

# KNOWLEDGE BASED SYSTEM FOR FAULT TREE ANALYSIS USING FUZZY LOGIC

## A DISSERTATION

*Submitted in partial fulfilment of the  
requirements for the award of the degree*

*of*

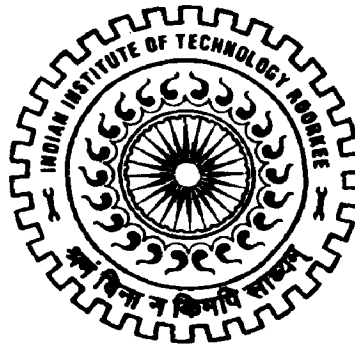
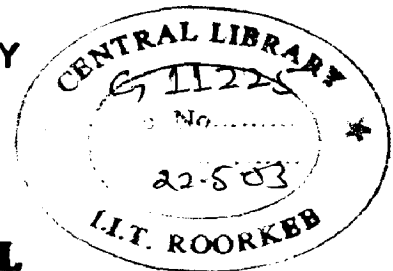
MASTER OF TECHNOLOGY

*in*

INFORMATION TECHNOLOGY

By

**SANDEEP SABBARWAL**



**ER & DCI  
NOIDA**

**IIT Roorkee-ER&DCI, Noida  
C-56/1, "Anusandhan Bhawan"  
Sector 62, Noida-201 307**

**FEBRUARY, 2003**

## CANDIDATE'S DECLARATION

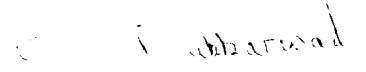
---

This is to certify that the work, which is being presented in this dissertation, entitled "KNOWLEDGE BASED SYSTEM FOR FAULT TREE ANALYSIS USING FUZZY LOGIC", in partial fulfillment of the requirements for the award of the degree of Master of Technology in Information Technology submitted in IIT, Roorkee – ER&DCI Campus, Noida, is an authentic record of my own work carried out from August 2002 to February 2003, under the supervision of Mrs. Pratibha Yadav, Scientist 'E', Scientific Analysis Group, Defense Research and Development Organization, Delhi.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

Date: 21-02-03

Place: Noida

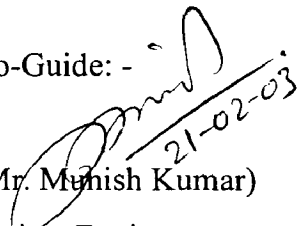
  
(Sandeep Sabbarwal)

---

## CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Co-Guide: -

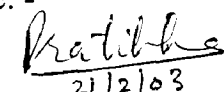
  
(Mr. Manish Kumar)

Project Engineer,

ER&DCI,

Noida

Guide: -

  
(Mrs. Pratibha Yadav)

Scientist 'E',

Scientific Analysis Group,

D.R.D.O., Delhi

## ACKNOWLEDGEMENT

---

I hereby take the privilege to express my deepest sense of gratitude to **Prof. Prem Vrat**, Director, Indian Institute of Technology, Roorkee, and **Mr. R.K. Verma**, Executive Director, Electronics Research & Development Center of India, Noida for providing me with this valuable opportunity to carry out this work. I am also very grateful to **Prof. A.K. Awasthi**, Programme Director and Dean, Post Graduate Studies and Research, **Prof. R.P. Agrawal**, course coordinator, IIT, Roorkee and **Mr. V.N. Shukla**, course coordinator, ER&DCI, Noida for providing the best of the facilities for the completion of this work and constant encouragement towards the goal. I owe special thanks to **Prof. C.E. Veni Madhavan**, Director, Scientific Analysis Group, Defense Research & Development Organization, Delhi for providing me an opportunity to carry out the dissertation at the organization.

I am grateful to my guides **Mrs. Pratibha Yadav**, Scientist 'E', SAG, DRDO and **Dr. P.K. Saxena**, Scientist 'G', SAG, DRDO, Delhi for their valuable guidance, advice, suggestions and constant encouragement through numerous discussions and demonstrations.

My sincere thanks are due to **Dr. S.S. Agrawal**, Emeritus Scientist (CSIR), Central Scientific Instruments Organization, Delhi and **Mr. Munish Kumar**, Project Engineer, ER&DCI, Noida for providing me with their valuable reference for securing the live project at DRDO, Delhi. I am thankful to **Dr. Poonam Rani Gupta**, Reader, ER&DCI, Noida for her continuous inspiration and support throughout the course of this dissertation.

However, I owe special thanks to my classmate Vivek Kumar, and all other friends who have helped me formulate my ideas and have been a constant support. Thanks to my parents who provided their support and enduring confidence in me during my entire life. Last but not the least, I thank almighty for being on my side from the conception of this idea to its implementation.



(Sandeep Sabbarwal)

Enrolment No. 019042

# CONTENTS

---

CANDIDATES DECLARATION	(i)
ACKNOWLEDGEMENT	(ii)
ABSTRACT	1
<b>1. INTRODUCTION</b>	<b>3</b>
1.1 Overview	3
1.2 Objective of the dissertation	3
1.3 Scope of the work	4
1.4 Organization of the thesis	4
<b>2. LITERATURE SURVEY</b>	<b>7</b>
2.1 What is Fault Tree	7
2.2 What is Fault Tree Analysis	7
2.3 Logic Symbols Of Fault Tree Analysis	9
2.4 Fault Tree Methodology	11
2.5 Basic Idea Of Fuzzy Logic	17
2.6 Fuzzy Set Representation	21
2.7 Fuzzy Set Operators	21
2.8 General Functional Blocks For Fuzzy Logic Applications	23
<b>3. ANALYSIS OF THE PROBLEM</b>	<b>25</b>
3.1 Analysis Of Problem Using Sample Fault Tree	25
3.2 Analysis Of Ammonia Storage Tank	27
3.3 Fault Tree For Ammonia Storage Tank	30
3.4 Logic Symbols Utilized In The Fault Tree For Ammonia Storage Tank	30
3.5 Failure Probabilities Of Basic Events	31
3.6 Minimal Cut Sets For The Fault Tree Of Ammonia Storage Tank	35

<b>4. DESIGN OF THE PROPOSED SOLUTION</b>	<b>37</b>
4.1 Design Approach	37
4.2 Design Assumptions	39
4.3 Design For File 1	40
4.4 Design For File 2	40
<b>5. IMPLEMENTATION ASPECTS</b>	<b>41</b>
5.1 Program Outline	41
5.2 Implementation Approach	41
5.3 Algorithm For Fault Tree Creation	42
5.4 Algorithm For Intermediate Processing At Basic Events	43
5.5 Algorithm For The Traversal Of The Fault Tree Performing Required Operation At Interior Nodes	44
<b>6. RESULTS AND DISCUSSION</b>	<b>47</b>
6.1 User Interface And Output Screens	47
6.2 Results And Discussion	53
<b>7. CONCLUSION</b>	<b>55</b>
<b>REFERENCES</b>	<b>57</b>



## ABSTRACT

---

Decision making in an imprecise and vague environment has been an area of research for dealing with many real life problems. The decision making process uses inference based on the input information. The information can be analyzed by either system alone or by the human assisted system.

Fault Tree is a graphical representation of the various conditions of equipment and human failure that can result in an accident or an undesirable event. The analysis of a Fault Tree provides one of the most credible means by which an undesired event may be identified. Although the fault tree provides useful information by displaying the interactions of equipment failures that could result in an accident, even an experienced analyst cannot identify directly all the combinations of equipment failures that can lead to the accident, from a given fault tree.

Present work is an attempt to develop a generalized tool for Fault Tree analysis using Fuzzy Logic for one of the laboratory dealing in Hazard quantification at Defense Research & Development Organization. The Software for Fault Tree Analysis generates Trapezium type of membership function for each of the Basic Events of the Fault Tree under consideration, traverses and computes the membership grade of top event occurrence, based on various fuzzy operations.

# INTRODUCTION

---

## 1.1 Overview

Decision making in an imprecise and vague environment has been an area of research for dealing with many real life problems. The concept of Fuzzy Logic and Fuzzy sets have been introduced as a means in this direction.

The concept of Fuzzy Logic was invented by Professor Lotfi A. Zadeh of the University of California at Berkeley in 1965. For the first 20 years, most engineers in the world paid little attention to this technology. However, since 1985, Japan has developed Fuzzy Logic fever, applying it to thousands of products ranging from auto-focus cameras to subway control systems. Today, Fuzzy Logic is used around the world in many application areas such as control, instrumentation, laboratory automation, factory automation, system identification, system modeling, pattern recognition, image database, decision support, data mining, stock selection, house hunting, investing, market survey, economic study, quality management, political analysis, insurance, resource management and much much more.....

In India, this technology is in its initial stages and thus, in this thesis, an attempt is made to contribute to the applications of the Fuzzy Logic technology by developing a soft computing tool for the Fault Tree analysis that can be used for any generalized data.

## 1.2 Objective of the Dissertation

The objective of this dissertation is to develop a soft computing tool for hazard monitoring, based on the input data through various sensors. The sensors are placed at various physical locations of a plant under consideration. They are also arranged in a hierarchical manner depending on their criticality in monitoring parameters. They essentially form a tree structure for decision making.

The task now is to compute the possibility of the failure of the top event[1] by analyzing the fault tolerance tree and making use of knowledge base for the failure probabilities[3] obtained from numerous observations.

### **1.3 Scope of the work**

As fuzzy logic[2] based analysis method is becoming popular day by day. The most important aspect of this approach is the way in which the imprecision[4] is handled. In the decision making process often inference is drawn based on the input information. The information is further analyzed by either :-

- (i) a system alone or
- (ii) by the human assisted system.

In the former case, which is often desirable for its automatic functioning, human knowledge is embedded in the system to make them intelligent.

Fuzzy logic again plays an important role in quantification of human knowledge. Hazard monitoring[5] is one area where minimum human involvement is desirable. For this purpose an intelligent monitoring and controlled system will be of immense use which can monitor, alarm and control the hazard with maximum efficiency.

One of the laboratory of “Defense Research & Development Organization” is actively involved in “Research & Development (R&D)” work in the area of hazard monitoring and development of models for their management.

The work proposed in this dissertation will be very useful for an ongoing “Defense Research & Development Organization” project and the modules developed here will be included in the final tool kit being developed.

### **1.4 Organization of the thesis**

The first chapter gives an overview of the dissertation and discussed the objective and scope of the work. The second chapter presents the essence of the literature surveyed and discusses relevant theoretical issues. The third chapter carries out a detailed analysis of the problem, the solution for which is to be developed. This is followed by chapter



four which presents the detailed design of the proposed solution. Chapter five gives the implementation of the solution. In chapter six, results obtained from the software developed are presented and discussed. Finally, chapter seven concludes the work, discussing some enhancements that may be incorporated in the software in future.

## LITERATURE SURVEY

---

This chapter deals with the theoretical aspects of the fault tree and fault tree analysis using Boolean logic and Boolean gates considering few examples in order to achieve better understanding of the concept. This is accompanied with the basic idea of Fuzzy logic and Fuzzy operators with the general functional blocks for fuzzy logic applications which would be applied to the Fault Tree Analysis.

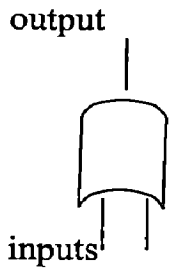
### 2.1 What is Fault Tree

Fault Tree[1] is a graphical representation of the various conditions of equipment and human failure that can result in an accident or an undesirable event. The solution of a Fault Tree provides the most credible means by which an undesired event may occur. In fact the solution is a list of the cut-sets of basic events which are together sufficient to result in the event of interest.

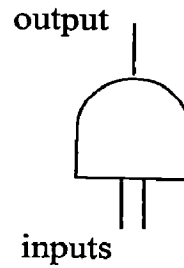
### 2.2 What is Fault Tree Analysis

Fault Tree Analysis[6] is unique amongst the methodologies used for hazard assessment in so far as it starts with the consequences of an event and seeks to determine the causes while other techniques start with the failure of a component and seek the consequences.

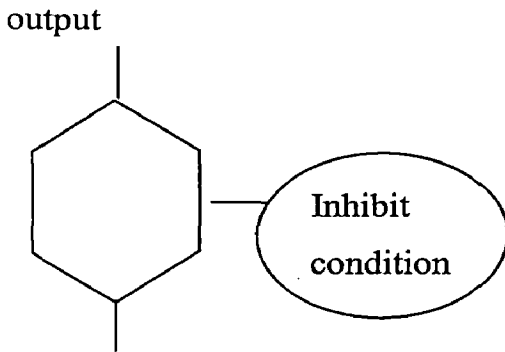
Fault Tree Analysis begins with the definition of a relevant undesirable event or the 'top event' that needs to be avoided and seeks the immediate causes of that event. Each of the immediate causes is analyzed to arrive at basic events or other intermediate events until the basic events responsible for each of the immediate causes are identified in terms of failure of process equipment or an error on the part of the operator. Fault Tree Analysis is therefore referred to as a 'backward looking' or 'top-down' approach.



