① Pgs 58
Ref 24

A FAULT TREE MANUAL

RESEARCH REPORT

Presented in Partial Fulfillment of the Requirements
For the Degree Master of Engineering, Industrial
Engineering Department of Texas A&M University

by

Thomas W. DeLong

Approved by

W L Johnston
Dr. W.L. Johnston
Co-Chairman

A W Wortham
Dr. A.W. Wortham
Co-Chairman

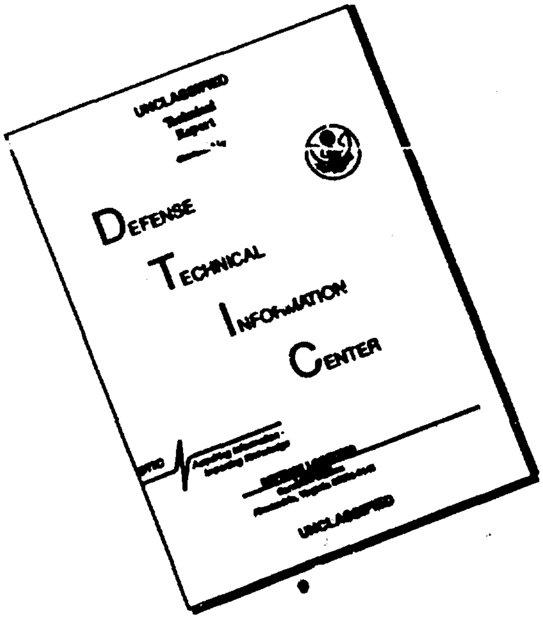
Vernon
Dr. R.J. Vernon

Elliott
Dr. R.W. Elliott

Texas A&M University

1970

DISCLAIMER NOTICE



UNCLASSIFIED

Technical Report

DEFENSE

TECHNICAL

INFORMATION

CENTER

UNCLASSIFIED

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

ABSTRACT

This paper is intended to give the reader a basic background in the system safety technique of fault tree analysis and related concepts. Four example problems illustrate the construction of the tree, combination of probabilities, and use of Boolean algebra. The description of simulation techniques provides a bridge from simple example problems to actual complex systems.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

USAMC Intern Training Center - USALMC
Red River Army Depot ATTN: AMXMC-ITC-E-S
Texarkana, Texas 75501

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

N/A

3. REPORT TITLE

A FAULT TREE MANUAL

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

N/A

5. AUTHOR(S) *(First name, middle initial, last name)*

Thomas W. DeLong

6. REPORT DATE

Dec 70

7a. TOTAL NO. OF PAGES

58

7b. NO. OF REFS

24

8a. CONTRACT OR GRANT NO.

N/A

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

N/A

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

N/A

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES

N/A

12. SPONSORING MILITARY ACTIVITY

Safety Office
Headquarters, US Army Materiel Command
Washington, D. C. 20315

13. ABSTRACT

This paper is intended to give the reader a basic background in the system safety technique of fault tree analysis and related concepts. Four example problems illustrate the construction of the tree, combination of probabilities, and use of Boolean algebra. The description of simulation techniques provides a bridge from simple example problems to actual complex systems.

ACKNOWLEDGMENTS

I wish to thank Dr. W. L. Johnston for help and guidance throughout this project. Dr. Johnston was a vital and necessary influence from start to finish.

Also, I was privileged and grateful to receive assistance from Dr. R. J. Vernon, one of the most inspiring teachers I have known.

CONTENTS

CONTENTS (continued)

FIGURES

TABLES

CHAPTER I

INTRODUCTION

The problems involved in the design of a complex system are almost overwhelming. A successful design effort includes accurately predicting characteristics such as reliability and performance, and skillfully anticipating problems of maintenance and safety. This paper describes one of the safety analysis techniques used in the design process. The technique, called fault tree analysis, is a valuable addition to other methods used in pre-production safety studies.

Safety analysis techniques include investigation of all energy sources for possible hazards, study of subsystem interfaces, alarm systems, and other parts of hardware for known hazards. Fault tree analysis uses a different approach. Instead of being hardware orientated, fault tree analysis is failure orientated; that is, it examines a particular system failure for all possible causes. Control of the system failure through knowledge of its causes is the analysis objective.

