A STUDY OF FAULT TREE ANALYSIS FOR SYSTEM SAFETY AND RELIABILITY

by

WEN-SHING LEE

B.S. (Industrial Management), National Cheng-Kung University,
Tainan, Taiwan, 1973

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by :

[illegible]
Co-Major Professor

[illegible]
Co-Major Professor

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his major professors, Dr. C.L. Hwang and Dr. Doris Grosh, for their valuable guidance and creative advice in the course of this work. He also acknowledges the encouragement and helpful comments from Dr. D.S. Chung.

The author wishes to extend his deep gratitude to his parents for their constant help and inspiration, to his wife, Sharon, for her continued support and patience throughout this work.

TABLE OF CONTENTS

CHAPTER CNE

INTRODUCTION

System safety analysis is an analytical process that identifies and analyzes potential safety and reliability problems existing within a system. Reliability is a measure of the system's capability to function during the system's mission under prescribed specifications. Safety is concerned with the risk or danger posed to personnel or to the public when the system performs its task.

Basically, the problem we wish to consider is how to deal with a large coalition of components which act as a unit to provide some desired function. We are concerned not only with the likelihood of failure of this system, but also with the possible causes of failure, and the various potential failure modes. There are two formalized methods in system safety and reliability, inductive and deductive analysis.

Inductive analysis involves postulating a possible state of components and/or subsystems and determining its overall effect on the system. Among the inductive approaches, event trees have attained wide usage, especially in the recent Reactor Safety Study (99). This is a methodology whereby an initiating event or signal is traced through a system, creating new branches each time that events with multiple outcomes are encountered. A single event, then, coupled with various subsequent occurrences, may create a number of paths leading to different final results. Another inductive method of modeling a system is the reliability block diagram (112). Here the system is reduced to a schematic type form in which the specific information flow paths connect the building blocks. The purpose of this type of diagram is to permit the tracing of a signal as it proceeds from beginning to end through the system in order to calculate the system reliability. A Preliminary Hazards Analysis (PHA) is a broad, all-encompassing inductive study performed at the conceptual stages of the system design (46). Its objectives are to identify hazardous conditions inherent in a system and to determine the effect of any potential accidents. A major goal of PHA is to prevent accidents that have occurred in identical or similar systems. Failure Modes and Effect Analysis (FMEA) is also a detailed inductive analysis performed at the design stages of a system (68). It systematically analyses all contributory component failure modes and identifies the resulting effect on the system. The purpose of FMEA is to identify areas in the design of hardware where improvements are required to ensure that the system will be reliable and safe for its intended use. One final inductive method of analysis is called Markov Analysis. While Failure Modes and Effect Analysis is a single-thread inductive analysis, i.e., the effect of each component state of the system is considered independently, Markovian analysis, on the other hand, considers multiple effects and in a multi-thread inductive analysis. This process can be used for operational simulations; however the complexity of the analysis makes hand calculation impractical, and the performance of accurate simulations requires expensive equipment. Markov analysis and its application to engineering problems are discussed in (103,108)

Deductive analysis, on the other hand, takes an opposite approach. It involves postulating a possible state of the overall system and identifying those component states that may contribute to its occurrence. Fault Tree Analysis(FTA) is the only deductive approach in general use.

Fault tree analysis is a technique of reliability analysis and is generally applicable to complex dynamic systems. Fault tree analysis provides an all inclusive, verstile, mathematical tool for analyzing complex system. Its application can include a complete plant as well as any of the systems and subsystems. Fault tree analysis provides an objective basis for analyzing system design, performing trade-off studies, analyzing common cause failures, demonstrating compliance with safety requirements, and justifing changes or additions. The fault tree itself is a graphical representation of Boolean logic associated with the development of a particular system failure, called the TOP event, to basic failures called primary events. For exemple, the TOP event could be the failure of the reactor scram system to operate during an excursion with the primary events being failures of the individual scram system components.

Fault tree analysis was first developed at Bell Telephone Laboratories to aid in the safety analysis of the Minuteman missile system in 1961 (12). Furthur development was done at the Boeing company in the mid-1960's(67), and since that time it was recognized that fault tree analysis could be successfully extended from the aerospace technology to nuclear reactor reliability, safety and availability technology, and to various other commercial operation such as the chemical processing industry . The importance of fault tree analysis in the nuclear industry is pointed out in the Reactor Safety Study (99), where a full 1300 pages deals specifically with fault tree analysis.

Fault tree analysis is of major value in (50):

1) Directing the analyst to ferret out failures deductively;

2) Pointing out the aspects of the system important with respect to the failure of interest;

3) Providing a graphical aid giving visibility to those

in system management who are removed from the system design changes;

4) Providing options for qualitative or quantitative system reliability analysis;

5) Allowing the analyst to concentrate on one particular system failure at a time;

6) Providing the analyst with genuine insight into system behavior.

Fault tree analysis consists of two major steps: 1) the construction of the fault tree, and 2) its evaluation. The evaluation of the fault tree can be qualitative, quantitative, or both, depending upon the scope and extensiveness of the analysis.

The objectives of this report are :

1) to present a critical review and classification of all fault tree analysis methodologies which have been proposed with their specific purposes in past two decades;

2) to illustrate the theoretical concept and the practical formula required or the various methodologies to make fault tree a tool for decision making in system analysis;

3) to provide a tutorial format of fault tree analysis to show an insight into its formal procedure and structure;

4) to relate the theory of mathematicl reliability pertinent to fault tree analysis with an emphasis on

engineering interpretations and applications.

A state-as-the-art review of the literature related to fault tree analysis is presented in chapter 2. The literature is classified as follows :

1) fault tree construction

2) fault tree evaluation: qualitative analysis

   minimal cut sets

   common-cause failure analysis

3) fault tree evaluation: quantitative analysis

   probabilistic evaluations of fault trees

   measures of importance of events and cut sets

Chapter 3 describes in detail the current usage of fault tree construction methodologies in the U.S. and elsewhere. Those methodologies discussed are :

1) synthetic tree model (STM) - J.B. Fussell in 1973 (50)

2) fault tree synthesis for chemical processes- G.J. Powers & F.C. Tompkins in 1974 (96)

3) computer-oriented approach to fault tree construction (CAT)-S.L. Salem, G.E. Apostolakis & D.Okrent in 1976 (105)

4) computer-aided synthesis of fault trees- S.A. Lapp & G.J. Powers in 1977 (80)

5) fault tree automatic synthesis from the reliability graph- P. Camarda, F. Corsi & Trentadue in 1978 (27)

The graphical symbology and general structuring process (67) of a fault tree will also be described previous to above discussion.

Chapter 4 discusses the methodologies of qualitative fault tree analysis. This is an integral part of fault tree evaluation. In this chapter, we first discuss the methods of finding minimal cut sets. That is usually an intermediate step for most of the quantitative fault tree analysis techniques. Those methods discussed are :

1) PREP- W.E. Vesely in 1969 (119)

2) MOCUS- J.B. Fussell in 1974 (56)

3) MICSUP- P.K. Pande, M.E. Spector, P.Chatterjee in 1975 (93)

4) ELRAFT- S.N. Semanderes in 1971 (110)

5) SETS- R.B. Worrell in 1974 (134)

Other methods with regard to prime implicants (minimum cut sets) of non-coherent fault trees will also be discussed.

The usefulness of qualitative procedures is not confined to determining minimum cut sets. Indeed, it is potentially useful through the evaluation process. For example, an area of recent concern is the effect upon a system of some common event, i.e., an event that does not appear in the fault tree but directly affects some of the basic event that do appear. Interest centers around the development of techniques for ferreting out such dependence and evaluating its impact on the operation of the system. Two common-cause analysis methodologies are presented in the second part of qualitative analysis :

1) COMCON- G.R. Burdick, N.H. Marshall, J.R. Wilson in 1976 (21)

2) NEW APPROACH- D.P. Wagner, C.L. Cate, J.B. Fussell in 1977 (126)

In chapter 5, we put fault tree evaluation in a quantitative analysis perspective. Recent emphasis on the quantification of performance and safety of nuclear power generation facilities and other industries as well, has resulted in an increasing need for probabilistic engineering system analysis. This concern is divided into two categories, probabilistic evaluations of fault trees and measures of importance of events and cut sets in fault trees.

For probabilistic evaluation of fault trees, we discuss the methods of evaluation in terms of coherent structure theory (10), time-dependent methodology of fault tree evaluation by Vesely (120), recent development in Germany (25), and Monte Carlo computer program for system reliability analysis (72). We will also discuss the reliability quantification technique used in the Reactor Safety Study (99), known as the Rasmussen study, and other evaluation codes.

As to the measures of importance of events and cut sets, we present the theory of probabilistic importance and the mathematical expressions that are required to compute importance. The purpose of computing probabilistic importance is to generate a numerical ranking for assessing weaknesses in a system. Such a ranking is analogous to a sensitivity analysis (77).

A brief conclusion and recommendations for future study is given in chapter 6.

To cover all the fault tree methodologies in a discussion of this sort is practically an impossibility, since fault tree analysis techniques are still in the process of evolution and we are continually learning more.

CHAPTER TWO

FAULT TREE ANALYSIS --- A STATE-OF-THE-ART REVIEW

2.1 Introduction

Fault tree analysis was first conceived by H.A.Watson of Bell Telephone Laboratories in connection with an Air Force contract to study the Minuteman Missile launch control system (12). At the 1965 Safety Symposium, sponsored by the University of Washingston and the Boeing company, several papers were presented that expounded the virtues of fault tree analysis (67). The presentation of these papers marked the beginning of a widespread interest in the possibility of using fault tree analysis as a system safety and reliability tool in the nuclear recator industry. In the past two decades, great efforts have been made in the solution of fault trees to obtain complete reliability information about relatively complex systems. The importance of fault tree analysis in the industry application is pointed out in Reactor Safety Study (99), where a full 1300 pages deals specifically with fault tree analysis.

The fundamental concept in fault tree analysis is the decomposition of a physical system into a logic diagram, or fault tree, in which certain specified causes lead to one specified TOP event of interest. This logic diagram is constructed using the symbols found in Figure 2-1 and 2-2.

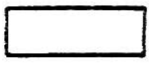
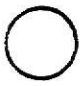
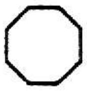
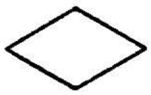
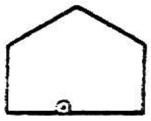
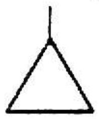
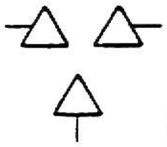
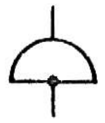
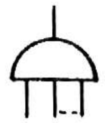
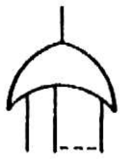
No.	Symbol	Denomination	Meaning
1		Rectangle	Variable Description
2		Circle	A primary variable belonging to an independent component.
3		Octagon	A primary variable belonging to a dependent component
4		Diamond	A non-primary variable which would require dissection in more basic variables, but that for some reasons has not been further dissected.
5		House	A variable whose sample space contains only one member, that is a variable which is constant and always takes either the value 1 or 0. Note: this symbol is used only as input to an AND gate.
6		Transfer IN	A connecting or transfer symbol indicating a variable entering the fault tree.
7		Transfer OUT	A connecting or transfer symbol indicating a variable going out from the fault tree.

Fig. 2-1. Table of variables(23)

No.	Symbol	Denomination	Boolean Notation	Output/Inputs Relationship	Rules for the Generation of the Truth Table
1		NOT	$B=\bar{A}$	$B=1-A$	Output takes the value 1 if predecessor takes the value 0 and vice versa.
2		AND	$B=\bigwedge_{i=1}^{n} A_i$	$B=\min(A_1;A_2\cdots;A_n)$	Output takes the value 1 if and only if all predecessors take the value 1, and the value 0 if at least one of the predecessors takes the value 0.
3		OR	$B=\bigvee_{i=1}^{n} A_i$	$B=\max(A_1;A_2\cdots;A_n)$	Output takes the value 1 if at least one of the predecessors takes the value 1, and the value 0 if and only if all predecessors take the value 0.

Note: A marked point at the input of the input of a gate means that the input variable is negated before entering the gate.

Fig. 2-2. Table of Basic Gates. (23)

The two basic units involved are the AND and OR gate. Another, less often used element is the NOT gate. TOP events can be taken from a preliminary hazard analysis, or by intuition. These events are usually undesired system states that can occur as a result of subsystem functional faults. A sample fault tree, along with its corresponding circuit diagram is shown in Figure 2-3.

The following four steps generally can be presented in a fault tree analysis (58).

1) system definition
2) fault tree construction
3) qualitative evaluation
4) quantitative evaluation

A state-of-the-art review of the literature related to fault tree analysis of each of these steps is discussed in this chapter. Table 2.1 gives the general classification of related references for fault tree analysis. Tables 2.2 and 2.3 present the references for fault tree construction and evaluation with regard to various methodologies. Table 2.4 summarizes the available computer codes for all the phases of fault tree analysis from construction to evaluation. Table 2.5 shows the applications of fault tree methodologies in real world problems.

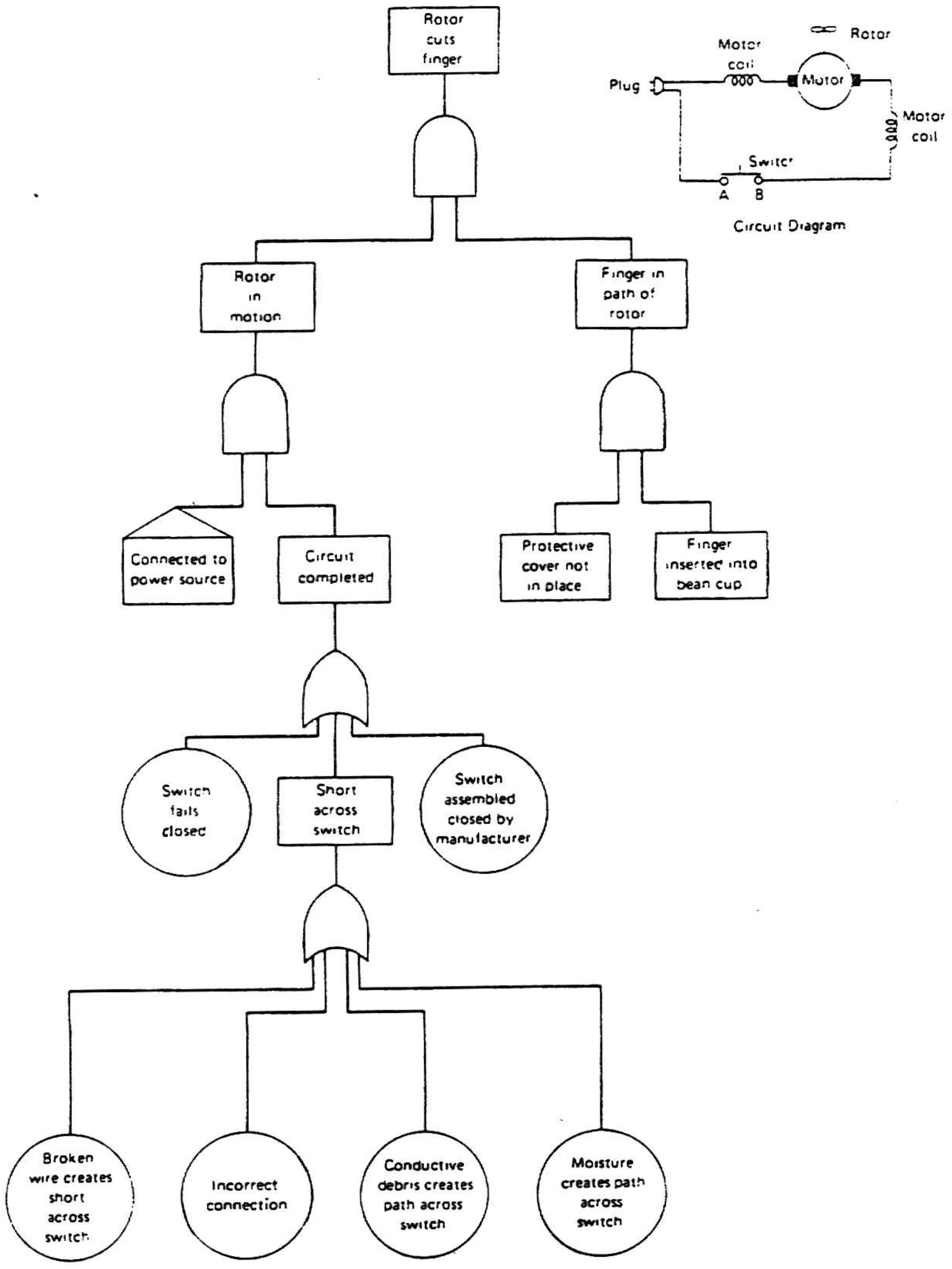

Rotor cuts finger
Rotor in motion
Finger in path of rotor
Connected to power source
Circuit completed
Protective cover not in place
Finger inserted into bean cup
Switch fails closed
Short across switch
Switch assembled closed by manufacturer
Broken wire creates short across switch
Incorrect connection
Conductive debris creates path across switch
Moisture creates path across switch
Rotor
Motor coil
Plug
Motor
Motor coil
Switch
A
B
Circuit Diagram

Fig. 2.3 Sample fault tree and its circuit diagram (69)

Table 2.1 General Classification

Class	References
Fault tree introduction	6, 7, 12, 19, 29, 43, 50, 58 68, 69, 75, 76, 79, 86, 100, 109, 138
Fault tree construction	1, 4, 51, 52, 54, 59, 67, 70 76, 78, 80, 81, 82, 83, 91, 95, 96, 97, 104, 105, 106, 107, 113, 115, 116, 117, 136 137
Qualitative evaluation	
Min. cut sets	2, 14, 23, 24, 32, 56, 60, 61, 62, 63, 71, 74, 84, 85, 90, 93, 98, 110, 118, 125, 127, 128, 129, 131, 132 133, 134, 135
Common_cause analysis	2, 20, 21, 30, 44, 87, 124, 126
Quantitative evaluation	
Probabilistic evaluation	3, 5, 8, 10, 13, 15, 17, 18, 22, 23, 25, 26, 28, 31, 33, 34, 35, 36, 37 38, 39, 40, 45, 47, 48, 53, 55, 57, 64, 72, 73, 76, 88, 89, 92, 101, 102, 119, 120, 121, 123, 125 130

Table 2.1 Cont'd

Measures of importance	9, 16, 53, 76, 77, 92
Fault tree application	12, 25, 26, 27, 41, 42, 43, 54, 57, 66, 69, 75, 76, 79, 80, 81, 96, 97, 99, 121, 122, 123

Table 2.2 Fault Tree Construction

Construction Technique	References
Haasll's structuring process	67
Fussell's STM (DRAFT code)	51, 52, 54
Powers and Tompkin's method	95, 96, 97
Salem et al.'s CAT	104, 105, 106, 107, 136
Lapp and Powers' computer-aid syn.	78, 80, 81, 82, 83, 137
Comarda et al.'s efficient algor.	1, 27
Taylor's CCD	91, 113, 115, 116, 117

Table 2.3 Fault Tree Evaluation

Technique	References
Qualitative evaluation : MCS	
1) Monte Carlo simulation	
PREP (FATE option)	125
2) Deterministic Method	
PREP (COMBO option)	125
MOCUS	56,60
ALLCUTS	118
MICSUP	32, 93
ELRAFT	110
FAUTRAN	131
SETS	132, 133
FATRAM	98
DICOMIC	62
Kumanoto & Henley	74, 84
Nakashima & Hattori	90
GO	129
Qualitative evaluation : common cause failure analysis	
COMCAN	20. 21
BACFIRE	30
Wagner et al's new approach	126

Table 2.3 Cont'd

Quantitative evaluation : prob. evaluation of fault tree	
Coherent structure theory	3, 6, 8, 10, 17, 33, 34, 35, 47, 47, 49
Monte Carlo simulation	
RELY 4	38, 72, 88
SAFTE	64
SAMPLE_WASH 1400	99
REDIS	73
CROSETTL'S code	38, 39, 40
Analytical method	
KITT	119, 120, 121, 125
Caldarola & Wickenhauser	22, 23, 24, 25, 26
Other methods	
ARMM	89
GO	65
NOTED	130
WAM_BAM	45
PARTEC	18
SALP	5
Diagraph Technique	31
Bit Manipulation	127
Quantitative evaluation : measures of importance	9, 16, 53, 76, 77, 92

Table 2.4 Available Computer Codes for Fault Tree Analysis

Computer Code	References
Fault tree construction	
DRAFT	51, 52, 54
CAT	104, 105, 106, 107, 136
L & P	78, 80, 81, 82, 83, 137
Taylor's CCD	91, 113, 114, 116, 117
Qualitative evaluation	
1) Minimal cut set	
PREP	125
MOCUS	56, 60
ALLCUTS	118
MICSUP	32, 93
ELRAFT	110
FAUTRAN	131
SETS	132, 133
FATRAM	98
DICOMIC	62
BAM_CUTS	45
BUP_CUTS	90
2) Common cause analysis	
COMCAN	20, 21
BACFIRE	30

Table 2.4 Cont'd

Quantitative evaluation	
1) Probabilistic evaluation	
RELY 4	72
SAFTE	64
SAMPLE	99
REDIS	73
CROSETTI'S code	38, 39, 40
KITT	119, 120, 121, 125
Caldarola & Wickenhauser	22, 23, 24, 25, 26
PL_MOD	92
ARMM	89
GO	65
NOTED	130
WAM_BAM	45
PARTEC	18
SALP	5
2) Measure of importance	
IMPORTANCE	76

Table 2.5 Applications of Fault Tree Analysis in Real World Problems

System analyzed by fault tree analysis	References
Aerospace safety study	12
Electrical system	27, 54
Chemical processing system	80, 81, 96, 97
Nuclear reactor safety study	25, 26, 42,
	57, 66, 79, 99
	121, 122, 123
Product safety	41, 43, 69
Decision making in system analysis	75, 76

2.2 System Definition

System definition is often the most difficult task associated with fault tree analysis. In order to perform a meaningful fault tree analysis, two basic types of system information are needed (106) :

1) component operating and failure modes : A description of how the output states of each component are influenced by the input states and internal operational modes of that component. Failure Modes and Effects Analysis can be applied here.

2) system chart : A description of how the various components of the system are interconnected, that is, how the inputs and outputs of each component are connected to other components. Of primary importance is a functional layout diagram of the system of interest showing all functional interconnections and identifyintg each system component. An example might be a detailed schematic diagram or a process flow chart.

A further step in the system description , then, is to establish the system boundary conditions (52). These define the situation for which the fault tree is to be drawn. A most important system boundary condition is the TOP event which is defined as the major system failure of interest. Initial conditions are then system boundary events that define the component configurations for which the top event is applicable.

System boundary conditions also include any fault event declared to exist or to be not-allowed for the duration of the fault tree construction. These events are called existing system boundary conditions or not-allowed system boundary conditions. An existing system boundary condition is treated as certain to occur and a not-allowed system boundary condition is treated as an event with no possibility of occurring. Finally, in certain cases, partial development of the TOP event, called the Tree Top, is also required as a system boundary condition. If the tree top system boundary condition is required, it is not considered as part of the fault tree construction process because it is obtained by inductive means.

2.3 Fault tree construction

Fault tree construction is generally the most tedious and most critical aspect of fault tree analysis. Since the computer codes of fault tree evaluation allow a rapid solution of fairly large fault trees, the bulk of time is currently spent in the actual construction of the fault tree. For instance, the construction of a recent fault tree for a nuclear safety system required over 25 man-years (99). Thus, several researchers have attempted to automate this phase of the analysis. Haasl (67) has described some general concepts for construction methodology . Fusell (51) presented a computer code, DRAFT, for electrical systems . Powers and Tompkins (97) developed a safety simulation language for chemical processes . Salem, Apostolakes, and Okrent (104) devised a new methodology for the computer-constructrion of fault trees (CAT). Lapp and Powers (80) proposed a computer aided synthesis of fault trees. Camarda, Corsi, and Trentadue (27) originated an efficient simple algorithm for fault tree automatic synthesis from the reliability graph. Finally, Taylor (116) initiated a failure mode analysis of control systems which is later used as a basis for cause-consequence analysis (91).

David Haasl formalized the thought process involved in the constrution of the fault tree. He devised a structuring process that establishes rules to determine the type of gate to use and imputs to the gate (67).

In the DRAFT code, Fussell (51) has employed a methodology called Synthetic Tree Model (STM), which is a systematic method for constructing fault trees either by hand, or via the DRAFT program. The basic idea behind STM is the modeling of each device in the system by a failure transfer function. Then, by tracing through the schematic, these transfer functions for various components are combined and edited to form the final fault tree.

Powers and Tompkins' method (96) is an automated fault tree construction method for chemical systems. Here the investigators are dealing not only with a system of mechnical components, but one also involving complex chemical interactions. Their approach, as in the other methodologies, is first to break down a system into constituent blocks, and define their operations via unit models, finally coupling these together systematically to form the tree.

Salem, Apostolakis and Okrent's CAT code (105) presented a general, computer-implemented approach for modeling nuclear and other complex systems involving mechanical, electrical, hydraulic and human interactions and common cause effects as well. This is based on the use of decision tables as component modeling (107), and a step by step editing procedure of coupling components and tracing through the system in order to construct the fault tree. Given sufficient information concerning the specific system to be analyzed, and a library of component models, this approach allows the rapid, systematic construction of a fault tree in standard form.

Lapp and Powers' Fault Tree Synthesis program (FTS) (80) is accomplished by first generating a diagraph (directed graph) for system representation, and then using fault tree synthesis algorithm to deduce the fault tree from the diagraph model of the system being analyzed. The diagaph for a complete system is constructed from a flow sheet of the system and input/ output models for the components in the system. The fault tree synthesis algorithm is performed by selecting the mode representing the top event and usin the appropriate operator (depending on whether negative feedback or feedforward loops pass through the current node) to logically connect the local causes until all the causes are developed to their primary events

Camarda, Corsi and Trentadue's simple efficient algorithm (27) for the automatic construction of fault trees concerning complex 2-state, non series-parallel system begins with the probabilistic graph of the system, i.e., a graph which shows all possible ways of correct system operation. The minimal tie sets of the reliability graph are then obtained and readily transformed to minimal cut sets. This results in a suitable form of fault tree that can be used in any available method of numerical evaluation .

Taylor's method (116) uses algebraic models for components with qualifiers to indicate which equations describe the operation or failure of the component. These qualified equations are then written for each component and the resulting collection forms the system model. This model can then be used to determine the consequences of any deviation in the input variables. The method involves more cause-consequence than fault tree analysis.

2.4 Qualitative Evaluation

Qualitative fault tree analysis consists of determining the minimal cut sets/minimal path sets and the common-cause failures. A minimal cut set is a set of basic events whose occurrence causes the top event to occur; it can not be reduced and still insure occurrence of the top event. A common-cause failure is a failure caused by a common event that does not appear in the fault tree but directly affect some of the basic events that do appear.

The minimal cut sets of simple fault trees may be obtained by inspection. However, large, complex fault trees involve hundreds of gates and events, and result in thousands of cut sets. An inspection procedure to determine the cut sets (and path sets) is not practical. In such cases systematic computer-aided procedures are necessary to reduce fault trees. Two major approaches used for computer reduction are Monte Carlo simulation and deterministic methods.

The Monte Carlo simulation procedure for finding minimal cut sets first assigns a time to failure for each component based upon an exponential failure distribution. These times to failure are choosen by first generating a uniformly distributed random number lying between 0 and 1, n for each component, and then finding t, the time to failure, from the following relation :

$$n_i = \frac{1 - e^{\lambda_i t_i}}{1 - e^{\lambda_i T}}$$

where T is a mission length which guarantees that t < T. In one Monte Carlo run, a time t is generated for each component, then the components are failed, one at a time in order of increasing t , until the top event is produced. This produces a cut set which is then reduced to a minimal cut set (125).

The basic idea behind the deterministic approach is direct expansion or reduction of the top event of a fault tree in terms of the constituent basic events using Boolean algebra. One of the earliest computer program using the deterministic method is the PREP program developed by Vesely and Narum (125). The program uses a direct, combination test algorithm by deterministically failing each component individually, then each pair, each group of three components, etc., but this rapidly becomes a huge problem for large numbers of components. Thus, Fussell and Vesely (60) developed an alternate algorithm which does not require the combination testing. It is based on the fact that an AND gate always increases the number of events in the cut set, and an OR gate always increases the numbe of cut sets. Fussell, Henry and Marshall (56) used this algorithm in their fault tree analysis program, MOCUS. This is a top-down oriented allgorithm and is designed to accept only AND

and OR gates. MICSUP (93) is, on the other hand, a bottom up algorithm. It starts with the lowest level gates that have basic events as input only, finds the minimal cut sets to these gates and then successively subsitutes these cut sets to these gates. The procedure is repeated until the minimal cut sets to the top event are found.

Semanderes (110), in the computer code ELRAFT , introduced the concept of prime number representation of basic events for reduction of fault trees. This concept is useful in storing the cut sets and eliminating the superset. The idea is to assign a unique prime number starting from 2 for each basic event. The cut sets are then represented by the product of the prime numbers of the basic events in the set. Each cut set is therefore stored as a unique number. Since the product can be factorized uniquely to the component prime numbers, the components of the cut sets are readily obtained.

While the above methods of finding minimal cut sets are applicable for coherent fault trees, i.e., the fault trees that are restricted to contain AND and OR gates only, the SETS computer code (132) finds the prime implicants to a non-coherent fault tree. The prime implicants are like minimal cut sets except that they may contain complemented basic events. Kumamoto and Henley (74) also developed a top-down algorithm for obtaining prime implicant sets of noncoherent fault trees.

Strictly, common cause failure is any occurrence or condition that results in multiple component failures. A significant common cause event is, then, a cause of secondary failure that is common to all basic events in one or more hardware minimal cut sets. The effect of common-cause events could be studied by altering the fault tree so that these events are explicitly represented in the tree, and then finding the minimal cut sets for the tree. Done in this way, the number of minimal cut sets would tend to increase substantially. Worse still, many of the minimal cut sets would be mixtures of basic events and common cause events that would be somewhat difficult to interpret (137). Therefore, two methodologies for locating common cause analysis have been developed.

The first one, called COMCAN (21), was developed by the Aerojet Nuclear Company for the U.S. Energy Research and Development Administration. The program requires as input whatever minimal cut sets have been selected from the fault tree and the generic cause susceptibility for each basic event in each category. The algorithm then searches for those minimal cut sets that are comprised of basic events that are all susceptible to the same generic cause, and this search is repeated for each category. Nevertheless, for complex systems, determining the list of minimal cut sets becomes a difficult and often an impossible task. Computer time and storage capacity become prohibitive. The above method that requires all the list of minimal cut sets as input is restricted. To overcome this difficulty a new procedure, proposed by Wagner et al., (126), was developed. This approach, without examining all the minimal cut sets, would locate minimal cut sets of any order which could fail due to common cause failure.

For quantitative analysis of common cause failures, W.E. Vesely (124) developed a statistical estimation technique by specializing the multivariate exponential Marshall-Olkin model (87).

2.5 Quantitative Evaluation

The fault tree is useful not only as a tool to visualize the various combinations of events in a system, but also as a convenient format for the probability evaluation of the undesired event and all the combinations of events that are most likely to cause the top event. This probability information can then be used to rank the various paths and to calculate the overall probability for the top event. Since the introduction of fault tree analysis, the area receiving the most research and deveopment effort has been the quantitative evaluation of fault trees.

The first step in the quantitative evaluation of a fault tree is to find the structural representation of the top event in terms of the basic events. Finding the minimal cut sets is one way of accomplishing this step. If the rate of occurrence and fault duration for all basic events are known and the statistical dependency of each basic event is known (or assumed), then the mathematical expectation or probability of the top event can be determined.

The Boolean representation of fault trees provides the link with coherent structure theory (10). When system success, rather than failure, is stressed, the coherent structure theory is the foundation of reliability theory. A coherent structure, in the context of fault trees, is nondecreasing in each basic event, i.e., that the occurrence of a basic event cannot cause a system transition from a failed state to an unfailed state. This implies that we do not allow complemented events; that is, no NOT type function are existed. A coherent structure then contains, by definition, all relevant basic events, i.e., the occurrence of each basic event must contribute in some way to the occurrence of the top event (10).

The minimal cut sets/path sets of a coherent fault tree can be obtained by using one of the available codes (56,93,125). The system unavailability can then be calculated either a) exactly by using the minimal cut sets/path sets to write the structure function of the tree as a sum of products of basic events provided that the basic events are not replicated in cut sets and all basic events are statistically independent, or b) approximately by using one of the following standard methods (10):

1) the inclusion-exclusion method of finding successive upper and lower bounds to the probability of the top event in terms of the minimal cut sets.

2) the minimal cut upper bound and min path lower bound given fault tree of statistically independent basic events.

3) the min-max bound for statistically dependent basic events, i.e., associated basic events.

Improved bounds for the above methods can sometimes be obtained by using modular decomposition (10,34).

The analysis of noncoherent fault trees proceeds in a similar way. Instead of finding the minimal cut sets in coherent structures, we identify the prime implicants in noncoherent fault trees. The prime implicants are like minimal cut sets except that they may contain complemented basic events Algorithms for obtaining the prime implicants are discussed in (74,132) . It is proved that all the methods applicable to coherent fault trees, except the minimal cut (path) bounds, can be extended to noncoherent fault trees (35).

By the late 1960's sophicated computer programs were available to obtain probabilistic information about the top event from probabilistic information about the basic events by using the Monte Carlo method. Such programs have been described by P. Crosetti (38) and H.E. Kongsoe (72).

A typical Monte Carlo simulation program involves the following steps (39) :

1) assignment of failure data to input fault events within the tree, and if desired, repair data.

2) representing the fault tree on a computer to provide quantitative results for performance of the overall system, subsystems and the basic input events.

3) listing of the failure that leads to the undesired event and identification of minimal cut sets contributing event results.

4) computation and ranking of basic input failure and availability performance results.

In accomplishing these steps, the computer program simulates the fault tree. Using the input data, it randomly selects various parameters from an assigned statistical distribution, and then tests whether or not the specified final event occurred within the specified time period. Each test is a trial, and these trials are run until the desired quantitative resolution is obtained. Thus, thousands or millions of trial years of performance can be simulated.

The simulation technique, as described above, is commonly called direct simulation, i.e., all the failure rates, repair rates and failure probabilities have the original values. The accuracy of the method increases with the number of trials but the method has the disadvantage that the required computation time for a given accuracy of the result increases tremendously, when the failure rates decrease or the number of basic events increases.

In order to reduce the computer run time to an acceptable level, yet the accuracy still holds. A statistical sampling procedure called importance sampling is used in Monte Carlo analyses (36). This technique depends on biasing the simulation through the use of another distribution so that the component or the combination of components that cause the unlikely event are emphasized in the sampling. So as not to bias the end result, corrections are made in the end.

The computer program RELY 4, developed by H.E. Kongsoe (72), consists of four different versions. Versions 1 and 3 use importance sampling and versions 2 and 4 use direct simulation.

In 1970, W.E. Vesely (119) made a most important advance in quantitative evaluation of fault trees by developing an analytic methodology, called Kinetic Tree Theory (KITT) , for fault trees containing repairable components. The output from computer programs exercising Vesely's method (125) contains complete quantitative information about the top event which includes the following, all as a function of time :

1) the probability of occurrence of the top event

2) the expected number of top event ocurrences per unit time

3) the hazard rate for the top event

4) the expected number of occurrences of the top event during the time interval 0 to t.

Similar information is also obtained for the minimal cut sets and primary (basic) events.

I. Caldarola and A. Wickenhauser (26) also developed an analytical computer program for fault tree evaluation . This program can evaluate coherent systems assuming binary component states with four different classes of components. The program locates minimal cut sets deterministically using a downward algorithm like that developed by Fussell (60). Computational techniques are based on those derived by Vesely (119). Besides having the capability to rapidly determine a variety of reliability characteristics for coherent systems, the program can perform a two-phase phased mission analysis. System unavailabvility is calculated for the first time period (phase) and system unreliability is calculated for the second phase. A second computer program is also developed for solving noncoherent systems with multistate components (23,24).

The application of reliability and fault tree analysis methods has rapidly increased in many industries that employ systems whose failure would have serious economic and safety consequences. Fault tree analysis is not necessarily a replacement for other forms of system reliability analysis but rather in many cases can be used in conjunction with inductive techniques to generate a unified system reliability and safety analysis as has been suggested by B.J. Garrick (64). In WASH-1400 Reactor Safety Study (99), fault trees were used for the individual system analysis, while event trees were used to construct the accident chains for nuclear power plant. Each defined system failure from the event trees served as a top event of a fault tree which was then constructed for the particular system. The result of the system reliability computation was an interval estimate of the reliability characteristic of interest. A Monte Carlo program called SAMPLE (99), was used to compute the uncertainty distribution on the system reliability characteristic of interest using the simplified mathematical model (based on exponential failure distribution for the system components) and using the uncertainty distribution on the parameters of the component failure and repair distribution .