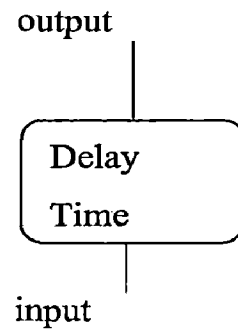
(i)  
OR GATE



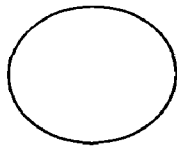
(ii)  
AND GATE



(iii)  
INHIBIT GATE



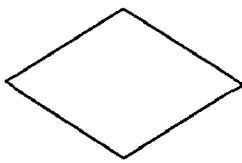
(iv)  
DELAY GATE



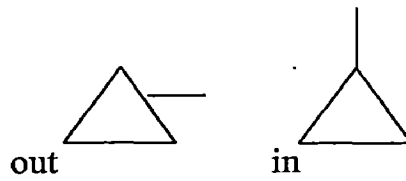
(v)  
BASIC EVENT



(vi)  
INTERMEDIATE EVENT



(vii)  
UNDEVELOPED EVENT



(viii)  
TRANSFER SYMBOL

Fig.2.3.1 Various Logic Symbols

## 2.3 Logic Symbols of Fault Tree Analysis

The following symbols as shown in figure 2.3.1 are used in the construction of a fault tree displaying the various relationships :-

- (i) **OR GATE** :- The OR gate indicates that the output event occurs if any of the input events occur.
- (ii) **AND GATE** :- The AND gate indicates that the output event occurs only when all the input events occur.
- (iii) **INHIBIT GATE** :- The INHIBIT gate indicates that the output event occurs when the input event occurs and the inhibit condition is satisfied.
- (iv) **DELAY GATE** :- The DELAY gate indicates that the output event occurs when the input event has occurred and the specified delay time has lapsed.
- (v) **BASIC EVENT** :- The BASIC event represents a basic equipment fault or failure that requires no further development into more basic faults or failures.
- (vi) **INTERMEDIATE EVENT** :- The INTERMEDIATE event represents a fault event that results from the interactions of other fault events that are developed through logic gates such as those defined above.
- (vii) **UNDEVELOPED EVENT** :- The UNDEVELOPED event represents a fault event that is not examined further because information is unavailable or because its consequence is insignificant.
- (viii) **TRANSFER SYMBOL** :- The TRANSFER IN symbol indicates that the fault tree is developed further at the occurrence of the corresponding TRANSFER OUT symbol.

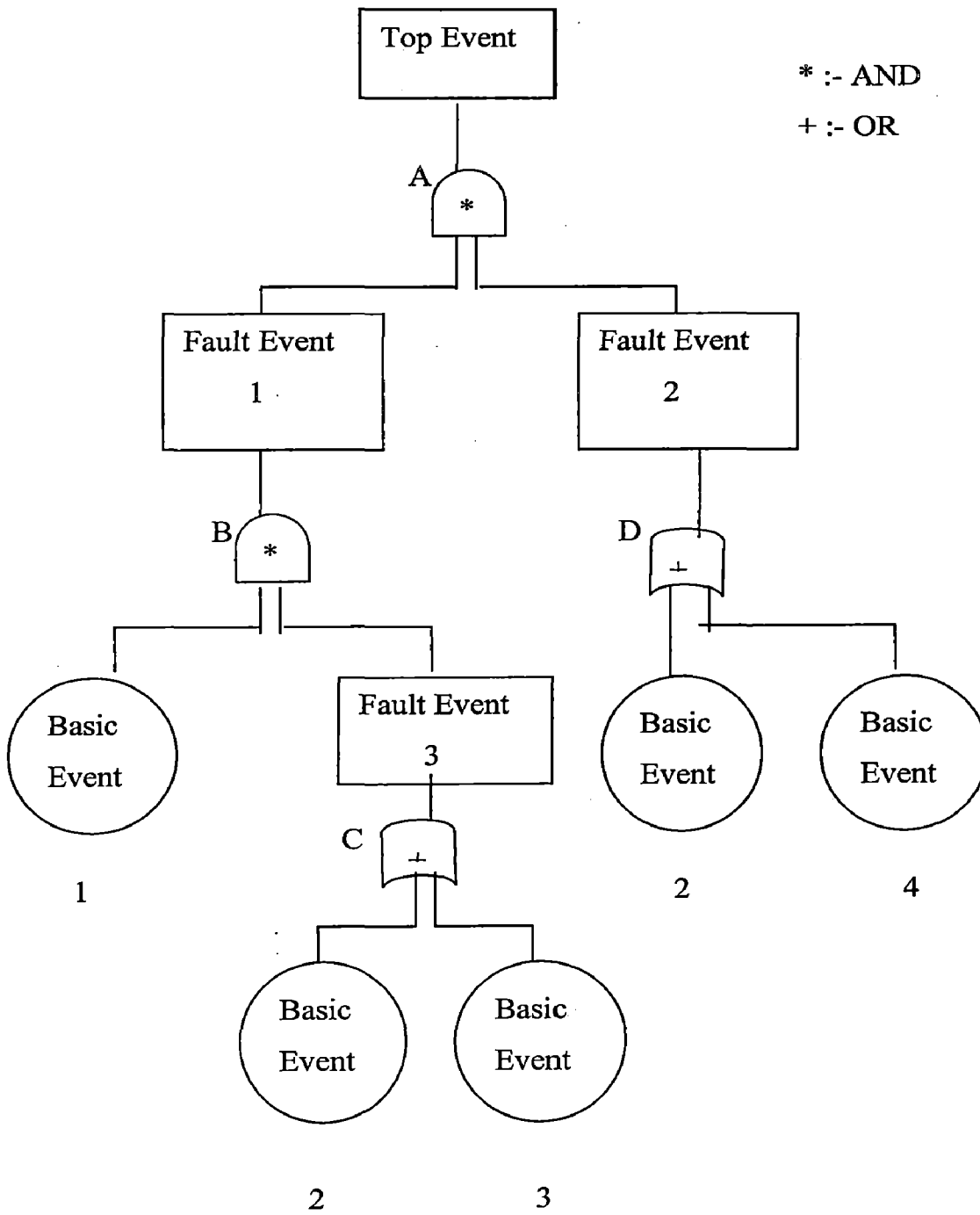


Fig. 2.4.2.1 Sample Fault Tree using various logic symbols

## 2.4 Fault Tree Methodology[7]

Four basic stages involved in the Fault Tree Analysis are described as under :-

### 2.4.1 Definition of Problem

A precise definition of the undesirable event or the 'top event' forms the basis of Fault Tree Analysis. The definition should identify the event in terms of what are the immediate consequences of the event (fire, toxic gas leakage or release of inflammable material), location of the event within a system (reactor, distillation column, compressor) and the conditions under which the event is expected to occur (normal operation, fire in adjacent unit). In other words to be meaningful the event description should necessarily encompass 'what', 'where', and 'when' of the event.

The level of resolution of a fault tree needs to be considered before its construction. Availability of failure rate data of various components in a system determines the details to be included in a fault tree. For example, a temperature measuring system can be described as several hardware items (sensor, transmitter, controller) if the failure rate data for its components is available. Otherwise the resolution can remain at temperature indicator level.

### 2.4.2 Construction of Fault Tree

A sample Fault Tree is shown in figure 2.4.2.1 using the symbols defined in section 2.3. The immediate causes of the top event are shown in the Fault Tree a level below the top event with their relationship to the 'top event'. If any one of the immediate causes results directly in the 'top event', the causes are connected to the top event with an OR logic gate. If all the immediate causes are required for 'top event' occurrence the causes are concluded to the 'top event' with an AND logic gate. Each of the immediate causes is then treated in the same manner as the top event and its immediate, necessary, and sufficient causes are determined and shown on the fault tree with appropriate logic gates. This development continues until all intermediate fault events have been resolved into their basic causes. Following basic rules are taken into consideration for ensuring consistency in the fault tree construction :-



- (i) **Fault Event Statements** :- The statement to be entered into the event boxes or circles must state precisely the nature of the fault ('what'), its location ('where') and circumstances under which it occurs ('when').
- (ii) **The 'No Miracle' Rule** :- Fault Tree Analysis does not permit assumptions in which the unexpected failure of an equipment interrupts or prevents an accident.
- (iii) **The 'Complete the Gate' Rule** :- All inputs to a particular gate should be completely defined before further analysis of any gate. The fault tree needs to be completed in levels and each level should be completed before beginning the next level.
- (iv) **The 'No Gate-to-Gate' Rule** :- Gate inputs should be properly defined fault events, and gates should not be connected to other gates.
- (v) **When there are two inputs to an AND gate** one input should be a probability and the other a frequency so that the product has the units failures per day or per year.

### **2.4.3 Fault Tree Solution**

Although the fault tree provides much useful information by displaying the interactions of equipment failures that could result in an accident, even an experienced analyst cannot identify directly from the fault tree all the combinations of equipment failures that can lead to the accident. This section discusses the method of "solving" the fault tree, or obtaining the minimal cut-sets for the fault tree. The minimal cut-sets are all the combinations of equipment failures that can result in the fault tree top event, and they are logically equivalent to the information displayed in the fault tree. The minimal cut-sets are useful for ranking the ways in which the accident may occur, and they allow quantification of the fault tree if appropriate data are available. The method described here will allow solution of many of the fault trees encountered in practice. The fault tree solution method has following four steps :-

- a. Uniquely identify all gates and basic events
- b. Resolve all gates into basic events
- c. Remove duplicate events within sets
- d. Delete all supersets (sets that contain another set as a sub set).

The result of the above procedure is the list of minimal cut-sets for the fault tree.

The above procedure is further explained with the help of a sample fault tree as shown in figure 2.4.2.1.

Step (a) :- To uniquely identify all gates and basic events in the fault tree. In figure 2.4.2.1, the gates are identified with letters and the basic events with numbers. Each identification should be unique, and if a basic event appears more than once in the fault tree, it should have the same identifier each time.

Step (b) :- To resolve all the gates into basic events. This is done in a matrix format, beginning with the top event and proceeding through the matrix until all gates are resolved. Gates are resolved by replacing them in the matrix with their inputs. The top event is always the first entry in the matrix and is entered in the first column of the first row as shown in figure 2.4.3.1. Two rules are used for entering the remaining information in the matrix i.e. the OR-gate rule, and the AND-gate rule.

Rule (i) :- The OR-gate rule :- When resolving an OR gate in the matrix, the leftmost input of the OR gate replaces the gate identifier in the matrix, and all other inputs to the OR gate are inserted in the next available row, one input per row. The next available row means the next empty row of the matrix. In addition, if there are other entries in the row where the OR gate appeared, these entries must be entered (repeated) in all the rows that contain the gate inputs.

Rule (ii) :- The AND-gate rule :- When resolving an AND gate in the matrix, the leftmost input to the AND gate replaces the gate identifier in the matrix, and the other inputs to the AND gate are inserted in the next available column, one input per column, on the same row that the AND gate appeared on. INHIBIT and DELAY gates are resolved in the same manner as AND gate.

A			

(a)

B	D		

(b)

1	D	C	

(c)

1	2	C	
1	4	C	

(d)

1	2	2	
1	4	C	
1	2	3	

(e)

1	2	2	
1	4	2	
1	2	3	
1	4	3	

(f)

Fig. 2.4.3.1. Matrix For Resolving Gates In Sample Fault Tree

These two rules are repeated until only basic event identifiers remain in the matrix. The sample fault tree shown in figure 2.4.2.1 can be solved using these two rules. Figure 2.4.3.1(a) shows the first entry in the matrix, gate A, which is the top event in our sample fault tree. The AND-gate rule is applied to resolve gate A into its inputs, gates B and D, as shown in figure 2.4.3.1(b). The next gate to resolve, is gate B. Gate B is also an AND gate, so its inputs are entered on the same row as gate B. This replacement is shown in figure 2.4.3.1(c). Next, Gate D is an OR gate, so its leftmost input replaces D and its second input is entered in the next available row, as shown in figure 2.4.3.1(d). Other components of row 1 (where gate D appeared) are also entered (repeated) on the next available row. Gate C is now the only gate left in the matrix, appearing on both row 1 and row 2. Each occurrence of gate C is resolved separately. First, on row 1, the OR gate rule is applied to gate C as shown in figure 2.4.3.1(e), resulting in a new set of entries in row 3. Similarly, the second occurrence of gate C is resolved as shown in figure 2.4.3.1(f). This completes the resolution of the gates in the matrix. The results of this step are four sets of basic events :-

Set 1 :- 1,2,2

Set 2 :- 1,2,4

Set 3 :- 1,2,3

Set 4 :- 1,3,4

Step (c) :- To remove duplicate events within each set of basic events identified in step(b). Only set 1 above has a repeated basic event in the results : Basic event 2 appears twice. When we remove this repeated event, the sets of BASIC events are :-

Set 1 :- 1,2

Set 2 :- 1,2,4

Set 3 :- 1,2,3

Set 4 :- 1,3,4

Step (d) :- To delete all supersets that appear in the sets of basic events. In the results above there are two supersets. Both set 2 and set 3 are supersets of set 1; that is, sets 2 and 3 each contain set 1 as a subset. Once these supersets are deleted, the remaining sets are the minimal cut-sets for the sample fault tree :-

Minimal Cut-Set 1 :- 1,2

Minimal Cut-Set 2 :- 1,3,4

#### **2.4.4 Minimal Cut Set Ranking**

Ranking the minimal cut sets is the final step of the fault tree analysis procedure.

For a qualitative ranking, two factors can be considered :-

- (i) **Structural importance** :- Structural importance is based on the number of the basic events that are in each minimal cut set. In this type of ranking, a one-event minimal cut set is more important than a two-event minimal cut set; a two-event set is more important than a three-event set; and so on. This ranking implies that one event is more likely to occur than two events, two events are more likely to occur than three events, etc.
- (ii) **Type of event** :- It considers ranking within each size of minimal cut set; for example, ranking the two-event minimal cut sets, based on what type of events make up the minimal cut set. The general rule that guides this ranking is :-
  - (a) Human error
  - (b) Active equipment failure
  - (c) Passive equipment failure

This ranking implies that human error are more likely to occur than active equipment failures (functioning equipment, such as a running pump) and that active equipment failures are more likely to occur than passive equipment failures (static, non-functioning equipment, such as a storage tank). Using this rule on a list of two-event minimal cut sets might result in the following ranking :-

Rank	Basic Event Type 1	Basic Event Type 2
1	Human error	Human error
2	Human error	Active equipment failure
3	Human error	Passive equipment failure
4	Active equipment failure	Active equipment failure
5	Active equipment failure	Passive equipment failure
6	Passive equipment failure	Passive equipment failure

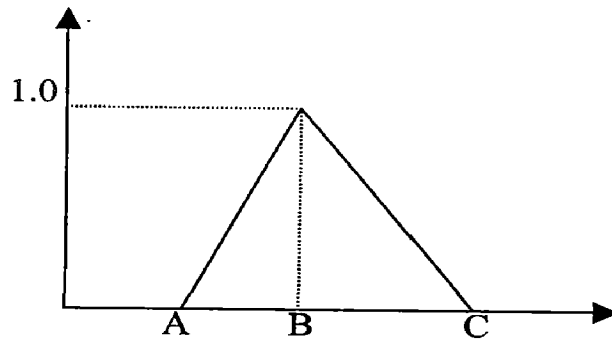
These rankings, although suggested by experience, may differ significantly from system to system based on such factors as the quality of equipment, the degree of operator training etc. The best qualitative ranking method is for an experienced analyst or engineer to examine the individual minimal cut sets and establish the most important sets based on actual operating experience.