Like other techniques, fault tree analysis begins at the conception stage and is completed before production begins so that safety improvements are designed into the system. Two outstanding strengths are: the clarity of subsystem interrelation shown by the tree, and the fact that the tree can be quantified. These two strengths will become obvious in the examples of Chapter V.

The fault tree technique was developed eight years ago and has been used in aerospace and military applications since then. The analysis is now taught in several system safety courses, such as those given periodically by the National Safety Council, Picatinny Arsenal, and the University of Washington. In addition, many excellent published articles give descriptions of the technique, but some are very general, others primarily describe the probabilities, or perhaps only the Boolean aspects of the mathematics. Few are well rounded and comprehensive, therefore it is difficult to get a good overall explanation of fault tree analysis by reading a single article. This manual presents a general basic background of the concept and associated techniques, including steps in construction of a tree, quantification with probabilities, math simplification of probabilities by classical methods and by Boolean algebra, and computer applications. This paper is designed to give a good fault tree background to safety engineering students.

Although the concepts of fault tree analysis are simple, the applications are complex. The Army Materiel Command will use the technique for analysis of systems having research and development and test and evaluation budgets of several million dollars. The examples in this manual are simple, yet they serve the purpose of this paper by presenting techniques necessary for larger trees.

At this point, a description of the tree may help orientate the reader (see Figure 1). The tree is a graphical representation of possible causes of a major failure which appears at the top of the tree. During construction, the tree grows downward and outward as failures and causes are described in increasing detail. When the tree is completed, probabilities are associated with the failures lowest on the tree. The assigned probabilities are combined as dictated by logic gates to give probabilities for events higher on the tree. The combination of probabilities continues until the complex top event has a probability calculated from the more accurate component data at the bottom of the tree. Two kinds of reasoning are involved: the thought processes involved in construction produce a downward flow, whereas the evaluation of probability and operation of the logic gates dictate an upward flow. Further discussion of the nature of the tree is postponed until Chapter III.

Chapter II considers the history of the technique and comments on references. Following that background information, Chapter III describes steps in construction of a tree. The two tree components, events and gates, are described verbally and illustrated graphically. Chapter IV explains math techniques including probability, approximations, and Boolean algebra. The four examples in Chapter V illustrate construction of a tree from a verbal description of a system, and the application of the math

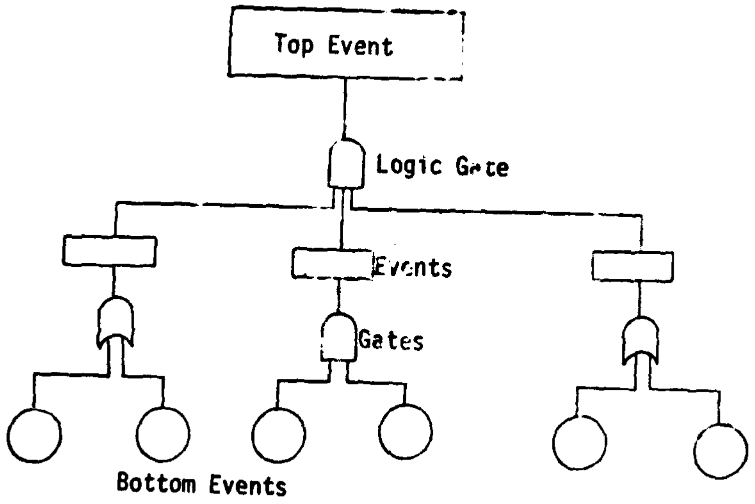


Figure 1. Example Tree

techniques described in Chapter IV. Chapter VI is a bridge to complex problems requiring computer simulation.

CHAPTER II

LITERATURE REVIEW

The fault tree method resulted from a contract between the Air Force Ballistics Systems Division and Bell Telephone Laboratories for the study of inadvertant launch in the Minuteman ICBM. The Launch Control Safety Study (1962) first described fault tree analysis in Volume I Section VII "Method of Inadvertant Launch Control Analysis". Minuteman I was in production when the study was completed, therefore no design changes resulted from the study, but results were so close to observed data that the technique was used in design of Minuteman II. Since then, fault tree analysis has been used in combination with other techniques to predict and improve safety performance in complex aerospace and military systems.