The SAFTE codes by B.J. Garrick (64) are also exapmles of Monte Carlo simulation programs. The SAFTE-1 program utilizes sampling techniques to simulate time dependent failure-repair fault tree models and to perform relevant systems reliability analysis, while the SAFTE-2 program has been written to consider the simpler case of no repair. SAFTE-3 is a Monte Carlo program capable of handling systems with or without redundancy and utilizes either direct or importance sampling techniques.

The REDIS program (73), developed in Denmark, is based on simulation of the direct type.The program is particularly designed for standby systems taking into consideration the various kinds of errors that can cause failure during both operating and standby conditions. It allows for uncertainties in data and produces a top event failure probability with a standard deviation. Furthermore, various types of interdependencies between components are allowed.

R.C. Erdmann et al. (45), also develop the WAM series of computer codes to provide flexibility as well as accuracy in the analysis of system reliability. By utilizing a common input deck, the WAM codes can provide information about systems modeled by any Boolean function. The information includes the point estimates of system as well. A drawing of the fault tree as input to the evaluation codes can also be obtained.

Several computer codes for different approaches are also available to analyze fault trees quantitatively. The SALP computer series , developed by M. Astolfi et al. (5), in Italy, are based on the use of list processing techniques for the direct manipulation of graphs. The algorithms introduce the NOT operator and qualitative information at the level of gates and primary events. While the SALP-3 is used for routine analysis of AND/OR fault trees, the SALP-4 can deal with AND-OR-NOT fault trees.

The PATREC code by A. Blin et al. (18), is also based on list processing techniques, which is realized by recognizing and replacing known subtrees or patterns by equivalent leaves with the corresponding unreliability/ unavailability. By repeatedly pruning the fault tree, it is finally reduced to a single leaf which represents the system unreliability for unrepairable systems and unavailability for repairable systems. The code is oriented towards direct computation of the top event availability without the use of minimal cut sets.

M.F. Chamow (31) suggests a new approach involving well-defined, closed-form methods for quantitative evaluations of fault tree logic. The method is based on directed graphs (diagraphs) and related matrix methods and depends in a major sense on the digraph representations developed for the basic OR and AND logic elements. The benefit of this method arises because the mathematical solutions are readily performed by standard matrix techniques, which can be implemented either manually or with the aid of a computer.

Finally, for the utilization of computer on computation and storage requirements, the fault tree analysis using bit manipulation suggested by D.B. Wheeler et al., (127), shows the effectiveness in producing minimal cut sets and top event probability through analysis of fault trees of various sizes.

The measure of importance of events and cut sets in fault trees is another important feature of quantitative fault tree analysis. While the evaluation of the top event provides us with system reliability/ availability information, the computation of probabilistic importance allows us to generate a numerical ranking to assess weaknesses in a system. Such a ranking is analogous to a sensitivity analysis. The practical applications of importance measures are for upgrading system designs, locating diagnostic sensors, and for generating checklists for system diagnosis (76).

Several probabilistic methods can be used to compute the importance of basic events and cut sets in the fault tree. A fundamental quantity in computing probabilistic importance is Birbaum's measure of importance (16). He defined the reliability importance of a component as the rate at which system reliability improves as the reliability of a component improves. In other words, Birbaum's measure of importance is the probability that the system is in a state in which the occurrence of event is critical. However, it is possible that a failure of a component can contribute to system failure without being critical. So Vesely and Fussell define the probabilistic importance as the probability that a component is contributing to system failure, given that the system has failed by specified time t. (53).

Barlow and Proschan (9) examined components as they fail sequentially in time . They consider the way components fail sequentilly in time to cause system failure, and then define the probabilistic importance as the conditional probability that a component causes the system to fail by time t. This is termed sequential measure of importance and gives specific information about the way system failure occurred.

Definitions of cut set importance are described by analogy to methods that determine component (basic event) importance.

H.E. Lambert (76) developed a computer code IMPORTANCE to compute various measures of probabilistic importance of basic events and cut sets to a fault tree. The code requires as input the minimal cut sets, the failure rates and the fault duration time (the repair times) of all basic events contained in the minimal cut sets. The output of the code includes seven measures of basic event importance and two measures of cut set importance by assuming statistical independence of basic events.

2.6 Available Computer Codes for Fault Tree Analysis

Numerous computer codes are available for processing fault trees. They are presented in Table 2.4. In the construction phase of the analysis, Fussell (51) pioneered the work with his DRAFT code for electrical systems. Salem et al. (105) produced the CAT code based on the application of the decision table. Lapp and Powers (80) developed the Fault Tree Synthesis (FTS) code for chemical processing system. Taylor and Hollo (116) use algebraic component models to construct a Cause-Consequence Diagram (CCD) which extends the fault tree methodology to better describe the the sequential effects of accident chains and to increase their visibility in the analysis procedure.

For qualitative evaluation, Vesely and Narum (125) made available a PREP code that obtained the minimal cut sets (or minimal path sets) for the fault tree. Because of the time consuming nature of the algorithms used in PREP, several newer and more efficient codes have been written employing faster deterministic routines not requiring Monte Carlo methods. The MOCUS code by Fussell (56) starts at the top of the fault tree and proceeds down while the MICSUP code by Pande et al. (93) starts at the bottom of the tree and proceeds up. In general, MICSUP requires less memory storage space in the computer than MOCUS since MICSUP stores all cut sets in a single array. For fault trees containing NOT gates (became non coherent), Worrell (132) developed the SETS computer code to find the 'prime implicants' for the fault tree. The prime implicants are like minimal cut sets except that they may contain complemented basic events. Other well-known deterministic programs for determining minimal cut sets are ALLCUTS (118), ELRAFT (110), FAUTRAN (131), FATRAM (90), DICOMIC (62), BAM-CUTS (45) and BUP-CUTS (90). For common cause failure analysis of qualitative fault tree evaluation, two computer codes have been developed using minimal cut sets as input : COMCAN (21), developed at INEL, and BACFIRE (30), developed at the University of Tennessee.

The early computer codes for quantitative fault tree evaluation were available to obtain probabilistic information about the top event by using the Monte Carlo method. Such codes are RELY 4 (72) and Crosetti's code (38). SAFTE (64), REDIS(73) and SAMPLE (99) can also be classified in this category. For analytic methodology of quantitative evaluation, Vesely and Narum (125) provided the KITT code for probabilistic fault tree evaluation starting from primary failure information to top failure information. Caldarola and Wickenhauser (26) also produced an analytical computer program similar to that of Vesely and Narum. The PL-MOD code by Olmos and Wolf (92) performed the step by step modularization of fault trees trhrough an extensive use of the list processing tools available in PL-1. Other computer codes developed by many industry users and research institutions and serving similar evaluation interests are ARMM (89), GO (65), NOTED (130), WAM-BAM (45), PARTEC (18), and SALP (5). Finally, for the measure of importance of events and cut sets in fault trees, Lambert (75) developed a very comprehensive computer code, IMPORTANCE, which computes various measures of probabilistic importance of basic events and cut sets to a fault tree.

2.7 Concluding Remarks

Fault tree analysis is a versatile reliability tool that has rapidly won favor with those involved in reliability and safety calculations. But fault tree models do have disadvantages. Probably the most outstanding one is the cost of development in first time application to a system. Some inductive analysis technique, like Failure-Mode-and-Effects Analysis (FMEA), is a much simplier and more cost effective technique to apply in analyzing small systems when a single failure analysis is adequate. However, as systems become more complex and the consequences of accidents become catastrophic, a technique such as fault tree analysis should be applied. Fault tree analysis can efficiently direct the efforts of an analyst in considering only those basic events that can contribute to system failure and represent the relationship of human error and environmental conditions in causing system failure. In addition, as the fast progress of automated fault tree analysis, this technique can be a more effectively and sophisticated analytical reliability tool.

A major difficulty with quantitative fault tree evaluation is the lack of pertinent failure rate data. Nevertheless, quantitative evaluations are particularly valuable for comparing systems designs that have similar components. The results are not as sensitive to the failure rate data as is an absolute determination of the system failure probability. Because of uncertainties in failure rate data, quantitative fault tree analysis has its greatest value when relative rather than absolute determinations are made. Fault tree analysis is then best applied during the detailed design stages of a system.

Fault tree analysis can be a most simple or a most sophisticated analytic reliability tool depending on the needs of the analyst. For system safety analyst, fault trees provide an objective basis for analyzing failure modes and probabilities and evaluating overall reliability. Simple logic applies to both systems and subsystems, and is an effective visualization tool for management as well as engineering; and for the process control or aerospace system analyst as well as the nuclear reactor design engineer.

CHAPTER THREE

FAULT TREE CONSTRUCTION

3.1 Introduction

A fault tree is a deductive logic model that graphically represents the various combinations of possible events, both fault and normal occurring in a system that lead to the top event. The goal of fault tree construction is to model the system conditions that can result in the undesired event. Before the construction of the fault tree can proceed, the analyst must acquire a thorough understanding of the system. In fact, a system description should be part of the analysis documentation. The analyst must carefully define the undesired event under consideration, called the top event. To make his analysis understandable to others, the analyst should clearly show all the assumptions made in the construction of the fault tree and the system description used. Practical considerations require that he scope the analysis, setting spatial and temporal bounds on the system. He should determine the limit of resolution, identify potential system interfaces and realize the constraints of the analysis in terms of the available resources, time and money.

Fault tree construction is commonly the most time consuming task. While much of the statistical and cut set analysis has been automated, actual construction of the fault tree is usually done by hand. Manual construction of the tree can be extremely tedious; a substantial fraction of 25 man-years of effort was required in the Nuclear Reactor Safety study (99). In addition to the time involved, the possibility exists that different analysts will produce different fault trees either by incorrect logic or ommission of certain events. A computer aided construction technique would prove valuable in alleviating both of the above problems. Any system of constructing fault trees should have the following four characteristics (80) :

1) Handle complex systems efficiently. A complex system is one with feedback or feedforward loops and over 20 components.

2) Consider system topology as well as actual components in constructing the tree.

3) Handle multivalued logic, i.e., consider the direction and magnitude of deviations in process variables in addition to component failures.

4) During fault tree construction, make checks to ensure consistency among events.

The automation of the construction phase of fault tree analysis has attracted considerable attention in recent years. Several methodologies have been proposed differing in the modeling of components or variables and in their objectives. Haasl (67) described some general structuring processes for fault tree construction. Fussell (51) developed the Synthetic Tree model for electrical systems. Powers and Tompkins (95-97) started the use of input/ output models for describing the local cause and effect relationships between variables and failure events for a single component of a system. Salem et al.,(104-107) used decision tables to aid in generation of fault trees. Lapp and Powers (80-82) developed the multiple edged digraph method of modeling of cause and effect, and invented an algorithm which converts the digraph into a fault tree. Camarda, et al., (27) initiated an algorithm for fault tree automatic synthesis from the reliability graph. Taylor (113-117) has also developed an algol-like language for describing the cause and effect relationships for a system component.

In the sections that follow, we will discuss the above methodologies of fault tree construction in detail.

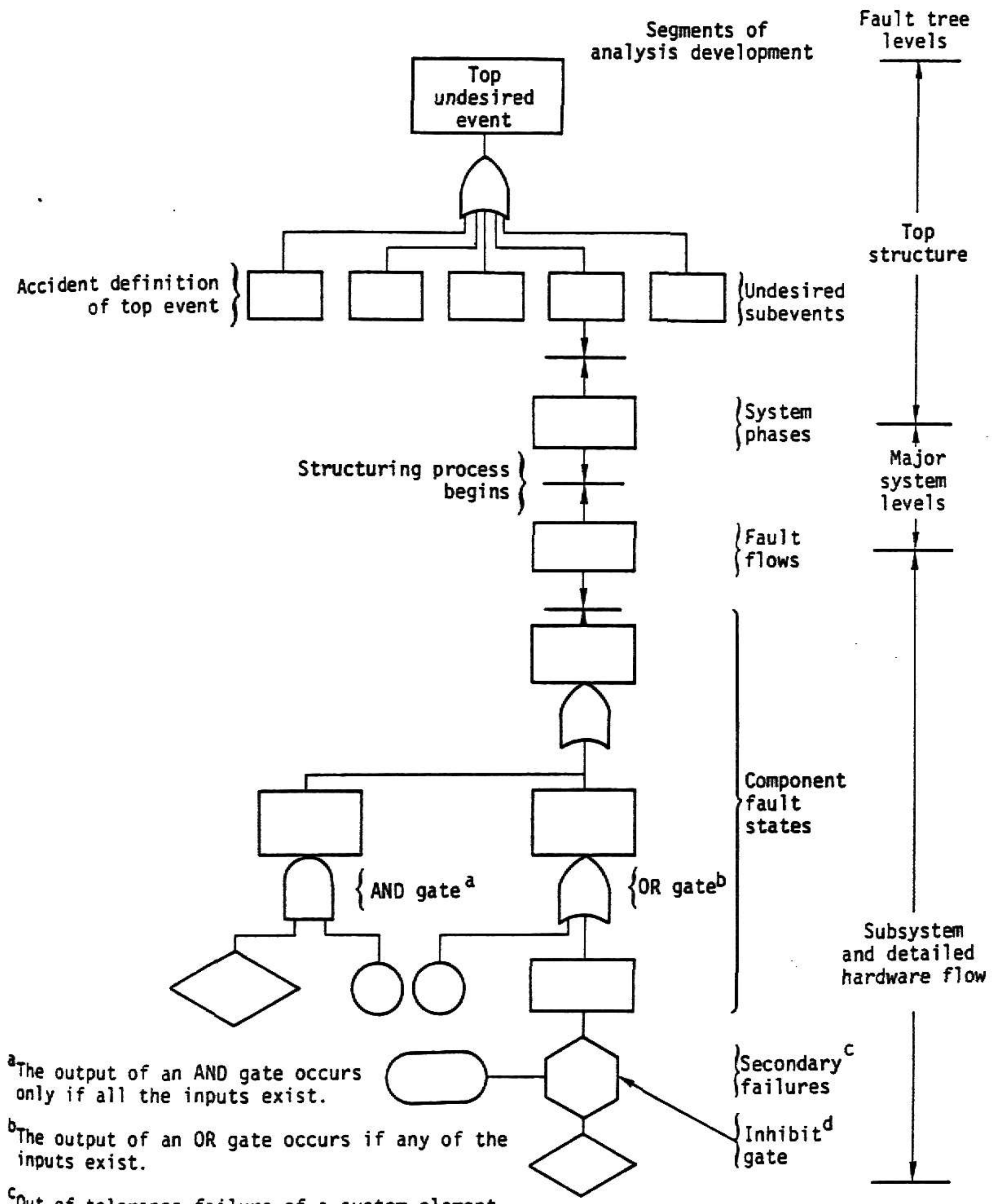
3.2 Haasl's Structuring Process for Fault Tree Constrution

3.2.1 Introduction

David Haasl (67) formalized the thought process involved in the construction of the fault tree. He devised a structuring process that established rules to determine the type of gate to use and inputs to the gate. The structuring process is used to develop fault flows in a fault tree when a system is examined on a functional basis, i.e., when failures of system elements are considered (see Figure 3.1). At this level, schematics, piping diagrams, process flow sheets, etc., are examined for cause-and effect types of relationships, to determine the subsystem and component fault states that can contribute to the occurrence of the undesired event. At this point, the flow of energy through the system is followed in a reverse sense from some undesirable outcome to its source.

3.2.1.1 Structuring Process

The structuring process requires that each fault event be written to include the description and timing of the fault event at some particular time. This means that each fault event must be written to include what the fault state of that system or component is and when that system is in the fault state. The established procedure answers two principle questions : 1) Is the event a state-of-component



[a]The output of an AND gate occurs only if all the inputs exist.

[b]The output of an OR gate occurs if any of the inputs exist.

[c]Out-of-tolerance failure of a system element — failure due to excessive operational or environmental stress.

[d]An inhibit gate is a special case of the AND gate. The oval indicates a conditional event

FIG. 3.1 Levels of Fault Tree Development (76)

or state-of-system fault ? 2) What is immediately necessary and sufficient to cause the event ?

In a state-of-component fault event, three failure mechanisms or causes are identified that can contribute to a component being in a failed state :

1) A primary failure is due to the internal characteristics of the system element under consideration.

2) A secondary failure is due to excessive environmental or operational stress placed on the system element.

3) A command fault is an inadvertent operation or non-operation of a system element due to failures of initiating elements to respond as intended to system conditions.

The above failure mechanisms describe the fundamental process involved in or responsible for a component failure mode.

We see that in the case of the first two failure mechanisms, the system element is nc longer able to perform its intended function (unless the element is repaired). In the case of the third failure mechanism, the system element can operate as intended if the initiating elements are returned to their normal states.

We use Figure 3.2 to demonstrate these failure-mechanism concepts. The primary event is indicated in circle. The command fault is shown in the rectangle. Some

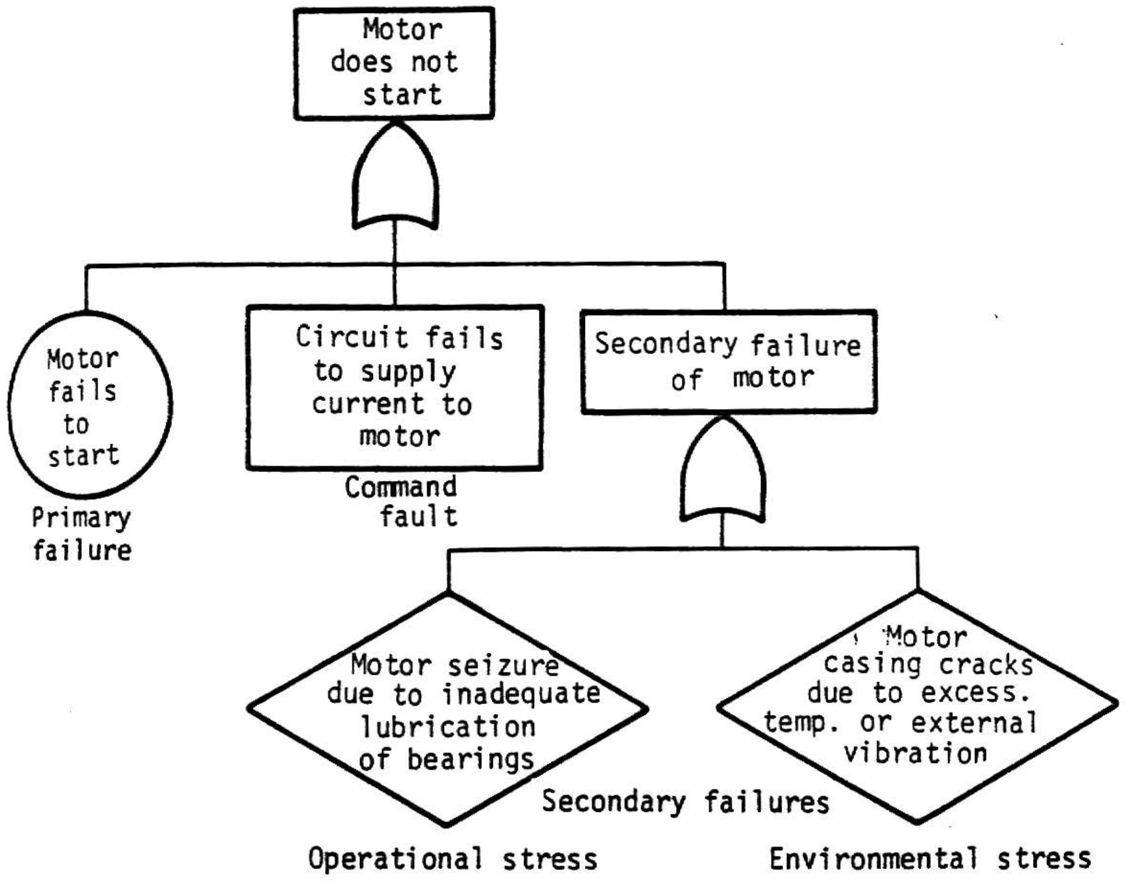

Motor does not start
Motor fails to start
Primary failure
Circuit fails to supply current to motor
Command fault
Secondary failure of motor
Motor seizure due to inadequate lubrication of bearings
Motor casing cracks due to excess. temp. or external vibration
Secondary failures
Operational stress
Environmental stress

FIG. 3.2 Fault Tree Showing Development of State-of-Component Fault Event(76)

out-of-tolerance failure mechanisms for the motor are 1) inadequate maintenance of motor and 2) excessive temperature or external vibration. The fault tree in Figure 3.2 is a simplified motorcircuit-switch system failure scheme.

Any fault event that can be described in terms of the failure mechanisms described above is said to be a state-of component fault event. In this case, the system element under examination is the sole cause of the fault event, i.e., the event results from the action of a single component.

An OR gate is always used to combine the input at a lower level which consist of the three failure mechanisms or causes as described above. Examples of state-of-component fault trees are, 1) failure of motor to start,2) failure of motor to turn off, 3) switch fails to open, and 4) switch fails to close. Events that have a more basic cause that cannot be described in terms of a simple component failure are termed state of system fault events. In this case an OR gate, AND gate, inhibit gate, or no gate at all can be used to combine the events at the next lower level. In state-of-system fault events, the immediately necessary and sufficient fault input events must be specified. For each newly developed event other than primary causes the structuring process is repeated until each event is developed to its limit of resolution.

3.2.1.2 Levels of Fault Tree Development

A complete or global safety analysis using the fault tree technique on an extensive system such as nuclear power plant or chemical processing plant normally requires three levels of fault tree development as shown in Fig. 3.1. The upper structure, called the top structure, includes the top event and the undesired subevents. These events such as fire, explosion, release of radioactivity are potential accidents and hazardous conditions and are immediate causes of top event. There is no structuring process at this level to tell the analyst what gate to use or what inputs are specified. The top structrure is actually a list of the functions whose loss constitutes a major accident as specified by the top undesired event. David Haasl claims that structuring the fault tree at the top level is an art in outlining.

The next level of the fault tree divides the operation of the system into phases and subphases, until the system environment remains constant and the system characteristics do not change the fault environment. In this second level of fault tree development, the analyst examines system elements from a functional point of view. Hence, the structuring process is used to develop fault flows within the system that deductively lead to subsystem and detailed hardware fault flow, which is the third level of the fault tree. At the third level, the analyst is faced with one of the most difficult aspects of fault tree analysis. He must show any external failure mechanisms that can simultaneously fail two or more system elements, and restructure the fault tree accordingly. The effects of common environmental or operational stresses are studied, as well as the effects of the human factor in the testing, manufacturing, maintainence, and operation of the system.

To illustrate further the concepts of the structuring process, a fault tree for a simple electrical system is given below.

3.2.2 Example

As an example, a sample system is given in Figure 3.3. This system is a standby system that is tested once every month. It consists of a battery, two switches in parallel, and a motor. To start the motor, two push buttons are pressed to close the two switch contacts 1 and 2. To stop the motor at the end of test, two push buttons are depressed. Periodically, say every six months, the operator must recharge the battery and perform routine maintenance on the motor.

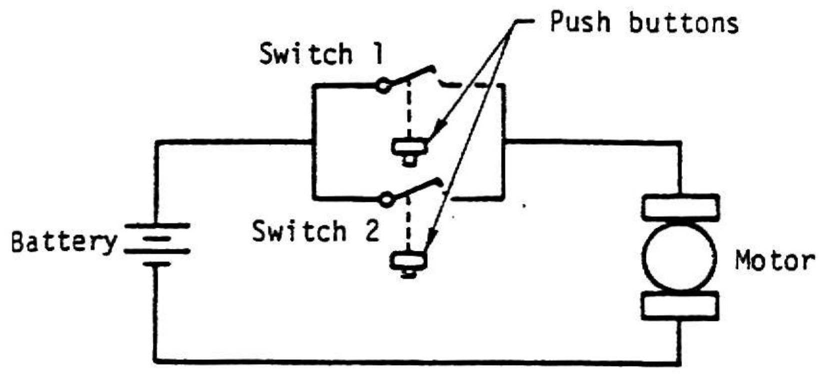


FIG. 3.3 Sample System

We assume that the wires or connections do not contribute to system failure. Pre-existing faults are allowed, e.g. the switch contacts may be failed closed as initial conditions. We also assume that all components are properly installed.

A detailed fault tree for the sample system is generated via the structuring process and is shown in Figure 3.4. The top event appears as failure of the motor to stat on test. Each gate event is labeled as to the event type, either state cf component or state of system fault event. We see that all command faults and secondary failures when developed are state-of-system fault events. An inhibit gate is shown in the development of the secondary failure, overrun of battery. It is interesting to note that two types of failure are shown for the switches. Switch 1 and 2 can fail to close upon demand or they can fail to open from the previous test and cause the battery to discharge. Close

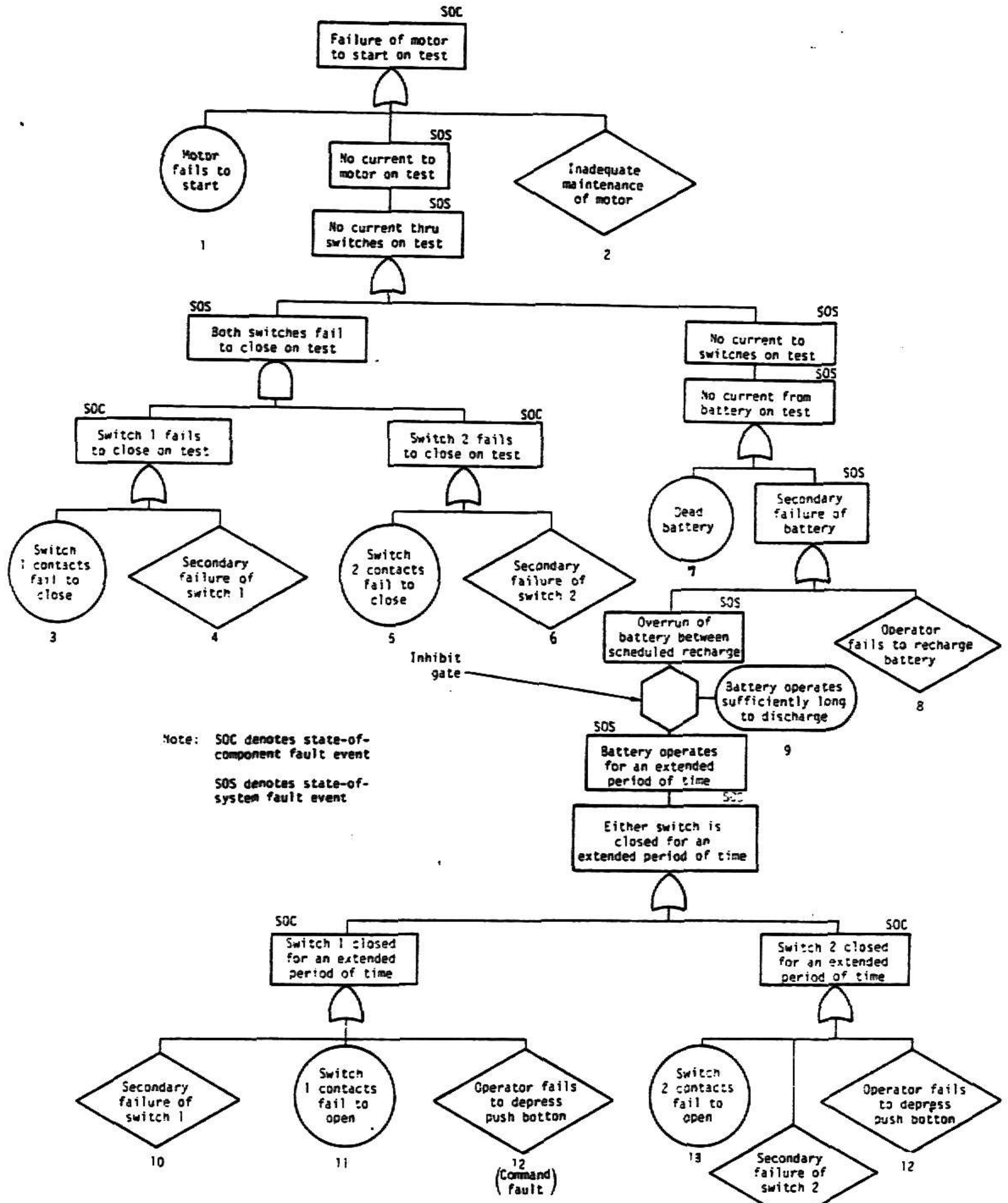

SOC
Failure of motor to start on test
Motor fails to start
1
SOS
No current to motor on test
SOS
No current thru switches on test
Inadequate maintenance of motor
2
SOS
Both switches fail to close on test
SOS
No current to switches on test
SOS
No current from battery on test
SOC
Switch 1 fails to close on test
SOC
Switch 2 fails to close on test
Dead battery
7
SOS
Secondary failure of battery
Switch 1 contacts fail to close
3
Secondary failure of switch 1
4
Switch 2 contacts fail to close
5
Secondary failure of switch 2
6
SOS
Overrun of battery between scheduled recharge
Operator fails to recharge battery
8
Inhibit gate
Battery operates sufficiently long to discharge
9
Note: SOC denotes state-of-component fault event
SOS denotes state-of-system fault event
SOS
Battery operates for an extended period of time
SOS
Either switch is closed for an extended period of time
SOC
Switch 1 closed for an extended period of time
SOC
Switch 2 closed for an extended period of time
Secondary failure of switch 1
10
Switch 1 contacts fail to open
11
Operator fails to depress push botton
12
(Command fault)
Switch 2 contacts fail to open
13
Secondary failure of switch 2
14
Operator fails to depress push botton
12

FIG. 3.4 Detailed Fault Tree of Sample System Generated via Structuring Process (76)

examination of the Fig.3.4 fault tree shows that human error can play a key role in system failure. The operator can forget to recharge the battery or fail to depress the push buttons after the test. Fault trees that include only hardware failures will overpredict the capability of performance of the system. Realistic assessments of system failure must include human error and secondary failures.

3.3 Fussell's Synthetic Tree Model (STM)

3.3.1 Introduction

Fussell's methodology for fault tree construction is programmed in a computer code called DRAFT (51) that automatically constructs fault trees of electrical schematics to the level of primary hardware failures. The basic building blocks of the methodology are component failure transfer functions. These are mini fault trees for components in a fault state. The information contained in them can be derived from a failure mode analysis which is independent of the particular system considered. With proper editing, the fault tree is automatically constructed from the component failure transfer functions. A hierachical scheme is developed that identifies fault events according to order. The information required as input to the code is 1) a schematic of the electrical system, 2) when applicable, the initial operating state of each component and 3) boundary conditions that can impose restrictions on the top event and events developed within its domain. The computer then finds the series circuit paths for each component in the schematic, called component coalitions, and identifies the order of each event requiring development. Events are considered up to fourth order. It then imposes new boundary conditions when necessary and then constructs the fault tree accordingly. The flow chart illustrating the methodology of

STM is given in Figure 3.5.

3.3.1.1 Event Description

In the STM, there are two parts to the event desription, a) the incident identification and b) the entity identification. The entity identification is the subject of the fault event and refers either to a component or to a component coalition. The incident identification describes a mode of failure or fault state. For example, consider the situation where current is inadvertently applied to the coil of a relay causing its contacts to close. In the fault statement relay contacts close inadvertently, the entity identification is relay contacts, and the incident identification is close inadvertently.

3.3.1.2 Component Failure Transfer Functions

There are several important facets of component failure transfer functions to be considered. First, the functions for any component must be dependent on the operational and failure characteristics of that component alone, and cannot depend upon the system itself. This insures that the transfer functions will be applicable regardless of the system in which the component is used. Actually, this is not only a requirement for STM, but for any fault tree analysis, unless component modeling is done only for a specific system. Secondly, the transfer functions consist of 6 parts,

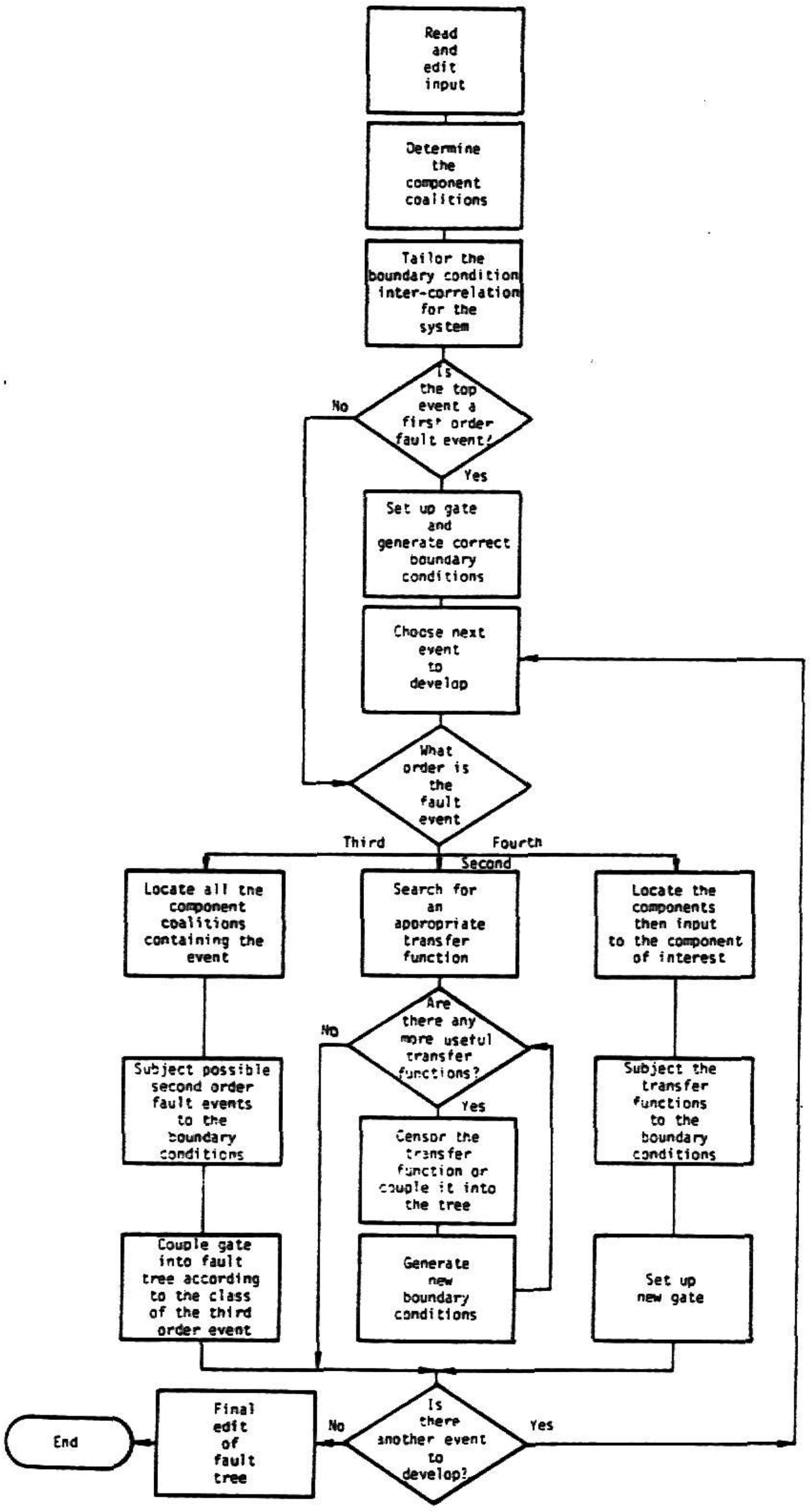

Read and edit input
Determine the component coalitions
Tailor the boundary condition inter-correlation for the system
Is the top event a first order fault event?
No
Yes
Set up gate and generate correct boundary conditions
Choose next event to develop
What order is the fault event
Third
Second
Fourth
Locate all the component coalitions containing the event
Search for an appropriate transfer function
Locate the components then input to the component of interest
Are there any more useful transfer functions?
No
Yes
Subject possible second order fault events to the boundary conditions
Censor the transfer function or couple it into the tree
Subject the transfer functions to the boundary conditions
Couple gate into fault tree according to the class of the third order event
Generate new boundary conditions
Set up new gate
End
Final edit of fault tree
No
Is there another event to develop?
Yes

FIG. 3.5 Flowchart for DRAFT Computer Code (51)

as indicated by Figure 3.6. The first element is the output event, or failure mode of the component. There is a unique failure transfer function for each failure mode of the component.