## 2.5. Basic Idea Of Fuzzy Logic[8]

Fuzzy Logic was invented by Professor L.A. Zadeh in 1965. Since 1985, fuzzy logic based controllers have been used in more than 2000 industrial and consumer products such as washing machines, vaccum cleaners, rice cookers, elevator control systems, air conditioners, and anti-lock braking systems. Commercial applications of fuzzy logic in other areas such as speech and image processing and decision support have also been available, although it is not as wide spread as in control systems.

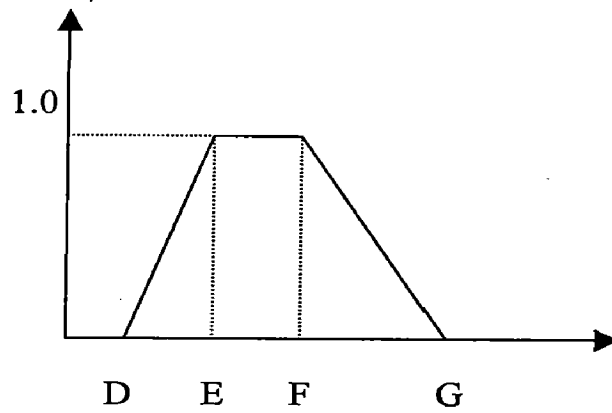
The idea of fuzzy logic is to allow one to represent fuzzy concepts. In the real world, we often have to deal with fuzzy concepts such as “high” speed, “low” temperature, “strong” signal, “tall” person, etc. A fuzzy concept can also stand for an abstract idea (such as desirability, intention, portability, etc.) which can not be directly and easily measured by physical and objective means.





(a)

A triangle membership function



(b)

A trapezoid membership function

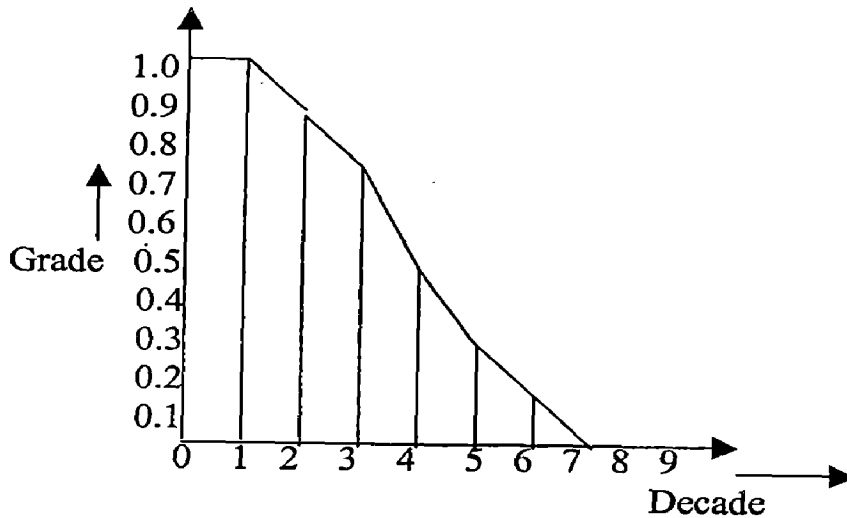
Fig. 2.5.1. Triangle and Trapezoid membership functions

A fuzzy concept such as “highness” in “high speed” is ill-defined. In this case, the fuzzy concept “highness” is related to “speed”. Given a value of speed, the value of “highness” may neither be a crisp true (represented by 1) nor a crisp false (represented by 0). The value may be a degree of truth in the interval from 0 to 1, and is defined by a membership function mapping from “speed” to the interval  $[0,1]$ . A value of membership function is called a grade of membership.

Each variable such as speed has a domain. Let  $[m,n]$  denote an interval from  $m$  to  $n$ . A fuzzy set for a variable is defined by a membership function mapping from the domain of the variable to  $[0,1]$ . For example, we may set the domain of the variable “speed” to be  $[0,100]$ , and define a fuzzy concept “high” by a membership function mapping from  $[0,100]$  to  $[0,1]$ . Note that we restrict ourselves to the case where fuzzy logic is applied to variables whose domains are numerical.

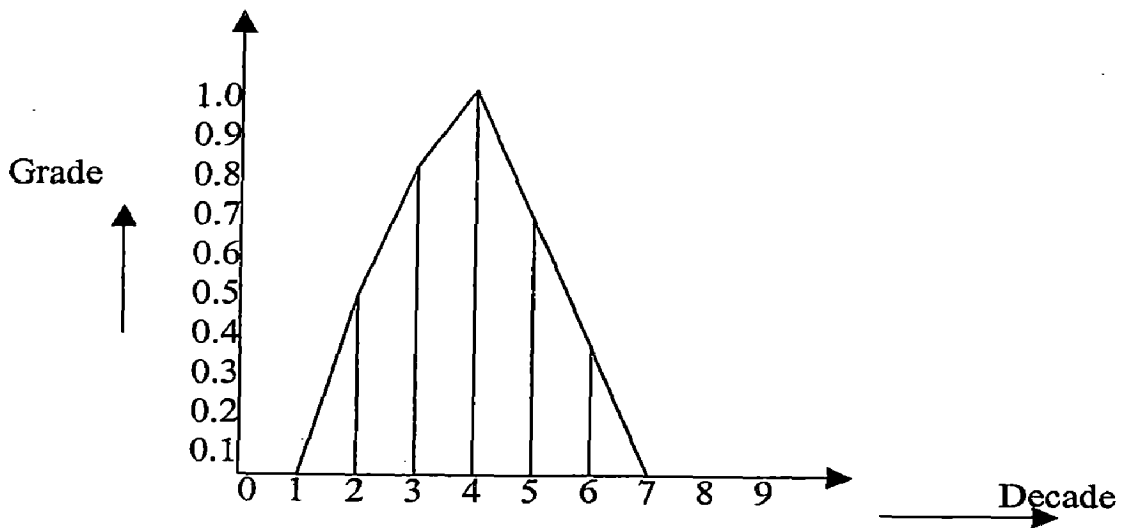
The two most popularly used forms of membership functions, namely triangle and trapezoid, are shown in figure 2.5.1.(a) and figure 2.5.1.(b) respectively. A triangle membership function, represented by triangle (A, B, C), is completely specified by the parameters A, B and C. Similarly, a trapezoid membership function, represented by trapezoid (D, E, F, G), is completely specified by the parameters D, E, F and G. The use of these parameterized membership functions is convenient for handling context-sensitive definitions of fuzzy concepts. For example, different parameters can be used to define a fuzzy concept “highness” for “speed” and a different fuzzy concept “highness” for “pressure”.

A fuzzy concept itself may be treated as a variable. We call it a fuzzy variable. For example, consider “safe speed”, where “safeness” is a fuzzy concept and “speed” is a variable. We can even treat “safeness” as a fuzzy variable, and talk as “safeness” is high, or “safeness” is low to mean, respectively, that the degree of “safety” is high, or the degree of “safety” is low.



(a)

Graphical representation of Fuzzy set  $X_1$   
(Young generation decade)



(b)

Graphical representation of Fuzzy set  $X_2$   
(Mid-age generation decade)

Fig. 2.6.1. Graphical representation of fuzzy sets  $X_1$  and  $X_2$

## 2.6. Fuzzy Set Representation

Fuzzy sets have unique way of its representation. Suppose a fuzzy set, say X, is discrete and finite it can be expressed as :-

$$X = \mu_x(u_1)/u_1 + \dots + \mu_x(u_n)/u_n$$

Where, '+' is not the summation symbol but the union operator,

'/' does not denote division but a particular membership value on the universe of discourse.

For example, Consider the universe of discourse as :-

$$U = \{0, 1, 2, \dots, 9\},$$

a fuzzy set  $X_1$  as 'young generation decade' and

a fuzzy set  $X_2$  as 'mid-age generation decade'.

A possible representation can be:-

$$X_1 = \{1.0/0 + 1.0/1 + 0.85/2 + 0.7/3 + 0.5/4 + 0.3/5 + 0.15/6 + 0.0/7 + 0.0/8 + 0.0/9\}.$$

$$X_2 = \{0.0/0 + 0.0/1 + 0.5/2 + 0.8/3 + 1.0/4 + 0.7/5 + 0.3/6 + 0.0/7 + 0.0/8 + 0.0/9\}$$

The above fuzzy sets can be represented graphically as shown in figure 2.6.1.(a) and figure 2.6.1.(b).

## 2.7. Fuzzy Set Operators

There are many operators used in fuzzy sets as there are in crisp sets. Some of the fuzzy set operators are explained with the help of an example. Consider

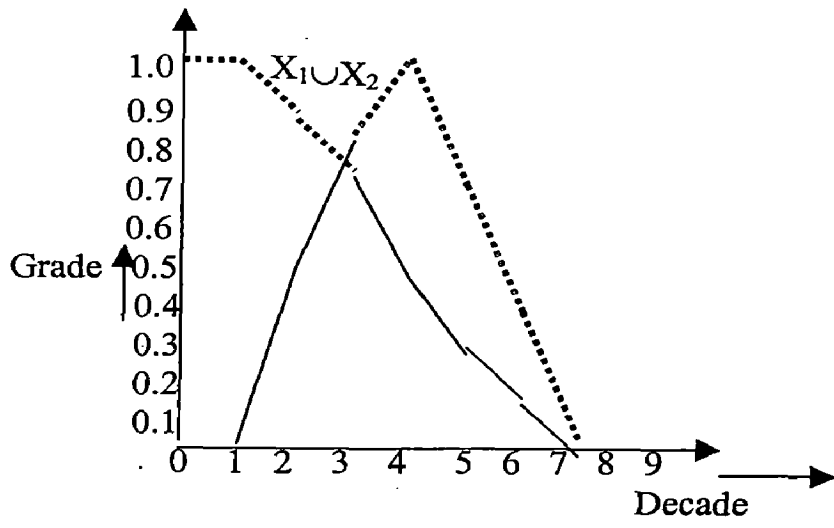
$$\mu_A = 0.8/2 + 0.6/3 + 0.2/4, \text{ and}$$

$$\mu_B = 0.8/3 + 0.2/5 \text{ as well as}$$

fuzzy sets  $X_1$  and  $X_2$  from section 2.6. above.

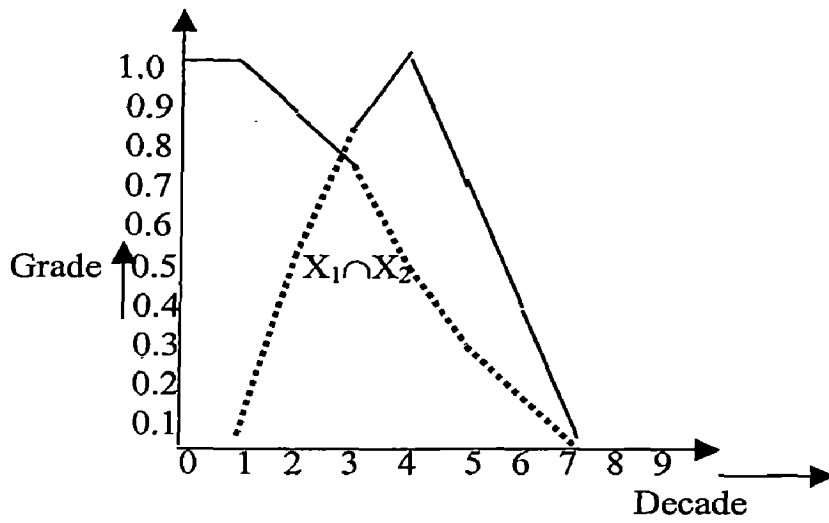
- (i) Set equality :-  $A = B$  if  $\mu_A(x) = \mu_B(x)$  for all  $x \in X$ .
- (ii) Set complement  $A'$  :-  $\mu_{A'}(x) = 1 - \mu_A(x)$  for all  $x \in X$ . This corresponds to the logic 'NOT' function.

$$\mu_{A'}(x) = 0.2/2 + 0.4/3 + 0.8/4$$



(a)

Fuzzy Union Representation



(b)

Fuzzy Intersection Representation

Fig. 2.7.1. Fuzzy union and fuzzy intersection representation

- (iii) Subset : -  $A \subseteq B$  if and only if  $\mu_A(x) \leq \mu_B(x)$  for all  $x \in X$ .
- (iv) Proper Subset :-  $A \subset B$  if  $\mu_A(x) \leq \mu_B(x)$  and  $\mu_A(x) < \mu_B(x)$  for at least one  $x \in X$ .
- (v) Set Union :-  $A \cup B$   $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$  for all  $x \in X$  where max is the join operator and means the maximum of the arguments. This corresponds to the logic 'OR' function.

$$\mu_{A \cup B}(x) = 0.8/2 + 0.8/3 + 0.2/4 + 0.2/5$$

- (vi) Set Intersection :-  $A \cap B$   $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$  for all  $x \in X$  where min is the meet operator and means the minimum of the arguments. This corresponds to the logic 'AND' function.

$$\mu_{A \cap B}(x) = 0.6/3$$

Fuzzy set union and fuzzy set intersection for the graphs shown in figure 2.6.1.(a) and figure 2.6.1.(b) can be represented graphically and is shown in figure 2.7.1.(a) and figure 2.7.1.(b) respectively.