After initial work at Bell Telephone Laboratories, development of fault tree continued at the Boeing Company, where the technique was applied to manned aircraft and simulation techniques were used extensively. Boeing and AVCO published fault tree reports on the Minuteman II system in March 1963 and January 1964 respectively. In June of 1965, Boeing and the University of Washington co-sponsored a System Safety Symposium in Seattle. Five of the presentations were fault tree articles by Boeing employees

(Feutz, Haasl, Kanda, Michels, and Nagel). A paper by Mearns of Bell Telephone Laboratories also described fault tree. These six papers and the Launch Control Safety Study are the main references cited in articles after 1965. (All articles mentioned in the text are referenced in the Bibliography. The Proceedings of the System Safety Symposium contain all the talks of the Symposium, including the six fault tree articles.)

Fault tree articles range widely in complexity and comprehensiveness. Recht's article (1966) reprinted from the National Safety News by the National Safety Council, is an excellent non-mathematical introduction to the technique. The articles by Hiltz (1965) and by Larsen (1968) both present Boolean algebra very early in the discussion, then use only AND and OR gates, which correspond to the Boolean operations lf "*" and "+". Hixenbaugh's article (1968) is an excellent reference for simulation processes and very specialized logic gates mentioned in no other references of the Bibliography. The paper by McDivitt, Goldberg, and Cornwell (1968) is an excellent example of application to a real system. The paper considers accident stimuli and damage potential in what the authors designate a hazard tree of failures associated with large rocket motors.

A recent suggestion to change the name of the technique from fault tree analysis to perhaps causal tree should be mentioned here. Since the purpose of analysis is not to find fault, but to systematically reduce causes of failures or hazards, perhaps several years from now causal tree will be the accepted name of the technique. This paper will use the term fault tree, which is used in the references, except for a few cases of hazard tree, which includes information about the degree of the hazard as well as the causes.

As in industry, use of the fault tree in the Army Materiel Command is still rather limited. The U. S. Army Missile Command uses the analysis much more than any other major commodity command. In the future, Army contracts will specify that fault tree analysis be done on complex systems of any type, missile, automotive, electronic, etc. This trend toward specification of safety in general and fault tree analysis specifically means that the safety engineers of the Army Materiel Command must be familiar with the use of various analysis techniques. These men serve a unique role by specifying safety analyses and performance in contracts, then reviewing engineering reports submitted according to those contracts.

This manual is intended to aid in the training of safety engineers of the Army Materiel Command by presenting many concepts associated with fault tree analysis. The first concept, of course, is the tree itself, discussed in the next chapter.

CHAPTER III

CONSTRUCTION OF A FAULT TREE

1. Introduction

The six steps of fault tree analysis given by Hixenbaugh (1968) are discussed in this and the following chapter. The steps are:

1. Define the event to be investigated by the tree
2. Gain an understanding of the system
3. Construct the tree
4. Collect quantitative data
5. Evaluate the probability of the event chosen in step 1
6. Analyze computer output

Steps one through four are discussed in this chapter, step five in the next, and techniques from both chapters are illustrated in the examples of Chapter V. Chapter VI discusses the sixth step. Before continuing with the elaboration of these steps, the reader should look at the structure of the samples in Chapter V to note the symbols, events, and probabilities.

2. Define the Event

The top event of the tree should be well defined in terms of operating mode of the system, environmental conditions and time limits. However, the failure must represent a major system malfunction which threatens personnel or equipment. The undesired failure in the original fault tree was an inadvertant launch of the Minuteman Intercontinental Ballistic Missile. Other possible examples are failure of a plane to remain on the runway before takeoff, or as in example one, failure to perform the system objective.

Care in choosing the event will be fruitful because the tree paves the way for improving the safety of the system (reducing the probability of the chosen failure) through redesign of subsystems and components which the analysis indicates are most likely causes of failure. Hence the only subsystems investigated are possible contributors to the failure chosen in this step. Selection of a minor failure in this step would exclude much of the system from scrutiny and the analysis would have lower significance.