Secondly, each transfer function has an output gate which couples the transfer function to the rest of the tree. Beneath the output gate, the transfer function itself is defined in terms of internal events and logic gates. Finally, these internal gates are coupled to the tree beneath via the input events. A sixth factor, the discriminator, is a flag which specifies which of the various transfer functions for a specific component can exist simultaneously in the fault tree. For instance, current and current too long can exist in a single component, but no current and current too long cannot.

Primary failures are always part of the component failure transfer functions . The logic gate used in the failure transfer function depends upon the type of failure considered for the component. For example, an electrical component such a fuse can fail in such a manner as to cause the output event to occur implying OR logic for the output gate. In another case, an electrical component can transmit an overload or inadvertently transmit current. Coexistence of another fault event is necessary for the output event to occur. In this case, the logic for the output gate is AND.

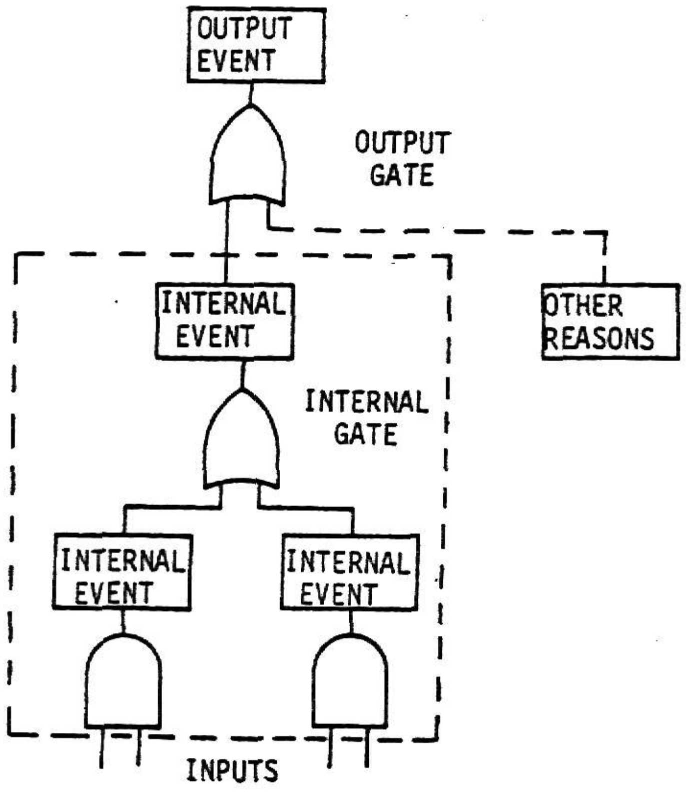


Fig. 3.6 Failure Transfer Function Structure

This situation is common with protective devices that fail in such a manner to allow out-of-tolerance conditions to exist, e.g. a fuse failing to open when a current overload exists within the circuit. Figure 3.7 illustrates failure transfer functions for electrical contacts. We can see that state-of-component fault events are embodied within these transfer functions.

3.3.1.3 Component Coalition Scheme

Within the context of the STM, a component coalition is a series circuit path in which components share an alliance with respect to current flow. If any one component fails to pass the current, then no current can flow in that coalition. However, any component may appear in more than one coalition. That component, then may receive and pass current via one coalition, even if no flow exists in the other. In sample system, fig.3.3, there are two component coalitions, 1) the battery, switch 1 and the motor and 2) the battery, switch 2 and the motor . This means there are two paths by which the motor can receive current from battery. If the switch 1 fails, no current flows in coalition 1, but may still flow in coalition 2 if the three appropriate components each allow current to pass. However, if the battery or the motor fail, no current can flow in either coalition, since these components are common to both. While this scheme is easily applied to electrical systems,

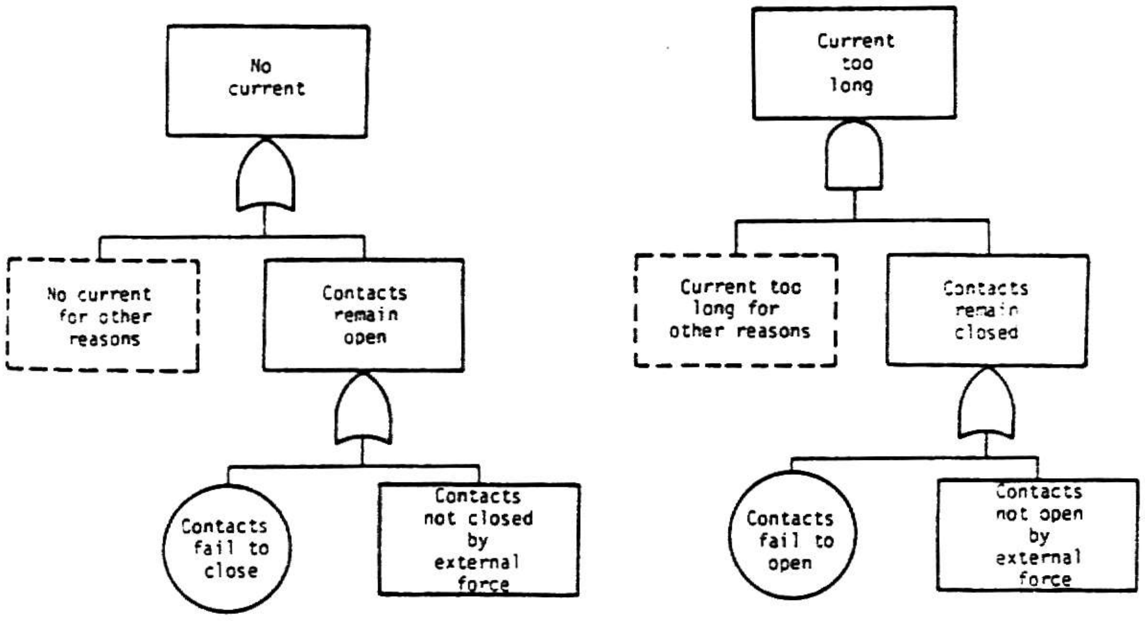

No current
No current for other reasons
Contacts remain open
Contacts fail to close
Contacts not closed by external force
Current too long
Current too long for other reasons
Contacts remain closed
Contacts fail to open
Contacts not open by external force

FIG. 3.7 Failure Transfer Functions for Electrical Contacts. (51)

it may be difficult to break down other types of systems into coalitions such as these.

3.3.1.4 Ordering of Fault Events

In contrast to Haasl's structuring process in which there are two basic fault events, state-of-component and state-of-system fault events, Fussell's methodology divides the construction of fault trees into four levels, or orders of fault events, 1) first-order, 2) second-order, 3) third-order and 4) fourth-order fault events. The following paragraphs describe the ordering of the fault events in the STM. It is helpful to refer to the flowchart in Figure 3.5.

Third and fourth-order fault events in the STM are command faults. For the development of third-order fault events, components are examined with respect to energy input from all series circuit paths that contain these components. This amounts to examining the state of each component coalition that is a source of energy or current to a given component. Events such as component receives no current when needed and component receives current inadvertently are examples of third order fault events. If a component is producing a fault event because of mechanical linkage with another component, such as a relay coil and its associated contacts or a pressure switch and its contacts, then such an event is referred to as a fourth-order fault event. Because of direct component interplay, fourth-order fault events always require component failure transfer functions as input events. Events such as no current in a component coalition, inadvertent flow of current in a component coalition are second-order fault events.

The development of third-order fault events always requires as input second-order fault events. In examining the fault state of each component coalition, we must examine each component in the coalition. Hence, the development of second-order fault events always requires as input failure transfer functions. If these failure transfer functions require third or fourth order fault events as input, then the above process is repeated until there are no more second-, third-, or fourth-order events that require development. The fault tree is completed when all events are developed to the level of primary hardware failures.

In some cases the top event is of first order,i.e., an event that requires development to the level of subsystem functional faults. In this case the analyst must manually construct the fault tree to the level where events are second order or higher. This procedure is analogous to the construction of the upper structure of the fault tree mentioned in the previous section. Fussell calls the upper structure the tree top boundary condition.

3.3.2 Example

As an example of the synthetic tree methodology, we again construct a fault tree for the sample system in Figure 3.3. In the STM, the initial conditions must describe the system in an unfailed state. The system boundary conditions must describe the system in an unfailed state. The system boundary conditions are :

TOP EVENT = Motor fails to start on test

Initial Conditions = Switches open

Not-allowed Events = Wiring or connection failures

Existing Conditions = None

The fault tree is shown in Figure 3.8. Note that a little more detail is shown on the switch contacts in Figure 3.8 in order to illustrate the development of fourth-order fault events. The hierarchical scheme illustrating the ordering of fault events is evident in Figure 3.8. Also, note that the circled events bear an almost exact resemblance to the component failure transfer function given in Figure 3.7 with initial conditions contacts open.

Second-order fault events such as no current in component coalition impose restrictions on events placed in their domain; e.g., if in the subsequent development of this event, we consider the component coalition again, events like current in component coalition are not allowed. Because of this restriction, component failure transfer functions

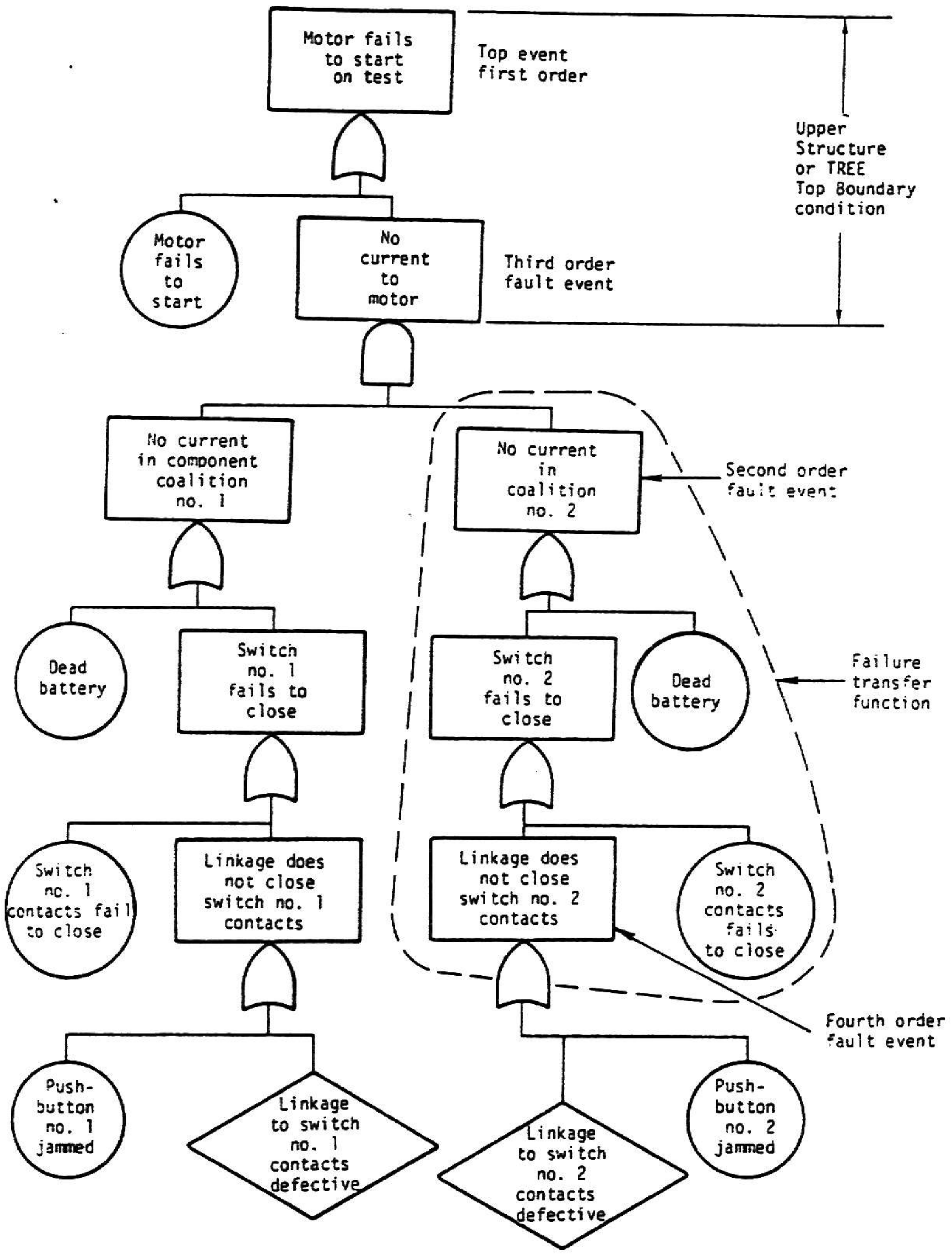

Motor fails to start on test
Top event first order
Upper Structure or TREE Top Boundary condition
Motor fails to start
No current to motor
Third order fault event
No current in component coalition no. 1
No current in coalition no. 2
Second order fault event
Dead battery
Switch no. 1 fails to close
Switch no. 2 fails to close
Dead battery
Failure transfer function
Switch no. 1 contacts fail to close
Linkage does not close switch no. 1 contacts
Linkage does not close switch no. 2 contacts
Switch no. 2 contacts fails to close
Fourth order fault event
Push-button no. 1 jammed
Linkage to switch no. 1 contacts defective
Linkage to switch no. 2 contacts defective
Push-button no. 2 jammed

FIG. 3.8 Fault Tree of Sample System Illustrating Synthetic Tree Methodology (76)

with output event current are equally not allowed. Fussell calls these restrictions, event boundary conditions. Such conditions are of consequence when we try to develop the secondary failure overrun of the battery. As we see in Figure 3.4, the battery discharges when the motor operates for an extended period of time. This further implies there is current in either component coalition 1 or 2. In the context of the STM we cannot place the secondary failure of the battery in the domain of the second-order fault events given in Figure 3.8. Instead, we must consider the system in a different operating state and construct a new fault tree with different tree top boundary conditions. The boundary conditions in this case are :

TOP EVENT = Battery operates for extended period of time

Initial Conditions = Switches closed

Not-allowed Events = Wiring or connetion failures

Existing Conditions = Motor operating

The tree top boundary condition and the fault tree are given in Figure 3.9.

Fussell further assigns third order fault events to classes. In Figure 3.9, a component (in this case, the motor) can inadvertently receive current (or an overload) from any coalition containing the component, implying OR logic as shown. This type of third order event is assigned to class I. On the other hand, in Figure 3.8, a component

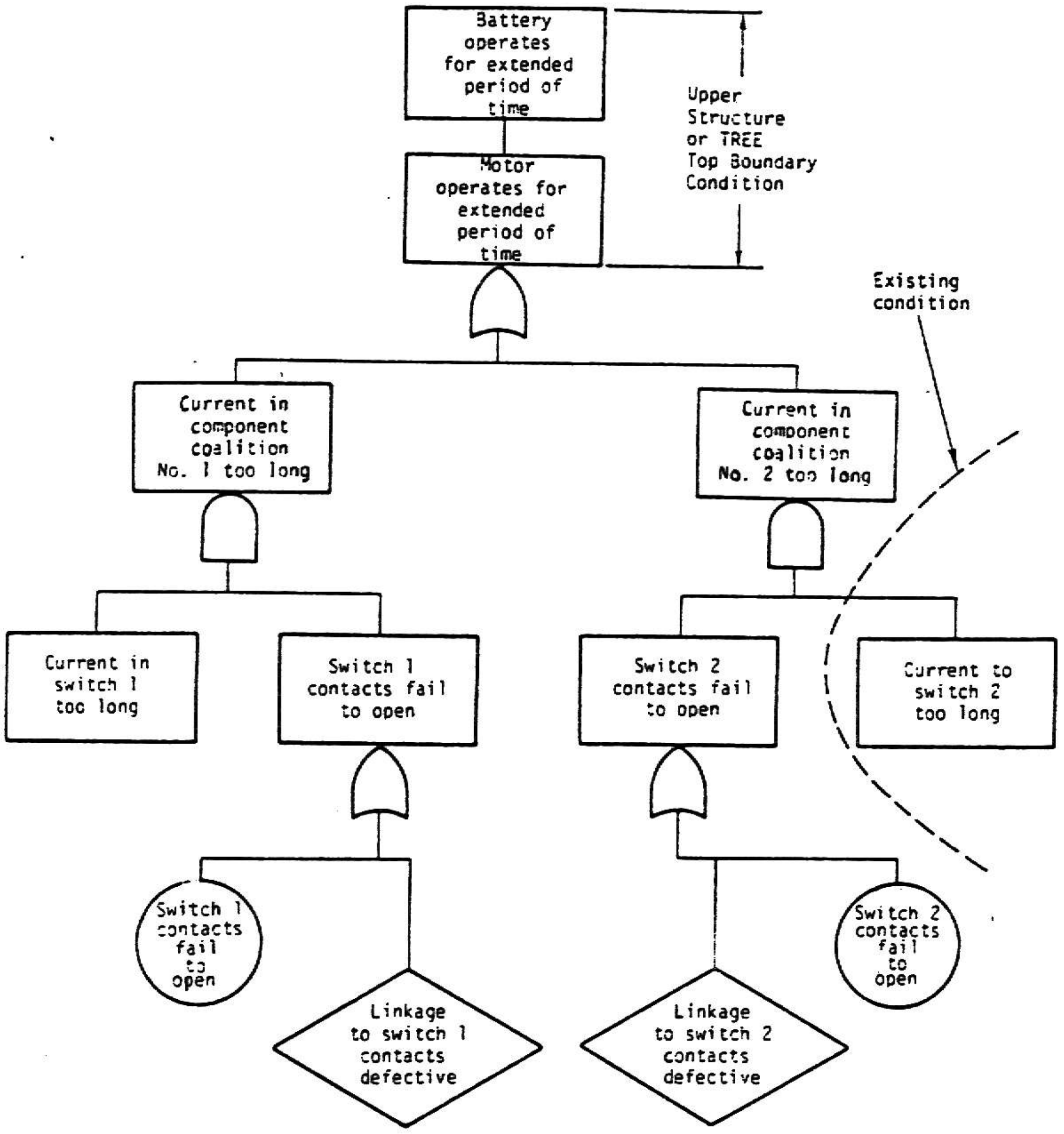

Battery operates for extended period of time
Motor operates for extended period of time
Upper Structure or TREE Top Boundary Condition
Existing condition
Current in component coalition No. 1 too long
Current in component coalition No. 2 too long
Current in switch 1 too long
Switch 1 contacts fail to open
Switch 2 contacts fail to open
Current to switch 2 too long
Switch 1 contacts fail to open
Linkage to switch 1 contacts defective
Linkage to switch 2 contacts defective
Switch 2 contacts fail to open

FIG. 3.9 Fault Tree for Secondary Failure of Battery (76)

receives no current when needed if all coalitions containing the component have no current, implying AND logic as shown. This type of third-order fault event is asigned to class II. In the DRAFT computer code, identification of the class of third-order event is necessary for determination of the proper logic gate to use, see Figure 3.5.

We see that for the sample system in Figure 3.3, if switch 1 or 2 is closed, we would expect the motor to operate. The event current to switch too long in Figure 3.9 is an existing condition and can be removed from the fault tree. The AND gate can also be removed; the fault then can simply be cascaded from one event to the other.

Additional editing concerns are also a part of STM and DRAFT. These deal with the replacement of identical events within a tree by transfers. Any gates which are repeated within a tree may be immediately eliminated and replaced by transfers, which greatly simplify the form and evaluation of complex fault trees.

The DRAFT code, however, was designed specifically to analyze electrical systems and, as such, may not be capable of handling certain types of mechanical components. The applicability of STM, and thus the DRAFT code, depends upon the ability to define events higher than first order fault events ,which is not always possible in any but electrical systems. Furthermore, the DRAFT code has the disadventage that the computer memory storage may be exceeded for large fault trees. This is due to the fact that the computer must store all the event boundary conditions that are generated during the course of fault tree development.

3.4 Powers and Tompkins' Fault Tree Synthesis for Chemical Processes

3.4.1 Introduction

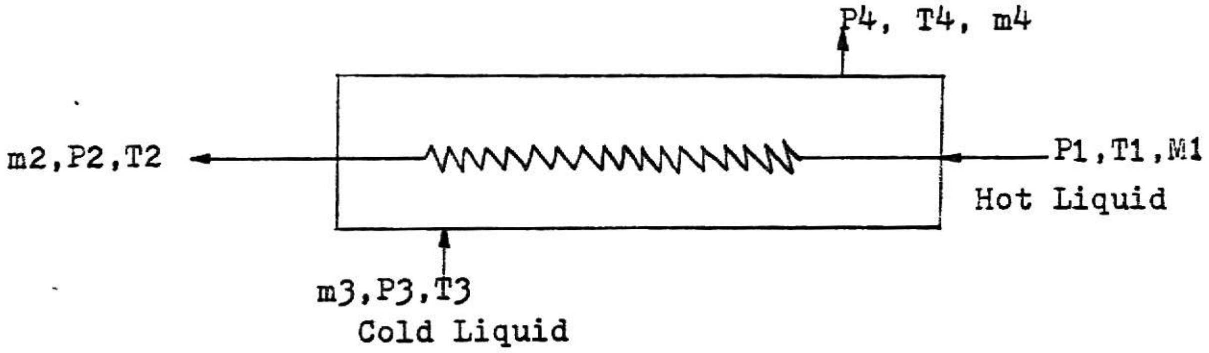
The work by Power and Tompkins is an automated fault tree construction method for chemical systems (95-97). The fault tree generation procedure uses information on 1) the description of the system (detailed flowsheet) 2) physical and chemical properties of materials in and around the system, and 3) unit models which describe the behavior of the units within the system and which are assembled to describe the behavior of the complete system. The unit models are connected to form an information flow structure for the complete processing system. Unit failure models are also defined for common chemical units. By systematically defining hazard states and searching the information flow structure for the system, it is possible to generate fault trees for the complete process.

3.4.1.1 Unit Modeling and Unit Failure Models

In handling chemical interactions, Powers and Tompkins have attempted to incorporate balance equations (material, momentum and energy), describing the behavior of the variables (species) which propagate throughout the system. In doing so, they have defined three levels by which the behavior of some output event (i.e., the products of some reactions) can be determined as dependent on input events (reactants). The simplest level is to specify merely whether the output is dependent upon any specific inputs. The next level allows one to ascertain the sign of the dependence. With this information it is possible to investigate the direction of variable changes which could lead to a hazardous event. Finally, the quantitative behavior of the output as function of the input constitutes the highest level of modeling of the equation describing the interactions. Although modeling of this level would undoubtedly lead to a large and complex program, the depth of the results might well justify such an effort.

As an example of the second level of modeling, an abbreviated matrix of coupled variables for a liquid/liquid countercurreent heat exchanger is shown in Figure 3.10. The sign indicates the change in the dependent variable for a positive change in the independent variable. With these models, it is possible to reveal a great deal about the safety behavior of a chemical processing system. The variable interactions and signs are sufficient for preliminary analysis. The sign and magnitude of the interaction are required for more detailed studies.

In order to perform the fault tree analysis, not only must models be generated for system performance as design, but models must be developed for describing failure modes.



Dependent Variables

	m1	m3	T2	T4	u
T1	0	0	+	+	0
T3	0	0	+	+	0
u	0	0	-	+	X
A	0	0	-	+	0
P1	+	0	+	+	+
P2	-	0	-	-	-
P3	0	+	-	-	+
P4	0	-	+	+	-

Fig.3.10 Variable coupling matrix for a liquid/liquid heat exchanger. (96)

In line with the modular approach, a failure model will be associated with each unit performance model. In terms of the descriptive framework for the unit models (steady state material, momentum, and energy balances), the failure modes may conveniently be classified for each unit as material, momentum, and energy balance failures. General failure modes for each category (material, Momentum and energy) are presented in Table 3.1. Note that the entries in Table 3.1 represent immediate causative factors, not necessary primary failures. The general failure modes provide the framework within which specific failure models for a particular unit (reactor separator, pump, etc.) may be formulated.

3.4.1.2 Fault Tree Generation

The general methodology for the generation of fault trees is the successive identification of predecessor events from the top node of the tree (the final hazard event) to the outermost nodes (the sequence-initiating or primal events). Fault tree generation starts with the definition of final hazard states. The first step in generating is then to obtain a mini-fault tree for the hazard in question. These fault trees have been defined for a wide range of possible hazards and are stored in the program's library. The fault tree for one type of explosion is shown in Figure 3.11. The fault tree indicates that for this type of explosion to occur, at the location in question, the correct

Table 3.1 General Failure Modes for Chemical Processing Equipmen

I. Material balance failures (species related)

   A. Routing--(wrong flow path)

   B. Flow

      1. Low

      2. High

      3. Wrong direction

   C. Leakage

      1. Internal

      2. External

   D. Species (not normally present in system or environs)

   E. Reaction

      1. Undesired side products

      2. Incorrect conversion

      3. Wrong location

      4. Wrong reaction

II. Momentum balance failures (pressure related

   A. Reaction

      1. Gas release

      2. Gas consumption

   B. Pressure sources

   C. Species (Vapor pressure)

   D. Pressure sinks

   E. Leakage

      1. Internal

Table 3.1 Cont'd

F. Phase changes

III. Energy balance failures (temperature related)

A. Reaction

1. Endothermic

2. Exothermic

B. Internal heat sources

C. Phase changes

D. External heat sources

E. Frictional heat sources (rotating equipment)

F. Leakage

1. Internal

2. External

G. Fouling of transfer surfaces

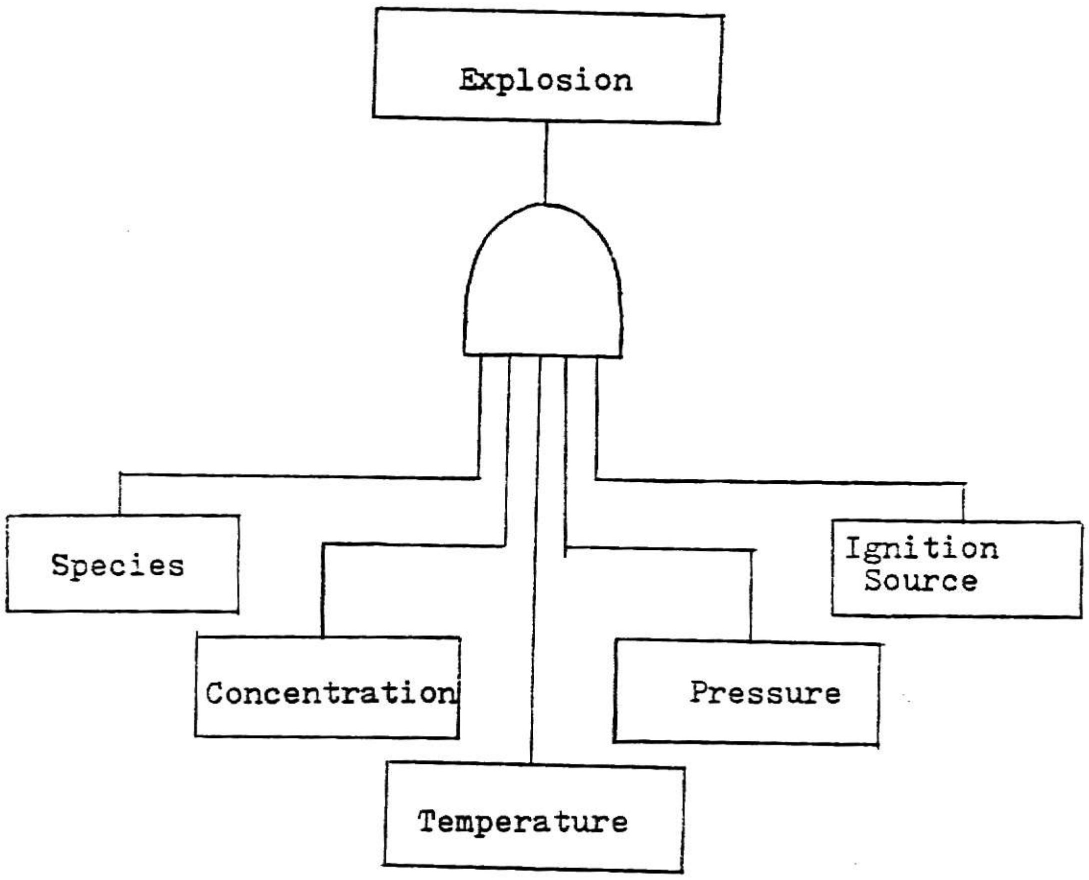


Fig. 3.11 Simplified hazard fault tree for an "Explosion"

species, concentrations, temperatures, pressures, and ignition source must be present. This version of the fault tree for explosion is simplified and more complicated versions can be utilized.

The top of the complete fault tree is hence defined by the logic required for the occurrence of the final hazard. The tree is then constructed downwards, as usual, by using the unit models to determine the species, temperatures and pressures required to produce the top. The sign of any dependencies can be obtained from the proper matrix, thus determining roughly the concentrations necessary. Then the species required can be trace backwards through the system to determine via what paths they could have arrived. This might involve both mechanical and chemical processes : opened valves, previous reactions to be backtracked etc.. Factors such as materials and age of components might be factored in, as well as the inclusion of data on reactivity of various chemical species under various conditions, and/or models to predict their behavior. During the investigation of these processes, the fault tree is being constructed, combining mini fault trees at each intermediate step as failure states are defined. Once the fault tree has been completed, it is in standard form employing AND and OR gates, and is readily evaluated by any of the fault tree evaluation codes described in chapters 4 and 5. The overal strategy for generating fault trees is shown in Figure 3.12.

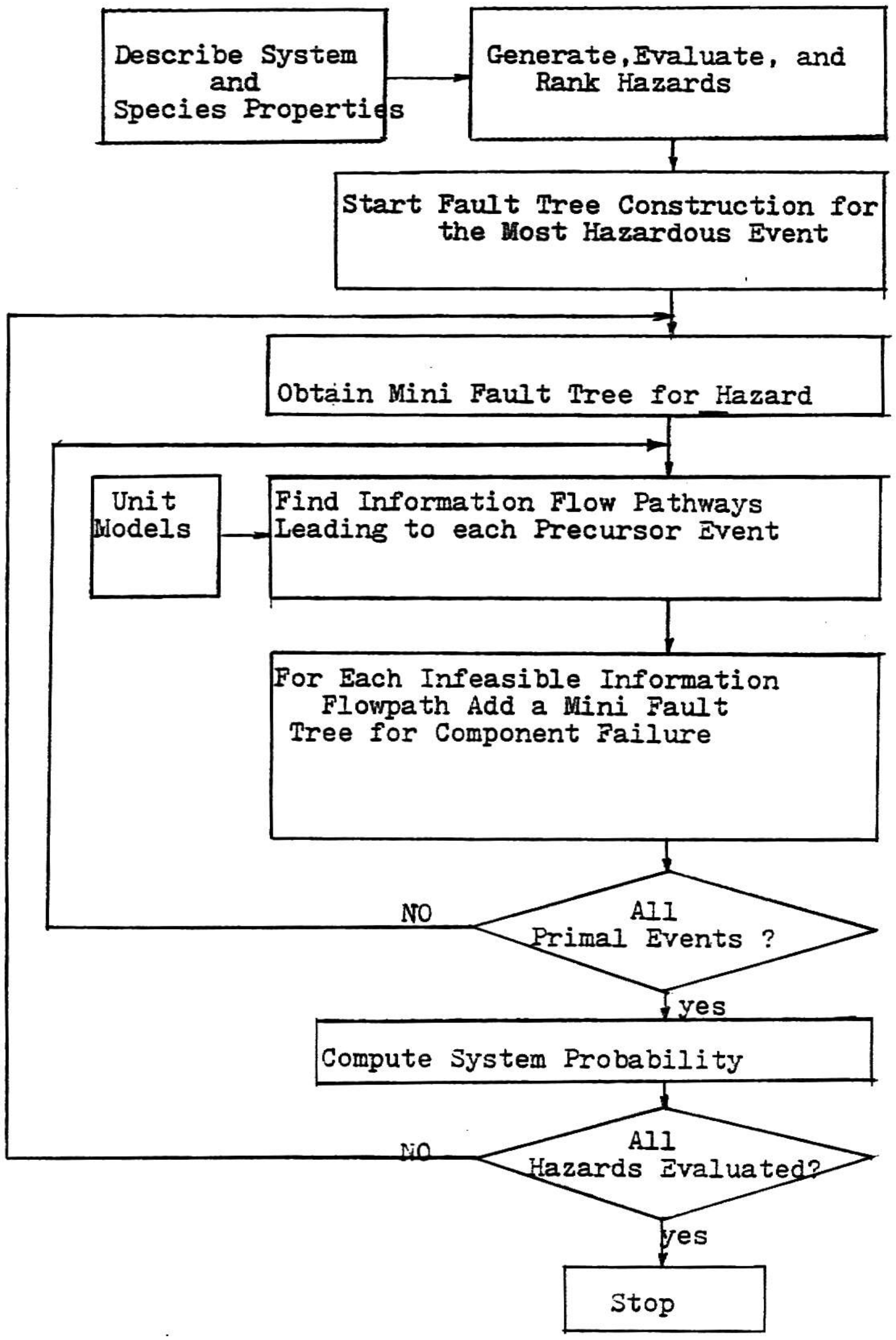

Describe System and Species Properties
Generate, Evaluate, and Rank Hazards
Start Fault Tree Construction for the Most Hazardous Event
Obtain Mini Fault Tree for Hazard
Unit Models
Find Information Flow Pathways Leading to each Precursor Event
For Each Infeasible Information Flowpath Add a Mini Fault Tree for Component Failure
NO
All Primal Events ?
yes
Compute System Probability
NO
All Hazards Evaluated?
yes
Stop

Fig.3.12 The overall strategy for generating fault trees. (96)

After definition of system hazards, a mini fault tree for the hazard is defined (see Figure 3.11). For precursor variables in the hazard, information flow paths for normal and failed states of the system are searched out. Mini fault trees which represent specific failure modes of equipment are used to bridge gaps in the information flow paths.

3.4.2 Example

Powers, Tompkins and Lapp developed a safety simulation language, called SESIL, for chemical processes by utilizing above discussed synthesis method (97). The language provides a general means for producing simulation models for a wide range of chemical processing systems. Boolean models which describe the safety performance of individual chemical processing units are linked together to describe the performance of complete processing systems. Data on properties of mateials and equipment within the process are used to formulate Boolean equations for potential system hazards (explosion, fire, overpressure, release of toxic material, etc.). These hazards are evaluated to determine their economic importance. The primal events (valve failure, power failure, etc.) which could cause each hazard are determined by symbolically solving the Boolean equations which simulate the process. This solution is a fault tree.

Given the process description and species data of the system, the first step in the analysis is to define potential hazards of interest for the process. Once these hazards are identified, they are converted into Boolean hazard equations. For example, it is known that butadiene and air will combine very rapidly under certain conditions to cause an explosion. These conditions are defined by the physics and the chemistry of this reaction. The usual conditions are :

1) butadiene and oxygen must be present;

2) the butadiene concentration must be above 2% (by volume) and less than 12%;

3) the phase must be vapor; and

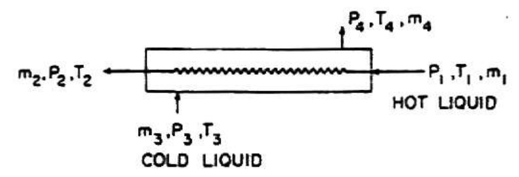
4) the temperature must be above 450 C or an ignition source present. These conditions can be formed into a Boolean hazard equation :

Explosion (at location X) = ((Butadiene Present) AND (Oxygen Present) AND (Concentration Butadiene greater than 2%) AND (Concentration Butadiene less than 12%) AND (Vapor Phase)) AND ((Temperature greater than 450 C) OR (Ignition Source))

This equation defines the top of the fault tree. The SESIL system contains over twenty types of hazard equations. These equations are directly linked to the property data file. The link is made so that the hazard equation describes the correct conditions required for the event.

The basis of the SESIL language is a set of unit models which describes how failures are initiated and propagated in each piece of equipment which might be found in a chemical process. These models are sets of Boolean difference equations which are derived directly from the steady state constitutive equations (mass, energy, and momentum balances) and from failure experience for the processing unit. An example is shown in Figure 3.13 for a simple counter-current shell and tube heat exchanger. The first set of equations describes how the output variables could deviate from their normal operating variables due to changes in input variables. The second set of equations describes how the output variables could deviate from their normal values due to events which are initiated within the heat exchanger. These models are formulated so that recycle of information, common mode failures, and human errors can be considered.