## 2.8. General Functional Blocks For Fuzzy Logic Application[8]

Most of the applications that are developed using fuzzy logic have the following general functional blocks :-

INPUT     $\longrightarrow$     DATA ANALYSIS     $\longrightarrow$     OUTPUT

The INPUT functional block gets data at a given time. Input can be single input or multiple input. If it is a single input, a single value is obtained. If it is a multiple input, more than one value is accessed. Input data may be obtained automatically by taking some measurements of a physical system such as a machine, or by asking the user to manually enter data into a form in a database system. To measure a physical quantity,



e.g., speed of a car, we use a sensor to convert it into an electrical signal which is then amplified by signal conditioning. The conditional signal is finally digitized by a data acquisition board for a computer to process.

The DATA ANALYSIS functional block analyses input data in order to get useful output. Depending upon applications, different analysis techniques are used.

The OUTPUT functional block takes output value(s) computed by DATA ANALYSIS functional block and use them for taking actions or making decisions. Different applications mean different decisions have to be made, either automatically or manually. For example, in a control system, a control decision means to set a control signal or turn on/off a switch. In pattern recognition, a recognition decision means to classify an input pattern into a class. In decision support, a selection decision means to select items from an output list of ranked items.

## ANALYSIS OF THE PROBLEM

---

### 3.1. Analysis Of Problem Using Sample Fault Tree

The problem under consideration is to develop a soft computing tool for hazard analysis, based on the input data through various sensors. For this purpose the sensors are placed at various physical locations of a plant under consideration. They are also arranged in a hierarchical manner depending on their criticality in monitoring parameters. They essentially form a tree structure for decision making.

The task now is to compute the possibility of the failure of the top event by analyzing the fault tolerance tree.

The soft computing tool developed is the contribution in the applications of the fuzzy logic which broadly uses the various fuzzy operations like MAX, MIN etc. The membership functions are defined for the basic events of the fault tree and the membership grades are allotted to the each basic event utilizing the knowledge base provided obtained after numerous observations and experiments.

For the further analysis, a hierarchical tree is formed mentioning the various conditions of equipment and human failure that can result in an accident or an undesirable event. This is stored in a tree kind of structure and depending on the severity of the hazard/fault the membership grades are computed for each parent node by traversing the fault tree in the proper fashion.

Using the various fuzzy operations like AND, OR, its effect is calculated on the parent node and finally on to the root which depicts the effect on the whole system. Diagrammatically a sample fault tree can be represented as shown in figure 3.1 :-

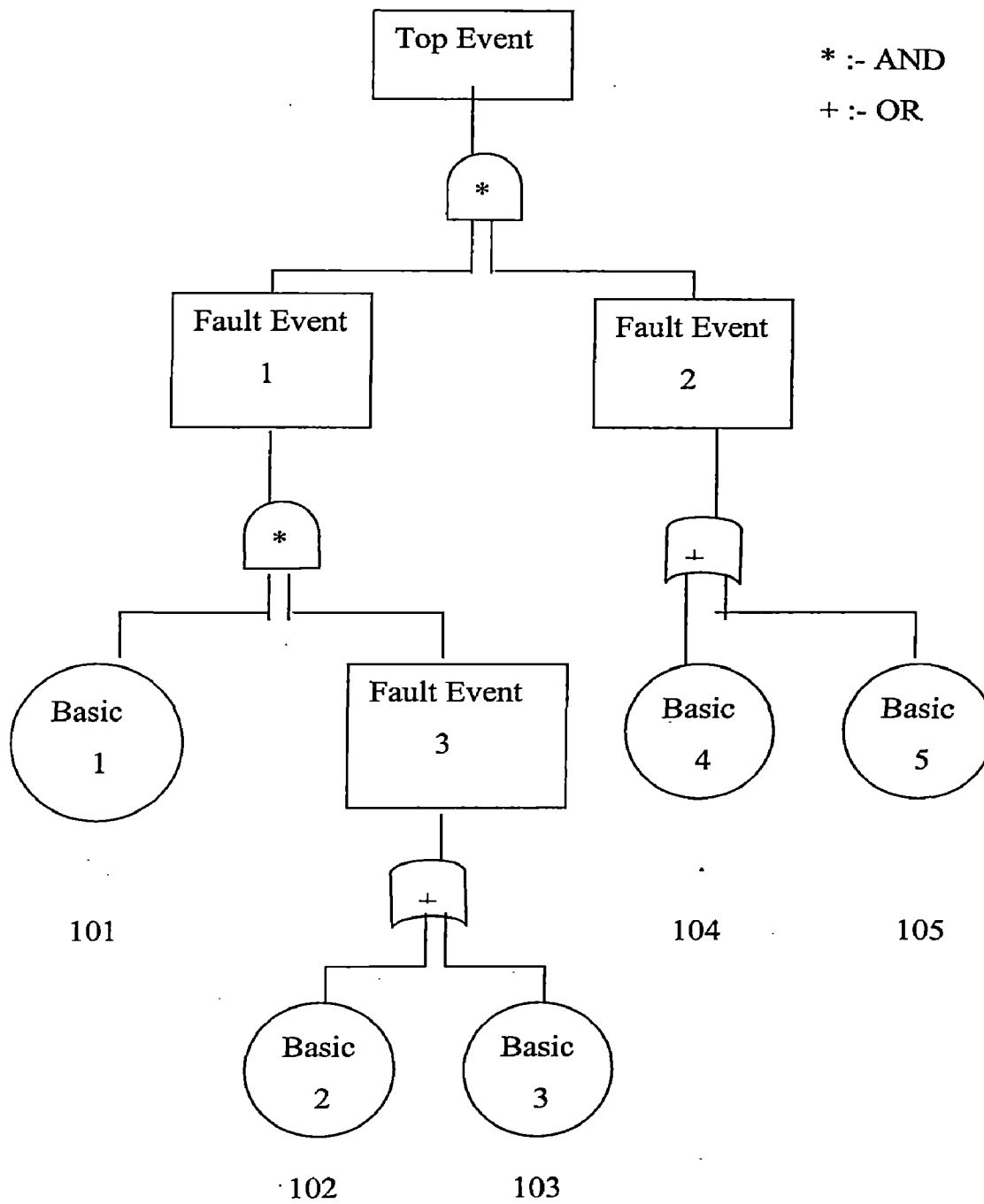


Fig.3.1.1. Sample Fault Tree using various logic symbols

A number of possible combinations known as cut-sets lead to the top event. The possible combinations or the cut-sets for the above sample fault tree are computed which are jotted as under depending upon the operation to be performed :-

- 1) 101,102,104
- 2) 101,103,104
- 3) 101,102,105
- 4) 101,103,105

Thus, for the proper functioning of tool it calculates the top event possibility for each of the possible combinations of the fault occurrence.

### **3.2. Analysis Of Ammonia Storage Tank**

This section deals with the actual data taken under consideration provided by one of the laboratories at “Defense Research & Development Organization”. One of the laboratories of “Defense Research & Development Organization” is actively involved in “Research & Development (R&D)” work in the area of hazard monitoring and development of models for their management.

Thus, the table 3.2.1 shows the Top Event, various immediate causes and the logical operation connecting immediate causes to the Top Event of Fault Tree for the Ammonia Storage Tank[9]. As mentioned above that Fault Tree Analysis starts with the consequences of an event and seeks to determine the causes while other techniques start with the failure of a component and seek the consequences. Fault Tree Analysis begins with the definition of a relevant undesirable event or the 'top event' that needs to be avoided and seeks the immediate causes of that event. As per the data provided by the laboratory, the ‘undesirable event’ or the ‘Top Event’ for which the failure possibility is to be computed is the rupture of the storage tank and the immediate causes leading to the ‘Top Event’ are :-

Serial No.	Top Event or Undesirable Event	Immediate Causes	Logical Operation
1	Rupture of Storage Tank	(a) Over Pressurization	
		(b) Over Filling	OR
		(c) Vacuum Formation	
2	Over Pressurization	(a) Pressure Safety Valve Fails	
		(b) Pressure Alarm Fails	AND
		(c) Pressure Rise	
		(d) Flare Fails	
3	Pressure Rise	(a) Flash Vessel Empty	OR
		(b) Temperature Shoot Up	
		(c) Refrigeration System Fails	AND
4	Flash Vessel Empty	(a) Level Indicator, Controller and Alarm Fails	AND
		(b) Flow Indicator and Controller Valve Fails	AND
		(c) Bypass Valve Fails	
		(d) Human Operator Fails	OR
5	Temperature Shoot Up	(a) Jetty Storage Tank Fails	
		(b) Heat Transfer From Pump Increases	OR
		(c) Cooling Water Pipe Leaks	
		(d) Cooling Water Pump-1 Fails	OR
		(e) Cooling Water Pump-2 Fails	AND

Serial No.	Top Event or Undesirable Event	Immediate Causes	Logical Operation
6	Refrigeration System Fails	(a) Compressor Fails	
		(b) Pump Fails	▶ AND
		(c) Power Source Fails	
		(d) Power Source Fails	▶ AND ▶ OR
		(e) Breaker Switch Fails	
		(f) Breaker Switch Fails	▶ AND
7	Over Filling	(a) Pump Fails	
		(b) Level Alarm Fails	
		(c) Human Operator (Overfilling) Fails	AND
		(d) Level Indicator Fails	
8	Vacuum Formation	(a) Vacuum Relief Valve Fails	
		(b) Pressure Transmitter Fails	AND
		(c) Manometer Out Of Order	

Table 3.2.1 Showing Top Event, Immediate Causes and Logical Operation for the Fault Tree of Ammonia Storage Tank



- (i) Over Pressurization
- (ii) Over Filling
- (iii) Vacuum Formation

Any of the above mentioned cause can lead to the rupture of the storage tank so all these causes are connected to the Top Event by the logical OR gate. Then, each of the immediate causes is analyzed to arrive at basic events or other intermediate events until the basic events responsible for each of the immediate causes are identified in terms of failure of process equipment or an error on the part of the operator. Fault Tree Analysis is therefore referred to as a 'backward looking' or 'top-down' approach.

Now, considering the above mentioned causes individually for the further development of the fault tree we get the list of the immediate causes and the logical operation connecting them as shown in table 3.2.1.

### **3.3. Fault Tree For Ammonia Storage Tank**

The above mentioned events summarized in table 3.2.1 are then arranged in a hierarchical manner depending on their criticality in monitoring parameters and essentially form a tree structure for decision making normally known as Fault Tree which is been shown in figure 3.3.1. The number attached with each node of the Fault tree denote its node number which is then utilized to compute the possible cut-sets of the fault tree in terms of the basic events.

### **3.4 Logic Symbols Utilized In The Fault Tree For Ammonia Storage Tank**

The following symbols are used in the construction of a fault tree displaying the various relationships :-

- (i) OR GATE :- The OR gate indicating that the output event occurs if any of the input event occurs and is denoted using '+' symbol.
- (ii) AND GATE :- The AND gate indicating that the output event occurs only when all the input event occurs and is denoted using '\*' symbol.
- (iii) BASIC EVENT :- The BASIC event representing a basic equipment fault or failure that requires no further development into more basic faults or failures which is denoted using a 'circle'.

(iv) INTERMEDIATE EVENT :- The INTERMEDIATE event representing a fault event that results from the interactions of other fault events that are developed through logic gates such as those defined above. This is denoted using a 'rectangular box'.

(v) UNDEVELOPED EVENT :- The UNDEVELOPED event representing a fault event that is not examined further because information is unavailable or because its consequence is insignificant. This is denoted in the Fault Tree using a 'rhombus'.

These logical symbols except the intermediate events are labeled using the node number to ensure that they can be differentiated.

