among actions based on the desirability of their immediate outcomes
- If agent may not know the current state and define RESULT(a) as a random variable whose values are the possible outcome states. The probability of outcome s , given evidence observations e, is written

$$P(RESULT(a) = s' \mid a, e)$$

where the a on the right-hand side of the conditioning bar stands for the event that action a is executed

- The agent's preferences are captured by a utility function, U(s), which assigns a single number to express the desirability of a state.

- The expected utility of an action given the evidence, EU (a|e), is just the average utility value of the outcomes, weighted by the probability that the outcome occurs:

$$EU(a|\mathbf{e}) = \sum_{s'} P(RESULT(a) = s' \mid a, \mathbf{e})U(s')$$

The principle of maximum expected utility (MEU) says that a rational agent should choose the action that maximizes the agent's expected utility:

$$action = \underset{a}{\operatorname{argmax}}\, EU(a|e)$$

In a sense, the MEU principle could be seen as defining all of AI. All an intelligent agent has to do is calculate the various quantities, maximize utility over its actions, and away it goes.

Basis of Utility Theory

- Intuitively, the principle of Maximum Expected Utility (MEU) seems like a reasonable way to make decisions, but it is by no means obvious that it is the only rational way.

- Why should maximizing the average utility be so special?
- What's wrong with an agent that maximizes the weighted sum of the cubes of the possible utilities, or tries to minimize the worst possible loss?
- Could an agent act rationally just by expressing preferences between states, without giving them numeric values?
- Finally, why should a utility function with the required properties exist at all?

Constraints on rational preferences

- These questions can be answered by writing down some constraints on the preferences that a rational agent should have and then showing that the MEU principle can be derived from the constraints

  $A \succ B$ the agent prefers A over B.

  $A \sim B$ the agent is indifferent between A and B.

  $A \succsim B$ the agent prefers A over B or is indifferent between them.

We can think of the set of outcomes for each action as a lottery—think of each action as a ticket. A lottery L with possible outcomes S1,...,Sn that occur with probabilities p1,...,pn is written

L = [p1, S1; p2, S2; ... pn, Sn] .

Constraints on rational preferences

- In general, each outcome Si of a lottery can be either an atomic state or another lottery. The primary issue for utility theory is to understand how preferences between complex lotteries are related to preferences between the underlying states in those lotteries.

To address this issue we list six constraints that we require any reasonable preference relation to obey:

- **Orderability:** Given any two lotteries, a rational agent must either prefer one to the other or else rate the two as equally preferable. That is, the agent cannot avoid deciding.

  Exactly one of $(A \succ B)$, $(B \succ A)$, or $(A \sim B)$ holds.

Constraints on rational preferences

- **Transitivity**: Given any three lotteries, if an agent prefers A to B and prefers B to C, then the agent must prefer A to C.

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$$

- **Continuity:** If some lottery B is between A and C in preference, then there is some probability p for which the rational agent will be indifferent between getting B for sure and the lottery that yields A with probability p and C with probability 1 − p.

$$A \succ B \succ C \Rightarrow \exists\ p\ [p, A;\ 1 - p, C] \sim B\ .$$

- **Substitutability:** If an agent is indifferent between two lotteries A and B, then the agent is indifferent between two more complex lotteries that are the same except that B is substituted for A in one of them. This holds regardless of the probabilities and the other outcome(s) in the lotteries.

$$A \sim B \Rightarrow [p, A;\ 1 - p, C] \sim [p, B;\ 1 - p, C]\ .$$

This also holds if we substitute $\succ$ for $\sim$ in this axiom.