3. Understand the System

The second step is understanding the system as it functions under conditions specified in step one. This includes the component interaction, hardware requirements, and necessary software. Several procedures helpful in this step are: listing the sequence of events necessary for successful operation of the system; enumeration of performance requirements and safety regulations; listing of failure modes and probabilities for each component; and construction of block diagrams of the system and system functions. Producing these lists often require design men from several disciplines including electronics, hydraulics, and mechanics. Often several different companies will be involved as subcontractors. This step is a good reminder of the magnitude of the effort and co-ordination required in fault tree analysis.

4. Construct the Tree

Tree construction is a logical process producing a graphic display of events such that all possible causes of a particular failure are shown below that failure. This process starts at the top of the tree, with the failure chosen in step one described in terms of major subsystem failures shown immediately below. These subsystem failures are further subdivided and explained in greater detail until the bottom of the tree is reached. The bottom events are failures of basic components which can be associated with a probability.

Since the fault tree is graphical as well as verbal, and several people at separate locations and at different times may contribute, generally accepted symbols are necessary to represent differences in events and logic relationships. The following sections describe events, logic gates, and special symbols, all of which are illustrated in Figures 2 through 5.

4.1 Events: Four kinds of events are represented by the four symbols in Figure 2. A circle represents a clearly defined failure of a basic component. In contrast to the clarity represented by the circle is the uncertainty associated with a diamond

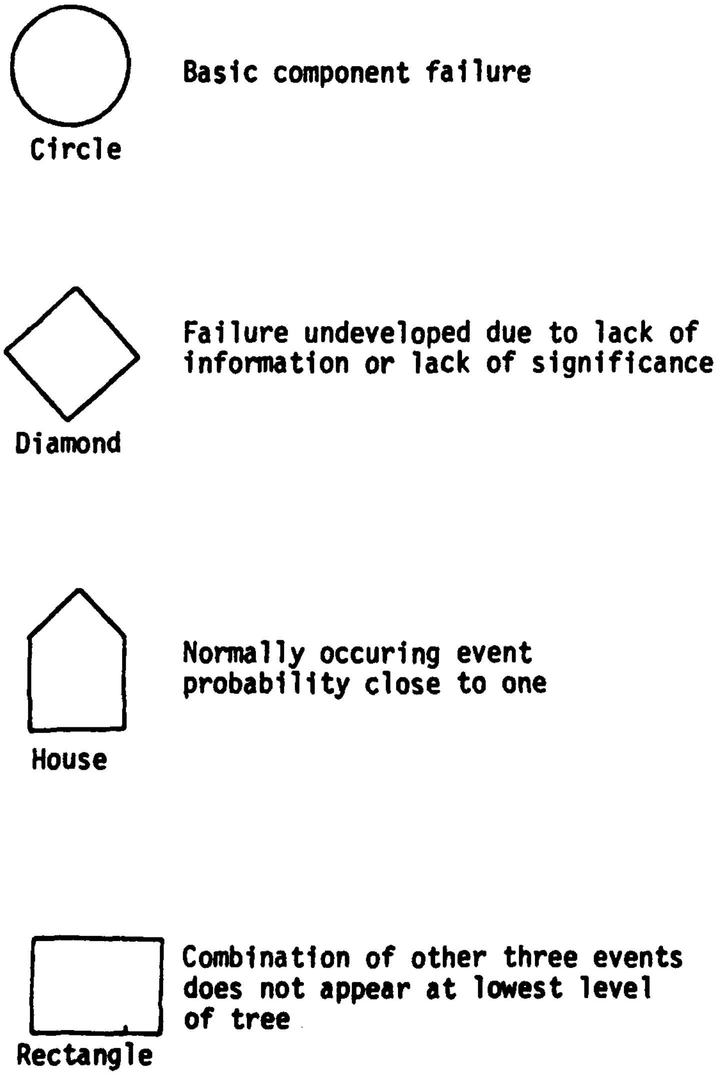

Basic component failure
Circle
Failure undeveloped due to lack of information or lack of significance
Diamond
Normally occuring event probability close to one
House
Combination of other three events does not appear at lowest level of tree
Rectangle

Figure 2. Events

event, which is a failure not well understood because of absence of information or significance. Normal, frequently occuring events, are symbolized by a house-shaped figure. Finally, several events combined together by a logic gate form a combination event represented by a rectangle.