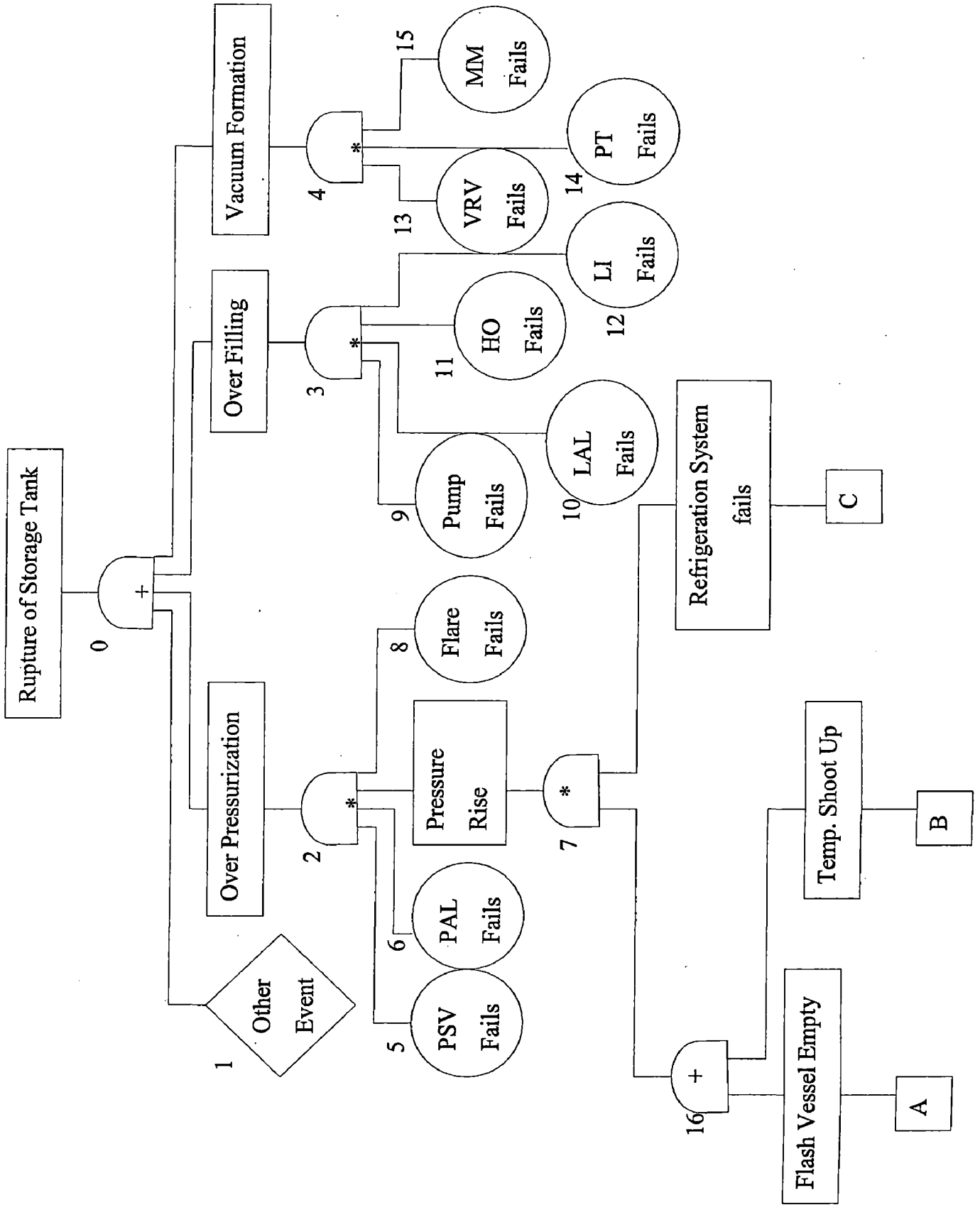
### 3.5. Failure Probabilities Of Basic Events

To collect, store and retrieve failure information effectively for devices of many types is an enormous task and requires an organization that is consistent with defined objectives. In order to carry out these studies for chemical plants a need for failure data of process control instruments[3] was recognized.

Thus, the data required for the failure probability computation was collected from the log books available at the fertilizer complex which maintains inter failure statistics of the majority of process control instruments. Failure probabilities[3] for the devices are computed using a formula :-

$$\text{Failure Probability} = \frac{\text{Number of Failures in a Year of that particular device}}{\text{Total Number of Operations in a Year of that device}}$$

The Lower Bound and Upper Bound of the failure probability is computed taking into consideration the best case and the worst case respectively. For example, consider the case for the cooling water pump, suppose in best case the cooling water pump fails for 1 time out of 20 operations in a year, then the lower bound of failure probability for cooling water pump can be computed as follows: -



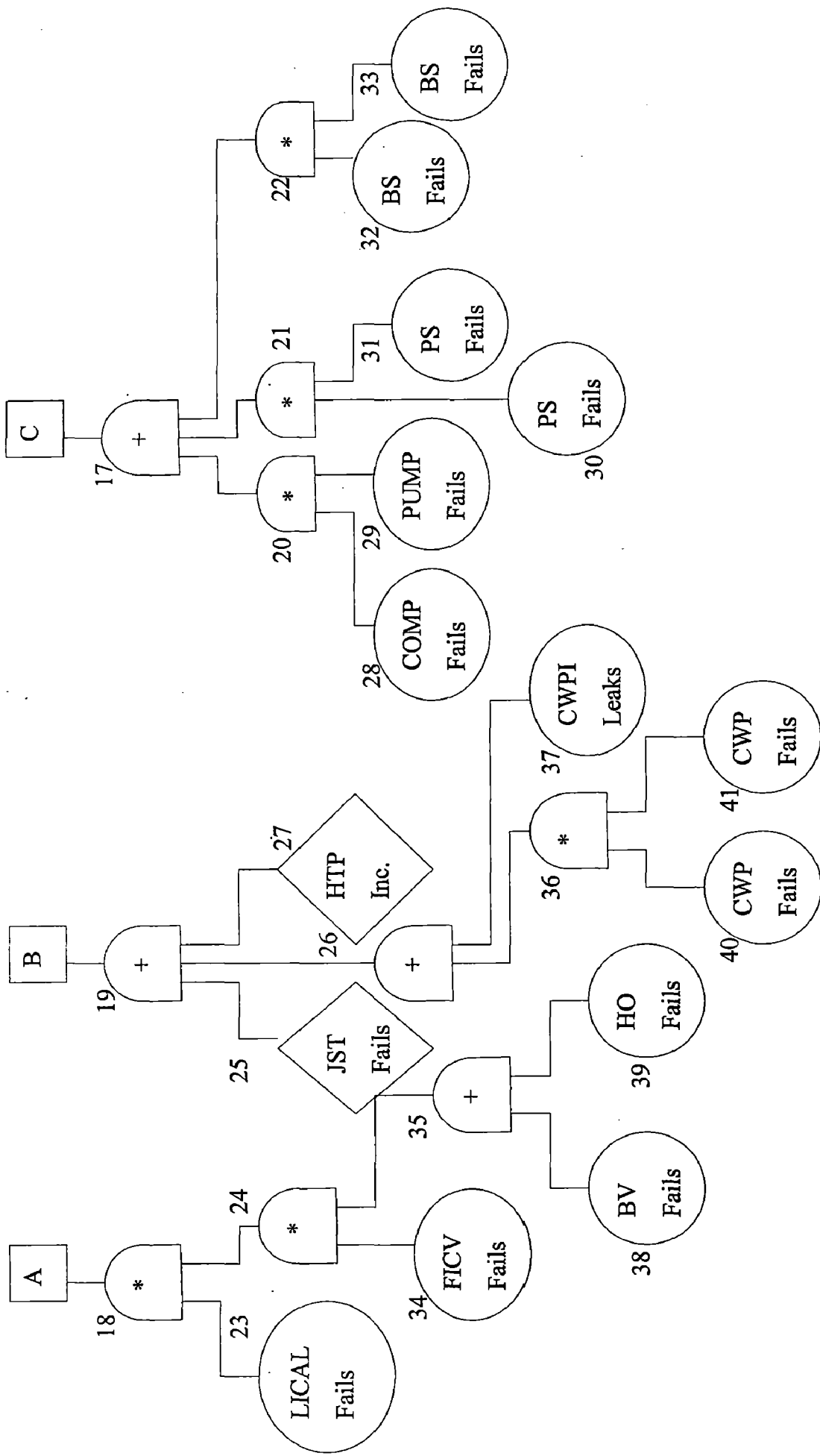


Fig. 3.3.1 Fault Tree Of Ammonia Storage Tank

Node No.	Abb.	Basic Event Name	Failure Probability	
			Lower Bound	Upper Bound
5	PSV	Pressure Safety Valve	0.05	0.525
6	PAL	Pressure Alarm	0.05	0.25
8		Flare Fails	0.0001	0.002
9		Pump	0.05	0.525
10	LAL	Level Alarm	0.15	0.25
11	HO	Human Operator (Over Filling)	0.4	0.8
12	LI	Level Indicator	0.0005	0.002
13	VRV	Vacuum Relief Valve	0.05	0.525
14	PT	Pressure Transmitter	0.35	0.72
15	MM	Manometer Out Of Order	0.0001	0.0002
23	LICAL	Level Indicator, Controller & Alarm	0.3	0.75
25	JST	Jetty Storage Tank	0.0001	0.002
27	HTP	Heat Transfer from Pump	0.0001	0.002
28	COMP	Compressor	0.89	0.95
29		Pump	0.05	0.525
30	PS	Power Source	0.005	0.015
31	PS	Power Source	0.005	0.015
32	BS	Breaker Switch	0.25	0.4
33	BS	Breaker Switch	0.25	0.4
34	FICV	Flow Indicator & Controller Valve	0.25	0.72
37	CWPI	Cooling Water Pipe	0.1	0.55
38	BV	Bypass Valve	0.0001	0.002
39	HO	Human Operator	0.0001	0.002
40	CWP	Cooling Water Pump-1	0.05	0.525
41	CWP	Cooling Water Pump-2	0.05	0.525

Table 3.5.1. Knowledge Base For Failure Probabilities Of Basic Events

Number of Failures in a year of cooling water pump in best case = 1

Total number of operations in a year of cooling water pump = 20

Lower Bound of Failure Probability for Cooling Water Pump =  $1/20$   
= 0.05

Similarly suppose, in worst case the cooling water pump fails for 10 times out of 20 operations in a year. So, the upper bound of failure probability for the cooling water pump can be computed as follows :-

Number of Failures in a year of cooling water pump in worst case = 10

Total number of operations in a year of cooling water pump = 20

Upper Bound of Failure Probability for Cooling Water Pump =  $10/20$   
= 0.5

Table 3.5.1 shows the computed values of failure probabilities of the instruments indicating upper and lower bounds.

### **3.6. Minimal Cut Sets For The Fault Tree Of Ammonia Storage Tank**

Minimal cut sets for the fault tree of Ammonia Storage Tank is computed using the previously described steps in section 2.4.3. Thus, the possible minimal cut sets ranked using the structural importance which is based on the number of the basic events that are in each minimal cut set are summarized as under. In this type of ranking, a one-event minimal cut set is more important than a two-event minimal cut set; a two-event set is more important than a three-event set; and so on. This ranking implies that one event is more likely to occur than two events, two events are more likely to occur than three events, etc.

Minimal Cut Set 1 :- (13 : 14 : 15)

Minimal Cut Set 2 :- (9 : 10 : 11 : 12)

Minimal Cut Set 3 :- (5 : 6 : 37 : 28 : 29 : 8)

Minimal Cut Set 4 :- (5 : 6 : 37 : 32 : 33 : 8)

Minimal Cut Set 5 :- (5 : 6 : 25 : 28 : 29 : 8)

- Minimal Cut Set 6 :- (5 : 6 : 27 : 28 : 29 : 8)
- Minimal Cut Set 7 :- (5 : 6 : 25 : 32 : 33 : 8)
- Minimal Cut Set 8 :- (5 : 6 : 27 : 32 : 33 : 8)
- Minimal Cut Set 9 :- (5 : 6 : 37 : 30 : 31 : 8)
- Minimal Cut Set 10 :- (5 : 6 : 25 : 30 : 31 : 8)
- Minimal Cut Set 11 :- (5 : 6 : 27 : 30 : 31 : 8)
- Minimal Cut Set 12 :- (5 : 6 : 40 : 41 : 30 : 31 : 8)
- Minimal Cut Set 13 :- (5 : 6 : 40 : 41 : 28 : 29 : 8)
- Minimal Cut Set 14 :- (5 : 6 : 40 : 41 : 32 : 33 : 8)
- Minimal Cut Set 15 :- (5 : 6 : 23 : 34 : 38 : 28 : 29 : 8)
- Minimal Cut Set 16 :- (5 : 6 : 23 : 34 : 39 : 28 : 29 : 8)
- Minimal Cut Set 17 :- (5 : 6 : 23 : 34 : 38 : 32 : 33 : 8)
- Minimal Cut Set 18 :- (5 : 6 : 23 : 34 : 39 : 32 : 33 : 8)
- Minimal Cut Set 19 :- (5 : 6 : 23 : 34 : 38 : 30 : 31 : 8)
- Minimal Cut Set 20 :- (5 : 6 : 23 : 34 : 39 : 30 : 31 : 8)

## DESIGN OF THE PROPOSED SOLUTION

---

### 4.1. Design Approach

Having analyzed the problem and identified the failure probabilities for basic events and minimal cut sets of the fault tree that are required for the software to be developed, the following solution is proposed :-

The software takes the input by the user for creating the fault tree in terms of the node number, operation performed at that node and the number of children of that particular node. As the fault tree reaches the leaf nodes or the basic events, the fault tree is created. Once the fault tree is created successfully, then the intermediate processing starts at the basic events for which the data for the failure probability in terms of the node number, lower bound and upper bound is stored in a file and the fault rate data for any particular minimal cut set is stored in a separate file in terms of the node number of the minimal cut set and the fault rate corresponding to that event. Then for each of the basic event the membership function is defined using the fuzzy logic concept and utilizing the failure probability and the fault rate data, the membership value is been computed. Thus, the intermediate processing at basic events are performed using the node number, lower bound, upper bound, fault rate, membership value computed and finally the permission to view the membership function for that particular basic event.

As the intermediate processing at the basic events is over, the fault tree is displayed showing the operation at the intermediate nodes and the membership grades at the basic event nodes. Finally, the fault tree is traversed using the bottom-up approach and the top event failure possibility is computed and is displayed in terms of the node number, operation at the node and the computed value of membership grade after applying the required fuzzy operation.