Constraints on rational preferences

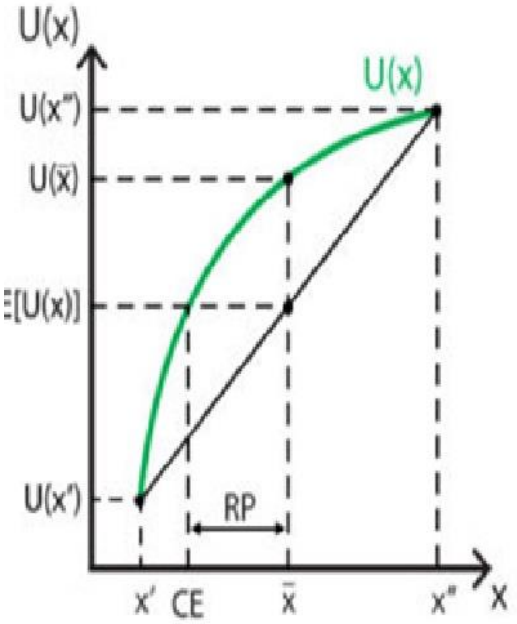
- **Monotonicity:** Suppose two lotteries have the same two possible outcomes, A and B. If an agent prefers A to B, then the agent must prefer the lottery that has a higher probability for A (and vice versa).

$$A \succ B \Rightarrow (p > q \Leftrightarrow [p, A; 1 - p, B] \succ [q, A; 1 - q, B])$$

- **Decomposability:** Compound lotteries can be reduced to simpler ones using the laws of probability. This has been called the "no fun in gambling" rule because it says that two consecutive lotteries can be compressed into a single equivalent lottery.

$$[p, A; 1 - p, [q,B; 1 - q, C]] \sim [p, A; (1 - p)q,B; (1 - p)(1 - q), C] .$$

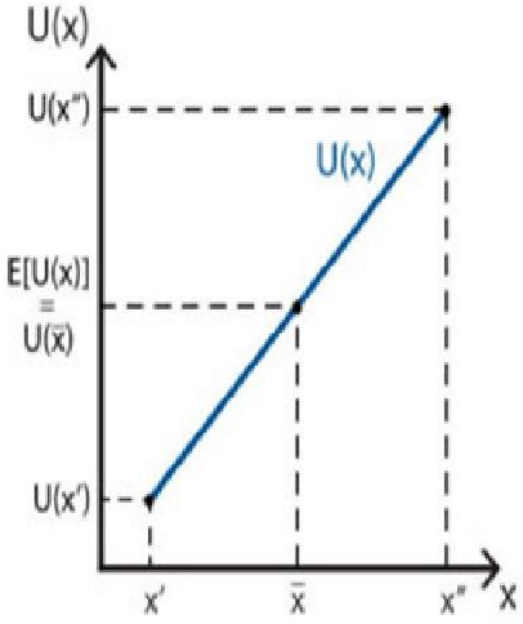
Expected Utilities



Risk averse individual

$E[U(x)] < U(\bar{x})$

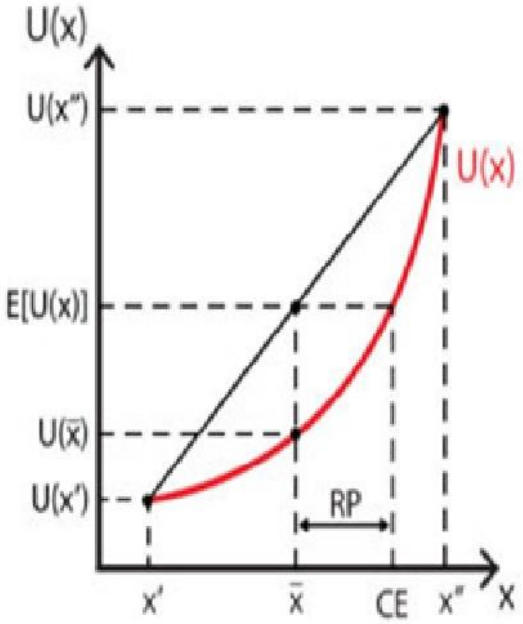
$CE < \bar{x}$



Risk neutral individual

$E[U(x)] = U(\bar{x})$

$CE = \bar{x}$



Risk loving individual

$E[U(x)] > U(\bar{x})$

$CE > \bar{x}$