For the fault tree synthesis, the variables in the hazard equation (temperatures, concentrations, pressures, phases, etc.) are the same as the variables in the Boolean equations for the unit models. These unit models have been linked together to form a larger set of Boolean equations for the complete process. A fault tree is the Boolean equation which relates the hazard to the primal events in the system. Hence one can solve the large set of Boolean equations which describes the process, for the variables in the hazard equation; a fault tree is the result. The SESIL system has two means for solving these equations; one is a



OUTPUT EVENT INPUT EVENTS

$$\text{ENERGY BALANCE} \left\{ \begin{array}{l} T2+ = T1+ \;U\; T3+ \;U\; M1+ \;U\; M3- \\ T2- = T1- \;U\; T3- \;U\; M1- \;U\; M3+ \\ T4+ = T1+ \;U\; T3+ \;U\; M1+ \;U\; M3- \\ T4- = T1- \;U\; T3- \;U\; M1- \;U\; M3+ \end{array} \right.$$

$$\text{MOMENTUM BALANCE} \left\{ \begin{array}{l} P2+ = M1+ \;U\; P1+ \\ P2- = M1- \;U\; P1- \\ M2+ = M1+ \\ M2- = M1- \\ M4+ = M3+ \\ M4- = M3- \end{array} \right.$$

where U = logical "or" operator

$$\text{MASS BALANCE} \left\{ \begin{array}{l} C_2^i + = C_1^i + \\ C_2^i - = C_1^i - \\ C_4^i + = C_3^i + \\ C_4^i - = C_4^i - \end{array} \right.$$

FAILURE EVENT UNIT FAILURES REQUIRED

$M2+ = EL_H^* \cap (P_H < P_E) \;U\; IL^* \cap (P_H < P_C)$

$M4+ = EL_C^* \cap (P_C < P_E) \;U\; IL^* \cap (P_C < P_H)$

$T2+ = EF^* \;U\; RI_H \;U\; RI_C \;U\; PL_C^* \;U\; PHS_C \;U\; EL_C \cap (P_E < P_C)$

$T4+ = EF^* \;U\; RI_H \;U\; RI_C \;U\; IL^* \cap (P_C < P_H)$

$P2+ = EL_H^* \cap (P_H < P_E) \;U\; IL^* \cap (P_H < P_C) + R3_H$

$P4+ = EL_C^* \cap (P_C < P_E) \;U\; IL^* \cap (P_C < P_H) + R3_C$

$C_2^i + = IL^* \cap (P_H < P_C) \cap (C_H^i < C_C^i) \;U\; EL_H^* \cap (P_H < P_E) \cap (C_H^i < C_E^i) \;U\; RP_H^i$

$C_4^i + = IL^* \cap (P_C < P_H) \cap (C_C^i < C_H^i) + EL_C^* \cap (P_C < P_E) \cap (C_C^i < C_E^i) \;U\; RP_C^i$

where U ≡ OR ; ∩ ≡ AND

EF ≡ EXTERNAL FACTOR (FIRE, WEATHER, ETC.)	PL ≡ PLUGGING
EL ≡ EXTERNAL LEAK	RI ≡ EXOTHERMIC REACTION
IL ≡ INTERNAL LEAK	R3 ≡ GAS RELEASE REACTION
PHS ≡ SOLID PHASE FORMATION	RP^i ≡ REACTION WHICH GIVES SPECIES i

H ≡ HOT STREAM	C ≡ COLD STREAM	E ≡ ENVIRONMENT
i ≡ SPECIES i	* ≡ PRIMAL EVENT	

FIG. 3.13 Boolean performance and failure model for a counter-current heat exchanger. (97)

path-finding method and the other is based on the d-algorithm for finding faults in a logic circuit. The detail of the algorithm is discussed in (97).

3.5 Salem et al's Computer-Oriented Fault Tree Construction

3.5.1 Introduction

The current approach of Salem, et al. (104-106) is a general computer oriented method of modeling complex systems of mechanical, electrical and hydraulic components, allowing for human interactions and common cause effects as well. This method is based on the use of decision tables for component modeling (107). These tables are used to describe each possible output state of a component as a complete set of combinations of states of inputs and internal operational or failed states. Given a method of describing the specific system configuration including initial system states, and a means of defining a top event of interest, the decision table models are used for the appropriate components within the system and are combined and edited to form a complete fault tree for the top event desired. The methodology has been implemented by the development of a computer code, called CAT, which, following several stages of editing, produces a fault tree in conventional format.

3.5.1.1 Component Modeling : Method of Decision Table

For the purpose of constructing fault trees, component models in terms of sub-fault trees, as used in the other automated approaches (51,96), would seem to be an immediate solution. However, a method which would involve a simple, tabular scheme, readily usable by those unfamiliar with fault tree analysis, would be especially appropriate. Moreover, a numerical rather than logic model might suggest alternate construction schemes, and possibly avoid some of the complications of other approaches.

Decision tables are an extension of truth tables in that they allow any number of states to be used for each entry in the table. Although the binary logic of the truth table is very simple, it is often insufficient for use in a complex system. For example, a three-state variable might be required to describe the water pressure in certain water-injection systems (no pressure, low pressure, and high pressure).

The decision table, then, would be used to describe each possible output state as a complete set of combinations of states of inputs and internal operational or failed states. As an example, a complete decision table for the fuse is shown in Table 3.2. We can see that :

Table 3.2 Complete Decision Table for Fuse

Row No.	Input State	Internal Mode	Output State
1	0	0	0
2	0	1	0
3	0	2	0

4	1	0	1
5	1	1	0
6	1	2	1
7	2	0	0
8	2	1	0
9	2	2	2

Signals : 0 = no signal, 1 = normal, 2 = overload Internal Modes : 0 = good, 1 = failed open, 2 = failed closed (shorted)

1) State 2 occurs at the output ih only one way : input overload and fuse failed shorted (row 9).

2) Output state 0 ocurs whenever the input is 0, regardless of the internal mode. Thus, the first three rows can be replaced by one row : 0 - 0, where the '-' means that the center column has no effect on the output.

3) State 0 also occurs whenever the fuse fails open, regardless of the input state. Thus, row 2,5 and 8 can be replaced by one row : - 1 0, showing that the input has no effect if the fuse has failed open.

Table 3.3 shows the result of using 2) and 3) to simplify Table 3.2. Now, we can consider the extension of these reduction methods to general, multi-state decision tables. In order to produce a 'don't care' entry, we must :

1) Search for rows with identical output states. If a

component has more than one output, all outputs must be checked.

2) For rows with identical output states search for those rows which agree in all but one column.

3) A 'don't care' situation then occurs if, in the remaining column, every possible state is represented by one of the rows found in step two. Thus, eliminate all but one of the rows and include a '-' state in the appropriate column of the one remaining row. One of the key points in this procedure is that in general only input and internal states can be reduced to 'don't care' states.

Now we will see how the reduced decision tables would be used in practice to generate fault trees. For the first step, some desired output state is necessary, for example, the event 'no output from fuse'. This might be the top event of a tree, or some intermediate event that would be required to produce a zero input to a succeeding component.

TABLE 3.3 REDUCED DECISION TABLE FOR FUSE

Row no.	Input state	Internal mode	Output state
1	0	-	0
2	-	1	0
3	1	0	1
4	1	2	1
5	2	0	0
6	2	2	2

Given the desired output state, a search is made for rows with the correct state, in this case row 1, 2 and 5 of the completely reduced table, Table 3.3. Since any of these rows has the correct output, they are connected by an OR gate, each row being a single input. Now the three input events must be developed. Row 1 has only a single defined state, 'no input to fuse', in addition to a 'don't care' state. Thus, row 1 is replaced by that single event. Row 2 has a 'don't care' input, and internal mode of 1 (failed open). This is a primary failure and thus becomes a direct primary input event. The development of row 5, however, indicates a more general situation. Here there are two states defined, both of which must be true for the output state to be zero. The result is an AND gate with the two appropriate inputs. The fault tree for this development is shown in Figure 3.14.

3.5.1.2 Definition of System

The method of decision tables has allowed the modeling of the operation and failure of each individual component in a system. Now it remains to determine a method of coupling components and tracing through the system in order to construct the fault tree. This can be done by defining a set of 'nodes' and their corresponding system states throughout the system.

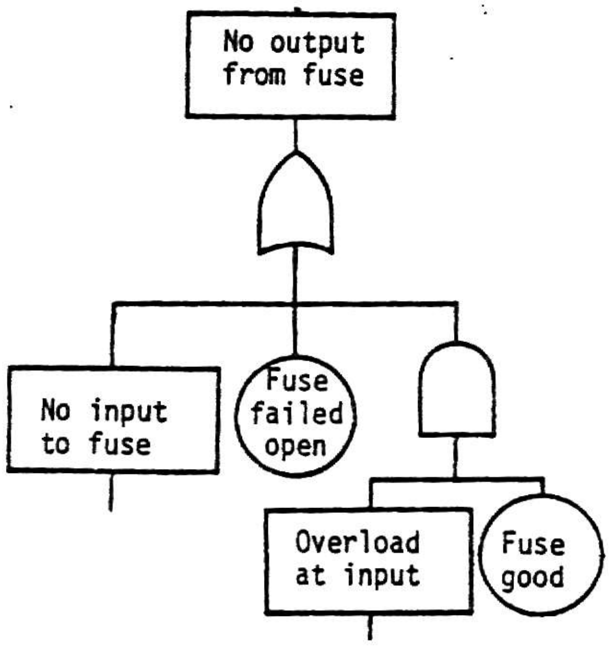


Figure 3.14 Sub-Tree for Event: "No Output from Fuse"

A node will be defined as any point in the system at which the output of a component is connected to the inputs of one or more succeeding components. That is, a node or junction represents the point of interconnection between component inputs and outputs. A system state which consists of a specific condition of the system will be defined at the node. Thus, in constructing the tree, all events being traced at a particular moment must be compatible with system states already defined. When a specific event is being developed, that event (a system state) is first checked and then set as a fixed state at the appropriate node, and all subsequent events are checked against the system states at the corresponding nodes. The concepts of system nodes and system states, in combination with the use of decision tables, will now be illustrated by a short example. Figure 3.15 shows a single power supply connected to two components, one directly, and one through a fuse. The numbers 1 through 4 indicate the system nodes, which could be identified systematically by the following scheme :

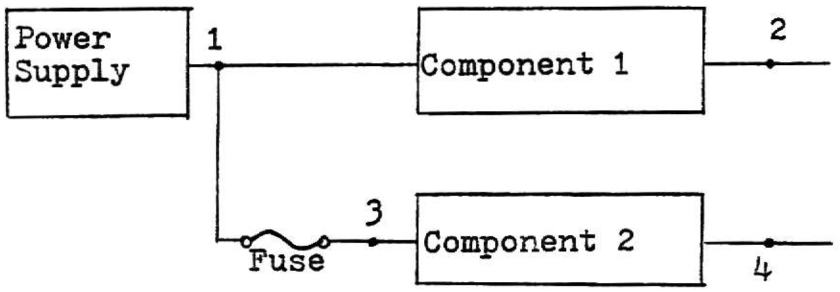


Fig. 3.15. Node numbering scheme for sample system.

Power Supply : Type = 101, Output = 1

Component 1 : Type = 102, Input = 1, Output = 2

Component 2 : Type = 102, Input = 3, Output = 4

Fuse : Type = 103, Input = 1, Output = 3

This is essentially all the input required by a code such as CAT, and defines both the interconnections of all input and output nodes, and specifies the type of each component. For example, Node 1 is defined as the output from the power supply, and input to both component 1 and the fuse. The type numbers would assign specific decision tables to each component and, in this example, would indicate that Component 1 and 2 were identical, modeled by the same decision table.

In addition to system states which are defined as the tree is generated, other states may have been defined as initial or boundary conditions. Although these terms represent slightly different concepts, their effct is the same, and the term boundary condition will be used for both. Both are states of the system which are defined initially and continue to exist throughout the entire fault tree; quite often they will be used to qualify the top event description. Meanwhile, the top event is essentially a set of boundary conditions which have the added function of starting the construction process.

3.5.1.3 Editing Concerns

In the CAT code, the editing is divided into three phases :

1) Editing while in the process of constructing a gate. This includes checking events for consistency with pre-defined system states and deleting them as required.

2) Removing excess, redundant, or contradictory gates or events after each gate or group of gates has been completed.

3) Final editing after the tree has been constructed. This includes checking for transfers, renumbering gates and other operations.

In the editing during construction of a gate, the basic concern is that no events be developed, or allowed to exit, which contradict existing events. The method of editing is based upon the utilization of component decision tables in conjunction with the system states as already defined.

The intermediate or 'post-gate' editing will take place each time the final branch of a gate has been completed. This editing generally consists of two phases. First is the elimination of single input gates which have resulted from the deletion of other inputs during previous editing steps.

Then follows a search for events which may have been rendered redundant or contradictory due to the removal of the single input gates.

Once the top gate of the fault tree has been completed and intermediate editing accomplished, the tree itself is finished and is in standard form suitable for further analysis.

3.5.2 Example

In order to demonstrate some of the techniques described above, an example tree will be constructed manually. The simple electrical circuit used will be in Figure 3.16. S1 and S2 will be signals, defined to be in existence as boundary (initial) events. 'P.S.' will be a power supply closing SW3, the relay switch defined by Table 3.4. The decision table for the power supply will be defined simply by power supply good gives signal out, power suplly failed produces no signal.

We will develop the decision table for switch 1 and 2. These will be standard two position (on/off) switches which will have two internal mechanisms : position (1 = off, 2 = on), and mechanical (0 = good, 1 = failed open, 2 = failed closed). For no signal input, we can immediately realize that there will be no signal output, regardless of the position or mechanical state. The rows remaining to be

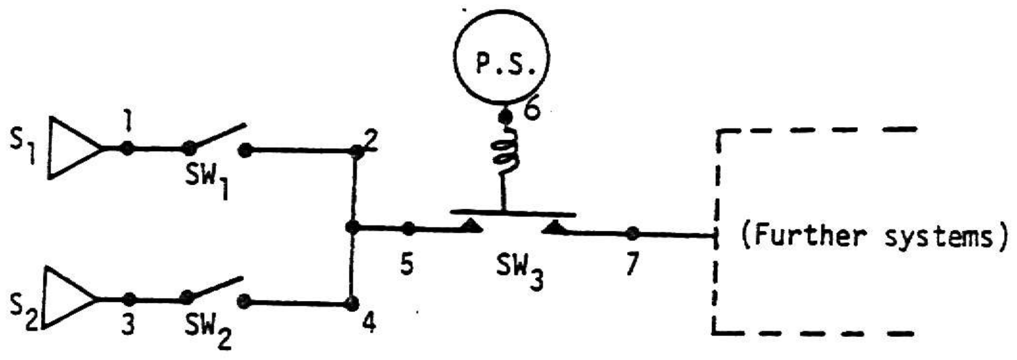


Fig 3.16 Sample System

TABLE 3.4 DECISION TABLE FOR RELAY SWITCH

Row No.	Signal Input	Coil Input	Internal Coil	Internal Contact	Output Signal
1	1	-	-	2	1
2	1	1	0	0	1
3	1	-	2	0	1
4	0	-	-	-	0
5	-	-	-	1	0
6	-	0	-	0	0
7	-	-	1	0	0

considered, then, will all begin with a signal input state. There will be 6 possible rows corresponding to the 2 x 3 = 6 possible combinations of internal states, and the result is Table 3.5. By applying the reduction rules of Section 3.5.1.1, the final reduced decision table for switches 1 and 2 is shown in Table 3.6. The junction which connects the outputs of switches 1 and 2 to the input of switch 3 is the OR gate of Table 3.7. That is , switch 3 receives a signal if either switch 1 or switch 2 transmits a signal.

Having labeled the system nodes in Figure 3.16, we now proceed to define the top event as no signal at point 7, which might actually be an inmtermediate event as required by some larger system, as shown schematically to the right. Furthermore, let us define switches 1 and 2 as being on, in addition to signals 1 and 2 being present.

We begin generating the tree with the TOP event which specifies node 7, the output from switch 3. Using decision Table 3.4 for the relay switch, we search for a zero output state, finding rows 4, 5, 6 and 7. Row 4 requires a zero signal input (node 5), row 5 has a contact failed open, row 6 requires no input to coil (node 6) and a good contact, while row 7 needs a coil failed open and good contact. That is :

Table 3.5 Decision Table for Switch 1 and 2

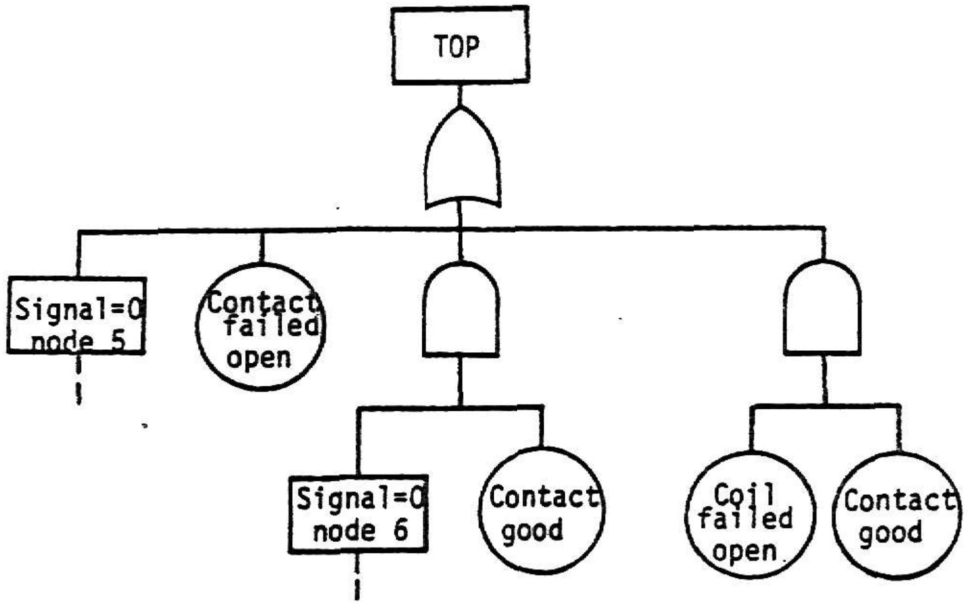
Row	Signal Input	Position	Mechanical	Output
1	0	-	-	0
2	1	1	0	0
3	1	1	1	0
4	1	1	2	1
5	1	2	0	1
6	1	2	1	0
7	1	2	2	1

TABLE 3.6 DECISION TABLE FOR ON/OFF SWITCH

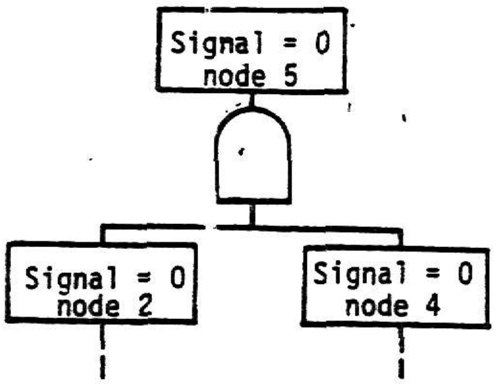
Row	Input	Position	Mechanical	Output
1	0	-	-	0
2	-	1	0	0
3	-	-	1	0
4	1	-	2	1
5	1	2	0	1

TABLE 3.7 DECISION TABLE FOR OR GATE

Input 1	Input 2	Output
0	0	0
1	-	1
-	1	1

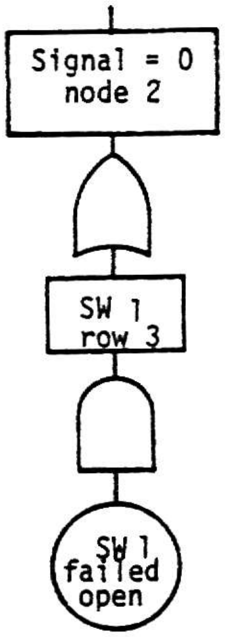


To evaluate branch 1, we note that node 5 is the output from the junction (OR gate), and from Table 3.7, no output requires no signal at both inputs (nodes 2 and 4). This produces :

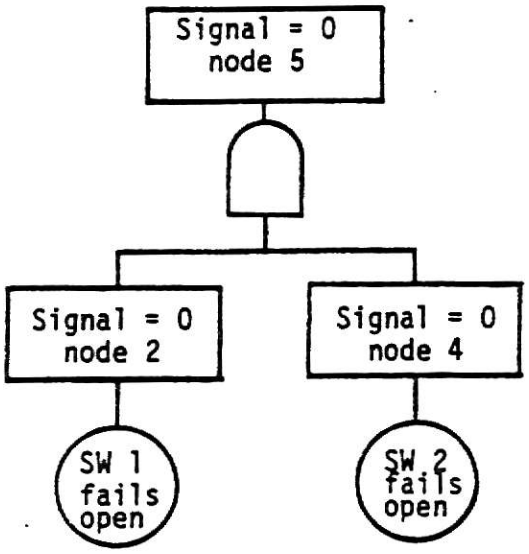


Tracing the left branch, node 2 is output from switch 1. Table 3.6 has three rows with an output of zero. Row 1, no signal at switch input (node 1), contradicts the initial condition signal at point 1. Row 2 requidres the switch to be off, which again contradicts an initial condition. Row 3,

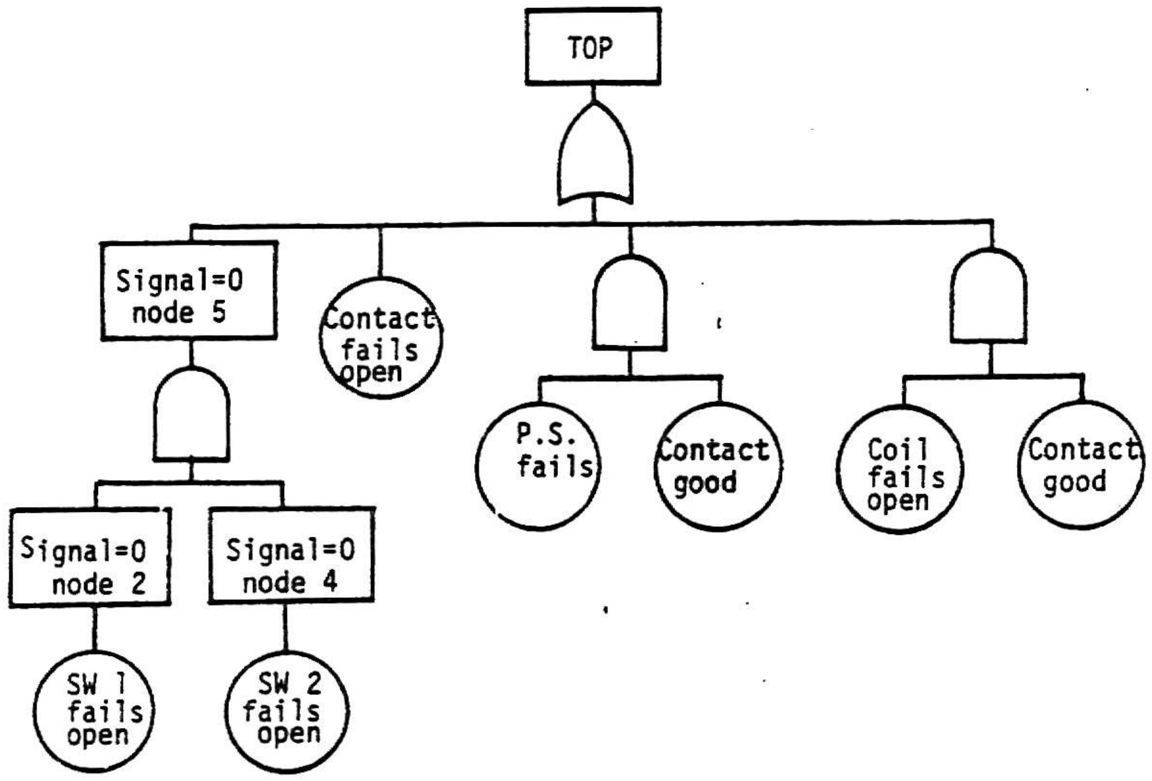
(switch failed open), however, is valid, and is the only row left, thus leaving the following :



These two single input gates, however, reduce to the single primary input itself. Furthermore, we develop the event signal = 0 at node 4 in the exact same manner, but for switch 2. The resulting branch 1 of the top gate is :



The final remaining branch is signal = 0 at node 6. Node 6 is the output from the power supply, and the event no output produces the single primary event power supply fails. Thus , the entire tree is :



This is the final form of the fault tree, unless it is desired to remove the event contact good. The approach may be employed if it is assumed that the probability of contact failure is sufficiently small (P<<1). In this case, the event contact good is defined as almost sure to occur and is deleted from the two AND gates. The gates, then, are both left as single input gates, and their inputs are inserted directly into the top OR gate. The final form of the fault tree for sample system is shown in figure 3.17.

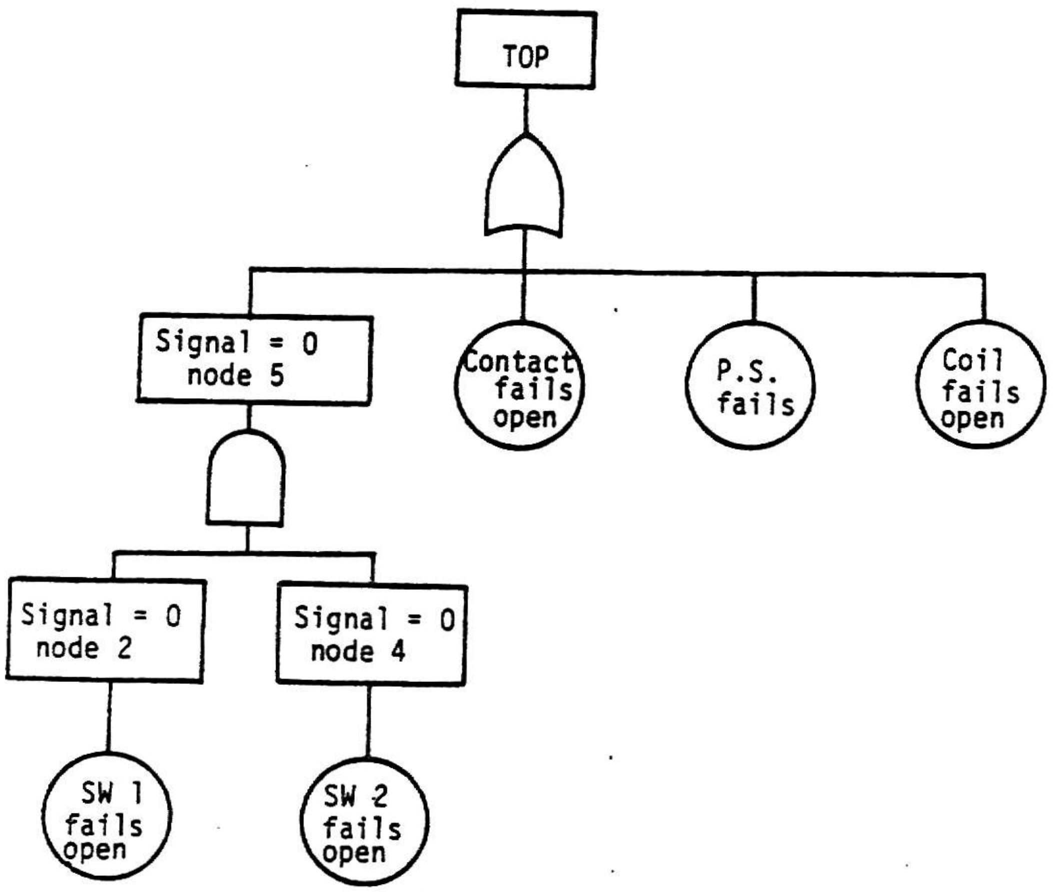


Fig 3.17 Final Tree for Sample System

3.6 Lapp and Powers' Computer-aided Synthesis of Fault Trees

3.6.1 Introduction

In order to synthesize fault tree automatically, it is first necessary to develop a satisfactory representation of the system under study. This representation must be general so that any type of process may be analyzed. At the same time, it must be suited for ease of computer processing. Lapp and Powers' synthesis methodology (80) uses a digraph (directed graph) as system representation to satisfy the needs. The digraph describes the normal, failed, and conditional relationships which exist between variables and events in the system. The fault tree is deduced directly from a digraph model of the system being analyzed. A computer program was developed to perform the fault tree algorithm.

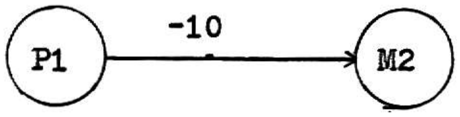
3.6.1.1 Process Representation-Digraphs

A digraph is a set of nodes connected by directed edges. The nodes of digraphs used in fault tree synthesis represent process variables and certain types of failures. In the case of chemical systems, these would be temperatures, pressures, flow rates, etc.. Relations among the various nodes are embodied in the edges connecting them. If a deviation in one variable cauase a deviation in a second variable, then a directed edge is drawn from the node representing the first variable to the node representing the second. A number is assigned to the edge depending on the direction and magnitude of the second deviation relative to the first. If a moderate deviation in the first variable causes a moderate deviation in the second, a value of '1' is assigned to the edge, on the other hand, if the second deviation is very large compared to the first, a value of '10' is assigned. If the second deviation is very small compared with the first, no edge connects the nodes. The sign of the number reflects the relative direction of the deviations. If they are in similar directions, the number is positive, otherwese it is negative.

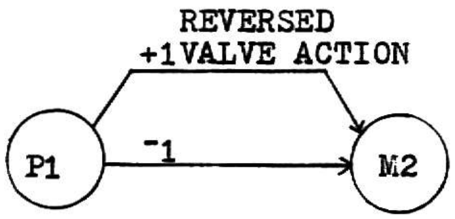
As an example, consider a control valve with spring action AIR TO CLOSE. We wish to represent the relationship between the air pressure on the valve (denoted by P1) and the flow path rate of fluid through the valve (M2). Since a positive deviation in P1 causes a negative deviation in M2, the diagraph is as follows :

$$\begin{tikzcd} \boxed{P1} \arrow[r, "-1"] & \boxed{M2} \end{tikzcd}$$

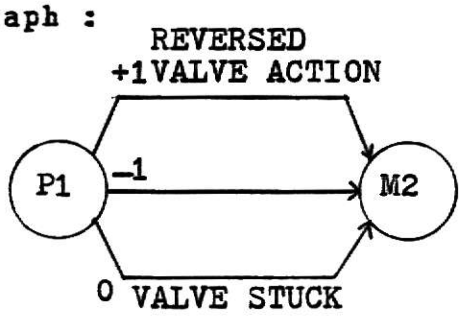
Suppose the valve has quick-closing characteristics. Then a positive deviation in P1 causes a very large negative deviation in M2, and the digraph is the following :



The number associated with the edge can be interpreted as a partial deviative ($\partial M2/\partial P1$) and is termed the gain between the variables. Thus far, we have concerned ourselves only with the usual relationship between variables. How do we model failures which alter these usual relationships? One answer is simply to include additional edges between the variables that represent the failed gains. Consider the valve again and suppose we wish to include the failure REVERSED VALVE ACTION. In this case, increasing P1 will increase M2. The following digraph embodies the additional failure :



In addition, if we wish to include the failure VALVE STUCK (increasing P1 has no effect on M2), then we would use the following digraph :



Edges with zero gains are drawn only if these edges represent failures. If zero-gain represents the normal relationship, no edge is drawn between the nodes.

The rules for constructing the system digraph from the component digraphs are :

1) start at the top event (output) variable.

2) get the component model from which the variable is the output

3) Work backwards through the component models to its inputs assembling the system digraph. (Do not trace any of the output variables to other variables at this time.)

4) For each input variable on the resulting digraph, repeat step 3 until variables are encountered which have no inputs (system boundary or failure modes).

5) The variables in the component digraph models are labeled by their location in the system flow diagram. External variables in the same location are labeled as the same variable. Components manufactured by the same company are given the same variable for certain failure modes. This leads to discovery of common-cause situations. Extension of the analysis to common systems such as power supply, instrument air, or a single operator results in numerous common variable situations. These are deduced directly from

the interconnections shown in the flow diagram.

6) If loops exist in the process, it is possible to pass through the same component digraph twice. The same rules given in steps 3 and 4 should be followed. Do not trace variables that have already been developed.

7) Variables that are conditions on edges are developed in the same manner as input variables.

3.6.1.2 Fault Tree Synthesis Algorithm

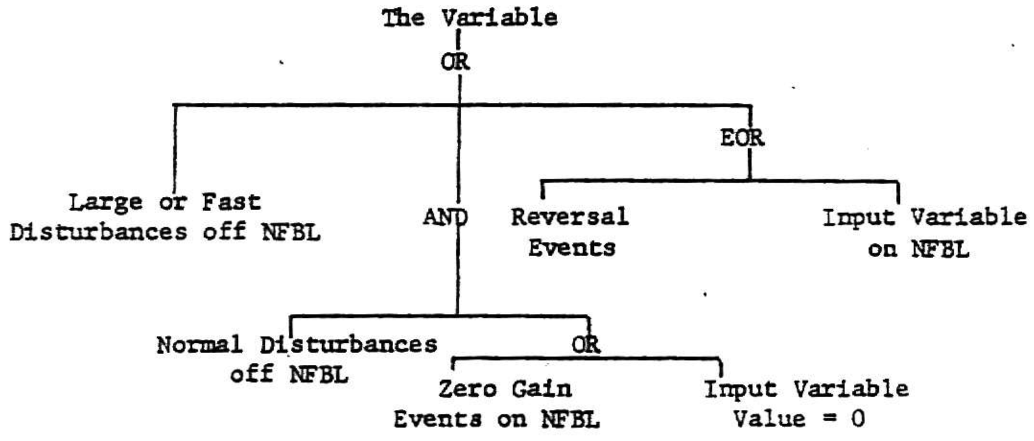
The process description is in the form of a digraph after the system digraph was constructed. The task now is directly deduce the fault tree from the digraph. This is accomplished by a synthesis algorithm which is based on local conversion of the digraph variables, events, and edges into a partial development of the fault tree. In order to do this, three basic conversions of digraphs into fault tree are performed :

1) They are based on whether negative feedback loops or negative feedforward loops pass through the digraph variable.

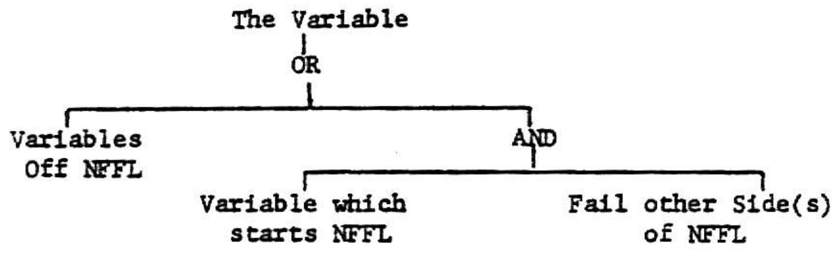
2) They depend on the value of the deviation of the output variable. The range of operation of the loops is used to select the operator.

3) The fault tree 'operators' are shown in FIgure 3.18. They are applied recursively until all nodes in the digraph have been developed.

FOR NEGATIVE FEEDBACK LOOP VARIABLES



FOR NEGATIVE FEEDFORWARD LOOP VARIABLE JUST BEFORE START OF LOOP



OTHERWISE

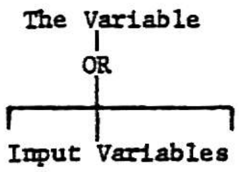


Fig 3.18 Boolean Expressions for Digraph Variables (81)

The consistency of loop variables are checked against conditions developed in the tree. (e.g. previous conditions and the domain of AND gates.)

A general algorithm is described as follows :

1) Generate digraph and find all negative feedback and feedforward locps.

2) Select node representing top event.

3) Determine local cause of this event by noting the inputs to the node of the digraph.

4) Delete any local causes which violate consistency.

5) Select the appropriate operator depending on whether negative feedback or feedforward loops pass through the current node. Use this operator to connect logically the remaining local causes. If negative feedback or feedforward loops are involved, store the appropriate event for later consistency checks.

6) Select a node corresponding to an undeveloped event and return to step 3. If only primal events remain, stop.