The following figure 4.1.1 presents the diagrammatic representation of the design discussed above :-



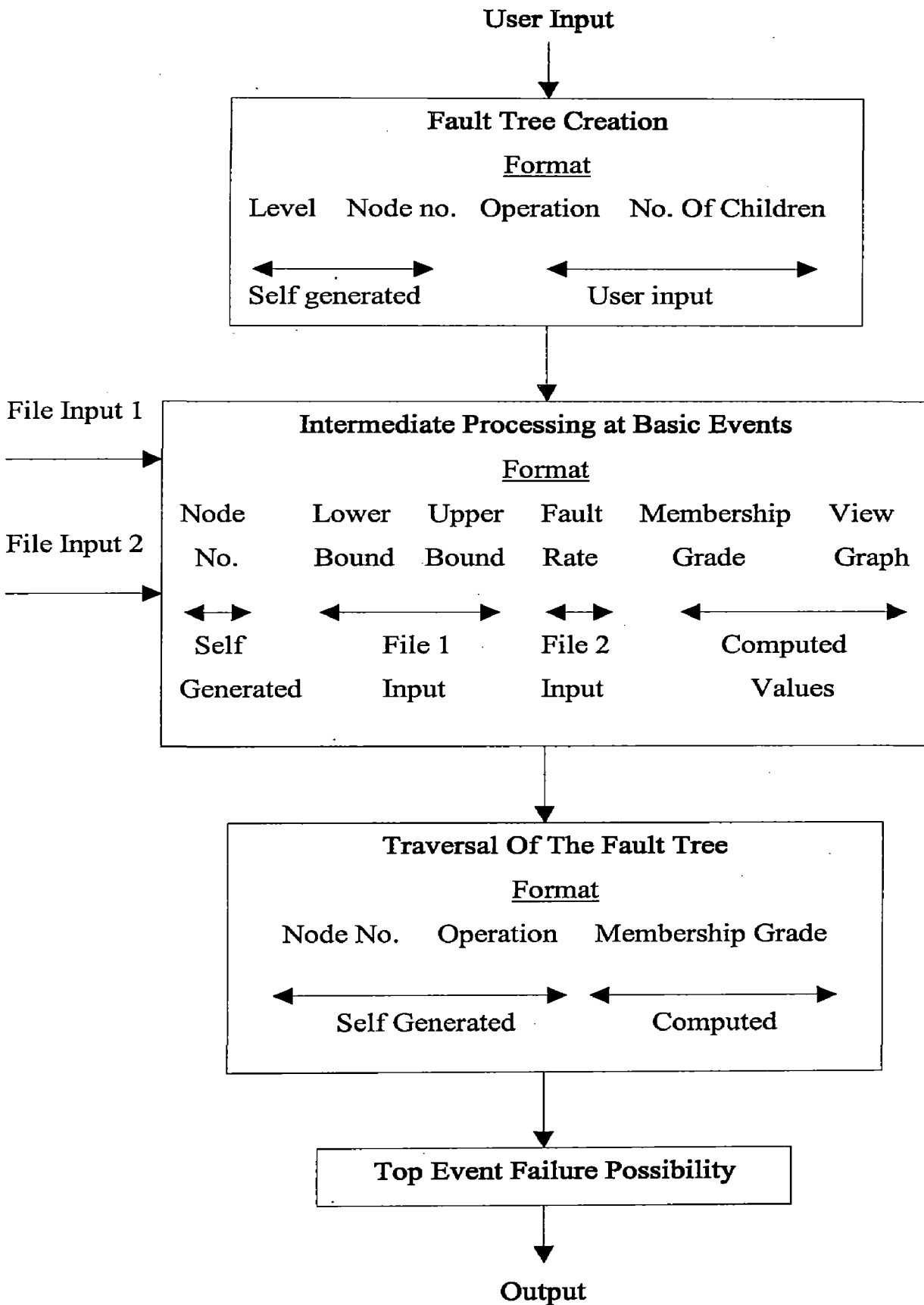


Fig. 4.1.1 Block Diagram Of The Design Proposed

## 4.2. Design Assumptions

i) Variables :-

- a) Depth of the Fault Tree
- b) Breath of the Fault Tree

ii) Operation at the Nodes :-

- a) Fuzzy AND :- Represented as '2'
- b) Fuzzy OR :- Represented as '3'
- c) Basic Event :- Represented as '0'

iii) Number of Data Files :- Two

- a) File 1 :- Stores Failure Probabilities of all Basic Events
- b) File 2 :- Stores Fault Rates of Basic Events corresponding to the Minimal Cut-Set taken into consideration

iv) Membership Function :-

- a) Trapezium Function :- The two known lower and upper failure probabilities represent the points  $a_1$  and  $b_1$  as shown in figure 4.2.1. The other two points  $a_2$  and  $b_2$  are then computed. Thus, the trapezium function can be defined as :-

$$\begin{aligned} \mu(x) &= 2x / (b_1 - a_1) + (b_1 - 3a_1) / (b_1 - a_1) \quad \text{for } (3a_1 - b_1) / 2 \leq x < a_1, \\ &= 1 \quad \text{for } a_1 \leq x \leq b_1, \\ &= 2x / (a_1 - b_1) + (a_1 - 3b_1) / (a_1 - b_1) \quad \text{for } b_1 \leq x \leq (3b_1 - a_1) / 2, \\ &= 0 \quad \text{otherwise} \end{aligned}$$

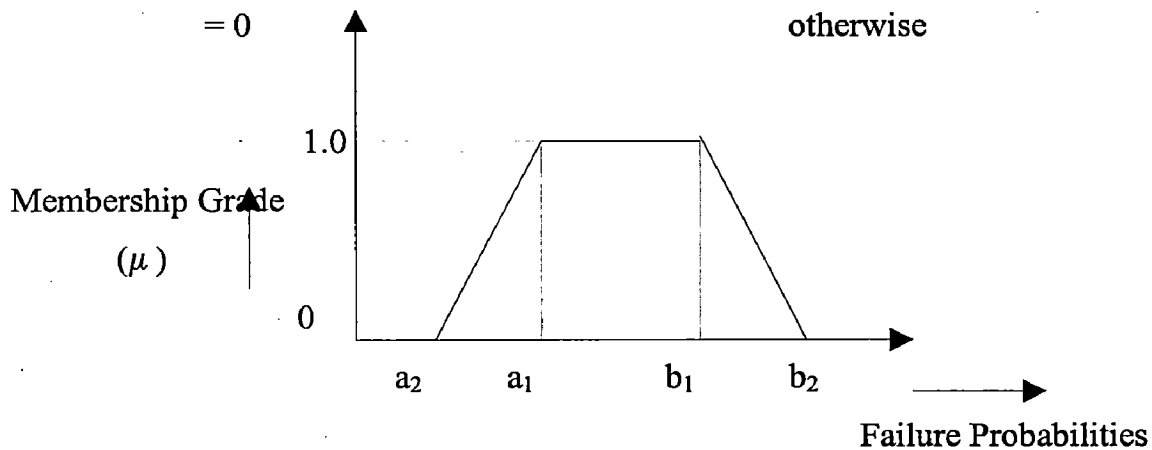


Fig. 4.2.1 Fuzzy Probability – Trapezoidal Representation

v) Fuzzy Operator :-

a) Fuzzy AND :-  $\text{MIN} ( \mu_1(x), \dots, \mu_n(x) )$

b) Fuzzy OR :-  $\text{MAX} ( \mu_1(x), \dots, \mu_n(x) )$

Where :-

$\mu(x)$  = Membership Grade corresponding to the Fault Rate considered

### 4.3. Design For File 1

i) This File stores the Failure Probabilities of Basic Events.

ii) The following is the format in which the values are stored in the File 1 :-

a) Column 1 :- Node Number

b) Column 2 :- Lower Bound

c) Column 3 :- Upper Bound

iii) Short-Cut Keys :-

a) C :- To Create File

b) V :- To View File

c) M :- To Modify or Update the File

d) Enter :- To Continue

### 4.4. Design For File 2

i) This File stores the Fault Rates of Basic Events.

ii) The following is the format in which the values are stored in the File 2 :-

a) Column 1 :- Node Number

b) Column 2 :- Fault Rate

iii) Short-Cut Keys :-

a) C :- To Create File

b) V :- To View File

c) M :- To Modify or Update the File

d) Enter :- To Continue

## IMPLEMENTATION ASPECTS

---

### 5.1. Program Outline

The program developed for the Fault Tree Analysis using Fuzzy Logic works as under :-

- i. Generates Trapezium type of membership function for each of the Basic Events of the Fault Tree under consideration.
- ii. Traverses The Fuzzy Fault Tree in a bottom-up fashion computing the intermediate results according to the AND and OR Fuzzy operations at the various interior nodes.
- iii. Finally the membership grade of the Top Event occurrence is generated which denotes the possibility of the Failure of the Top Event.

### 5.2. Implementation Approach

- i. The information on the lower and upper bounds of the failure probabilities for each basic event of the Fault Tree is the input to the program which is stored in a file besides the input for the creation of the Fault Tree.
- ii. Trapezoidal membership function for each basic event is assumed in which the two known lower and the upper bounds of the failure probability viz.  $a_1$  and  $b_1$  are assigned membership grade 1 while the points  $a_2$  and  $b_2$  which are not known are assigned a membership grade of 0.
- iii. Using some standard method, the two outer points are obtained depending upon the tolerance affordable in the problem, resulting in a trapezium membership function.
- iv. A Trapezoidal representation is obtained for each Basic Event of the Fault Tree under consideration.
- v. Fuzzy AND and Fuzzy OR operations are used to compute the intermediate results as per AND and OR gates in the Fault Tree at the various interior nodes, finally resulting into a Top Event Possibility (TEP).

### 5.3. Algorithm For Fault Tree Creation

- i. Take the number of levels in the Fault Tree as input by the user.
- ii. Now, start with the root node and ask for the number of children of the root node.
- iii. Create the number of nodes corresponding to the number of children of the root node.
- iv. Push the so created child nodes into a queue data structure.
- v. Pop the first node from front as per the definition of queue and then consider it as the root node.
- vi. Repeat from step 2 to step 5 until the queue is exhausted.
- vii. As the queue is exhausted the Fault Tree is created successfully.

Thus, the structure defined for the creation of the Fault Tree are mentioned below:-

a) Data structure defined to store information about the nodes of Fault Tree :

```
typedef struct faulttree
{
    int operation,nodenum;
    float newoperation;
    struct faulttree *leftmostchild ,*rightsibling;
} faulttree;
```

b) Structure that stores the child nodes in a linked list :-

```
typedef struct node
{
    struct faulttree *fptr;
    int nodenum;
    struct node *next;
} node;
```

c) Linked list defined to keep track of the number of nodes at each level :-

```
typedef struct numofnodesatlevel
{
    int num;
    struct numofnodesatlevel *next;
} numofnodesatlevel;
```

#### 5.4. Algorithm For Intermediate Processing At Basic Events

- i. Traced the Basic Event nodes of the Fault tree corresponding the operation '0'.
- ii. Takes the input as lower and upper bound of the failure probability from the File 1 that stores the Failure probabilities for each basic event.
- iii. Again, takes an input as Fault Rate of the Basic Event from the File 2 that stores the Fault Rates of the Basic Events corresponding to the Minimal Cut Set taken into consideration.
- iv. Trapezoidal membership function is generated in which any point within the two known lower and upper bounds of failure probabilities are assigned membership grade '1' while any point outside the two outer points which are taken according to the tolerance affordable, are assigned a membership grade of '0'.
- v. The tolerance factor is computed to the left of lower bound and right of upper bound using the formula mentioned below :-

$$\text{Tolerance Factor} = (\text{upper bound} - \text{lower bound}) / 2$$

which is then added at both ends of lower bound and upper bound respectively leading to a trapezium membership function.

- vi. Now, the input from the File 2 is then used to compute the membership grade corresponding to the fault rate of basic event of the minimal cut set taken into consideration.
- vii. The membership function using the above mentioned method is generated and user is given the right to view the membership function if he/she desires to.

The structure defined for the handling of the File are given as under :-

a) Shows the format for the File 1 as well as File 2 :-

```
typedef struct format
{
    int nodeno;
    float lowerbound,upperbound;
    float faultrate;
} format;
```

b) Linked list structure for storing the data of the file :-

```
typedef struct formatnode
{
    format data;
    struct formatnode *next;
} formatnode;
```

### **5.5. Algorithm For The Traversal Of The Fault Tree Performing Required Operation At Interior Nodes**

- i. The pointer of the root node is stored from the previous algorithm in order to have track of the various nodes for traversal.
- ii. The root node is checked for the leftmostchild, while it is not NULL the pointer of the parent node is pushed into a stack and the leftmostchild is then made the current node.
- iii. Then, the rightsibling of the current node is checked, if the rightsibling is not NULL then the rightsibling is made the current node.
- iv. Considering current node to be the root node, repeat step 2 and step 3 until both the conditions are satisfied.
- v. Once, the conditions are satisfied, pop the pointer from the stack and make its leftmostchild its childpointer

- vi. While, the childpointer is not NULL, perform the required operation on the siblings and store it in pointer of the current node.
- vii. If the operation to be performed is the Fuzzy AND operation, then take the minimum value of the membership grades so, obtained.
- viii. If the operation to be performed is the Fuzzy OR operation, then take the maximum value of the membership grades computed previously.
- ix. Repeat the procedure until the stack becomes empty.
- x. Once the stack is empty the pointer of the current node points to the root node and the value obtained is the Top event possibility (TEP).

The stack structure used for the traversal of fault tree into consideration is given as under :-

a) Stack data structure defined for the traversal of the faulttree

```
typedef struct stack
{
    faulttree *tpr;
    struct stack *next;
}stack;
```



## RESULTS AND DISCUSSION

### 6.1. User Interfaces And Output Screens

This chapter shows some of the user interface screens and the output screens taking into consideration the Fault Tree for the Ammonia storage Tank and the Minimal Cut Set 1 defined in section 3.5 composed of node numbers 13, 14, and 15 which corresponds to the vacuum relief valve failure, pressure transmitter failure and manometer out of order.