4.2 Logic Gates: Many different logic gates are used to combine events, but three simple ones are sufficient. These three, AND, OR and INHIBIT are illustrated in Figure 3 with their electric switch analogy. Note that the inputs enter from below and the output comes from the top of the gate. The AND gate produces an output if all of the inputs exist simultaneously. The OR gate produces an output when at least one of the input conditions occur. These two gates are the same as ordinary usage of the words "and" and "or". The INHIBIT gate produces an output when the input is present and the specified condition exists. In other words, the output is "inhibited" by lack of the stated condition. The INHIBIT gate can be compared to FORTRAN's logical IF statement. The FORTRAN statement "IF (A EQ. B) GO TO 1030" states that if the condition A equals B is satisfied, go to statement number 1030. If the condition is not satisfied, continue in normal sequence.

A RANDOM EXHIBIT gate is a variation of the INHIBIT gate. In this variation, the functional condition is replaced by a probability that the input will produce an output. The probability is stated in an ellipse attached to the hexigon.

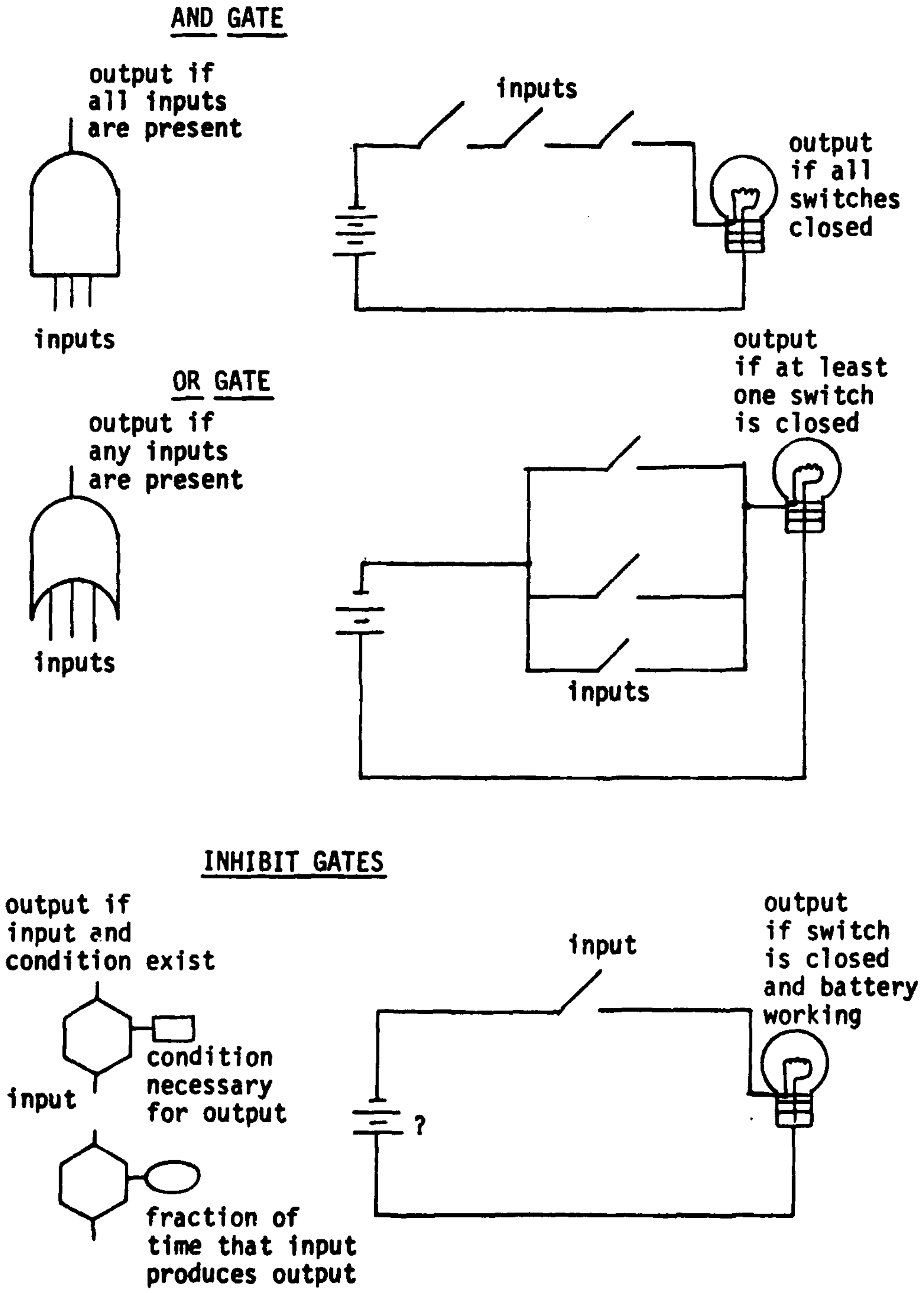

AND GATE
output if all inputs are present
inputs
inputs
output if all switches closed
OR GATE
output if any inputs are present
inputs
output if at least one switch is closed
inputs
INHIBIT GATES
output if input and condition exist
condition necessary for output
input
fraction of time that input produces output
input
output if switch is closed and battery working
?

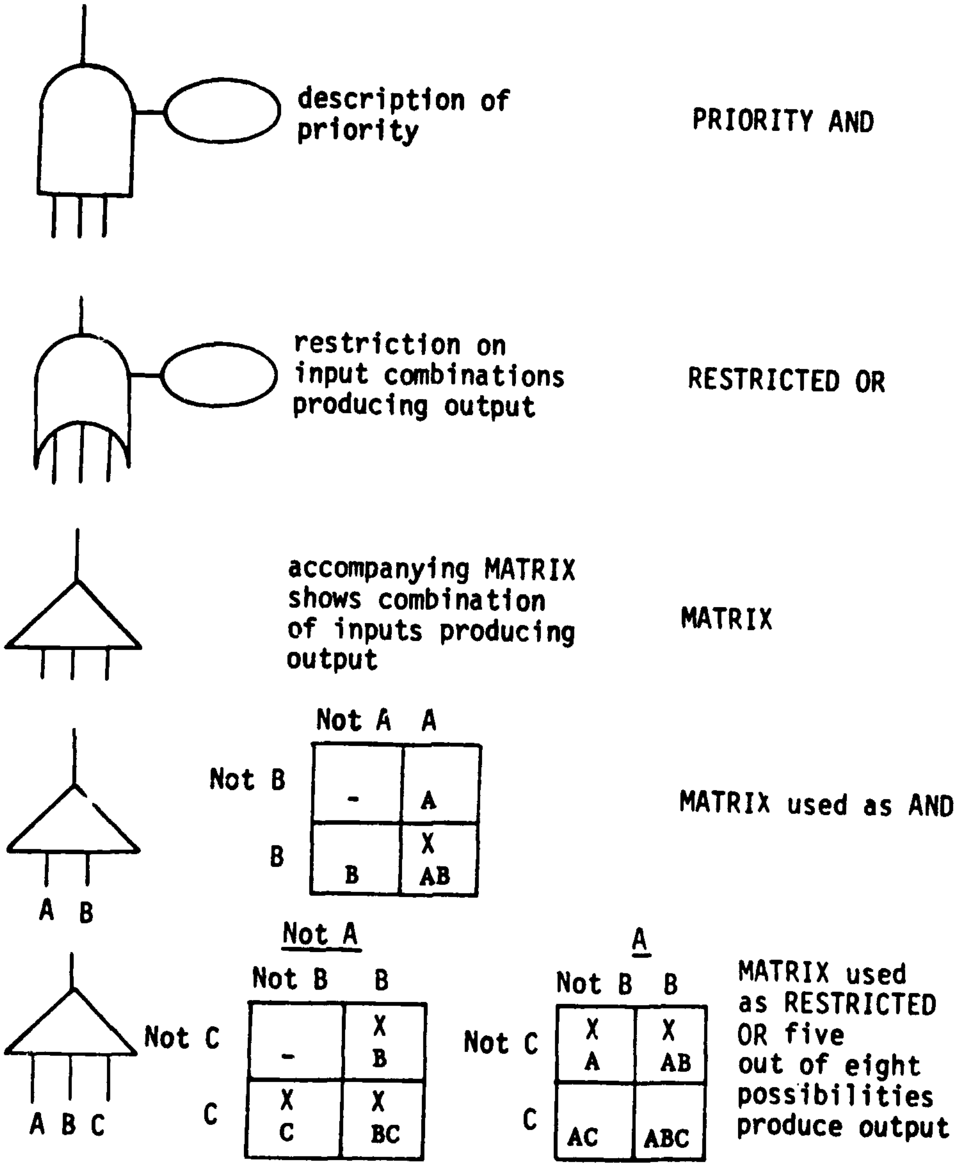
Figure 3. Basic Gates and Electric Switch Analogies