3.6.2 Example

Consider the flow control loop shown in Figure 3.19. The system senses the flow and adjust the valve position to maintain flow at the set point value. The controller may be placed on manual. When on manual the controller

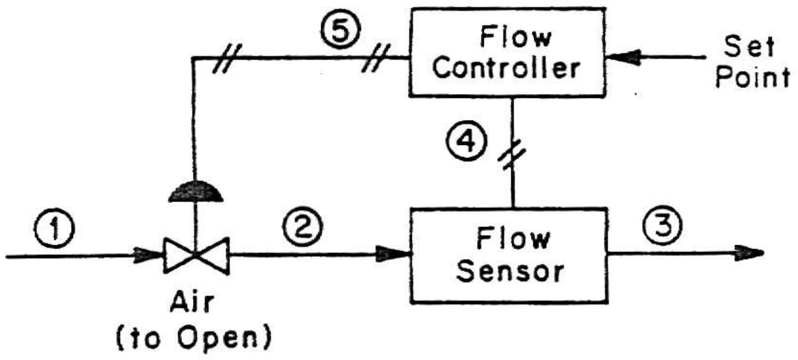


Fig 3.19 Flow Control System

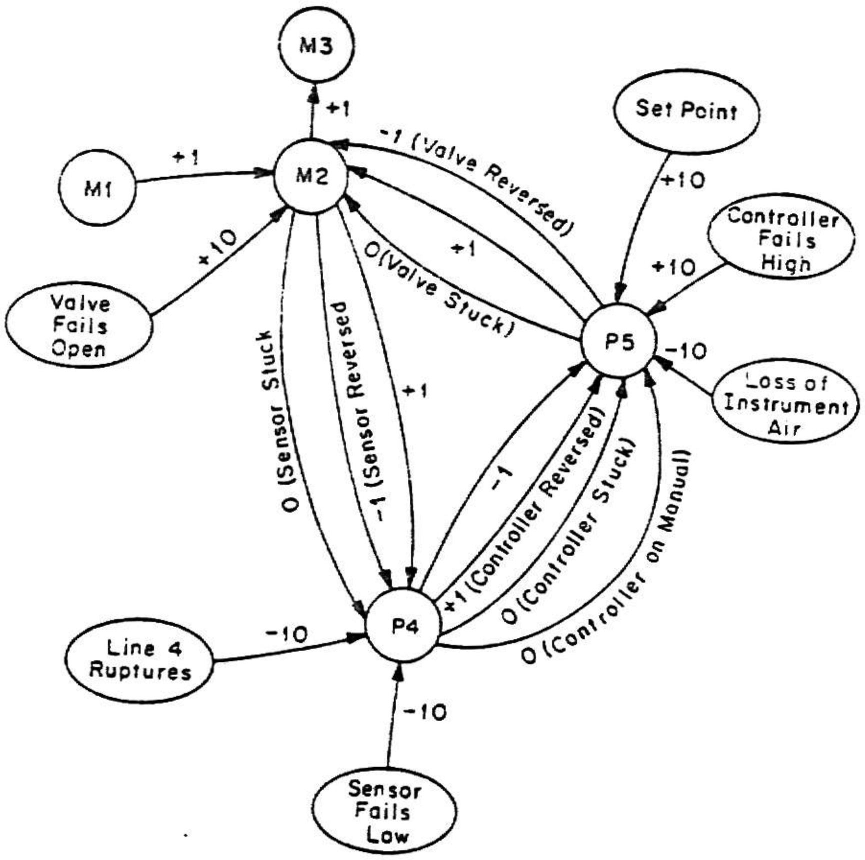


Fig. 3.20 Digraph for a Flow Control System (81)

maintains the output of the controller at its last value. Changes in the input to the controller do not cause changes in the output when in manual operation. The controller may also be made reverse acting by either explicitly changing the action by activating a switch on the contrcller or by increasing the loop gain or dead time (or reset rate or derivate rate if integral or derivate action is present in controller). The digraph for the flow control system is shown in Figure 3.20. The digraph was contructed by starting at the TOP event (in this case flow out of the system M3) and working backwards through the digraph models for each component in the system. The procedure is illustrated in section 3.6.1.1.

The algorithm of fault tree synthesis is then, as described in secticn 3.6.1.2, based on the logical (AND, OR, EOR) combinations of digraph variables (and their values) which could cause a particular deviation in an output variable. The Boolean expressions for a digraph variable given in Figue 3.18 were based on a detailed study of how negative feedback loops and negative feedfoward loop might fail. The negative feedback loop Boolean expression has three major terms. The leftmost branch shown in Figure 3.18 indicates that a large or fast deviation in an input variable to a negative feedback loop will pass through the loop. That is, an OR gate is indicated. The second term (center branch) denotes the fact that if a normal deviation in an input variable enters a negative feedback loop that it is also necessary (AND) to fail the feedback loop by inactivation. The third term (right branch) of the Boolean expression for a negative feedback loop indicates that the loop can directly cause the deviation in output variable by becoming a positive feedback loop. As noise is always present in these loops we assume that reversal of the loop is all that is necessary to cause the loop to go unstable. However, if two reversals occur they cancel each other and the loop remains negative. Hence the exclusive OR gate indicates on reversal event or another but not both.

The negative feedforward loop Boolean equation has two major terms (branches). The leftmost branch indicates that if the disturbance variable entering the loop does not send signals down all sides of the negative feedforward loop, the loop will not cancel out the disturbance (hence an OR gate). The right hand term indicates that if a deviation in an input varible activates all sides of the negative feedforward loop. It is necessary to have the disturbance AND fail the other sides of the loop.

If the digraph variable is not on any negative feedback or feedforward loops, use an OR gate.

The fault tree derived from the flow control digraph (Figure 3.20) is shown in Figure 3.21. Gates 3, 4, and 5 are due to the feedback loop operator.

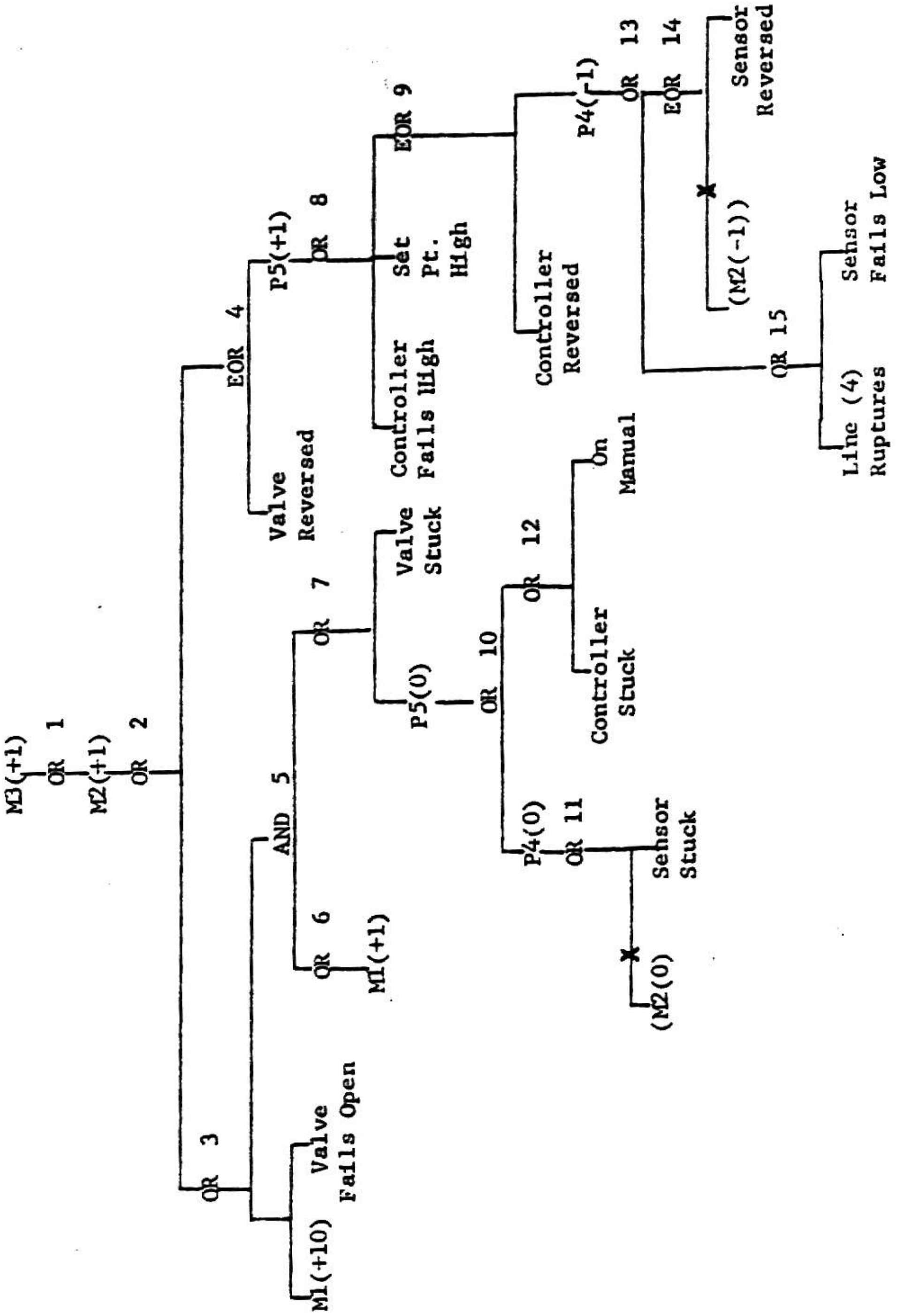


Fig 3.21 Fault Tree for a Flow Control System (81)

3.7 Camarda et al's Efficient Algorithm of Fault Tree Synthesis from the Reliability Graph

3.7.1 Introduction

The fault tree automatic synthesis introduced by Camarda et al (27), deals with 2-state systems which have a considerable number of components. The approach begins with the probabilistic graph of the system, i.e., a graph which shows all possible ways of correct system operation, and results in a suitable form of fault tree that can be used in any available method of numerical evaluation. The general algorithm for this approach is described as follows:

1) Represent the system by means of its reliability graph G. Choice of terminal node S & T corresponds to selecting the particular system failure of interest (TOP event).

2) Investigate all minimal tie-sets between S & T in G. To perform this, a simple method is employed for removing the nodes of the reliability graph one by one and resulting in a transmission function F* which is the union of all the tie-sets between S & T.

3) Simplify F*, eliminating all nominal tie-sets, and obtain F as a function of all the minimal tie-sets. The indempotence and absorption law of Boolean algebra are used here.

4) Logically invert the S-O-P (sum of product)

expression of F by means of DeMorgan's law yielding F as a P-O-S (product of sum) expression.

5) Order the minimal tie sets according to the number of their components and then gather those components that are common to the greatest number of minimal tie-sets. This will lead to the final multilevel expression, F (no longer in P-O-S form) fcr the fault tree.

3.7.2 Example

The system has reliability block diagram and corresponding graph as shown in Figure 3.22a and b.

1) The branches a, b, d, are unidirectional; c, e are bidirectional. Nodes 1 and 4 are the input and output nodes respectively.

2) Remove node 2 from the graph; the modified graph is shown in Figure 3.23a. Remove node 3; this is the final result and is shown in Figure 3.23b. The tie-sets of the system are :

a.b, c.d, c.e.b, a.e.d, a.e.c.b

3) Simplify the nonminimal tie-sets; the minimal tie-sets are :

a.b, c.d, a.e.d, c.e.b

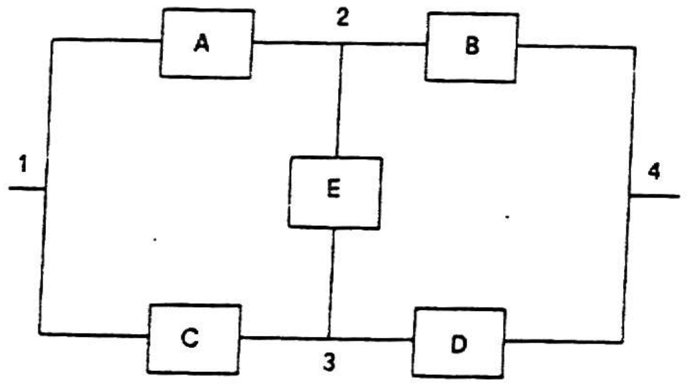
F in the S-O-P expression is :

$F = a.b + c.d + a.e.d + c.e.b$

4) Logically invert F

$\bar{F} = (\bar{a} + \bar{b}).(\bar{c} + \bar{a}).(\bar{a} + \bar{e} + \bar{d}).(\bar{c} + \bar{e} + \bar{b})$

(a)



(b)

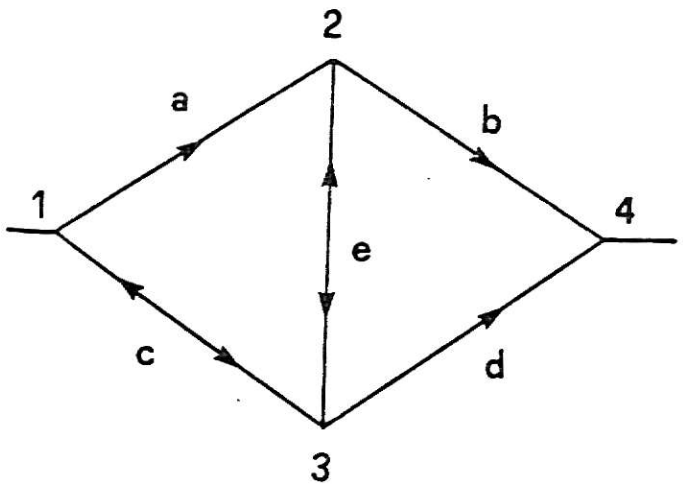
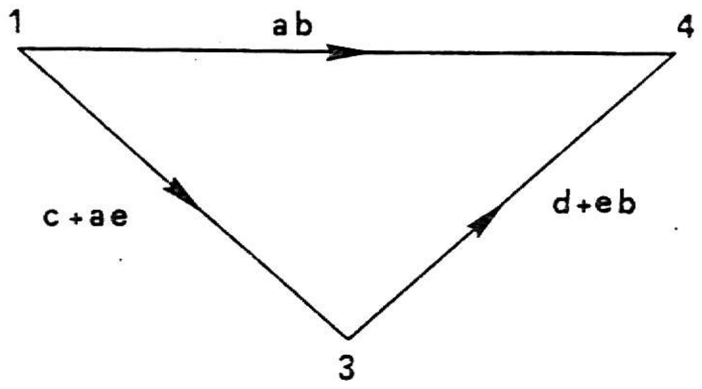


Fig.3.22 Reliability Block Diagram and Reliability Graph (27)

(a)



(b)

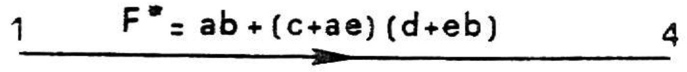


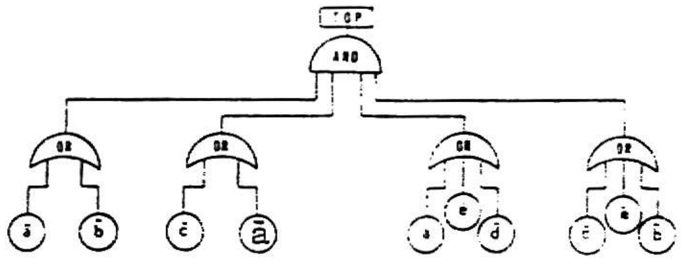
Fig.3.23 Reliability Graph after Removal of (a) node 2 (b) node 2 and 3 (27)

5) Factor $\bar{a}$ from factors 1 and 3 in above equation, and $\bar{c}$ from factors 2 and 4.

$\bar{F} = (\bar{a} + \bar{b}.(\bar{e} + \bar{d})).(\bar{c} + \bar{d}.(\bar{e} + \bar{b}))$

This form corresponds to a 4-level representation of the fault tree, as shown in Figure 3.24a. Figure 3.24b is a two-level fault tree obtained from expression (4).

(a)



(b)

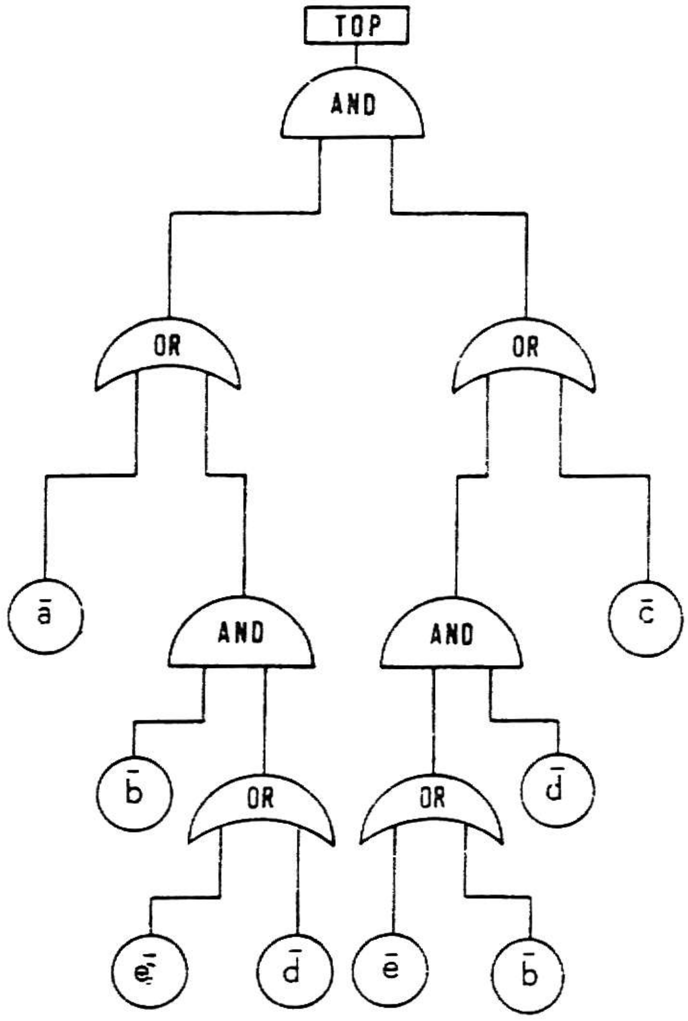


Fig. 3.24 (a) Two Levels Fault Tree for the Example Considered

Fig. 3.24 (b) Multilevel Fault Tree (27)

3.8 Concluding Remarks

In order to reduce the cost and time of adequate fault tree construction and to avoid oversights of some failure sources, automated treatment is required. On the other hand, it has some disadvantages, e.g. human errors and environmental effects usually cannot be considered, but it can be a rapidly executed initial procedure, to be followed by a more detailed fault tree analysis.

Generally the automatic fault tree development requires three main steps :

1) to find a proper system or component modeling method which is suitable for computer programming;

2) to develop an algorithm for fault tree construction;

3) to implement these algorithm cn computers.

Up to the present, algorithms published on automated fault tree construction are rather limited. As for the methodologies discussed above, Hassl's structuring process (67) marks a starting point of fault tree construction technique. No computer program was provided at this stage. Fussell (51) pioneered the work of automatic construction with his Synthetic Tree Model (STM). Features were :

1) primarily for electrical system;

2) logic models used as component transfer functions;

3) Discriminator flags used to ensure internal consistency;

4) Computer code produced.

Powers and Tompkins (96) concentrated primarily on defining a methodology for synthesis of process-plant fault trees. The basic theme was similar to Fussell's STM except that a functional model was defined which showed the interactions between process variables. No computer code was produced.

Salem et al.(105) produced the computer Automated Tree code (CAT). Features are :

1) Decision table models contains multiple input and output states suitable for non-binary systems.

2) The code can synthesize trees containing AND gates. Earlier codes do not do this.

Lapp and Powers (80) produced the Fault-Tree Synbthesis (FTS) code. Features are :

1) A 2-step method bases on constructing a fault tree from a digraph which represents the system interactions. A computer program is now being developed which will aid in generating the digraph;

2) Multiple-state values considered for both nodes and links;

3) Automatic detection of feedback and feedforwqrd loops and use of this information in the synthesis.

Nevertheless, some controversy exists concerning this approach. A number of papers have been written commenting on various aspects of the L & P algorithm (70,78,82.83).

Camarda et al(27) proposed an efficient algorithm for fault tree automatic synthesis from the reliability graph. The reliability graph representation is, for large systems, generally much easier to obtain than the fault tree, because the ways in which a physical system can operate are much fewer than those in which it can fail. However, this approach can only handle those complex 2- state, non series-parallel systems, and no computer code was provided.

Finally Taylar and Hollo(116) use algebraic component models to construct a Cause-Consequence diagram (CCD) (91). The Cause-Consequence Diagram is the most comprehensive representation and has recently received widespread interest as a method for reliability and safety analysis of complex systems. It extends the fault tree methodology to better describe the sequential effects of accident chains and to increase their visibility in the analysis procedure. The CCD methods is based upon the fact that the paths from several independent fault events pass to their consequences through focal nodes representing 'focal events', which very often have been identified during plant design. The 'focal events', therefore, will generally release som -limiting action. By selecting expedient 'critical events' as 'focal events' a cause search, as well as a consequence search, will be facilitated in a systematic way.

The cause diagram of CCD is the same as the conventional fault tree. The fault tree (cause diagram) construction algorithm suggested by Taylor and Hollo (116) is similar in principle to those described by Fussell (51) and by Powers and Tompkins (96). It works by tracing the course of events backwards through a component network. At each stage it takes a given event established in the fault tree and looks among component transfer functions for a combination of an input event and component variable conditions, which can lead to the already established event. For detailed explanation, consult references (113-117).

There has been considerable interest in the use of computer codes for fault tree construction because it is a complex and time-consuming task. The work on fault propagation and its representation has reached a point where the basic methodology is reasonably well understood and tested. However, there are a lot of difficulties in fault tree synthesis which still remain unsclved (4). The published algorithms are not yet of sufficiently high quality for general use. Criteria need to be established so that algorithms can be evaluated.

CHAPTER FOUR

QUALITATIVE EVALUATIONS OF FAULT TREES

4.1 Introduction

The fault tree can be used as a visual medium in communicating and supporting decisions based on the analysis. Either the analyst or the management can inspect a fault tree and determine by engineering judgement the most likely sets of basic events leading to the top event. A qualitative judgement can be made regarding the safety of the system and the identification of critical system elements if the system is to be upgraded. A qualitative evaluation can also take into account many practice considerations and assumptions that at times may be difficult to incorporate in quantitative calculations. Moreover the qualitative evaluation usually serves as an intermediate step to the quantitative evaluations. The results of a qualitative evaluation, however, are less manageable due to the subjective nature of decisions based on qualitative judgement. Certain qualitative procedures that were developed as a part of fault tree analysis are presented in this chapter. They are divided into two categories : finding minimal cut sets path sets, and common cause failure analysis.

4.2 Minimal Cut Sets / Path Sets

The first step in a qualitative evaluation is to determine the minimal cut sets and the minimal path sets of the fault tree. A cut set is a set of basic failures which are collectively sufficient to cause the undesired event. If all the numbers of a cut set are not only sufficient, but also necessary to cause the undesired event, then such a cut set is called a minimal cut set. A path set is a set of basic successes which are collectively sufficient to assure the undesired event. If all the numbers of the set are not only sufficient, but also necessary to assure such a success, then such a path set is called a minimal path set. The minimal path sets of a fault tree are the same as the minimal cut sets of the dual fault tree, which is constructed by interchanging the AND and OR logic gates and changing the basic failure to basic successes in the orginal tree.

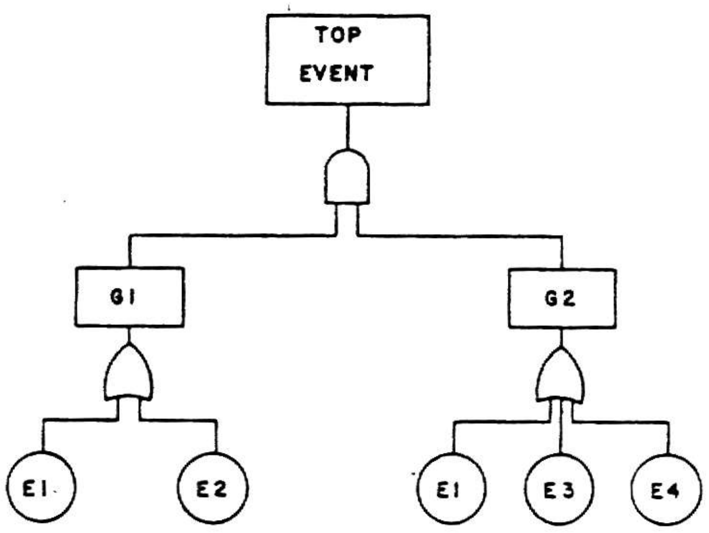


Fig.4.1 Fault tree example

The minimal cut sets of simple fault trees may be obtained by inspection. Consider the fault tree given in Figure 4.1. By inspection the following combinations of basic events (cut sets) are found to cause the top event:

CS1 : E1, E1

CS2 : E1, E3

CS3 : E1, E4

CS4 : E2, E1

CS5 : E2, E3

CS6 : E2, E4

All the above cut sets, however, are not minimal cut sets and the supersets have to be eliminated. This also may be accomplished by inspection.

In CS1, the event E1 is duplicated, and it may be reduced to E1. CS2, CS3 and CS4 are supersets of CS1. CS5 and CS6 are neither supersets of CS1 nor of each other, and so they are retained. Thus, the minimal cut sets of this simple fault tree are :

MCS1 : E1

MCS2 : E2, E3

MCS3 : E2, E4

Large complex fault trees involve hundreds of gates and events, and result in thousands of cut sets. An inspection procedure to determine the cut sets and path sets is not practical. In such cases systematic computer-aided procedures are necessary to reduce fault trees. Two methods used for computer reduction are Monte Carlo simulation and deterministic methods.

4.2.1 Monte Carlo Simulation

The Monte Carlo simulation procedure for finding minimal cut sets/ path sets may be summarized as follows :

Step 1: generate n independent random numbers R (i=1,2,....,n), uniformly distributed between 0 and 1. (n is the number of basic events)

Step 2: if $0 < r < P$ (j=1,2,....n), the jth event E is triggered where P is the probability of occurrence of E

.

Step 3: decide if the combination of the triggered events caused the top event.

Step 4: if the top event occurs, store the combination of the triggered events; this is a cut set. If the top event does not occur discard the combination of the triggered events. Step 1 to Step 4 constitute a trial.

Step 5: after a sufficient number of trial are performed, compare the stored cut sets with each other and retain only the minimal cut sets.

In order to illustrate the above procedure, consider the fault tree shown in Figure 4.1. Let the probabilities of occurrence of the basic events by:

P1 : 0.4

P2 : 0.55

P3 : 0.5

P4 : 0.6

Ten trials are conducted, and their outcomes are given in Table 4.1. The cut sets are given in the last column. After eliminating the supersets, the minimal cut sets found are :

MCS1 : E1

MCS2 : E2, E4

Comparing these minimal cut sets with those obtained by inspection, it becomes evident that one of the minimal cut sets, namely, (E2, E3) was missed by the simulation procedure. If, however, more trials are conducted, the above minimal cut set also may be obtained. This is illustrated in Table 4.2 in which ten more trials are conducted. From the total of 20 trials, all the cut sets are obtained.

In the above example, the fault tree is simple and the event failure probabilities are high. Therefore, all the minimal cut sets are obtained in just 20 trials. But most practical problems involve large fault trees and small failure probabilities, and therefore thousands or even millions of trials may be required to determine the dominant

TABLE 4.1 (61)

TRIAL	r_1	r_2	r_3	r_4	E_1 $P_1 = ·4$	E_2 $P_2 = ·55$	E_3 $P_3 = ·5$	E_4 $P_4 = ·6$	TOP	CUT SET
1	·98275	·21216	·47006	·25488	N	Y	Y	Y	Y	E_2 E_3 E_4
2	·93487	·17776	·25025	·48054	N	Y	Y	Y	Y	E_2 E_3 E_4
3	·77638	·33408	97394	·82975	N	Y	N	N	N	—
4	·13771	·85457	·84468	·68318	Y	N	N	N	Y	E_1
5	·82422	·07865	·17314	·41471	N	Y	Y	Y	Y	E_2 E_3 E_4
6	·79394	·90396	·82875	·98281	N	N	N	N	N	—
7	·15063	·18243	·81691	·61175	Y	Y	N	N	Y	E_1 E_2
8	·30831	·21955	·55957	·47123	Y	Y	N	Y	Y	E_1 E_2 E_4
9	·61512	·24620	·44361	·14503	N	Y	Y	Y	Y	E_2 E_3 E_4
10	·52270	·15054	·92854	·25843	N	Y	N	Y	Y	E_2 E_4

Y EVENT OCCURS
N EVENT DOES NOT OCCUR
THE RANDOM NUMBERS ARE TAKEN FROM RANDOM TABLE

TABLE 4.2(61)

TRIAL	r_1	r_2	r_3	r_4	E_1 $P_1=.4$	E_2 $P_2=.55$	E_3 $P_3=.5$	E_4 $P_4=.6$	TOP	CUT SET
11	.69103	.84932	.06307	.78502	N	N	Y	N	N	—
12	.73582	.08117	.71713	.91705	N	Y	N	N	N	—
13	.91009	.72733	.93842	.34805	N	N	N	Y	N	—
14	.57300	.28176	.51982	.81231	N	Y	N	N	N	—
15	.92344	.93644	.55786	.81226	N	N	N	N	N	—
16	.92749	.68882	.60100	.89024	N	N	N	N	N	—
17	.42831	.45484	.12684	.97088	N	Y	Y	N	Y	E_2 E_3
18	.07745	.02923	.63854	.53621	Y	Y	N	Y	Y	E_1 E_2 E_4
19	.45498	.87859	.31865	.32266	N	N	Y	Y	N	—
20	.09182	.98937	.44177	.56282	Y	N	Y	Y	Y	E_1 E_3 E_4

Y EVENT OCCURS
N EVENT DOES NOT OCCUR
THE RANDOM NUMBERS ARE TAKEN FROM RANDOM TABLE

cut sets. However, this problem may be overcome by incorporating an appropriate importance sampling procedure (88) in the simulation. Vesely's PREP code (125) used the Monte Carlo simulation with importance sampling in finding the minimal cut sets for fault trees. The method has the feature that the most important minimal cut sets, those most likely to occur, are found first.

However, the Monte Carlo method does have its disadvantages :

1) even for a qualitative analysis, basic failure probabilities are needed.

2) the method does not guarrentee the determination of all the minimal cut sets.

In spite of these drawbacks, it is still a very powerful technique, and is capable of analyzing complex fault trees involving logics which are not amenable to deterministic methods.

4.2.2 The Deterministic Approach

The basic idea behind the deterministic approach is direct expansion or reduction of the top event of a fault tree in terms of the constituent basic event using Boolean algebra. In the example given in Figure 4.1, it is easy to show :

TOP EVENT = G1 + G2

$$= (E1 + E2) \cdot (E1 + E3 + E4)$$

$$= E1 + E1\ E3 + E1\ E4 + E1\ E2 + E2\ E3 + E2\ E4$$

The right hand side of the equation gives the cut sets of the top event. The elimination of the supersets reduces the above equation to the minimal cut sets. The Boolean expansion given above is from top down. The cut sets could also have been determined by Boolean reduction of the tree from bottom up.

Determination of the cut sets of large fault trees would be impractical without the use of a computer. Several computer codes for finding minimal cut sets have been developed in past two decades.

4.2.2.1 PREP (COMBO option)

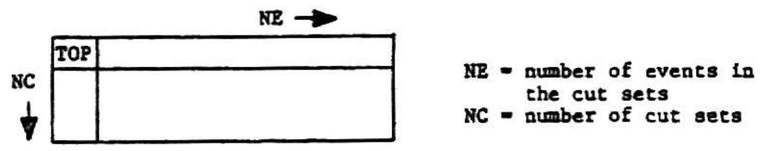
One of the earliest computer programs using the deterministic methods is the PREP program developed by Vesely and Narum (125). The program, except the Monte Carlo option (FATE) described above, uses a direct, combinatin testing algorithm (COMBO) for its deterministic approach. First, one by one the basic events are tested to see if they could cause the undesired event. Those that could do so are one event minimal cut sets. Then the two-event combinations (e.g., E1 and E2, E2 and E3, E1 and E3, etc.) are tested. In the two-event combination testing, the events which are already found to be one-event minimal cut sets are not included, since any combination containing them could only be a superset. The procedure is repeated with three-event combinations and so on, until all the possible combinations are tested. The user may stop the testing after a specified level of combinations, say two-event combination, are tested. In such a case the algorithm will give all the minimal cut sets to that level.

Though the combination testing algorithm is straight forward and gives all the minimal cut sets directly, the number of combinations to be tested becomes unmanageably large in a large-sized fault tree. For example, the number of two event combinations for a tree with 2000 basic events is about 2 million (2000!/2!(2000-2)!=1999000). The number of three event combinations for the same fault tree is over 1.3 billion. Testing such numbers of combinations can be prohibitive. Hence the combination testing algorithm is not suited for large trees having high redundancy.

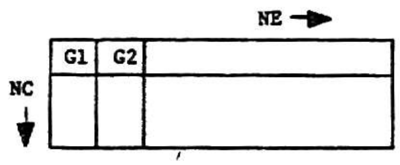
4.2.2.2 MOCUS

Fussel and Vesely (60) developed an alternate algorithm which does not require the combination testing. It is based on the fact that an AND gate always increases the order of the cut sets (number of events in the cut set), and an OR gate always increases the number of cut sets. The cut sets obtained by the algorithm are called Boolean Indicated Cut Sets (BICS). The BICS are not necessarily minimal cut sets, and hence they have to be compared with each other and the supersets eliminated. Though the determination of the BICS by this algorithm is relatively fast, the elimination of the supersets may require considerable amount of computer time. Fussell, Henery and Marshall (56) used this algorithm in their fault tree analysis program, MOCUS. The same algorithm was independently conceived by Diven, Griffing, Thorpe and Van Slyke (118) and was used in their program ALLCUTS. ALLCUTS produces not only the minimal cut sets, but also the probabilities of occurrence of each, which can be used to gauge their relative importance. The top event probability, however, must still be computed separately.

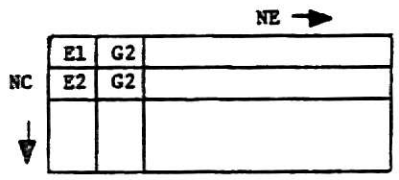
The determination of the BICS may be illustrated by analyzing the sample fault tree of Figure 4.1. The solution is started by forming a matrix with the top event at the first location :



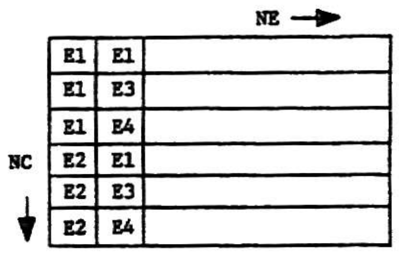
Next the top event is replaced by the gates and events below it (here, G1 and G2). Since it is an AND gate, it would increase the number of events in the cut sets. So G1 and G2 are entered in the first row of the matrix.



Then the gate G1 is replaced by the events below it, namely E1 and E2. Since G1 is an OR gate, it would increase the number of cut sets. So E1 and E2 are entered in two different rows. The gate G2 which accompanied G1 in the matrix above, should accompany both E1 and E2 in the new matrix.



Next the gate G2 is replaced by the gates and events below it.



Since there are no more gates, but only basic events in the matrix, the procedure is complete. The final matrix indicates that there are six BICS, namely :

CS1 : E1, E1

CS2 : E1, E3

CS3 : E1, E4

CS4 : E2, E1

CS5 : E2, E3

CS6 : E2, E4

The supersets may now be eliminated by comparing the cut sets with each other.

4.2.2.3 MICSUP

MICSUP (93), however, is a program that works from the bottom up. The algorithm starts with lowest level gates, those with only components as inputs. The cut set for a low level AND gate is the set of all inputs. For a low level OR gate, each input component become a separate cut set. The process continues for higher level gates, i.e., those with only gates or gates and components as inputs. The cut sets for a higher level AND gate are formed by taking the union of all combinations between the input sets. The cut sets for OR gates are all the set inputed. Taking again as an example the tree in Figure 4.1., the process is as follow :

```
step 1 : G1 (E1) (E2)
         G2 (E1) (E3) (E4)

step 2 : Top Event (E1, E1)
                   (E1, E3)
                   (E1, E4)
                   (E2, E1)
```

(E2, E3)

(E2, E4)

The final step in the process is reduction. This step may be performed after each gate, or, to obtain the BICS, only after the top gate. The sets from the previous example are BICS. Through reduction, we obtain the minimal cut sets.