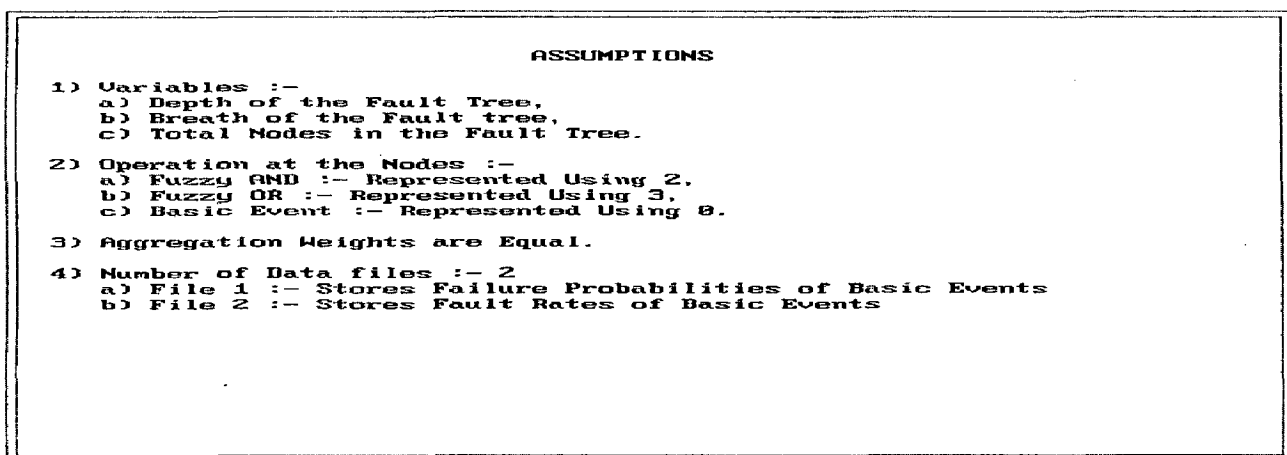


Fig. 6.1.1 Showing the assumptions made in the software

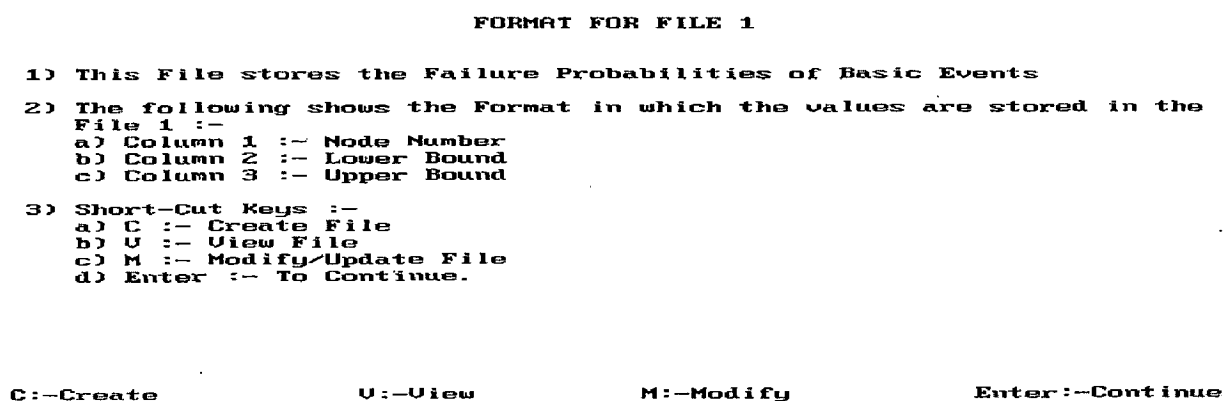


Fig. 6.1.2 Showing the format for the File 1 with the short-cut keys.

Node Number	Lower Bound	Upper Bound
6	0.050000	0.250000
8	0.000100	0.002000
9	0.050000	0.525000
10	0.150000	0.250000
11	0.400000	0.800000
12	0.000500	0.002000
13	0.050000	0.525000
14	0.350000	0.720000
15	0.000100	0.002000
23	0.300000	0.750000
25	0.000100	0.002000
27	0.000100	0.002000
28	0.890000	0.950000
29	0.050000	0.525000
30	0.005000	0.015000
31	0.005000	0.015000
32	0.250000	0.400000
33	0.250000	0.400000
34	0.250000	0.720000
37	0.100000	0.550000
38	0.000100	0.002000
39	0.000100	0.002000
40	0.050000	0.525000
41	0.050000	0.525000

Fig. 6.1.3 View of the File 1 storing the node number, lower bound and upper bounds of the basic events

```

FAILURE PROBABILITIES :- MODIFICATION

Node Number :- 5
Current Lowerbound :- 0.050000
Current Upperbound :- 0.525000

Any Modification Required ? (y/n) :-n

Node Number :- 6
Current Lowerbound :- 0.050000
Current Upperbound :- 0.250000

Any Modification Required ? (y/n) :-n

Node Number :- 8
Current Lowerbound :- 0.000100
Current Upperbound :- 0.002000

Any Modification Required ? (y/n) :-n_

```

Fig. 6.1.4 Interface for the user to modify the failure probability data and enter the new lower bound and upper bound against that node number

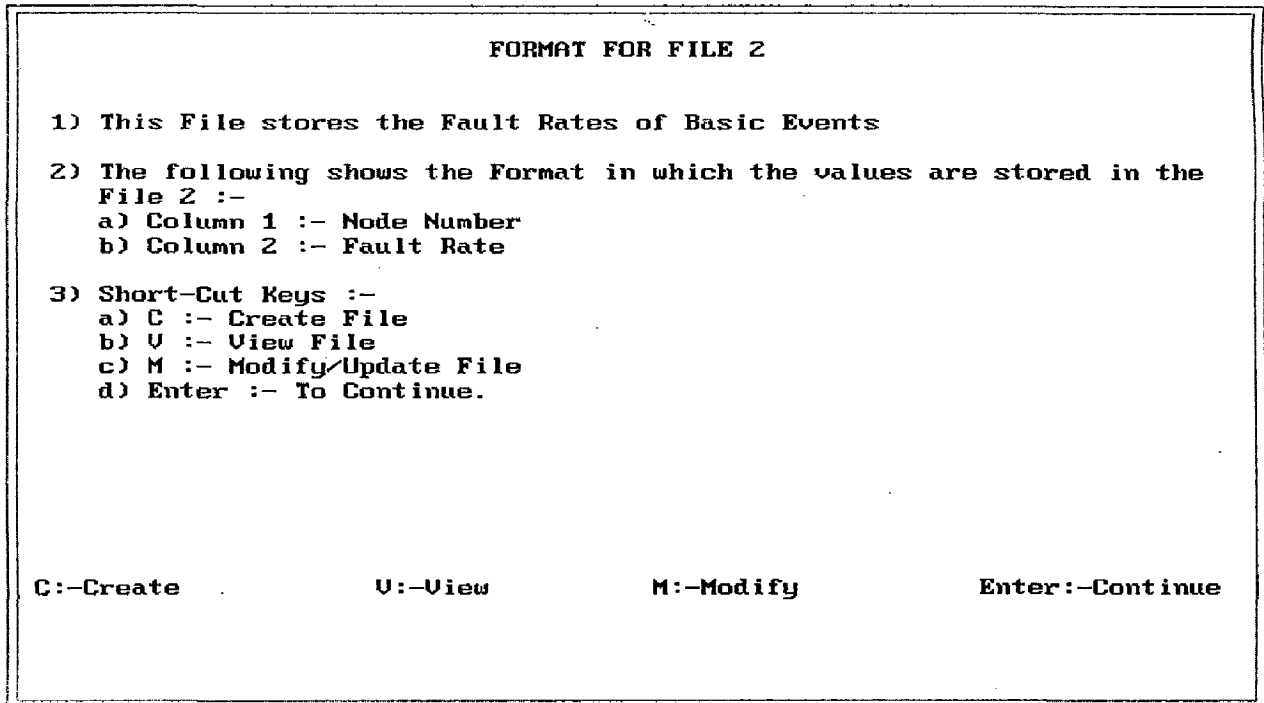


Fig. 6.1.5 Showing the format for the File 2 with the short cut keys

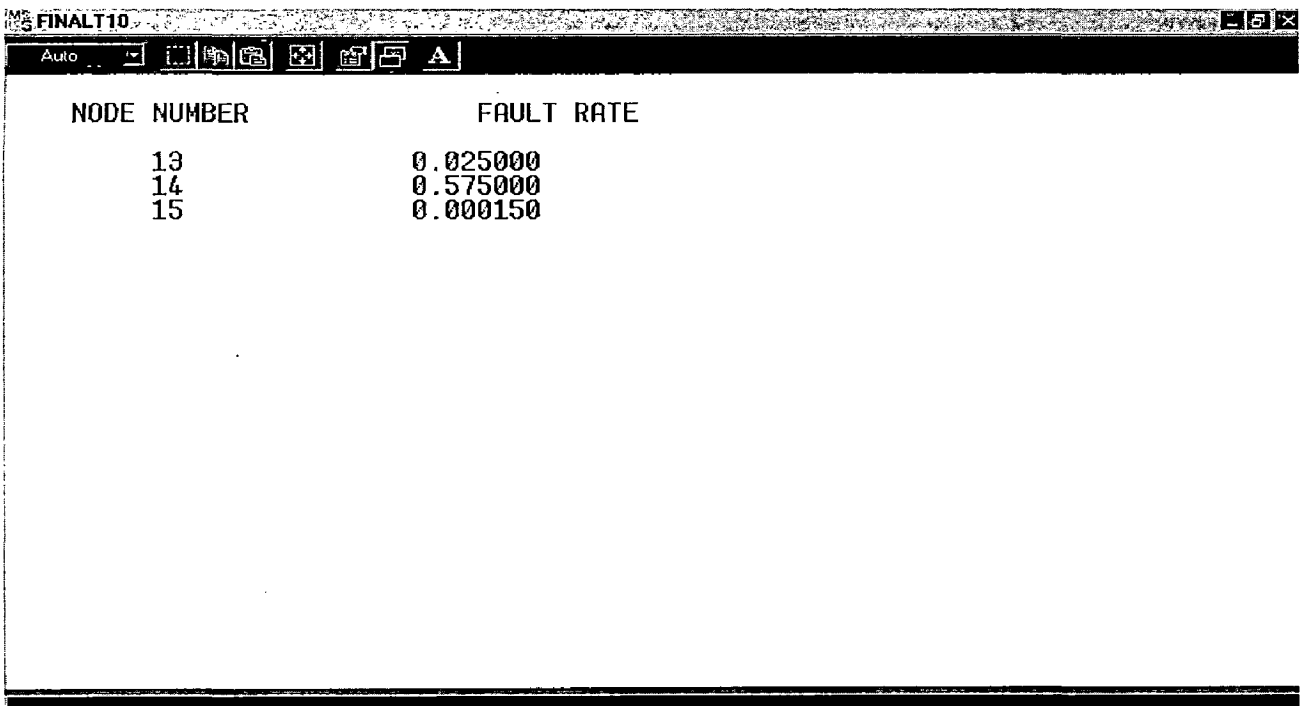


Fig. 6.1.6 View of the File 2 showing the node number and fault rate for the minimal cut set 1 taken into consideration

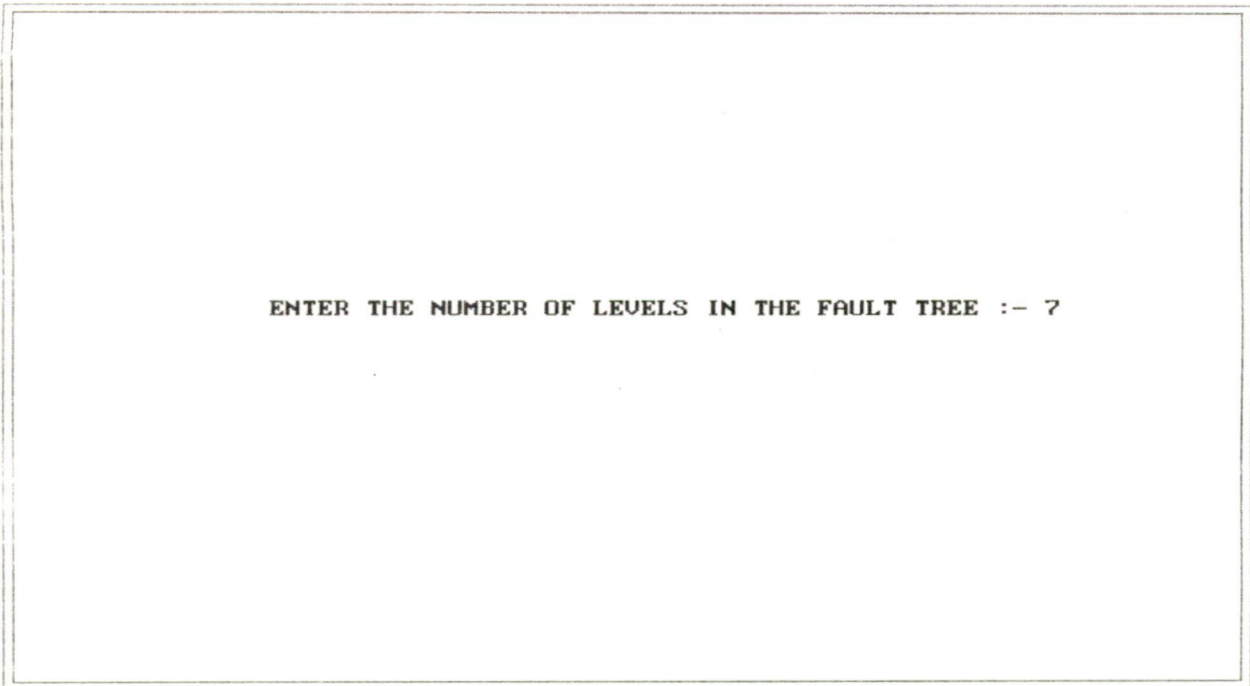
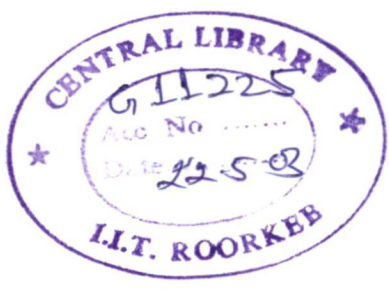


Fig. 6.1.7 Takes the number of levels in the fault tree as input from the user

LEVEL	NODE NUMBER	OPERATION	NUMBER OF CHILDREN
0	0	3	4
1	1	0	0
1	2	2	4
1	3	2	4
1	4	2	3
2	5	0	0
2	6	0	0
2	7	2	2
2	8	0	0
2	9	0	0
2	10	0	0
2	11	00	0
2	12	0	0
2	13	0	0
2	14	0	0
2	15	0	0
3	16	3	2
3	17	3	3
4	18	2	2
4	19	3	3

Fig. 6.1.8 Interface for the program of fault tree creation taking operation and number of children as input from user at each node.