For many situations, the three primary gates are awkward, although sufficient. Several modifications, created for convenience, are shown in Figure 4 and described below.

The PRIORITY AND gate puts a restriction on the sequence of inputs to an AND gate which produce an output. An excellent example is a device which requires a pulse and a direct current voltage to produce an output. If the pulse occurs (voltage rises and falls) before the direct current is turned on, no output is produced because the inputs did not co-exist. On the other hand, if the voltage is applied before the pulse, an output occurs. A PRIORITY AND gate would specify that the direct current voltage must come before the pulse for output.

The OR gate is modified by restricting certain specified combinations of inputs from producing outputs. A general form of the RESTRICTED OR gate is called the MATRIX gate, which produces an output for input combinations shown in an accompanying matrix. At the bottom of Figure 4, the MATRIX gate is used as an RESTRICTED OR gate for which no output occurs for the three conditions: no input, all three inputs, and inputs A and C without B. The MATRIX gate is also shown being used as an AND gate for two inputs.

4.3 Special Symbols: Shown in Figure 5, three special symbols representing parts of trees are used to reduce redundancy. These are the last set of symbols presented for construction of a


description of priority
PRIORITY AND
restriction on input combinations producing output
RESTRICTED OR
accompanying MATRIX shows combination of inputs producing output
MATRIX
MATRIX used as AND
A B
MATRIX used as RESTRICTED OR five out of eight possibilities produce output
A B C