Step 3 : Top Event MCS1 : E1

MCS2 : E2, E3

MCS3 : E2, E4

The important merit of the upward algorithm over its predecessor, MOCUS, a downward method, is the ability to give the minimal cut sets of any intermediate gate without further processing. This allows the user to break up large trees into smaller, more manageable segments. Another merit is the efficiency of MICSUP over MOCUS in both time and space requirement. Having the cut sets at each gate, allows reduction which eventually reduces processing time by removing extraneous sets. Also knowing from TREEL, a preprocessing program, the number of times a gate is replicated, allows reduction in storage space by removing all gates below the current processing level which do not appear again. A mathematical derivation and proof of MICSUP can be found in Chatterjee's paper (32).

4.2.2.4 ELRAFT

Semsnderes (110) introduced the concept of prime number representation of basic events for reduction of fault trees. This concept is useful in storing the cut sets and eliminating the supersets. The idea is to assign a unique prime number starting from 2 for each basic event. The cut sets are represented by the product of the prime numers of the basic events in the set. Each cut set is therefore stored as a unique number. Since this product can be factorized uniquely to the component prime numbers, the components of the cut sets are readily obtained. The prime number representation also makes the reduction of the cut sets to minimal cut sets easy. The reduction procedure consists of two steps. First, the cut set numbers (product of the basic event prime numbers) are reduced to the product of distinct prime numbers only, thus eliminating any duplicate events in the cut sets. Second, if a cut set number is divisable by another cut set number, the latter is eliminated as a superset of the former. Semanderes used this prime number manipulation technique in his ELRAFT program. This technique also forms the basis for Wong's FAUTRAN program (131).

The storing of cut sets and the elimination of supersets by prime number representation may be illustrated via the example fault tree of Figure 4.1. First the four basic events are represented by the first four prime number starting from 2, i.e.,

E1 -- 2

E2 -- 3

E3 -- 5

E4 -- 7

G1 and G2 are OR gates and, therefore,

G1 = 2, 3

G2 = 2, 5, 7

The combination of these events through the AND gate G1, gives the following results :

CS1 : E1, E1 -- 2 x 2 = 4

CS2 : E1, E3 -- 2 x 5 = 10

CS3 : E1, E4 -- 2 x 7 = 14

CS4 : E2, E1 -- 3 x 2 = 6

CS5 : E2, E3 -- 3 x 5 = 15

CS6 : E2, E4 -- 3 x 7 = 21

Elimination of the duplicate events by reducing the cut set numbers as the product of distinct prime numbers yields,

CS1 : E1 -- 2

CS2 : E1, E3 -- 10

CS3 : E1, E4 -- 14

CS4 : E2, E1 -- 6

CS5 : E2, E3 -- 15

CS6 : E2, E4 -- 21

The supersets are eliminated by discarding those cut sets that are divisable by others. For example :

CS2/CS1 = 10/2 = 5 (integer)

Hence CS2 is a superset of CS1, and so discarded. Similarly CS3 and CS4 may also be eliminated. So the minimal cut sets are :

MCS1 : CS1 -- 2 =E1

MCS2 : CS5 -- 15 = 3 x 5 = E2, E3

MCS3 : CS6 -- 21 = 3 x 7 = E2, E4

The advantages of the prime number representation are :

1) the cut sets may be stored as a single number, instead of a number of basic events.

2) the superset eliminations are accomplished by simple mathematical operations.

Again, this algorithm begins at the bottom of the tree, working upwards, forming intersections and unions of events, as appropriate for the gate types encountered.

4.2.2.5 SETS

In a coherent fault tree, the logic gates are restricted to AND and OR gates and the minimal cut sets can be obtained easily by the usual fault tree methodologies described above. The trees of complex systems, however, often include gates other than simple AND and OR, e,g., EOR and NOT. Cut sets do not apply since the trees no longer have monotone properties, i.e., they become non-coherent. As stated in a stateof-the-art discussion on fault trees (58), the minimal cut set concepts, under this situation, should be replaced by a set of literals in a prime implicant in Boolean algebra. The set can be called a prime implicant set.

The SETS computer code (133) developed by Worrell, is designed to find the prime implicants to a fault tree. The prime implicants are like minimal cut sets except that they may contain complemented basic events. The Set Equation Transformation System (SETS) (132) allows the generation of set equations directly, or by logical combination of other set equations through a process of substitution. The execution of the program is carried out in three major steps. First, it will be necessary to read an input representation of a fault tree, establish an equation for each intermediate event as a function of its input events, and then make these equations available for further processing. Next, we shall want to change the form of an equation using a substitution process that allows us to control the literals that will occur in the equation. In particular, we must be able to replace any literal in the right part of an equation with the right part of the equation for that literal, if it exists. Finally, it must be possible through the application of identities to reduce any equation that has been generated, and ultimately to determine the set of all prime implicants for it.

Take the sample fault tree in Figure 4.1 for example. We first obtain the set equations for all intermediate events.

$$G1 = E1 \vee E2$$

$$G2 = E1 \vee E3 \vee E4$$

$$\text{Top Event} = G1 \wedge G2$$

Now, we start the substitution process. As the right part of the equation is processed from left to right, each literal is replaced by the right part from its equation, if one exists. Thus, we create the following Boolean expression for the top event.

$$\text{Top Event} = (E1 \vee E2) \wedge (E1 \vee E3 \vee E4)$$

Then, the right part of the equation for the top event is expanded by the distribution law into a disjunctive normal form, while at the same time applying the identities $X \wedge X = X$ and $X \vee X \wedge Y = X$. This will result the sum of the prime implicants that we seek :

$$\text{Top Event} = (E1 \wedge E1) \vee (E1 \wedge E3) \vee (E1 \wedge E4) \vee (E2 \wedge E1) \vee (E2 \wedge E3) \vee (E2 \wedge E4)$$

$$= E1 \vee (E1 \wedge E3) \vee (E1 \wedge E4) \vee (E2 \wedge E1) \vee$$
$$(E2 \wedge E3) \vee (E2 \wedge E4)$$
$$= E1 \vee (E2 \wedge E3) \vee (E2 \wedge E4)$$

And the minimal cut sets are obtained from the final expression of the top event shown above.

4.2.2.6 Other Algorithms and Computer Codes

New algorithms and computer codes for finding minimal cut sets/path sets of fault trees are continuously being developed by many industrial users and academic and research institutions. A computer program FAITREE (85) developed by the Foster Wheeler Development Corporation uses the concept of binary bit manipulation to generate the minimal cut sets. Since digital computers are very efficient in binary operations, the representation of events in binary bit strings and their reduction by the use of binary logic operation inherent in the computer give a substantial reduction of computer time.

Rasmuson and Marshall (98) also developed the code FATRAM for determining the minimal cut sets. The method, which was developed to make more efficient use of computer memory, is a top-down algorithm similar to MOCUS. The gates are resolved in a deterministic manner according to the following rules : 1) AND gates and OR gates with gate inputs are resolved first, and 2) OR gates with only basic event inputs are resolved last. When comparing its computer memory requirements and the execution time with MOCUS, the FATRAM algorithm does show the adventage on saving computer core memory.

The DICOMICS code, developed by Garribba et al, (62) is based on segmenting the original tree into a fully equivalent forest of subtrees and on constructing, rather then searching, minimal cut sets for the tree starting from minimal cut sets of subtrees. The major advantage of the method is the direct determination of minimal cut sets up to any order requird. There are no size or complexity limitations and processing time is almost independent of size of the tree, number of minimal cut sets, and maximum order of minimal cut sets.

There are numerous algorithms for obtaining prime implicant sets of a Boolean function. These algorithms can be applied to non-coherent fault trees. The drawbacks of the algorithms are that they assume initially a sum of product or a product of sum expression of the top event in terms of the primary events. These expressions involve a large number of terms, and their reduction generally requires a large number of operations or large computer memories. Kumanoto and Henley (74) presented a top-down algorithm for obtaining prime implicant sets of non-coherent fault trees and avoided sum of product expressions of top event. Some discussions and controversy regarding the algorithm are found in (84).

Finally, Nakashima and Hattori (90) proposed an efficient bottom-up algorithm for enumerating minimal cut sets of fault trees. This method aims to improve the conventional bottom-up algorithm so as to obtain all minimal cut sets more quickly. The improvement is to reduce the number of checks of redundant terms for the logical product of two reduced sum-of-oriduct forms. A computer program BUP-CUTS is produced.

4.3 Common-Cause Failure Analysis

Common cause failure analysis, also called common mode failure analysis, is an integral part of a complete system safety and reliability analysis. A common cause failure is any occurrence or condition that results in multiple component failures. Epler (44) reported that 'the common mode failure may be dominant by as much as a factor of 10'. Taylor (114) reported on the frequency of common cause failures in the U.S. power reactor industry. 'Of 379 component failures or groups of failures arising from independent causes, 78 involved common mode failure of two or more components.' The reported dominance and frequency of common cause failure events illustrates the need for an effective approach to common cause failure analysis.

In the context of fault tree analysis, common cause failure analysis deals with identifying the mechanisms that are external to the system elements and can cause simultaneous failure of a number of elements or paths. In the context of a command fault, we are concerned with system interface conditions that result in an unrecognized dependence on a control element. This means identification of human as well as hardware functional interdependencies. In the context of secondary failure, we are concerned with unforeseen environmental or operational stresses that can simultaneously fail two or more system elements.

A current approach to computer aided common cause failure analysis consists of five basic steps (126) :

1) Determine the list of hardware minimal cut sets for the system being analyzed.

2) Obtain the qualitative failure characteristics for each basic event in the minimal cut sets.

3) Search the complete list of hardware minimal cut sets for common cause condidates using a computer.

4) Include the probabilistic effects of the common cause condidates in the quantitative system analysis.

5) Form conclusions and recommendations based on the results of the qualitative and quantitative analysis.

Two programs have been developed for common cause failure analysis using minimal cut sets as input; COMCAN (21), developed at INEL, and BACFIRE (30), developed at the Univrsity of Tennessee. A new procedure for automated common cause failure analysis of complex system is also proposed by Wagner et al. (126). The approach locates common cause failures without examining all the minimal cut sets which are usually enormous and often an impossible task for a large fault tree.

4.3.1 COMCAN

Strictly, common cause failure is any occurrence or condition that results in multiple component failures. For fault tree analysis the term secondary failure will be used to identify the categories of component malfunction pertinent to common cause failure analysis. A significant common cause event is a cause of secondary failure that is common to all basic events in one or more hardware minimal cut sets. If a minimal cut set has a significant common cause event and, if all the components represented by the bsic events in that minimal cut set share a 'common physical location.' that minimal cut set is called a common cause candidate.

A large number of secondary failure causes can be found that would cause component failures. Therefore the analyst is directed toward the generic cause of component failure rather than the specific event that results in the component failure. For example, the events 'water hammer' and 'pipe whip' can be represented by the cause 'impact'. These causes are termed generic causes. To further aid the analyst, three broad categories of generic causes have been suggested (21): mechanical-thermal, electrical-radiation, and chemical-miscellaneous. Table 4.3 shows potential failure causes of a mechanical-thermal nature.

A common cause analysis (COMCAN) program was written to implement the methodology. The program requires as input whatever minimal cut sets have been selected from the fault tree and the generic cause susceptibility for each basic

Table 4.3 Generic Causes_Mechanical/Thermal(134)

Generic Cause Symbol	Generic Cause	Specific Secondary Causes
I	Impact	Pipe whip,water hammer,missiles earthquake, structural failure
V	Vibration	Machinery in motion,earthquake
P	Pressure	Explosion,out_of_tolerance syst changes(pump overspeed,flow blockage)
G	Grit	Airborn dust,metal fragments generated by moving parts with inadequate tolerances
S	Stress	Thermal stress at welds of dis-similar Metals
T	Temperature	Fire,lightning,welding equip_ment,coolant system faults, electrical short circuits

event in each category. If the location option is used, the domain of each generic cause must also be supplied as input. In essence, the algorithm then searches for those minimal cut sets that are comprised of basic events that are all suspectible to the same generic cause, and this research is repeated for each category. Suppose, for example, that four min cut sets have been selected (BE1,BE3.BE4), (BE1,BE2), (BE3,BE6) and (BE2,BE4,BE5,BE6), where BEi, 1<i<6, represent basic events. Now assume that the location option is not being used and that the Mechanical/Thermal Generic cause susceptibility for the basic events is as indicated in Table 4.4. The COMCAN program would find that :

(BE1,BE3,BE4) is susceptible to P (pressure),

(BE1,BE2) is susceptible to I (inpact) and G (grit)

(BE3,BE6) has no Mechanical/Thermal susceptibility, and

(BE2,BE4,BE5,BE6) is susceptible to G (grit).

Provided in COMCAN is a cause ranking scheme whereby the analyst can assign a numerical ranking (0 to 9) for event susceptibilities. The higher the number, the greater is the susceptibility. An option is provided for printout of prime common-cause candidates containing events all ranked greater than or equal to N for some integer N, 0<N<9, selected by the analyst. This option allows the analyst to limit the size of the output to those prime common-cause candidates most susceptible to the generic causes.

Table 4.4 Example Mechanical/Thermal Susceptibility (134)

Basic Event	Generic Cause Symbols	Generic Cause Susceptibility
BE1	I,P,G,T	Impact,Pressure,Grit,Temperature
BE2	I,G	Impact,Grit
BE3	P	Pressure
BE4	P,G	Pressure,Grit
BE5	G	Grit
BE6	G,T	Grit,Temperature

The format used for COMCAN is compatible with the input format used with computer programs for qualitative and quantitative reliability and safety analysis such as PREP, KITT, and MOCUS.

A summary of COMCAN options is provided in Table 4.5.

4.3.2 New Approach for Automated Common-Cause Analysis

A problem in computer aided common cause failure analysis of complex system, both COMCAN and BACFIRE codes, is that all minimal cut sets must be determined for the top event. The difficulty is, when large trees are involved, the number of minimal cut sets is usually enormous. Computer codes for obtaining minimal cut sets are usually limited to locating those of small order because of the long running time required to obtain all the minimal cut sets. Thus a common cause failure analysis requiring minimal cut sets as input is usually incomplete since minimal cut sets of higher order have been discarded. This problem is overcome by a new approach suggested by Wagner et al (126), which does not require all such minimal cut sets be determined although the results are the same as if these cut sets were determined.

The basic idea of the new approach is to dissect the fault tree and determine minimal cut sets for individual branches and then synthesize common cause failure analysis results for the top event. The minimal cut sets found for

Table 4.5 COMCAN Inputs and Outputs (20)

Standard Input		Outputs
Primary Event Descriptions Generic Cause Subsceptibilities and cut sets	▶	List of all Cut Sets Sharing a Common Cause
Optional Inputs		
Manufacturer of Each Component	▶	List of all Cut Sets Sharing a Common Cause or Common Manufacturer
Location of Each Component and Barner Map Delineating Common Locations	▶	List of all Cut Sets Sharing a Common Cause and Common Location
Susceptibility Index Representing the Relative Failure Probability due to the Effect of the Generic Cause on the Component in the Appropriate Failure Mode	▶	List of all Cut Sets Sharing a Common Cause Ranked According to the Impact of the Common Cause on the Cut Set
Similar Flag Turned on	▶	List of all Cut Sets Sharing a Common Cause or Containing Similar Type Components
Type Flag Turned on	▶	List of all Cut Sets Sharing a Common Cause and containing Similar Type Components (according to some analysis thes basic requirements must exist for any dependent failure analysis)

Table 4.5 Cont'd

	Keyword	Controls Printout of
Various Printer Options ▶	Area	Input Locations (barrie map)
	Causes	Prime Common Cause Cand dates Recordered by Cau and Location
	Generic	Generic Cause and Commo Link Tables
	Card Images	Basic Event Description Information in Card Image Form
	Events	Basic Event Description Information Formatted f Ease of Reading
	Singles	Single Event Cut Sets
	Storage	Size of Internal Arrays
	Rank N	Prime Common Cause Cand dates with Basic Events Ranked Greater than or Equal to the Interger N

the individual branches of the fault tree are termed intermediate minimal cut sets. Common cause failure analysis of the intermediate minimal cut sets yields intermediate common cause candidates. A dummy event must then be defined to describe all the intermediate common cause candidates for a particular branch of the fault tree. The dummy event contains all the information necessary for analysis of the next level of the fault tree. At the completion of the analysis, the dummy events are expanded and common cause candidates for the top event are constructed. The new procedure is a step by-step analysis, advancing from the bottom to the top of the fault tree through its individual branches. This procedure is equivalent to an analysis using all the minimal cut sets for the top event.

There are two guidelines in deciding the point of dissection in the fault tree :

1) The number of Boolean Indicated Cut Sets (BICS) of the dissected branch must be small enough to allow determination of the intermediate minimal cut sets.

2) The intermediate minimal cut sets preferably should contain two or more basic events.

By keeping the number of BICS low, only minimum computer requirements are necessary. By obtaining two event or larger intermediate cut sets and analyzing them for common cause failure, all intermediate minimal cut sets which are not candidates for common cause failure may be eliminated. This further reduces the amount of information which must be carried to the next step of the analysis.

Once the fault tree has been properly dissected, the intermediate minimal cut sets may be found by conventional means. Treating the dummy events as ordinary basic events, standard computer programs will correctly determine the intermediate minimal cut sets for all segments of the dissected fault tree. Whenever the top segment of the dissected fault tree has been analyzed for intermediate common cause candidates, the dummy events in the results must b be expanded to obtain the common cause candidates for the top event. Expansion of the dummy events produces common cause candidates for the top event in terms of the minimal cut sets for the original fault tree.

Constructing common cause candidates for the top event consists of three steps :

1) Expanding the dummy event to the intermediate common cause candidates which it represents.

2) Determining potential common cause candidates for the top event.

3) Obtaining the common cause candidates for top event.

A flow chart for the above described is shown in Figure 4.6.

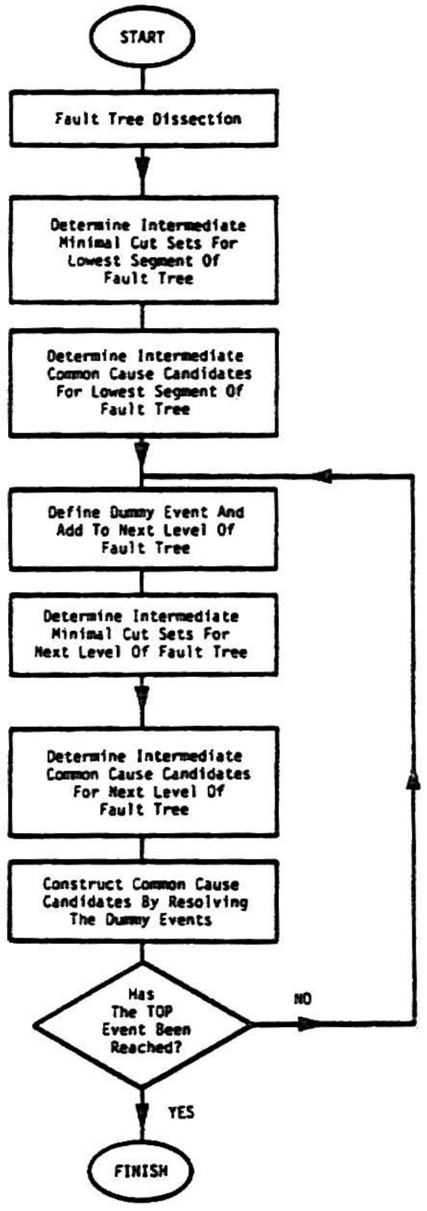

START
Fault Tree Dissection
Determine Intermediate Minimal Cut Sets For Lowest Segment Of Fault Tree
Determine Intermediate Common Cause Candidates For Lowest Segment Of Fault Tree
Define Dummy Event And Add To Next Level Of Fault Tree
Determine Intermediate Minimal Cut Sets For Next Level Of Fault Tree
Determine Intermediate Common Cause Candidates For Next Level Of Fault Tree
Construct Common Cause Candidates By Resolving The Dummy Events
Has The TOP Event Been Reached?
NO
YES
FINISH

Fig 4.6 Flow Chart for Synthesis Procedure for Common Cause Failure Analysis (126)

4.3.3 Quantitative Considerations for Common Cause Failure Analysis

Most generic causes of secondary failure can be assigned a time dependent occurrence rate in each domain. Often these cccurrence rates are identical to hardware failures (usually hardware external to the system being analyzed) that result in the generic failure cause. The occurrence rate for the generic cause temperature could correspond to the rate of cooling system failure. Unlike component reliability characteristic, the common cause candidates' reliability characteristics are not completely described by the time- dependent occurrence rate of the failure cause. Each ocurrence of a cause of secondary failure will not cause system failure. Common cause candicates are ranked by their sensitivity to each cause of secondary failure. The sensitivity ranking is used to determine the fraction of the occurrences of a cause of secondary failure that actually cause system failure. The failure rate of the common cause candidate is the occurrence rate of the cause of secondary failure weighted by the fraction of system failures which result from the cause of secondary failure. The common cause candidate sensitivity rank is determined by the sensitivity rank of the least sensitive component since all components in a minimal cut sets must fail for system failure.

A time dependent repair rate can also be determined for the new cut set that results from the common cause event. The repair of the secondary failure must be considered in addition to the repair of the components implied by the common cause candidate.

Some special conditions cannot be represented by an occurrence rate. Special conditions such as manufacturer and similar parts are examples. These links among all the events in a hardware minimal cut set increase the failure probability of that minimal cut set. They are usually treated as having a constant probability of contributing to system failure. The reputation of the manufacturer or the similar part in common influences the probabilistic effects of the common cause candidate on the system reliability characteristics.

Vesely (124) developed statistical estimation techniques for common cause failure analysis. The multivariate exponential Marshall-Olkin model (87) is specialized to produce an efficient estimation technique for dealing with the sparse data usually available for quantitative common cause failure analysis. Implicit in the use of the Marshall-Olkin model are the assumptions that each failure cause has an exponential distribution for its first time of occurrence and all possible failure causes are assumed to be competing; i.e., the observed component failures are determined by the failure cause which first occurs. The Marshall-Olkin model is specialized by assuming that the component population is homogeneous in the sense that it consists of components which are subject to similar failure causes. Two cases have been considered within the homogeneous model : failure rates for common causes which are 1) constant, and 2) binomial.

CHAPTER FIVE

QUANTITATIVE EVALUATIONS OF FAULT TREES

5.1 Introduction

A major goal of fault tree analysis is to calculate the probability of occurrence of the top event. However, it may also be useful to calculate the importance of minimal cut sets to the top event or the importance of specified basic events to the top event. In this chapter, we first review the most commonly used methods for calculating the probability of occurrence of the top event. Probabilistic evaluation of fault trees in the context of coherent structure theory will be described. Monte Carlo and analytical methods for obtaining probability characteristics of top event will also be discussed. We then present a survey of the available methods that quantitatively rank basic events and cut sets according to their importance. Such a ranking permits identification of events and cut sets that significantly contribute to the occurrence of the top event.

5.2 Probabilistic Evaluations of Fault Trees

Fault tree construction provides a systematic procedure to identify and record the various combinations of component fault states and other events that can result in an undesired state of a system. The resultant logic diagram shows the various component failure events as they combine through a set of Boolean gate operators leading to the top tree event. It is a well known fact that such logic diagrams may be mathematically represented in terms of coherent structures. The system unavailability can then be calculated either exactly, by using the minimal cut sets to write the structure function of the tree as a sum of products of primary inputs, or approximately by using the standard methods of probability bounds (10).

Other than the probabilistic evaluations of fault trees in the context of coherent structure theory, the evaluation of the occurrence probability of the top event of a fault tree can also be carried out by means of Monte Carlo simulation methods or by means of analytical methods. Monte Carlo simulation allows reliability information to be obtained for systems of almost any degree of complexity. However, this method provides only estimates and no parametric relation can be obtained. Analytical methods give more insight and understanding because explicit relationships are obtainable. Results are also more precise because these methods usually give the exact solution of the problem.

5.2.1 Coherent Structure Theory for Fault Tree Evaluation

It has been observed by reliability theorists that many of the quantities computed by fault tree analysis can also be computed using the concepts and techniques of reliability theory. Coherent structure theory (10) has been used for a long time in the reliability context. It has contributed heavily towards the development of the mathematical aspects of fault tree analysis. Barlow and Chatterjee (6) developed a mathematical theory of fault tree analysis using many of the concepts of coherent structure theory.

Consider a fault tree with n basic events, the ith event having a binary indicator variable Yi, such that

$$Yi = \begin{cases} 1 \text{ if basic event i occurs} \\ 0 \text{ otherwise} \end{cases}$$

The top event is associated with a binary indicator variable (Y), such that

$$\varphi(\underline{Y}) = \begin{cases} 1 \text{ if the top event occurs} \\ 0 \text{ otherwise} \end{cases}$$

where $\underline{Y} = (Y1, Y2, \ldots\ldots Yn)$ is the vector of basic event outcomes. We are assuming that the state of the system $\varphi(\underline{Y})$, can be expressed completely in terms of the indicator variables. $\psi(\underline{Y})$ is known as the structure function for the top event.

We now limit ourselves to Boolean structures, $\psi(\underline{Y})$, that are monotonic or coherent. A coherent structure, $\psi(\underline{Y})$, by definition, is nondecreasing in each argument Yi, i.e., that the occurrence of basic event cannot cause a system transition from a failed state, $\psi(\underline{Y}) = 1$, to a unfailed state, $\psi(\underline{Y}) = 0$. This implies that we do not allow complemented events. A coherent structure contains, by definition, all relevant basic events, i.e., the occurrence of each basic event must contribute in some way to the occurrence of the top event. Given the complete set of minimal cut sets Kj (j=1,2,....k) and minimal path sets Pj (j=1,2,......p) for a fault tree, which can be obtained by the computer code described in 4.1, its coherent structure may be expressed in terms of minimal cut sets or minimal path sets.

<u>Minimal Cut Representation</u>

Let K1,K2,.....Kk be the minimal cut sets of basic events for a specified fault tree. Then

$$\psi(\underline{Y}) = \coprod_{s=1}^{k} \prod_{i \in k_s} Y_i \tag{5. 1}$$

is the so-called minimal cut representation for ψ.

Minimal Path Representation

Let P1,P2,.....Pp be the minimal path sets of basic events for a specified fault tree. Then

$$\psi(\underline{Y}) = \prod_{r=1}^{p} \coprod_{i \in P_r} Y_i \qquad (5.\ 2)$$

is the so-called minimal path representation for ψ.

It is obvious from either the minimal cut or the minimal path representation that is coordinatewise nondecreasing.

Now, let us examine the system at one point in time. We assume that the state of the ith basic event is described by a random variable, Yi. Yi is a Bernoulli random variable, its probability of occurrence, Fi, is given by the mathematical expectation of Yi, denoted as E(Yi), where by definition

P(Yi=1) = E(Yi) = Fi

Likewise, $\psi(\underline{Y})$ is a Bernoulli random variable, the probability of the top event, P(Top Event) being given by

P(Top Event) = E($\psi(\underline{Y})$) = P ($\psi(\underline{Y})$ = 1)

The probabilistic evaluations of top event can then be obtained by following cases with respect to the structural characteristics of basic events of the fault tree.

1) Exact Solution : If basic events are not replicated in minimal cut sets and all basic events are statistically

independent, then

$$P\left[\text{Top Event}\right] = \coprod_{s=1}^{k} \prod_{i \in k_s} F_i \qquad (5.\ 3)$$

If there are no event replications among minimal path sets and basic events are statistically independent. We also write

$$P\left[\text{Top Event}\right] = \prod_{r=1}^{p} \coprod_{i \in P_r} F_i \qquad (5.\ 4)$$

2) <u>Min Cut and Min Path Bounds</u> :In general, basic events are replicated and expressions (5.3) and (5.4)are not valid. Esary and Proschan (47) proved, however, that the following bounds always hold

$$\prod_{r=1}^{p} \coprod_{i \in P_r} F_i \leqslant P\left[\text{Top Event}\right] \leqslant \coprod_{s=1}^{k} \prod_{i \in k_s} F_i \quad (5.\ 5)$$

when the basic events are statistically independent. The upper bound is known as the min cut upper bound and, in general, it is quite close to the exact value when Fi's are small, which is the usual situation.

3) <u>The Inclusion-Exclusion Method</u> : This method provides successive upper and lower bounds on top event probability which converge to the exact probability for fault trees. Let Es be the event that all basic events in min cut set Ks occur. We also assume all basic events are statistically independent. Then

$$P[E_s] = \prod_{i \in k_s} F_i$$

The top event corresponds to the event $\bigcup_{s=1}^{k} E_s$ if the fault tree has K min cut sets. Hence

$$P[\text{Top Event}] = P\left[\bigcup_{s}^{k} E_s\right]$$

Let $S_r = \sum_{1 \leq i_1 \leq i_2 \leq \cdots \leq i_r \leq k} P\left[E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_r}\right]$

By the inclusion-exclusion principles (49)

$$P[\text{Top Event}] = \sum_{r=1}^{k} (-1)^{r-1} S_r \qquad (5.\ 6)$$

and $$P[\text{Top Event}] \leqslant S_1 = \sum_{s=1}^{k} \prod_{i \in k_s} F_i$$

$$P[\text{Top Event}] \geqslant S_1 - S_2$$

$$P[\text{Top Event}] \leqslant S_1 - S_2 + S_3$$

and so on. Although it is not true in general that the upper and lower bounds will converge in a monotone fashion, in practice it may be necessary to calculate only a few Sr's to obtain a close approximation.

4) <u>Min-Max Bounds</u> : If ocurrence of basic events are not statistically independent, then the previous methods, based on assumed independence of basic events, are no longer

valid. In a great many reliability situations, the random variables of interest are not independent, but rather are 'associated'. For instance, the analyst may know that certain components in his system are subjected to a common environment or share a common load, so that a failure of a component results in increased load on the remaining components. In some cases, it may be difficult or tedious to show this dependency explicitly in terms of a secondary failure development in the fault tree. However, it is possible to incorporate statistical dependency in a quantitative evaluation by assuming that basic events are positively dependent. Barlow and Proschan (10) show that if indicator random variables are associated, then the following bounds always hold :

$$\max_{1<s<k} \prod_{i\in k_s} F_i \leqslant P\left[\text{Top Event}\right] \leqslant \min_{1<r<P} \coprod_{i\in P_r} F_i \qquad (5.\ 7)$$

The above expression tells us that the path set with the lowest failure probability is an upper bound for the probability of the top event, when basic events are associated.

5) <u>Improved Bounds by Modular Decomposition</u> : Defined in terms of the reliability network diagram, a module is a group of components which behaves as a 'super component'. In the context of fault trees, an intermediate gate event is a module to the top event if the basic events contained in the

domain of this gate event do not appear elsewhere in the fault tree, i.e., the gate event is a disjoint subtree. Decomposing a tree into modules is useful in reducing the computation required for probabilistic evaluation of fault trees.

A formal definition of a module (10) in terms of coherent structure theory is given as follows :

Let ψ be the indicator variable for the top event depending on a set of basic events N. Let M be a subset of N with complement M, χ be a coherent structure on M, then if

$$\psi(\underline{Y}) = \Gamma(\chi(\underline{Y}^{M}), \underline{Y}^{M^c}) \quad (5.8)$$

where $\underline{Y}^{M}$ means that the arguments are restricted to M, the set M with structure function χ is a module of ψ.

Barlow and Proschan (10) prove under the assumption of statistical independent that the minimal cut upper bound in expression (5.5) is a better bound when fault trees (or network diagrams) are decomposed into modules.

Chatterjee (34) proposes algorithms to find what he calls the finest modular decomposition of a fault tree. He uses game theory results by Shapley and Billera (15) to develop his algorithm. The connection with game theory is interesting. Basic events in a fault tree are analogous to the players in n-person game theory. Minimal cuts (paths) correspond to losing (winning) coalitions of players. A modular set or independent branch of a fault tree is analogous to a committee of players in the game theory context. However, this decomposition method requires all the minimal cut sets to be found as input, which is usually not feasible for a large fault tree. In contrast with this, Olmos and Wolf (92) devise an algorithm which derives the modular composition of a fault tree directly from its diagram description. A computer program, PL-MOD (92), written in PL-1, is prepared to perform the modularization algorithm. This program is capable of modularizing fault trees containing replicated components and replicated modular gates, and finds the probabilities of modules and top event as well.

6) Noncoherent Structure Case : Normally, fault trees have been used to analyze physical systems where improved performance of individual components does not degrade the performance of a system as a whole. Such systems are called coherent structures. Recently, however, there has been a demand to analyze noncoherent systems. Noncoherence occurs whenever NOT gates are introduced into a fault tree structure. This is most prevalent whenever human decisions play a role in instrument functioning or, more generally, whenever subsystems are forced into conflict with one another.

It has been proved (35) that all the methods applicable to coherent fault trees, except the minimal cut (path) set bounds, can be extended to noncoherent fault trees. When the fault tree is modularized, applying the min-max bounds or the inclusion-exclusion upper bound to both the modules and the organizing function yields the same bound that is obtained without modular decomposition. However, if the organizing function or the modules are simple enough, exact calculations can be performed at either level to give an improved bound. Alesso and Benson(3) present concepts that will allow the decomposition of noncoherent systems into coherent subsystems, they also provide a bounding condition for noncoherent systems that will not decompose in a desjoint manner.

5.2.2 Monte Carlo Simulation for Fault Tree Evaluation

The Monte Carlo method is generally accepted as a very powerful tool for calculation of reliabilty or systems, due to its versatility and, in particular for large complicated systems, relatively short computation time if used in connection with a technique for reduction of the variance of the result, like importance sampling (88).

Basically, the Monte Carlo approach is a procedure in which trials of the fault tree are simulated. In each trial, primary failures are made to occur and are repaired according to their failure and repair probabilities. The top failure is checked at various time points to determine whether it has occurred. For every top failure occurrence, a 'success' is tallied in the appropriate tally counter. The average of the successes over many trials yields an estimate of the probability of the top failure occurring. A typical Monte carlo simulation program involves the following steps:

1) Assign failure rate data to input fault events within the tree, descriptive mission data, and if desired, repair rate data.

2) Represent the fault tree on a computer to provide quantitative results for the overall system performance, subsystem performance, and the basic input event performance. These results can include the specified final event probability of failure and success, total failure information, availability, and downtime results.

3) List the failures which lead to the undesired event and identify critical path contributing event results.

4) Compute and rank basic input failure and availability performance results.

The Monte Carlo simulation is applicable to systems of any complexity and can theoretically handle any prescribed failure and repair distributions. However, the Monte Carlo simulation requires a fairly large amount of computer time, and to obtain results in reasonable time, the failure and repair distributions assigned to the primary failures must be limited to simple forms. Further, the Monte Carlo simulation yields statistical estimates for results, and there is always a disturbing possibility that estimates may be in considerable error, which is not shown by the accompanying error estimates.

There are several computer programs available to obtain probabilistic information about the top event from probabilistic information about the primary events by using the Monte Carlo method :

1) RELY 4 : The computer program RELY 4 (72) calculates both reliability and availability for any system including systems with standby units and, in addition, the standard deviations on both results. The method is based on the assumption that the condition of failure for the system to be analyzed can be expressed by a Boolean expression in terms of failures of a number of units which fail independently with known failure rates. The basic principle or operation is quite simple :

The times to failure and to repair for each component are simulated and it is analyzed whether the system has failed at any time during or at the end of the period considered. This process is then repeated a certain number of times (F) depending upon the required accuracy of the results. Then it is counted how many times the system has failed 1) at least once during the period considered (FP),

2) at the end of the period (FE). From these figures (multiplied by weighting factors for importance sampling), the availability of the system at the end of the period considered and the reliability are found as FE/F and FP/F respectively.

There are four different versions of the program. Versions 1 and 3 use importance sampling and versions 2 and 4 use direct simulation. While direct simulation is a simulation method by which all failure rates, repair rates and failure probabilities have the original values, the importance sampling is to concentrate the distribution of simulated system failure probability to the area of most importance-near the correct value.

The accuracy of the direct simulation method increases with the number of trials, but the method has the disadvantage that the required computation time for a given accuracy of the results increases tremendously when the failure rates decrease or the number of units increases. Monte Carlo simulation with importance sampling (88), therefore, is made on the basis of increased failure rates with a subsequent correction of the failure probability of the system multiplication with a weighting factor. This way a given number of Monte Carlo trials will cause a relatively high number of concentrated weighted estimates of the probability of system failure. Thus the reliability estimates will relatively soon achieve a small statistical variation.

In version 1, the times to repair for each unit can be distributed according to either a gamma or exponential distribution function. In versions 2 and 3 both the times to failure and the times to repair are exponentially distributed, while in version 4 they are distributed according to a Weibull distribution function.

From experience, version 3 has proved to be the most generally applicable version and it gives the highest accuracy for a given computer time.