NODENUMBER	LOWERBOUND	UPPERBOUND	FAULTRATE	MEMBERSHIPVALUE	VIEWGRAPH (Y/N)
1			NULL	0.000000	n
5	0.050000	0.525000	NULL	0.000000	n
6	0.050000	0.250000	NULL	0.000000	n
8	0.000100	0.002000	NULL	0.000000	n
9	0.050000	0.525000	NULL	0.000000	n
10	0.150000	0.250000	NULL	0.000000	n
11	0.400000	0.800000	NULL	0.000000	n
12	0.000500	0.002000	NULL	0.000000	n
13	0.050000	0.525000	0.025000	0.894737	
14	0.950000	0.720000	0.575000	1.000000	n
15	0.000100	0.000200	0.000150	1.000000	n
23	0.300000	0.750000	NULL	0.000000	n
25	0.000100	0.002000	NULL	0.000000	n
27	0.000100	0.002000	NULL	0.000000	n
28	0.890000	0.950000	NULL	0.000000	n
29	0.050000	0.525000	NULL	0.000000	n
30	0.005000	0.015000	NULL	0.000000	n
31	0.005000	0.015000	NULL	0.000000	n

Fig. 6.1.9 Showing the intermediate processing at basic events taking input as lower bound, upper bound and fault rate from File 1 and File 2 respectively and then computing the membership value for each basic event of minimal cut set 1 taken into consideration with the permission to view the function to the user if desired so.

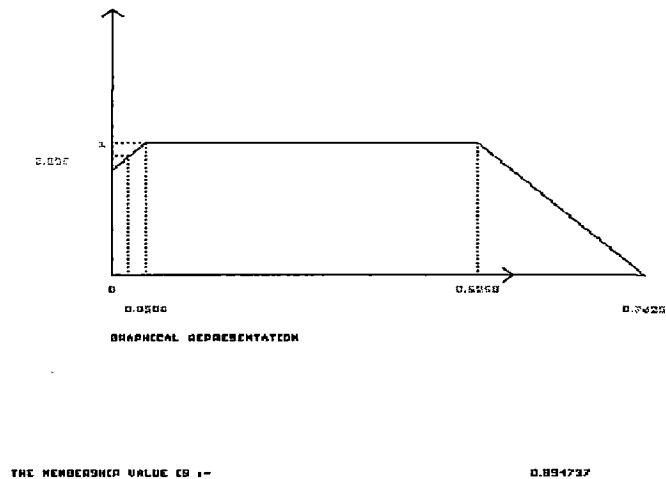


Fig. 6.1.10 Membership function of node number 13 showing the membership grade of 0.894737 at the fault rate of 0.025

NODE NUMBER	OPERATION	RESULTANT MEMBERSHIP VALUE
35	3	0.000000
24	2	0.000000
18	2	0.000000
36	2	0.000000
26	3	0.000000
19	3	0.000000
16	3	0.000000
20	2	0.000000
21	2	0.000000
22	2	0.000000
17	3	0.000000
7	2	0.000000
2	2	0.000000
3	2	0.000000
4	2	0.894737
0	3	0.894737

Fig. 6.1.11 Screen showing the traversal of fault tree carrying the required operations at the interior nodes giving final Top Event possibility as 0.894737 for the minimal cut set 1 taken into consideration.

TOP EVENT FAILURE POSSIBILITY = 0.894737

Fig. 6.1.12 Exclusively giving the Top Event Failure Possibility for the minimal cut set 1 comprising of node number 13, 14, and 15 as 0.894737

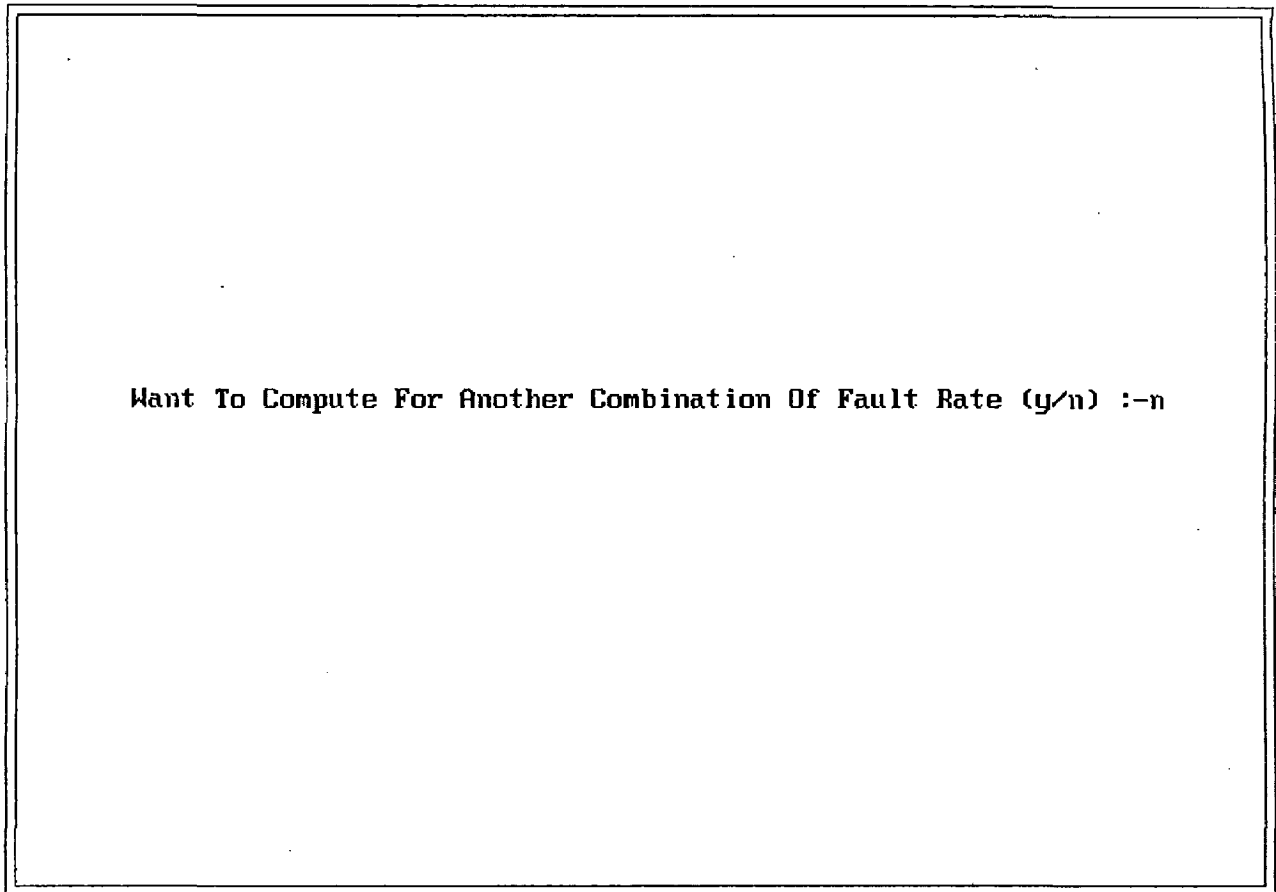


Fig. 6.1.13 Interface asking for permission from user for the computation of Top Event Failure Possibility for another minimal cut computed in section 3.5

## 6.2. Results And Discussion

The Top Event Failure Possibility for each of the minimal cut sets defined in section 3.5 are computed and are listed as under taking into consideration the structural importance of the minimal cut set ranking :-

<u>Minimal Cut Set Ranking</u>	<u>Top Event Failure Possibility</u>
Minimal Cut Set 1 :- (13 : 14 : 15)	0.894737
Minimal Cut Set 2 :- (9 : 10 : 11 : 12)	0.0
Minimal Cut Set 3 :- (5 : 6 : 37 : 28 : 29 : 8)	0.65
Minimal Cut Set 4 :- (5 : 6 : 37 : 32 : 33 : 8)	0.65
Minimal Cut Set 5 :- (5 : 6 : 25 : 28 : 29 : 8)	0.65

<u>Minimal Cut Set Ranking</u>	<u>Top Event Failure Possibility</u>
Minimal Cut Set 6 :- (5 : 6 : 27 : 28 : 29 : 8)	0.65
Minimal Cut Set 7 :- (5 : 6 : 25 : 32 : 33 : 8)	0.65
Minimal Cut Set 8 :- (5 : 6 : 27 : 32 : 33 : 8)	0.65
Minimal Cut Set 9 :- (5 : 6 : 37 : 30 : 31 : 8)	0.02
Minimal Cut Set 10 :- (5 : 6 : 25 : 30 : 31 : 8)	0.02
Minimal Cut Set 11 :- (5 : 6 : 27 : 30 : 31 : 8)	0.02
Minimal Cut Set 12 :- (5 : 6 : 40 : 41 : 30 : 31 : 8)	0.02
Minimal Cut Set 13 :- (5 : 6 : 40 : 41 : 28 : 29 : 8)	0.65
Minimal Cut Set 14 :- (5 : 6 : 40 : 41 : 32 : 33 : 8)	0.65
Minimal Cut Set 15 :- (5 : 6 : 23 : 34 : 38 : 28 : 29 : 8)	0.65
Minimal Cut Set 16 :- (5 : 6 : 23 : 34 : 39 : 28 : 29 : 8)	0.65
Minimal Cut Set 17 :- (5 : 6 : 23 : 34 : 38 : 32 : 33 : 8)	0.65
Minimal Cut Set 18 :- (5 : 6 : 23 : 34 : 39 : 32 : 33 : 8)	0.65
Minimal Cut Set 19 :- (5 : 6 : 23 : 34 : 38 : 30 : 31 : 8)	0.02
Minimal Cut Set 20 :- (5 : 6 : 23 : 34 : 39 : 30 : 31 : 8)	0.02

The results thus obtained partially follow the structural importance aspect which is based on the number of the basic events that are in each minimal cut set. In this type of ranking, a one-event minimal cut set is more important than a two-event minimal cut set; a two-event set is more important than a three-event set; and so on. This ranking implies that one event is more likely to occur than two events, two events are more likely to occur than three events, etc. Thus, the Top Event Failure Possibility obtained for various cut sets are in decreasing order except the few exceptional cases in which the type of the event plays the vital role.



## CONCLUSION

---

The case study on Fuzzy Fault Tree Analysis using the available inter failure statistics of process control instruments brings out its utility over the conventional or crisp approach. Top Event Possibility (TEP) of hazardous event (accidental release of ammonia) using Fuzzy Fault Tree Analysis (FFTA) approach has been computed for the various possible minimal cut sets obtained after analyzing the fault tree.

It could be inferred that the Top Event Possibility (TEP) for the Fault tree of ammonia storage tank varies from 0 to 0.894737 with most likely value between 0.02 to 0.65 . Thus, the available lower and upper failure rates of the process control instruments or the basic events and their logical connections using Fuzzy operation results in Fuzzy Top Event Possibility value.

The software has scope for further analysis of the Fault Tree in following areas :-

- i. The minimal cut sets thus obtained after analyzing the Fault Tree can be generated automatically with the help of an efficient computer program which will then make the job of the analyst much easier.
- ii. Defuzzification :- The Top Event Possibility thus obtained can be further defuzzified using any of the standard techniques so as to get the range or the single value of the failure probability of the Top Event. This leads to the computation of the Top Event Probability instead of Top Event Possibility.

## References:

- [1] Dr. K.V. Raghavan and Dr. A.A. Khan, "Methodologies for risk and safety assessment in chemical process industries", Commonwealth science council, 1990.
- [2] George J.Klir and Tina A. Folger, "Fuzzy Sets, Uncertainty and Information", Prentice Hall, 1988.
- [3] Deshpande A.W. and Khanna P., "Fuzzy Fault Tree Analysis : Case Studies" National Environmental Engineering Research Institute (NEERI), Nagpur
- [4] P.K.Maji, R. Biswas and A.R. Roy, "Fuzzy Soft Sets", The Journal of Fuzzy Mathematics, Vol. 9, No. 3, 2001.
- [5] Olaniya R.S., Raje D.V. and Deshpande A.W., "Approaches to Fault Tree Evaluation", National Environmental Engineering Research Institute, Nagpur
- [6] Deshpande A.W., Deshpande U.A. and Khanna P., "Fuzzy Fault Tree Analysis : A Case Study", International Conference on Fuzzy Logic and Neural Networks, July 17-22, 1992, Japan
- [7] "Guidelines for Hazard Evaluation Procedures", American Institute of Chemical Engineers.
- [8] Chin-Liang Chang, "Fuzzy-Logic-Based programming", Advances in Fuzzy systems-Application and Theory Vol. 15, World Scientific, 1997
- [9] NEERI report on "Hazard Study and Quantitative Risk Assessment of RCF Complex", Chembur Bombay, Vol. 1, New Ammonia Plant (Trombay) November 1990.

- [10] Olaniya R.S., Mathurkar H.N., Deshpande A.W., "Probabilistic Risk Assessment Of Fertilizer Plants : A Case Study", National Environmental Engineering Research Institute, Nagpur
- [11] A. Kaufman and M.M. Gupta, "Introduction to Fuzzy Arithmetic Theory and Applications", Van Nostrand Reinhold, New York, 1984
- [12] Tanaka H., Fan L.T. and Toguch K., "Fault Tree Analysis by Fuzzy Probability", IEEE Trans. On reliability, 32(5) (1983) pp 455-457
- [13] K.B. Mishra and G.G. Weber, "Use Of Fuzzy set Theory For Level-1 studies in Probabilistic Risk Assessment", Fuzzy Sets and Systems, 37, pp 139-160, 1990
- [14] K.B. Mishra and G.G. Weber, "A New Method for Fault Tree Analysis", Microelectron Reliability, Vol. 29, No. 2, pp 195-215, 1989.