2) SAFTE (SAFTE-1,2,3) : These programs are identified as the SAFTE series (64) (System Analysis by Fault Tree Evaluation). The SAFTE-1 program performs a Monte Carlo fault tree simulation for systems of redundant, repairable components with time dependent probabilities of failure. The principle results obtained are the probability density function, the cumulative distribution of system first failures, the mean time to system failure, and the mean time to system repair. The estimated probable error in these results is obtained.

The SAFTE-2 program has been written to consider the more simple case of no repair. It, like SAFTE-1, is a Monte Carlo fault tree simulation of systems with and without redundant components. The principle outputs of the program are probability density function and cumulative distribution of the system TTF's (time to failure) plus representative values, as well as mean and variance of system TTF's. The number of times each component fails is computed.

The SAFTE-3 program has been written to compute the probability of system failure at any time t based on steady state repair (fixed, rather than random). SAFTE-3 is capable of handling systems with or without redundancy and utilizes either direct or importance sampling techniques. In addition to probability of system failure at any time t, the program provides as output the fraction of time each component is in the failed state.

3) SAMPLE : The SAMPLE code (99), developed in WASH-1400, is a Monte Carlo program which evaluates the complete Boolean algebraic expression for the fault tree. It uses distributions instead of point probabilities for the component failure probabilities, and produces a distribution and confidence bounds on the top event. That is, because of the uncertainties in much component data, a highly simplified mathematical model was determined based on exponential failure distributions for the system components. The parameter of each failure distribution and the mean of each repair distribution were not assumed known with certainty. A lognormal distribution that represented the

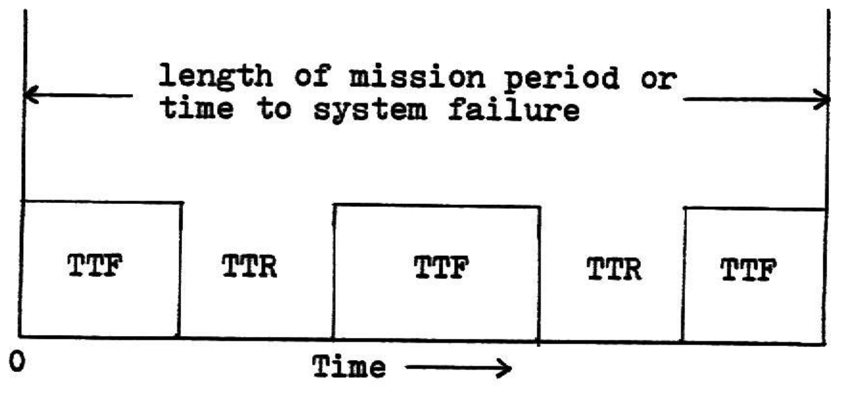
uncertainty in each parametere was determined. The SAMPLE program then allows inputs in the form of a median value and error limits on the failure pdrobability for each component. Therefore, by randomly choosing failure probability for each component, determined by the corresponding distribution, a point probability for the top event occurrence is computed, along with a set of confidence limits on the numbers. Typically, 90% limits are used, which means that, due to errors and uncertainties in data, there is 10% probability that the true value for the top event lies outside the limits predicted by the code.

4) REDIS : REDIS (73) is a Monte Carlo type code by direct simulation. The program combines some of the features of the SAMPLE code with those of several other approaches. Although it deals with a cut set approach, the code, as SAMPLE allows for uncertainties in data, and produces a top failure probability with a standard derivation, rather than confidence limits. Furthermore, various types of interdependencies between components are allowed, such as the failure of one component increasing the stress, and thus the failure probability, of others.

5) Crosetti's Code : This is one of the early Monte Carlo codes for fault tree analysis (38). This program, written in FORTRAN V, assumes an exponential failure distribution for basic events and a choice of either

normally (Gaussian) distributed or constant repair time. The program views the system represented by the fault tree as a statistical assembly of independent basic input events, each characterized by an exponential failure distribution and, if used, a constant or normal repair distribution.

The program computes a randomly determined time-to-failure (TTF) for each basic input event, based on the assigned mean-time-to-failure (MTTF) values. The system is then tested, as each basic input event fails, to determine if a failure occurs at the system level during the specified mission period. A time-to-repair (TTR) is generated (for variable repair) based on the mean-time-to-repair (MTTR) values and detection times, and a new TTF value is assigned to each failed basic input event to permit failure after repair as shown below.



Basic Input Event Failure and Repair Cycle Example

This process continues until either system failure occurs, or the end of the final mission period is reached. Then a new set of randomly determined values are assigned to each component for the next trial. By testing a sufficiently large population of systems trials in this manner, system probability of failure, probability of success, and more significant subsystem and component contributions to system failure are identified.

The program is also capable of simultaneously analyzing two different system mission periods where both mission periods start at the same point in time for each trial. This capability permits comparing system performance for two mission periods during each computer run.

5.2.3 Analytical Method for Fault Tree Evaluation

In 1970, Vesely made a most important advance in quantitative analysis of fault trees by developing an analytical methodology for fault trees containing repairable components. The particular analytical technique called 'Kinetic Tree Theory (119)' with the associated computer program (125) , is used to obtain quantitative probabilistic characteristics regarding the safety and reliability of these systems. Once the fault tree is drawn, the kinetic tree theory technique automatically yields detailed time-dependent information for every component, for every minimal cut sets and for the system itself (top event).

Caldarola and Wickenhauser (26) also developed a very fast analytical computer program for fault tree evaluation at the German nuclear research center of Karlsruhe. The Karlsruhe computer program for the evaluation of the availability and reliability of complex repairable coherent systems is essentially based on Vesely's theory with some additional important and fundamental improvements. Particular features of this computer program are 1) capability to identifiy the whole set of minimal cut sets and to list them in order of importance, and 2) capability to analyze systems characterized by two phases one following the other in time (two time axis).

1) Vesely's Kinetic Tree Theory

There are two assumptions for Kinetic Tree Theory. The first assumption is that the primary failures, or basic events, of the fault tree are independent; one primary failure many occur at any number of places in the fault tree, but those primary failures which are unique are assumed independent. The second assumption is that the minimal cut sets of the fault tree are known. Given above assumptions and the failure (hazard) probabilities and repair probabilities of primary failures of the fault tree, this method allows the complete probabilistic information to be obtained for each primary failure of the fault tree first, then for each minimal cut set, and finally for the top failure itself.

A) Primary Failure (basic event) Information :

Consider a single primary failure of the fault tree. Let

$z^p(t)\,dt$ = the probability of the failure occurring in time t to t+dt given the failure is not existing at time t (hazard probability)

$u^p(t)\,dt$ = the probability of the failure being repaired in time t to t+dt given the failure is existing at time t (repair probability)

The quantities $z^p(t)$ and $u^p(t)$ are basic data in terms of fault tree evaluation or reliability theory and are termed the hazard rate and repair rate for the primary failure respectively. It will be assumed that $z^p(t)$ and $u^p(t)$, or their equivalents, are known for every primary failure of the fault tree.

From $z^p(t)$ and $u^p(t)$, other probabilistic quantities may be obtained which quantify, or characterize, the particular primary failure. The probability of the primary filure first occurring in time t to t+dt given it does not exist at time t' is

$$a(t',t)\,dt = \exp\left(-\int_{t'}^{t} z^p(t'')\,dt''\right) z^p(t)\,dt;\quad t' \leq t \qquad (5.9)$$

with regard to repair, the probability that the primary failure is repaired at time t to t+dt, given it exists at time t' is

$$b(t',t)dt = \exp(-\int_{t'}^{t} u^{\rho}(t'')dt'')u^{\rho}(t)dt;\ t' \le t \qquad (5.10)$$

The quantities a(t',t) and b(t',t) are termed the first occurrence distribution (or first failure distribution) and the repair distribution respectively, Besides the above, there are two other primary failure characteristics which are essential for any reliability study or fault tree evaluation. The first characteristic is the primary failure intensity (failure density function) $f^{P}(t)$, which is defined such that

$\underline{f^{P}(t)}$ = the expected number of times the primary failure occurs at time t per unit time

An equation for $f^{P}(t)$ in terms of the data for primary failure can then be obtained from balance considerations of failure and repair :

$$f^{P}(t) = a(0,t) + \int_{0}^{t} f^{\rho}(t'')dt'' \int_{t''}^{t} b(t'',t')\, a(t',t)\, dt' \quad (5.11)$$

The first term on the right hand side of equation (5.11) is the contribution to $f^{P}(t)$ from the first occurrence of the primnary failure. The second factor is the contribution to $f^{P}(t)$ from the failure occurring at time t'', being repaired at t', and then reoccurring at time t. For the case of the failure being non-repairable, for example $b(t',t)=0$ and equation (5.11) becomes

$$f^P(t)=a(0,t);\ \text{nonrepairable primary failure} \tag{5.12}$$

In general, equation (5.11) can be solved using Laplace transformation techniques, or simple numerical integration tchniques can be used.

The second primary failure characteristic of interest is the primary failure existence probability $F^P(t)$;

$\underline{F^P(t)}$= the probability that the primary failure exists at time t

The non-existence probability, or the probability of the primary not existing at time t is merely $1-F^P(t)$. From the definition of $Z^P(t)$ and $f^P(t)$, it is apparent that

$$Z^P(t) = f^P(t)/(1-F^P(t)) \tag{5.13}$$

or

$$F^P(t) = 1- f^P(t)/Z^P(t) \tag{5.14}$$

For a specific failure intensity $f^P(t)$ and hazard rate $Z^P(t)$, $F^P(t)$ is simply obtainable from equation (5.14).

The quantities $f^P(t)$ and $F^P(t)$ along with the basic data $Z^P(t)$, $\mu^P(t)$, $a(t',t)$ and $b(t',t)$, are definitive functions which characterize the probabilistic behavior of the primary failure for all time. The characteristics $f^P(t)$ and $F^P(t)$, obtained for every primary failure are important in themselves since they show the effects oof repair, maintenance, and changes in environment and show these effects as functions of time. Moreover, with $f^P(t)$ and $F^P(t)$ determined for all the primary failures of the fault tree, the probabilistic characteristics for the minimal cut sets and for the top event can be obtained.

B) <u>Minimal Cut Set Failure Information</u>

By definition, a minimal cut set is a smallest set of primary failures such that if all these primary failures exist at time t the minimal cut failure (and top failure) exists at time t. Consider a particular minimal cut set. Let it consist of n primary failures and let these constituent primary failures be designated with indices from 1 through n. The first characteristic obtained for the minimal cut set will be the minimal cut failure existence probability $F^m(t)$ defined as ;

<u>$F^m(t)$</u> = the probability that the minimal cut set failure exists at time t

Since the minimal cut set failure exists at time t if and only if all its primary failures exist at time t,

$$F^m(t) = \prod_{i=1}^{n} F_i^p(t) \qquad (5.15)$$

where $F_j^p(t)$ is the existence probability for the jth primary failure of the minimal cut set (equation (5.14)). The minimal cut set failure nonexistence probability is then just $1-F^m(t)$ and is the probability of the minimal cut set not existing at time t.

The existence probability $F^m(t)$ is significant to the top failure to which the particular minimal cut set contributes. If the minimal cut set failure exists at time t then the top failure exists at time t.

The second quantity of interest characterizing the minimal cut set is termed the minimal cut set failure intensity $f^m(t)$;

$\underline{f^m(t)}$ = the expected number of times the minimal cut set occurs at time t per unit time

For the minimal cut set failure to occur, one or more of the primary failures must not exist at time t and these primary failures not existing must all simultaneously occur between time t to t+dt. Validly neglecting orders of dt greater than or equal to two, $f^m(t)$ is thus

$$f^m(t) = \sum_{j=1}^{n} f_j^p(t) \prod_{\substack{l=1 \\ l \neq j}}^{n} F_l^p(t) \qquad (5.16)$$

And, again, by defintion ;

$Z^m(t)\,dt$ = the probability of the minimal cut set failure occurring in time t to t+dt given the minimal cut set does not exist at time t

Therefore, the minimal cut set hazard rate becomes:

$$Z^m(t) = \frac{\sum_{j=1}^{n} f_j^p(t) \prod_{l=1,\, l\neq j}^{n} F_l^p(t)}{1 - F^m(t)} \tag{5.17}$$

The quantities $F^m(t)$, $f^m(t)$, and $Z^m(t)$, which characterize the minimal cut set failure are thus all simply determinable from the characteristics $f^p(t)$ and $F^p(t)$ of the primary failures which comprise the minimal cut set. The probabilistic characteristics for the minimal cut set are important in themselves since they quantify each minimal cut set as function of time and lead to the determination of the characteristics of the top failure of the fault tree.

C) Top Failure Information

Assume the minimal cut sets of the fault tree are known, and let there be N such minimal cut sets. Assume also the initial condition that at t=0 all the primary failures of the fault tree are non-existent. The top failure characteristic most simply obtained is the top failure existence probability $F^T(t)$;

$F^T(t)$ = the probability that the top failure exists at time t

Since the top failure exist if and only if one or more of the minimal cut sets failure exist

$$F^T(t) = \bigcup_{i=1}^{N} F_i^m(t) \tag{5.18}$$

By expansion,

$$F^T(t) = \sum_{i=1}^{N} F_i^m(t) - \sum_{i=2}^{N} \sum_{j=1}^{i-1} \prod^{+i,j} F^p(t)$$
$$+ \ldots . + (-1)^{N-1} \prod^{+i,\ldots,N} F^p(t) \tag{5.19}$$

where the product symbol is defined such that

$\prod^{+i,\cdots,N}$ = the product of unique primary failure quantities where the primary failure occurs in at least one of the minimal cut sets 1,....N.

The exact value for $F^T(t)$ can be determined straightforwardly by using equation (5.19). However, when the fault tree becomes complicated and hence the minimal cut sets increase, the above calculation could be tedious. A breaking procedure is suggested and is a particularly efficient method of obtaining successively tighter envelopes for $F^T(t)$, for primary failure existence probabilities $F^p(t)$, much less than 1, which is generally the case.

$$F^T(t) \leq \sum_{i=1}^{N} F_i^m(t) \qquad (5.20)$$

$$F^T(t) \geq \sum_{i=1}^{N} F_i^m(t) - \sum_{i=2}^{N} \sum_{j=2}^{i-1} \prod^{+i,j} F^p(t) \qquad (5.21)$$

$$\vdots$$

etc.

For those situations in which a simple but accurate approximation is desired for $F^T(t)$, the probability bounds developed by Esary and Proschan (47) for coherent structures can be applied here and result in the following equation;

$$F^{T(t)} \leq 1 - \prod_{i=1}^{N}(1 - F_i^m(t)) \qquad (5.22)$$

Equation (5.22) gives an upper bound and hence a safe and conservative estimate for $F^T(t)$.

Having obtained $F^T(t)$, the characteristic next determined is the top failure intensity, $f^T(t)$;

$\underline{f^T(t)}$ = the expected number of times the top failure occurs at time t per unit time

For the top event failure to occur in t to t+dt, all the minimal cut set must not exist at time t and then one or more of the minimal cut set must occur in t to t+dt. Hence, we can define the following equation ;

$$f^{T}(t)\,dt = f^{T(1)}(t)\,dt - f^{T(2)}(t)\,dt \qquad (5.23)$$

The first contribution to $f^{T}(t)\,dt$ is the contribution from one or more of the minimal cut sets occurring. Whenever a minimal cut set occurs, the top failure occurs. However, the second term $f^{T(2)}(t)\,dt$ must be substracted from $f^{T(1)}(t)\,dt$ which accounts for those cases in which one or more min cut set failures occur while other minimal cut sets are already existing. By applying probabilistic expansion theory for above two terms, we get

$$f^{T(1)}(t) = \sum_{I=1}^{N} f_i^{m}(t) - \sum_{i=2}^{N} \sum_{j=1}^{i-1} f(t;i,j) \underset{\bullet}{\overset{+i,j}{\prod}} F^{p}(t) + \sum_{i=3}^{N} \sum_{j=2}^{i-1} \sum_{k=1}^{j-1} f(t;i,j,k) \underset{\bullet}{\overset{+i,j,k}{\prod}} F^{p}(t)$$

$$+ \ldots\ldots\ldots \qquad (5.24)$$

where $f(t;1,\ldots\ldots.m)$ is the failure intensity for a failure mode which has as its primary failures which are common members to all the minimal cut sets $1,\ldots\ldots.m$, (this can be obtained in the similar way as equation 5.16)

and $\underset{\bullet}{\overset{+1,\cdots,m}{\prod}}$ = the product of unique primary failure quantities where the primary failure occurs in at least one of the minimal cut set $1,\ldots\ldots,m$, but is not common member in all of them

meanwhile

$$f^{T(2)}(t) = \sum_{i=1}^{N} f_B\,(t;i)$$

$$- \sum_{i=2}^{N} \sum_{j=1}^{i\ 1} f_B(t;i,j)+\ldots. \qquad (5.25)$$

where

$$f_B(t;\ i_1,\ldots,i_n) = \sum_{i'=1}^{N} f(t;i_1,\ldots,i_{\bar{n}}\ i') \prod_{i_1\ldots i_n}^{i'} F^p(t)$$

$$- \sum_{i'=2}^{N} \sum_{j'=1}^{i'-1} f(t;i_1\ldots i_{\bar{n}}\ i',j') \prod_{i_1\ldots i_n}^{i',j'} F^p(t)$$

and f (t; 1,...m-1,...n) = the failure intensity for a failure mode which has as its primary failures the primary failure common to all m minimal cut sets 1,...,m deleted from which are those primary failure also in any of the minimal cut sets 1,...,n.

while $\prod_{1,\ldots,m}^{1,\ldots,n}$ = the prduct of unique primary failure quantities, where the primary failure is a member of any of the minimal cut sets 1,...,n or is a member of the minimal cut set 1,...,m, but is not a common member of these m minimal cut sets.

For fault trees with a large number of minimal cut sets, the backeting procedure again is an extremely efficient method of obtaining an tight as enveloping as desired for $f^T(t)$. For scoping calculations, and in fact for many calculations, the following upper bound is of sufficient accuracy for a determination of $f^T(t)$;

$$f^T(t) \leq \sum_{i=1}^{N} f_i^m(t) \qquad (5.26)$$

This approximation gives a conservative estimate of $f^T(t)$, and can be simply computed from the minimal cut set intensities $f_i^m(t)$, and is usually within three significant figures of the true value of $f^T(t)$.

With the failure intensity $f^T(t)$ and the existence probability $F^T(t)$ determined, the remaining top failure characteristic, the top failure hazard rate, $Z^T(t)$, is simply obtainable. The top failure hazard rate $Z^T(t)$ is defined in a completely analogous manner to the primary and minimal cut set hazard rate;

$\underline{Z^T(t)\,dt}$ = the probability that the top failure occurs in time t to t+dt given it is not existing at time t.

thus, again by definition, we get,

$$Z^T(t) = f^T(t) / (1-F^T(t)) \qquad (5.27)$$

or

$$Z^T(t) < \sum_{i=1}^{N} f_i^m(t) \Big/ \prod_{i=1}^{N} (1- F_i^m(t)) \qquad (5.28)$$

The top failure hazard rate is thus determined, whether exactly or by enveloping

The top failure characteristics $F^T(t)$, $f^T(t)$, and $Z^T(t)$ are consequently all determined.

The codes PREP and KITT to execute above probabilistic theory are written in FORTRAN IV for the IBM 360/75 computer and can handle fault trees consisting of up to 2000 gates and up to 2000 components (primary failures) and inhibit conditions. PREP and KITT can handle any assortment of repairable and nonrepairable components on the fault tree. The codes can treat 'single phase' problems, where the component failure rates and repair times remain constant with time, or can treat 'multi-phase' problems where the component failure rates and repair times are constant within a time phase, but may vary from phase to phase. For these multiphase problems, each component may have its own unique phases (up to 50) and may be repairable in some phases and non-repairable in others. The codes can handle constant repair times, exponential repair distributions, or the multiphase combinations of these two types of repair. Inhibit conditions with either constant probailities or time-dependent probabilities can be included with the component.

2) Caldarola and Wickenhauser's Analytical Method

Caldarola and Wickenhauser (26) also developed an analytical computer program for fault tree evaluation at the German nuclear research center of Karlsruhe. This program can solve coherent systems in binary logic. Four different classes of components can be handled by the program :

1) Unrepairable components

2) Repairable components with failures which are immediately detected (revealed faults)

3) Repairable components with failures which are detected upon demand (faults remain unrevealed until next demand occurs)

4) Repairable components with failures which are detected upon inspection (faults remain unrevealed until next inspection is carried out)

The program can perform also time dependent calculations. In particular the program can analyze systems characterized by two phases one following the order in time (two time axis). The calculation takes place in two steps. The first step is to identify the minimal cut sets. The algorithm to identify the minimal cut sets is the so called 'downward algorithm' already described in MOCUS (56) by Fussell. The second step for the evaluation of the availability and reliability of complex repairable coherent systems is based on Vesely's theory with some additional important and fundamental improvements.

The calculation of the system unavailability and unreliability is described as follows :

First the unavailability and the failure intensity of each component Ki as a function of time are calculated for both initial conditions: component intact (Vu) and component failed (Vd) at the initial time. The equations of Vu and Vd for each class of components are given in Table 5.1. With reference to Table 5.1 the development of the equations is described in the article (22) by Caldarola. The unavailability V of a component is given by the following equation :

$$V = (1-P)\ Vu + P\ Vd \qquad (5.29)$$

where P is the probability that the component is unavailable at the initial time. The failure intensity h of a component is given by the following equation

$$h = \lambda(1-v) \qquad (5.30)$$

where λ is the component failure (hazard) rate (assumed to be constant). The unavailability Ujk and the failure intensity Hjk of a general cut set Cjk are given respectively by the following equations

Table 5.1
Component unavailability (22)

Class	Type of component	INITIAL STATE			
		Intact (V_u)		Failed (V_d)	
1	Unrepairable	$1-e^{-\lambda t}$		1	
2	Repairable (revealed faults)	$\frac{\lambda}{\lambda+\mu}[1-e^{-(\lambda+\mu)t}]$		$\frac{\lambda}{\lambda+\mu}+\frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}$	
3	Repairable (faults detected upon demand) $\epsilon=\left(\frac{\lambda+\mu+\nu}{2}\right)^2-(\lambda\mu+\lambda\nu+\mu\nu)$	$\epsilon<0$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}\left\{1-e^{-(\lambda+\mu+\nu)t/2}\left[\cos(t\sqrt{\lvert\epsilon\rvert})+\frac{1}{2\sqrt{\lvert\epsilon\rvert}}\left(\frac{\mu^2+\nu^2}{\mu+\nu}-\lambda\right)\sin(t\sqrt{\lvert\epsilon\rvert})\right]\right\}$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}+\frac{\mu\nu}{\lambda\mu+\lambda\nu+\mu\nu}e^{-(\lambda+\mu+\nu)t/2}\left(\cos(t\sqrt{\lvert\epsilon\rvert})+\frac{\lambda+\mu+\nu}{2\sqrt{\lvert\epsilon\rvert}}\sin(t\sqrt{\lvert\epsilon\rvert})\right)$	
		$\epsilon=0$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}\left\{1-e^{-(\lambda+\mu+\nu)t/2}\times\left[1+\frac{t}{2}\left(\frac{\mu^2+\nu^2}{\mu+\nu}-\lambda\right)\right]\right\}$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}+\frac{\mu\nu}{\lambda\mu+\lambda\nu+\mu\nu}e^{-(\lambda+\mu+\nu)t/2}\left[1+\frac{\lambda+\mu+\nu}{2}t\right]$	
		$\epsilon>0$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}\left\{1-e^{-(\lambda+\mu+\nu)t/2}\left[\cosh(t\sqrt{\lvert\epsilon\rvert})+\frac{1}{2\sqrt{\lvert\epsilon\rvert}}\left(\frac{\mu^2+\nu^2}{\mu+\nu}-\lambda\right)\sinh(t\sqrt{\lvert\epsilon\rvert})\right]\right\}$	$\frac{\lambda(\mu+\nu)}{\lambda\mu+\lambda\nu+\mu\nu}+\frac{\mu\nu}{\lambda\mu+\lambda\nu+\mu\nu}e^{-(\lambda+\mu+\nu)t/2}\left[\cosh(t\sqrt{\lvert\epsilon\rvert})+\frac{\lambda+\mu+\nu}{2\sqrt{\lvert\epsilon\rvert}}\sinh(t\sqrt{\lvert\epsilon\rvert})\right]$	
4	Repairable (faults detected upon inspection) $q=\lg(2-\lg\theta\lambda)$ $\lambda_{\text{eff}}=2[1-\Gamma(1/q)/q]\theta/\eta^2+\lambda\frac{\eta-\theta}{\eta}\left(\frac{\eta-\theta}{\eta}+\frac{2\tau}{\eta}\right)$	$t>m\eta_+$	$1-e^{-(t-m\eta)\lambda_{\text{eff}}[1-e^{-((t-m\eta)/\theta)^q}]}$	$0<t\leqslant\eta_-$	1
		$t=m\eta_-$ $(m>0)$	$1-e^{-\eta\lambda_{\text{eff}}[1-e^{-(\eta/\theta)^q}]}$	$t>m\eta_+;m>0$	$1-e^{-(t-m\eta)\lambda_{\text{eff}}[1-e^{-((t-m\eta)/\theta)^q}]}$
				$t=m\eta_-;m>1$	$1-e^{-\eta\lambda_{\text{eff}}[1-e^{-(\eta/\theta)^q}]}$

Nomenclature of the symbols used. V_u = component unavailability with component being intact at the initial time, V_d = component unavailability with component being failed at the initial time, t = time, λ = component failure rate (constant), μ = component repair rate (constant), ν = average demand frequency (constant), η = time interval

$$U_{jk} = \prod_{i=1}^{n} (V_i)^{\alpha_{i,jk}} \qquad (5.31)$$

and

$$H_{jk} = \sum_{i=1}^{n} \alpha_{i,jk} \lambda_i (1-V_i) \prod_{q=1, q \neq i}^{n} (V_q)^{\alpha_{q,jk}} \qquad (5.32)$$

where n=total number of components belonging to the fault tree, ijk=1 if Ki Cjk, and i,jk=0 if Ki Cjk. As we can see, equations (5.31) and (5.32) are derived as the same fashion as those suggested by Vesely in his Kinetic Tree Theory (119

The unavailability U and failure intensity H of the system (top failure) are given respectively by the two following equations :

$$U = \sum_{K=1}^{L_1} \sum_{jk=1}^{L_k} (-1)^{k-1} U_{jk} \qquad (5.33)$$

and

$$H = \sum_{k=1}^{L_1} \sum_{jk=1}^{L_k} (-1)^{k-1} H_{jk} \qquad (5.34)$$

The computer program then considers the cut sets of order 1, 2, and 3 only.

$$U \cong \sum_{k=1}^{3} \sum_{jk=1}^{L_k} (-1)^{k-1} U_{jk} \tag{5.35}$$

$$H \cong \sum_{k=1}^{3} \sum_{jk=1}^{L_k} (-1)^{k-1} H_{jk} \tag{5.36}$$

The system unreliability Q is approximated by the integral of the system failure intensity.

$$Q(t) \cong \int_0^t H(t')\,dt' \tag{5.37}$$

The Karlsruhe program thus can perform a limited (two-phase) phase mission analysis. System unavailability is calculated for the first time period (phase) and system unreliability is calculated for the second phase. Such computations have application to safety systems which first must be available at the onset of their mission and then, given that they are available, must perform reliably through an additional period of time.

A new computer program is also being developed at Karlsruhe. This computer program will be able to analyse non-coherent systems with multistate components. (23-24).

5.2.4 Other Computer Codes for Fault Tree Evaluation

There are several other computer codes not belonging to the above categories (Monte Carlo or analytical method) are also applicable for formal fault tree evaluation. Some relate to the evaluation of block diagrams, while others feature different approaches on the evaluation technique.

1) ARMM

ARMM (89) is a general purpose computer program for deriving and solving a mathematical model of the reliability of complex systems. It is based on a sequential application of the conditional probability theorem to the probability of system failure. The program selects those combinations of component failures which cause a system failure, and derives and solves the reliability mathematical model for computing failure probabilities. The output includes 1) the probability of successful system operation; 2) the probability of each failure combination; 3) the probability of function failure, its percentage of system failure, and the rank of each function of the system; 4) each component's contribution to the probability of system failure, i.e., its percentage of the total probability and rank among the componets; and 5) the component combinations which are the major contributors to system unreliability. Other useful aspects of the program are a built-in capability to handle dependent components and mutually exclusive failure modes, and the use of input data requirements which are simplified for engineers not familiar with programming methods.

2) GO and NOTED

Both GO (65) and NOTED (130) differ from the block diagram (like ARMM) and fault tree analysis codes in that the system analyzed includes not only logic gates, but also switches, delay, etc. For instance, a signal (or event probability) is allowed to pass through a switch only if that switch is closed. Thus the probability of an output from the switch is the product of the probability of an input signal and the probability that the switch is closed. A delay is simply a 'block' which retards the passage of a signal through it. If a signal arrives at time t+T, where T is the delay time.

Both codes begin with a system chart or diagram consisting of a logic interconnection of switches, delays and gates of special type handle internally by the programs; they result in the output of probabilities of events specific endpoints. Both are similiar in that they numerically evaluate the system in one step as the signals are traced from beginning to end. This means that change in component probabilities requires a complete rerun, even though the system remains unchanged. However, the GO includes the option of doing several job runs at one time.

Both approaches also rely upon the investigator to initially decompose a complex system of components into a few basic switch types which can be handled by the program. (This is analagous to constructing the tree for fault tree codes.) Whereas the GO code divides the time scale into several discrete points and defines switch in terms of fix probabilities of failure, NOTED defines switch types in terms of continuous functions, including exponential, lognormal, normal, Weibull and forms including repair times. Thus the output of GO consists of probabilities of discrete output events, while NOTED produces a graph of the cumulative probability as a function of time, at any of several points in the system. Aside from its discrete nature, the GO code also differs in that its output can be in the form of a joint distribution, for example, the distribution of times to arrival of signals at different endpoints in the system. (For instance, an example of the GO code produces the joint density function for time to arrival at two points in a system.)

The similarities in both these approaches, then, include a forward marching algorithm through systems of signals and switches, although the one code deals with fixed probabilities and the other with distributions. However, with both codes one must be careful in preparing the system chart to be analyzed. Since often complex components must be manually decomposed into the simpler switch types handled internally. Thus, it may be required to decomposed a complex unit into several independent switches and gates in order to model the various interdependent mechanisms and interactions within it.

3) BAM

BAM (45) code is a new approach to fault tree analysis. It uses some of the methodology of GO. The idea has been to describe the fault tree in terms of truth tables rather than by directly manipulating AND and OR gates. The advantages of this procedure have allowed a code which not only can handle NOT gate logic, but also dependent events. Furthermore, the storage requirements have been minimized, as in GO, by computing event probabilities at intermediate points and then deleting unnecessary terms. Thus the BAM code in one step, provides the top event probability without ever requiring minimal cut set information. However, by directly evaluating the fault tree, BAM produces a point probability rather than a time history of the system.

4) PARTEC and SALP

The computer program PATREC (18) relies on the recognition of subtree patterns whose probability combination laws have been previously stored in the library of the computer code. The subtree is then replaced by a super-component with an associated occurrence probability equal to that of the recognized subtree. By repeating this process the whole tree is eventually transformed into a single super-component whose occurrence probability corresponds to that of the top tree event. By using the programming language PL-1 for its list processing capabilities, the PATREC code can take into account several present day problems : multi-dependencies treatment, uncertainties in the reliability data parameters, influence of common mode failures, etc..

The SALP (5) code is also based on direct manipulation of graphs by listprocessing techniques. The SALP series can handle the routine analysis of AND/OR fault trees and AND-OR-NOT trees as well.

5.3 Measures of Importance of Events and Cut Sets in Fault Trees

A system is an orderly arrangement of components that performs some task or function. It is clear by the arrangement of these components that some are more critical with respect to the functioning of the system than others. While the probabilistic evaluation of the top event of a fault tree gives the quantitative information for the system failure been mostly concerned, the probabilistic measures of importance of events and cut sets are equally needed for system analysis. Measuring the relative importance of components may

1) Identify components that merit additional research and development, thereby improving the overall reliability at minimal cost or effort.

2) Suggest the most efficient way to diagnose system failure by generating a repair checklist for an operator to follow.

A survey of the available probabilistic methods that quantitatively rank basic events and cut sets according to their importance will be presented in the following section. Each probabilistic method that computes importance can be expressed in terms of a 'g' function that computes the probability of the top event in terms of the basic event probabilities. It is assumed that all basic events are statistically independent.

5.3.1 Probabilistic Expressions that Measure Importance (of components)

We first introduce three measures of importance computed in terms of g(F(t)), a function that measures the age of the system at t and describes system behavior at one point in time. Then, we introduce two other measures of importance that describe system in terms of sequences of component failures that cause the system to fail in time. These measures are functions of the past behavior of the system while the first three are not.

1) Birnbaum's Measure of Importance

In 1969, Birnbaum (16) introduced the concept of importance of a coherent system. He defined the reliability importance of a compontnt i as the rate at which system reliability improves as the reliability of component i improves. For the evaluation of a fault tree where the top event is system failure and the basic events are component failures, Birnbaum's definition of component importance becomes

$$\frac{\partial g(\underline{F}(t))}{\partial F_i(t)} = g(1_i, \underline{F}(t)) - g(0_i, \underline{F}(t)) \overset{def}{\equiv} \Delta g_i(t) \qquad (5.38)$$

Stated in other terms, $\Delta g_i(t)$ is the probability that the system is in a state at time t in which the functioning of component i is critical : the system functions when i functions, then system fails when i fails. The failure of i is critical at time t when $\psi(1i,Y(t)) = \psi(0i,Y(t)) = 1$.

Birnbaum's definition of importance is also known by two other names, 1) marginal importance, and 2) the partial derivative.

2) <u>Criticality Importance</u>

Birnbaum's definition of importance is a conditional probability in the sense that the state of the ith component is fixed. The probability that the system is in a state at time t in which component i is critical and that component i has failed by time t is

$$\left\{ g(1i,F(t)) - g(0i,F(t)) \right\} \; Fi(t) \qquad (5.39)$$

If we make this conditional to system failure by time t, then the above expression becomes

$$\frac{\left\{ g(1_i, \underline{F}(t) - g(0_i, \underline{F}(t)) \right\} \; F_i(t)}{g(\underline{F}(t))} \overset{def}{\equiv} I_i^{CR}(t) \qquad (5.40)$$

The above expression is defined as the criticality importance of component i. Note that $I_i^{CR}(t)$ is a function of Fi(t) while gi(t) is not.

3) <u>Vesely-Fussell Definition of Importance</u>

It is possible that a failure of a component can contribute to system failure without being critical. Component i contributes to system failure if a cut set containing i has failed. Let the probability that component i contributes to system failure be denoted as g (F(t)). The probability that component i contributes to system failure, given that the system has failed by time t, is given by

$$\frac{g_i(\underline{F}(t))}{g(\underline{F}(t))} \overset{def}{\equiv} I_i^{VF}(t) \qquad (5.41)$$

This concept of importance was introduced by Vesely and Fussell (53). Chatterjee call I_i^{VF} (t) the diagnostic importance of i.

4) <u>Barlow-Proschan Measure of Importance</u>

Barlow and Proschan (9) examined components as they failed sequentially in time. They assume that if two or more components have a vanishingly small probability of occurring at the same instant, then one component must have caused the system to fail. The probability that event i cause the system to fail during a differential time interval of t', where t'<t, is

$$\left\{g(1i,F(t'))-g(0i,F(t'))\right\} \; dFi(t') \qquad (5.42)$$

integrating between 0 and t

$$\int_0^t \left\{g(1i,F(t'))-g(0i,F(t'))\right\} \; dFi(t') \qquad (5.43)$$

We get the probability that component i cause the system to fail in (0,t).

It can be shown (9) that

$$\sum_{i=1}^{n} \int_0^t \left\{g(1_i, \underline{F}(t')- g(0_i, \underline{F}(t'))\right\} dFi(t') = g(F(t)) \quad (5.44)$$

i.e., (5.44) is the probability that the system fails before t, where n is the number of components comprising the system.

The conditional probability that a component i causes the system to fail by the time t is then the Barlow-Proschan measure of importance

$$\frac{\int_0^t \left\{g(1_i, \underline{F}(t'))- g(0_i, \underline{F}(t'))\right\} dF_i(t')}{\sum_{i=1}^{n} \int_0^t \left\{g(1_i,\underline{F}(t')- g(0_i, \underline{F}(t'))\right\} dF_i(t')} \overset{def}{\equiv} I_i^{BP} \; (t) \quad (5.45)$$

5) <u>Sequential Contributory Importance</u>

It might be interesting to assess the role of the failure of a component i when another component, say j, causes the system to fail. The failure of i is a factor in this case only if i and j are contained in at least one minimal cut set. The probability that component i is contributing to system failure when j causes the system to fail is

$$\frac{\int_0^t \left\{ \mathbf{g}(1_i, 1_j, \underline{F}(t')) - \mathbf{g}(1_i, 0_j, \underline{F}(t')) \right\} F_i(t') dF_j(t')}{\mathbf{g}(\underline{F}(t))} \tag{5.46}$$

and, in general, the probability that component i is contributing to system failure when another component causes the system to fail is

$$\frac{\sum_{\substack{j \\ i \neq j}} \int_0^t \left\{ \mathbf{g}(1_i, 1_j, \underline{F}(t')) - \mathbf{g}(1_i, 0_j, \underline{F}(t')) \right\} F_i(t') dF_j(t')}{\mathbf{g}(\underline{F}(t))}$$

$$\overset{def}{\equiv} \quad I_i^{sc}(t) \tag{5.47}$$

Where the sum over j is to include only those components that appear in at least one minimal cut set with component i. Expression (5.47), I_i^{sc} (t), shall be called the sequential contributory importance of component i.

5.3.2 Cut Set Importance

Definitions of cut set importance are described by analogy to methods that determine component importance.

In the Vesely-Fussell definition, the importance of a cut set Kj is the probability that cut set Kj contributes to system failure. It is given by

$$\frac{\prod_{i \in kj} F_i(t)}{\mathbf{g}(\underline{F}(t))} \tag{5.48}$$

The Barlow-Proschan definition of the importance of a cut set Kj is the probability that a cut set Kj causes the system to fail. For a cut set Kj to have caused the system to fail, a basic event contained in the cut set must have caused the system to fail and all other events in the cut set must have failed prior to the event that caused the system to fail. Barlow-Proschan's measure of importance of a cut set Kj is

$$\frac{\sum_{i \in kj} \int_0^t \left(\mathbf{g}(\underline{1}^{k_j}, \underline{F}(t)) - \mathbf{g}(0_i, \underline{1}^{k_j - \{i\}}, \underline{F}(t))\right) \prod_{\substack{l = i \\ l \in Kj}} F_l(t)\,dF_i(t)}{\mathbf{g}(\underline{F}(t))} \tag{5.49}$$

where $\underline{1}^{K_j}$ means that Yi is equal to 1 for each basic event i contained in cut set Kj. Since $g(\underline{1}^{K_j}, \underline{F}(t))=1$, the above expression becomes

$$\frac{\displaystyle\sum_{i\in k_j} \int_0^t \left(1 - g\left(0_i, 1^{k_j - \{i\}}, \underline{F}(t)\right)\right) \prod_{\substack{l=i \\ l\in k_j}} F_l(t)\, dF_i(t)}{g(\underline{F}(t))}$$

(5.50)

Vesely-Fussell's definition of cut set importance always assigns more importance to a cut set of a lower order than a cut set of a higher order when basic event probabilities are equal. This is not always true, however, with Barlow-Proschan's measure of importance (76). A summary of importance measures for events and cut sets is listed in Table 5.2.

5.3.3 Importance of components when Repair is Permitted

Each of the methods previously described can also assess the importance of components when repair is permitted. In every importance expression except Barlow and Proschan's, the limiting unavailability, $\bar{A}i$, can be substituted for Fi(t) without any change in probabilistic meaning (76).

5.3.4 Computer Codes for Measures of Importance

A computer code called IMPORTANCE (76) was developed by

TABLE 5.2 Summary of Importance Measures (76)

IMPORTANCE MEASURE	PROBABILISTIC EXPRESSION	MEANING
BIRNBAUM Basic Event Importance	$g(1_i, \underline{q}(t)) - g(0_i, \underline{q}(t))$	Probability that the system is in a state in which the occurrence of event i is critical.
CRITICALITY Basic Event Importance	$\dfrac{[g(1_i, \underline{q}(t)) - g(0_i, \underline{q}(t))]\, q_i(t)}{g(\underline{q}(t))}$	The probability that event i has occurred and is critical to system failure.*
UPGRADING FUNCTION Basic Event Importance	$\dfrac{\lambda_i(t)}{g(\underline{q}(t))} \quad \dfrac{\partial g(\underline{q}(t))}{\partial \lambda_i(t)}$	Fractional reduction in the probability of the top event when $\lambda_i(t)$ is reduced fractionally.
VESELY-FUSSELL Basic Event Importance	$\dfrac{g_i(q(t))}{g(\underline{q}(t))}$	Probability that event i is contributing to system failure.*
BARLOW-PROSCHAN Basic Event Importance	$\int_0^t \{g(1_i, \underline{q}(t)) - g(0_i, \underline{q}(t))\}\, w_{f,i}(t)\, dt$	Expected number of failures caused by basic event i in [0, t].
CONTRIBUTORY SEQUENTIAL Basic Event Importance	$\sum_{\substack{j \\ i \neq j \\ i \& j \in K_\ell \text{ for some } \ell}} \int_0^t \{g(1_i, 1_j, \underline{q}(t)) - g(1_i, 0_j, \underline{q}(t))\}\, q_i(t) dw_{f,j}(t)$	The expected number of system failures in [0,t] caused by min cut sets that contain basic event i with basic event i occurring prior to system failure.

*Given that system failure has occurred

TABLE 5.2 Cont'd

IMPORTANCE MEASURE	PROBABILISTIC EXPRESSION	MEANING
STEADY-STATE BARLOW-PROSCHAN Measure of Basic Event Importance	$\dfrac{[g(1_i, \underline{\bar{A}}) - g(0_i, \underline{\bar{A}})]/(\mu_i+\tau_i)}{\sum_{i=1}^{n}[g(1_i, \underline{\bar{A}}) - g(0_i, \underline{\bar{A}})]/(\mu_i+\tau_i)}$	Probability that event i causes system failure in the steaJy state.†
FIRST FAILURE RATE OF BREAKDOWN, T* Basic Event Importance	$\dfrac{g_{s,i}(t)}{\sum_i g_{s,i}(t)}$	Probability that event i causes first system failure approximated by T* method.†
FIRST FAILURE RATE OF BREAKDOWN, SS Upper Bound, Basic Event Importance	$\dfrac{\ln[1-\Delta g_i]/(\mu_i+\tau_i)}{\sum_{i=1}^{n}\ln[1-\Delta g_i]/(\mu_i+\tau_i)}$	Probability that event i causes first system failure approximated by steady-state upper bound method.†
VESELY-FUSSELL Cut Set Importance	$\dfrac{\prod_{i\in K_\ell} q_i(t)}{g(\underline{q}(t))}$	Probability that min cut set K_ℓ is contributing to system failure.*
BARLOW-PROSCHAN Cut Set Importance	$\sum_{i\in K_i}\int_0^t [1 - g(0_i, \underline{1}^{K_i-\{i\}}, \underline{q}(t))] \prod_{\substack{j\neq i\\ j\in K_i}} q_j(t)\, dw_{f,i}(t)$	Expected number of system failures caused by min cut set K_i.

*Given that system failure has occurred
†Maintained system

Lambert to rank basic events and cut sets according to various importance measures described above. The IMPORTANCE computer code is capable of handling time-dependent fault trees under the assumption that each basic component be statistically independent and that its failure and repair distribution be exponential in time. It requires as input the minimal cut sets; the failure rates and fault duration times of all basic events. The code computes as output the following measures of bsic event importance, 1) Birnbaum 2) Criticality, 3) upgrading Function (proposed by Lambert), 4) Vesely-Fussell, 5) Barlow-Proschan, 6) Sequential Contributory and two measures of cut set importance, 1) Barlow Proschan and 2) Vesely-Fussell.

The PL-MOD code (92) developed by Olmos and Wolf, also provides an option to command the calculation of modular and component Vesely-Fussell importances following the structural analysis and probabilistic evaluation of the fault tree.

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

6.1 Conclusions

Fault tree analysis has proved to be a useful analytical tool for the reliability and safety analysis of complex systems. Inductive analysis like preliminary hazards analysis (PHA) or failures modes and effects analysis (FMEA) can become extremely inefficient when analyzing complex system due to the large number of component states that must be considered. Fault tree analysis can efficiently direct the effects of an analyst in considering only those basic events that can contribute to system failure, i.e., to the occurrence of the top event. The information contained in the evaluation of the fault tree can further assist an analyst in simulating system failure, identifying subsystem functional faults, and upgrading system designs to improve their safety or reliability.

Fault tree models do have disadvantages. Probably the most outstanding one is the cost of development in first-time application to a system (50). However, as in the development of engineering drawings, the cost is somewhat offset by future application of the models in accident prevention, maintenance scheduling, and system modifications.

Another disadvantage to fault tree analysis is the possibility of oversight and omission (43). Automated fault tree construction can eliminate the possibility of omitting the routine failure modes. The automated approach can standardize fault tree analysis and eliminate the confusion associated with the seemingly different ways analysts can manually construct fault trees.

A major pitfall with quantitative fault tree evaluation is applying poor or imapplicable failure data, or poor assumptions to highly complex systems (66). In standard reliability analysis, point values (i.e.,'best estimates') are generally used for both data and results in quantifying the system model. Nevertheless, the lack of pertinent failure rate data can be overcome by making use of statistical theorem in data specialization. Furthermore, quantitative fault tree evaluations are particularly vluable for comparing systems designs that have similar components. The results are not as sensitive to the failure rate data as in an absolute determination of the system failure probability.

The main points to consider in applying fault trees to any situation are:

1) The fault tree analysis process is iterative so it should begin by trying to provide a basic understanding which can be expanded and embellished upon as the process

continues;

2) The fault tree analysis investigation should have the latitude to answer questions about failure importance; and

3) The fault tree analysis technique is of engineering value only when used to increase system understanding as well as producing the fault tree diagram.

6.2 Possible Future Work

A problem in fault tree modeling is that it is difficult to apply Boolean logic to describe failures of system components that can be partially successful in operation and thereby have effects on the performance of the system. Leakage through a valve is a good example. Research efforts should be directed toward construction of Boolean failure models for chemical and physical processes.

Since the published methodologies for fault tree analysis are numerous and not yet of sufficiently high quality for general use. It would be useful be establish some criteria so that methods can be evaluated.

It would also be useful to develop certain statistical techniques for deriving failure rate data to be used as input of basic events for fault tree evaluations. Quantitative techniques of common cause failure analysis should also be developed for general use.

Finally, fault tree analysis is in many cases costly and time consuming; consequently, is difficult to apply in real world situation. An interesting research problem would be to establish the feasibility and usefulness of an adequate fault tree model for system safety and reliability analysis in real world systems.

REFERENCES

1. K. K. Aggarwal, "Comment on an efficient simple algorithm for fault tree automatic synthesis from the reliability graph", IEEE Trans on Reliability Vol. R_28, No. 4, Oct. 1979.

2. R. N. Allan, I. L. Ronsiris, and D. M. Fryer, "An efficient computational technique for evaluating the cut/tie sets and common_cause failures of complex systems, "IEEE Trans on Reliability, Vol. R_30, No. 2, June 1981,pp. 101_109.

3. H. P. Alesso and H. J. Benson, "Fault tree and reliability relationships for analyzing noncoherent two_state system", Nuclear Engineering and Design, Vol. 56,1980, pp. 309_320.

4. P. K. Andow, "Difficulties in fault_tree synthesis for process plant", IEEE Trans on Reliability Vol. R_29, No. 1, April, 1980, pp. 2_9.

5. M. Astolfi, S. Contini, C. L. Van der Muyzenberg, and G. Volta, "Fault tree analysis by list_processing techniques", in Synthesis and Analysis Methods for Safety and Reliability Studies, edited by G. Apostolakis, 1978, pp. 5_32.

6. R. E. Barlow and P. Chatterjee, "Introduction to fault tree analysis". Operations Research Center, U. C. Berkeley, ORC 73_30, Dec. 1973.

7. R. E. Barlow and H. E. Lambert, "Introduction to fault tree analysis". Reliability and Fault Tree Analysis, R. E. Barlow and J. B. Fussell, editors, SIAM, 1975, pp. 7_35.

8. R. E. Barlow, and F. Proschan, "Availability theory for multi component systems", Multivariate Analysis III, P. R. Kriah_ naiah, editor, Academic Press N.Y.,1971.

9. R. E. Barlow and F. Proschan, "Importance of system compon-
-ents and fault tree analysis", Operation Research Center, Univ. of Calif., Berkeley, Rept. ORC 74-3, 1974.

10. R. E. Barlow and F. Proschan, "Statistical theory of relia-
-bility and life testing", Holt, Rinehart, and Winston, New York,1975.

11. L.Bass et al, "Fault tree graphics", in Reliability and Faul Tree Analysis, edited by R. E. Barlow and J. B. Fussell, SIAI 1975, pp. 913-927.

12. Bell Telephone Laboratories, "Launch control safety study", Section VII, Vol. 1, Bell Telephone Labs., Inc., Murray Hill, N. J. 1961.

13. N. N. Bengiamin, B. A. Bowman, and K. F. Schenk, "An efficier algorithm for reducing the complexity of computation in faul tree analysis", IEEE Trans on Nuclear Science, Vol. NS-23, Oct. 1976, pp. 1442-1446.

14. R. G. Bennetts, "On the analysis of fault trees", IEEE Trans on Reliability, Vol. R-24, No. 3, Aug. 1973, pp. 175-185.

15. L. J. Billera, "On the composition and decomposition of clutters", Journal of Combinatorial Theory, Vol. 11, pp.234-245, 1971.

16. Z. W. Birnbaum, "On the importance of different components in a multicomponent system", Multivariate Analysis II, P. R. Krishnaiah, editor, Academic Press, New York, 1969.

17. Z. W. Birnbaum, J. D. Esary and S. C. Saunders, "Multi_ component systems and structures and their reliability", Technimetrics, Vol. 3, No. 1, Feb. 1961.

18. A. Blin, A. Carline, et al., "PATREC, a computer code for fault tree calculations", in Synthesis and Analysis Method for Safety and Reliability Studies, edited by G. Apostola_ kis, pp. 33_43, 1978.

19. D. B. Brown, "Fault tree analysis", Systems Analysis and Design for Safety, Prentice_Hall, Inc., Englewood Cliffs, N.J., PP. 152_193, 1976.

20. G. R. Burdick, "COMCAN_ a computer code for common_cause analysis", IEEE Trans on Reliability, Vol. R_26, No. 2, June, 1977.

21. G. R. Burdick, N. H. Marshall, and J. R. Wilson, "COMCAN_ a computer code for common_cause analysis", ANCR_1314, May, 1976.

22. L. Caldarola, "Unavailability and failure intensity of com_ ponents", Nuclear Engineering and Design 44, pp. 147_162, 1977.

23. L. Caldarola, "Fault tree analysis with multistate compone_ nts", in Synthesis and Analysis Methods for Safety and Rel_ iability Studies, NATO, pp. 199_248, 1978.

24. L. Caldarola, "Coherent systems with multistate components", Nuc. Eng. and Des. Vol. 58, pp. 127_139, 1980.

25. L.Calderola and A. Wickenhauser, "Recent advancements in fault tree methodology at Karlsruhe" in Nuclear Systems

Reliability Engineering and Risk Assessment, J. B. Fussell and G. R. Burdick, editors, SIAM 1977.

26. L. Caldarola and A. Wickenhauser, "The Karlsruhe computer program for the evaluation of the availability and relia- bility of complex repairable systems", Nuclear Engineering and Design 43, 463-470, 1977.

27. P. Camarda, F. Corsi, and A. Trentadue, "An efficient simple algorithm for fault tree automatic synthesis from the rel- iability graph", IEEE Trans on Reliability Vol. R-27, No. 3, pp. 215-221, Aug. 1978.

28. J. M. Cargal, "An alternative fault-tree algebra", IEEE Trans on Reliability, Vol. R-29, No. 3, pp. 269-272, Aug. 1980.

29. A. Carnino, "Safety analysis using fault trees", NATO Advan- ced Study Inst. on Generic Techniques of System Reliability Assessment, 1973.

30. C. L. Cate and J. B. Fussell, "BACFIRE- a computer program for common cause failure analysis", The University of Tennessee, NERS-77-02,1977.

31. M. F. Chomow, "Directed graph techniques for the analysis of fault trees", IEEE Trans on Reliability, Vol. R-27, No. 1, pp. 7-15, April 1978.

32. P. Chatterjee, "Fault tree analysis : min cut set algorithms ORC 74-2, Operations Research Center, University of Califor- nia, Berkeley, California, Jan. 1974.

33. P. Chatterjee, "Fault tree analysis : reliability theory and systems safety analysis", Operations Research Center, U. C. Berkeley, ORC 74-34, Nov. 1974.

34. P. Chatterjee, "Modularization of fault trees : a method to reduce the cost of analysis", in Reliability and Fault Tree Analysis, SIAM, pp. 101-126, 1975.

35. T. L. Chu, G. Apostolakis, "Methods for probabilistic analysis of noncoherent fault trees", IEEE Trans on Reliability, Vol. R-29, No. 5, Dec. 1980.

36. C. E. Clark, "Importance sampling in Monte Carlo analysis", Operation Research, Sept/Oct., pp. 603-620, 1961.

37. A. G. Colombo, "Uncertainty propagation in fault tree analysis", in Failure Prevention and Reliability presented at the Design Engg. Technical Cong., Chicago, Ill, Sep. 26-28, pp. 95-103, 1977.

38. P. Crosetti, "Computer program for fault tree analysis", Douglas United Nuclear, Inc., Richard, Wash., DUN-5508, Apr. 1969.

39. P. A. Crosetti, "Fault tree analysis with probability evaluation", IEEE Nuclear Power Systems Symposium, Nov. 1970.

40. P. A. Crosetti, "Fault tree analysis for systems reliability", Instrumentation Technology, pp. 52-56, August 1971.

41. P. A. Crosetti and R. A. Bruce, "Commercial application of fault tree analysis", Proc. Reliability and Maintainability Symposium, Vol. 9, pp. 230-244, 1970.

42. G. E. Cummings, "Application of the fault tree technique to a nuclear reactor containment system", in Reliability and Fault Tree Analysis, edited R. E. Barlow et al., SIAM, Philadelphia, pp. 805-825, 1975.

43. R. L. Eisner, "Fault tree analysis to anticipate potential

failure", presented at the Design Eng. Conf. Amer. Soc. Mechanical Eng., May 8-11, 1972.

44. E. P. Epler, "Common mode failure considerations in the design of systems for protection and control", Nuclear Safety, Vol. 10, No. 1, pp. 38-45,1969.

45. R. C. Erdmann, J. E. Kelly, H. R. Kirch, F. L. Leverenz, and E. T. Rumble, "A method for quantifying logic models for safety analysis ", in Nuclear Systems Reliability Engineering and Risk Assessment, edited by J. B. Fussell et al., SIAM PP. 732-754, 1977.

46. C. A. Ericson, "System safety analytical technology- preliminary hazards analysis", the Boeing Co., Seattle, Rept. D2-113072-1, 1969.

47. J. D. Esary and F. Proschan, "Coherent structures of non-identical components", Technometrics, Vol. 5, No. 2, May 1963.

48. J. D. Esary and H. Ziehms, "Reliability analysis of phased missions". in Reliability and Fault Tree Analysis, SIAM, pp. 213-236, 1975.

49. W. Feller, "An introduction to probability theory and its applications", Vol. I, 3rd Ed. New York, John Wiley and Sons, 1968.

50 J. B. Fussell, "Fault tree analysis- concepts and techniques", NATO Advanced Study Inst. on Generic Techniques of system Reliability Assessment, 1973.

51. J. B. Fussell, "Synthetic tree model - a formal methodology for fault tree construction", ANCR-1098, March, 1973.

52. J. B. Fussell, "A formal methodology for fault tree construction", Nuclear Eng. and Design, 52, pp. 337-360,1973.

53. J. B. Fussell, "How to hand-calculate system reliability and safety characteristics", IEEE Trans on Reliability, Vol R-24, No. 3, pp. 169-174, Aug. 1975.

54. J. B. Fussell, "Computer aided fault tree construction for electrical systems", in Reliability and Fault Tree Analysis R. E. Barlow and J. B. Fussell, editors, pp. 37-56, SIAM, 1975.

55. J. B. Fussell, G. R. Burdick, D. M. Rasmuson, J. R. Wilson and J. C. Zipperer, "A collection of methods for reliability and safety engineering", ANCR-1273, 1976.

56. J. B. Fussell, E. B. Henry and N. H. Marshall, "MOCUS- a computer program to obtain minimal sets from fault trees" ANCR-1156, Aerojet Nuclear Company, Idaho Falls, Idaho, March, 1974.

57. J. B. Fussell and H. E. Lambert, "Quantitative evaluation of nuclear system reliability and safety characteristics", IEEE Trans on Reliability Vol. R-25, No. 3, pp 178-183, Aug. 1976.

58. J. B. Fussell, G. J. Powers and R. G. Bennetts, "Fault trees - a state of the art discussion", IEEE Trans on Reliability R-23, No. 1, pp. 51-55, April 1974.

59. J. B. Fussell and W. E. Vesely, "Elements of fault tree construction_ a new approach", Trans Amer. Nuc. Soc. pp. 794, 1972.

60. J. B. Fussell and W. E. Vesely, "A new methodology for obtaining cut sets for fault trees", Trans ANS, Vol. 15, p. 262, 1972.

61. A. C. Gangadharan, M. S. M. Rao and C. Sundarajan, "Com_puter methods for qualitative fault tree analysis", in Failure Prevention and Reliability, edited by S. B. Bennett et al., pp. 251_262, 1977.

62. S. Garribba et al., "DICOMICS, an algorithm for direct co_mputation of minimal cut sets of fault trees", EUR_5481e, 1975.

63. S. Garribba et al., "Efficient construction of minimal cut sets from fault trees"?, IEEE Trans on Reliability , Vol. R_26, No. 2, Jun. 1977.

64. B. J. Garrick, "Principles of unified system safety analy_sis", Nuclear Engineering and Design 13, pp. 245_321, 1970.

65. W. Y. Gately, D. W. Stoddard and R. L. Williams, "GO, a computer program for the reliability analysis of comples systems", Daman Science Corporation, Colorado Springs, Col_lorado, KN_67_704(R), April, 1968.

66. C. W. Griffin, "The fault tree as a safety optimization design tool", presented at the Topical Meeting on Water_Reactor Safety, Mar. 1973.

67. D. F. Haasl, "Advanced concepts on fault tree analysis", System Safety Symposium, The Boeing Company, Seattle, Washington, June 8_9, 1965.

68. W. Hammer, "Fault tree analysis", Handbook of System and Product Safety, Prentice_Hall, Inc. Englewood Cliffs, N.J. pp. 238_246, 1972.

69. W. Hammer, "Fault tree analysis", Product Safety Management and Engineering, Prentice_Hall, Inc.,Englewood Cliffs, N.J. pp. 204_228, 1975.

70. E. J. Henley and H. Kumamoto, "Comment on : computer_aided synthesis of fault trees", IEEE Trans on Reliability, Vol. R_26, pp. 316_317, Dec. 1977.

71. B. L. Hulme and R. B. Worrell, " A prime implicant algori_thm with factoring", IEEE Trans on computers, Vol. C_24, pp 1129_1131, Nov. 1975.

72. H. E. Kongsoe, "RELY 4 : a Monta Carlo computer program for systems reliability analysis", Danish Atomic Energy Commis_sion, RISØ_M_1500, June 1972.

73. H. E. Kongsoe, "REDIS, a computer program for system relia_bility analysis by direct simulation", International Sympo_sium on Reliability of Nuclear Power Plants, Insbruck, Aus_trin, April 14_18, 1975.

74. H. Kumanoto and E. J. Henley, "Top_down algorithm for obt_aining prime implicant sets of noncoherent fault trees", IEEE Trans on Reliability, Vol. R_27, pp. 242_249, Oct. 1978.

75. H. E. Lambert, "System safety analysis and fault tree analysis", UCID_16238, Lawrence Livermore Lab. , Livermore, California, May, 1973.

76. H. E. Lambert, "Fault trees for decision making in system analysis", Lawrence Livermore Laboratory, University of California, Livermore, UCRL_51829, Oct. 1975.

77. H. E. Lambert, "Measures of importance of events and cut sets in fault trees",in Reliability and Fault Tree Analysis edited by R. E. Barlow et al., SIAM , pp. 77_100, 1975.

78. H. E. Lambert, "Comment on the Lapp_Powers computer_aided synthesis of fault trees", IEEE Trans on Reliability, Vol. R_28, No. 1, pp. 6_9, April 1979.

79. E. E. Lewis, "Fault trees", Nuclear Power Reactor Safety, John_Wiley and Sons, N.Y., pp. 87_91, 1977.

80. S. A. Lapp and G. J. Powers, "Computer_aided synthesis of fault trees", IEEE Trans on Reliability, pp. 2_13, April 1977.

81. S. A. Lapp and G. J. Powers, "The synthesis of fault trees" in Nuclear Systems Reliability Engineering and Risk Assessment, edited by J. B. Fussell and G. R. Burdick, SIAM, pp. 778_799, 1977.

82. S. A. Lapp and G. J. Powers, "Update of Lapp_Powers fault tree synthesis algorithm". IEEE Trans on Reliability, Vol. R_28, No. 1, pp. 12_14, April 1979.

83. M. O. Locks, "Synthesis of fault trees:an example of noncoherence", IEEE Trans Reliability , Vol. R_28, pp. 2_5, Apr. 1979.

84. M. O. Locks, "Fault trees, prime implicants and noncoherence ", E. I. Ogunbiyi,"Author reply #1", H. Kumamoto and E. J. Heneley"Author reply #2", M. O. Locks, "Rebuttal", IEEE Trans on Reliability, Vol. R_29, pp. 130_135, June 1980.

85. M. S. Madhava Rao, "FALTREE_ a computer program for fault tree analysis", Engineering Science and Technology Dept. Letter Report, EST_77_1, Foster Wheeler Development Corporation, Livingston, N. J. 1977.

86. S. W. Malasky, "Fault tree analysis". System Safety, Hayden Book Co. Inc., Rochelle Park, N. J. pp. 142_194, 1974.

87. A. W. Marshall and I. Olkin, :A multivariate exponential distribution", JASA, 62, pp. 30_44, 1967.

88. M. Mazumdar, "Importance sampling in reliability estimation ", Reliability and Fault Tree Analysis (editors : R. E. Barlow et al.,) SIAM 1975.

89. C. W. Mcknight, et al., "Automatic reliability mathematical model", North American Aviation, Inc., Downey, California, NA 66_838, 1966.

90. K. Nakashima and Y. Hattori, "An efficient bottom_up algorithm for enumerating minimal cut sets of fault trees", IEEE Trans on Reliability, Vol. R_28, Dec. 1979.

91. D. Nielsen, "Use of cause_consequence charts in practical system analysis", in Reliability and Fault Tree Analysis, edited by R. E. Barlow and J. B. Fussell, SIAM, pp. 849_880, 1975.

92. J. Olmos and L. Wolf, "A moduler representation and analysis of fault trees", Nuclear Engineering and Design, Vol. 48, Aug 1978.

93. P. K. Pande, M. E. Spector and P. Chatterjee, "Computeri_zed fault tree analysis " TREEL and MICSUP", ORC 75_3, Ope_ration Research Center, University of California, Berkeley, April 1975.

94. S. L. Pollack, "Decision tables : theory and practice", Wiley_Interscience, N. Y. 1971.

95. G. M. Powers and F. C. Tompkins, "Computer_aided synthsis of fault trees for complex processing systems", NATO Adva_nced Study Inst. on Generic Techniques of System Reliability Assessment, pp. 307_314, 1973.

96. G. J. Powers and F. C. Tompkins, "Fault tree synthesis for chemical process", AICHE Journal, Vol. 20, No. 2, pp. 376_387, March 1974.

97. G. J. Powers, F. C. Tompkins and S. A Lapp, "A safety simulation language for chemical processes : a procedure for fault tree synthesis", in Reliability and fault tree analysis, R. E. Barlow and J. B. Fussell, editors, pp. 57-75, SIAM 1975.

98. D. M. Rasmuson and N. H. Marshall, "FATRAM_ a core efficien cut_set algorithm ", IEEE Tran. on Reliability, Vol. R_27, No. 4, pp. 250_253, Oct. 1978.

99. Reactor Safety Study_ An Assessment of Accident Risk in U. S. Commercial Nuclear Power Plants, WASH_1400 (NUREG_75/014), U. S. Nuclear Regulatory Commission, Washington, D.C., 1975, October.

100. J. L. Recht, "System safety analysis : the fault tree", National Safety News, April 1966.

101. A. Rosenthal, "Decomposition Methods for Fault Tree Analysis IEEE Trans on Reliability, Vol. R_29, No. 3, June 1980.

102. E. T. Rumble, F. L. Leverenz and R. C. Erdmann, "Generalized fault tree analysis for reactor safety", Electric Power Research Inst., Palo Alto, California, EPRI_217_2_2, June 1975.

103. N. H. Roberts, "Mathematical models in reliability engineer_ing", McGraw_Hill, N.Y.,p243, 1964.

104. S. L. Salem and G. Apostolakis, "The CAT methodology for fault tree construction", in Synthesis and Analysis Method for Safety and Reliability Studies, edited by G. Apostolakis et al., NATO, pp. 109_128, 1978.

105. S. L. Salem, G. E. Apostolakis and D. Okrent,"A computer_oriented approach to fault tree construction", EPRI NP_288, Electric Power Reaearch Institute, Nov. 1976.

106. S. L. Salem, G. E. Apostolakis and D. Okrent, "A new metho_dology for the computer_aided construction of fault tree", Annals of Nuclear Energy, Vol. 4, pp. 417_433, 1977.

107. S. L. Salem, J. S. Wu and G. E. Apostolakis, "Decision tabl development and application to the construction of fault trees", Nuclear Technology, Vol. 42, pp. 51-64, Jan. 1979.

108. G. H. Sandler, "System reliability engineering", McGraw-Hill, N.Y., p. 243, 1964.

109. R. J. Schroder, "Fault tree for reliability analysis", Proc 1970 Annual Symposium on Reliability, Feb. 3-5, Los Angeles pp. 198-205, 1970.

110. S. N. Semanderes, "ELRAFT, a computer program for the efficient logic reduction analysis of fault trees", IEEE Trans on Nuclear Science, Vol. NS-18, No. 1, pp. 481-487, Feb. 1971.

111. M. L. Shooman, "Probabilistic reliability : an engineering approach", McGraw-Hill Book Company, 1968.

112. C. O. Smith, "Introduction to reliability in design", N.Y. McGraw-Hill, 1976.

113. J. R. Taylor, "A formalisation of failure mode analysis of control systems", Danish Atomic Energy Commission, RISØ-M-1654, Sept. 1973.

114. J. R. Taylor, "A study of failure causes based on U. S. power reactor abnormal occurrence reports", Reliability of Nuclear Power Plants, IAEA-SM-195/16, 1975.

115. J. R. Taylor, "Sequential effects in failure mode analysis" in Reliability and Fault Tree Analysis, pp. 881-894, SIAM 1975.

116. J. R. Taylor and E. Hollo, "Algorithm and programs for consequence diagram and fault tree construction", Rept. No. RISØ_M_1907, Danish Atomic Energy Commission, Roskilde, Denmark, 1977.

117. J. R. Taylor and E. Hollo, "Experience with algorithms for automatic failure analysis", in Nuclear Systems Reliability Engineering and Risk Assessment, edited by J. B. Fussell and G. R. Burdick, pp. 759_777, SIAM 1977.

118. W. J. Van Slyke and D. E. Griffing, "ALLCUTS, a fast comprehensive fault tree analysis code", Atlantic Richfield Hanford Company, Ricland, Washington, ARH_ST_112, July 1975.

119. W. E. Vesely, "Analysis of fault trees by kinetic tree theory", IN_1330, Idaho Nuclear Corp., Idaho Falls, Oct., 1969.

120. W. E. Vesely, "A time_dependent methodology for fault tree analysis", Nucl. Engr. and Design, Vol. 13, No. 2, pp. 337-360, August 1970.

121. W. E. Vesely, "Reliability and fault tree applications at NRTS", Proc. 1970 Reliability and Maintanability Symposium Vol. 9, pp. 472_480, 1970.

122. W. E. Vesely, "Reliability quantification techniques used in the Rasmussen study", in Reliability and Fault Tree Analysis, SIAM, pp. 775_803, 1975.

123. W. E. Vesely, "Time dependent unavailability analysis of nuclear safety system", IEEE Trans on Reliability, Vol. R_26, No. 4, pp. 257_260, Oct. 1977.

124. W. E. Vesely, "Estimating common cause failure probabilities in reliability and risk analysis : Marshall-Olkin specialization", in Nuclear Systems Reliability Engineering and Risk Assessment, pp. 314-341, SIAM 1977.

125. W. E, Vesely and R.E. Narum, "PREP and KITT computer code for the automatic evaluation of a fault tree", Idaho Nuclear Corporation, Idaho Falls, Kdaho, IN-1349, 1970.

126. D. P. Wagner, C. L. Cate and J. B. Fussell, "Common cause failure analysis methodology for complex systems",in Nuclear Systems Reliability Engineering and Risk Assessment edited by J. B. Fussell and G. R. Burdick, pp. 289-313, SIAM 1977.

127. D. B. Wheeler et al., "Fault tree analysis using bit manipulation", IEEE Trans on Reliability, Vol. R-26, No. 2, pp. 95-99, June 1977.

128. R. L. William and W. Y. Gateley, "Use of the GO methodology to directly generate minimal cut sets", in Nuclear Systems Reliability Engineering and Risk Assessment, edited by J. B. Fussell and G. R. Burdick, pp. 825-849, SIAM 1977.

129. R. R. Willie, "Computer-aided fault tree analysis : FTAP", Operations Research Center, U. C. Berkeley, ORC 78-14, Aug. 1978.

130. E. R. Woodcock, "The calculation of reliability of systems : the program NOTED", UKAEA Authority Health and Safety Branch, Risley, Warrington, Lancashire, England, AHSB(S) R. 153, 1971.

131. P. Y. Wóng, "FAUTRAN_ a fault tree analyzer", AECL_5182, Atomic Energy of Canada Limited, Chalk River Nuclear Lab. Chalk River, Ontario, Canada, 1975.

132. R. B. Worrell, "Set equation transformation system (SETS)" SLA_73_0028A Sandia Laboratories, Albuquerque, New Mexico, May 1974.

133. R. B. Worrell, "Using the set equation transformation system in fault tree analysis", in Reliability and Fault tree analysis, SIAM Conference Volume PP. 165_185, 1975.

134. R. B. Worrell, "Qualitative analysis in reliability and safety studies", IEEE Trans on Reliability, Vol. R_25, No. 3, August 1976.

135. R. B. Worrell, D. W. Stack and B.L. Hulme, "Prime implicants of noncoherent faiut trees",IEEE Trans on Reliability, Vol. R_30, No. 2, pp. 98_100, June 1981.

136. J. S. Wu, S. L. Salem and G.E. Apostolakis, "The use of decision tables in the systematic construction of fault trees", in Nuclear Systems Reliability Engineering and Risk Assessment, edited by J. B. Fussell and G. R. Burdick, pp. 800_824, SIAM 1977.

137. T. W. Yellman, "Comment on computer_aided synthesis of fault trees", IEEE Trans on Reliability, Vol. R_28, No. 1, pp. 10_11, April 1979.

138. J. Young,"Using the fault tree analysis technique", Rel_iability and Fault Tree Analysis, R. E. Barlow and J. B. Fussell, editors, pp. 827_848, SIAM 1975.

A STUDY OF FAULT TREE ANALYSIS FOR SYSTEM SAFETY AND RELIABILITY

by

WEN_SHING LEE

B.S. (Industrial Management), National Cheng_Kung University,
Tainan, Taiwan, 1973

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1982

ABSTRACT

This study presents a review and classification of fault tree analysis methodologies developed in the past two decades for the interests of system safety and reliability.

A state-of-the-art review of the literature related to fault tree analysis is presented in chapter 2. Chapter 3 describes in detail the current use fault tree construction methodologies in the U.S. and elsewhere. Chapter 4 discusses the methodologies of qualitative fault tree evaluation of both finding minimal cut (path) sets and common cause failure analysis. Chapter 5 illustrates the major quantitative fault tree analysis techniques for probabilistic evaluation of fault trees and measures of importance of events and cut sets. A brief summary and possible future study is given in chapter 6.

Fault tree analysis has proved to be a useful analytical tool for the reliability and safety analysis of complex systems. The literature on fault tree analysis is, for the most part, scattered through conference proceedings and company reports. Therefore, we feel that a readable, logical introduction to this subject is very much needed.