	Not A	A
Not B	-	A
B	B	X AB

Not A

	Not B	B
Not C	-	X B
C	X C	X BC

A

	Not B	B
Not C	X A	X AB
C	AC	ABC

Output for

A, B, C, A & B, or B & C

No output for

no input, A & C or all three inputs

Figure 4. Modified Gates

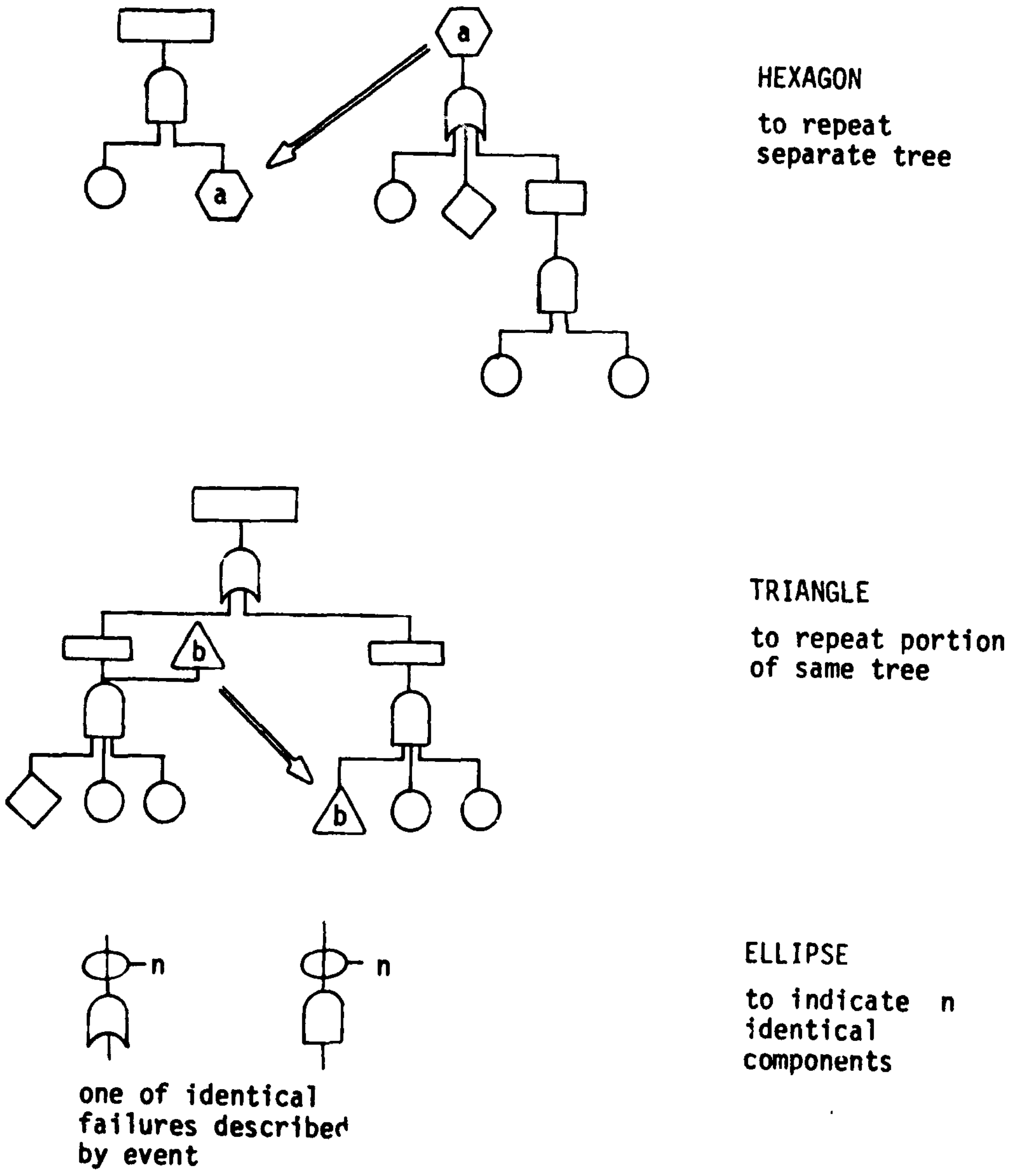

a
a
HEXAGON
to repeat separate tree
b
b
TRIANGLE
to repeat portion of same tree
n
n
ELLIPSE
to indicate n identical components
one of identical failures described by event

Figure 5. Special Symbols

tree.

The hexagon refers to another fault tree which is substituted where the symbol appears. A good use for this symbol would be in example one of Chapter V at the diamond which states "receiver fails". This failure would be elaborated in a complex fault tree headed by a hexagon. The diamond of example one would be changed to a hexagon and both six sided figures would bear the same letter.

To repeat another portion of the same tree, a pair of triangles is used as illustrated. The portion of the tree below the triangle on the left is substituted at the point where the triangle on the right appears.

The last special symbol indicates identical components either in series or parallel. In this case only one component is mentioned and the redundancy is shown by an ellipse around the output. The number of components is written beside the symbol.

5. Collect Quantitative Failure Data

The above symbols and gates are sufficient for construction of any elementary fault tree. The next analysis step is the collection of probabilities for the lowest events of the tree. These probabilities come from experience, tests, published data (such as MIL HDBK 217A Reliability Stress and Failure Rate Data for Electronic Equipment, U.S. Government Printing Office) or engineering judgment. These probabilities are combined up the tree by logic gates to calculate probabilities for each event. Techniques of combining and simplifying probabilities are covered in the next chapter.

CHAPTER IV

MATHEMATICS OF SIMPLIFICATION

1. Introduction

After probabilities are assigned to the bottom events, the next task is to combine the probabilities until the top event is reached. This chapter discusses two general categories of combination techniques. The first, applicable to independent events, consists of classical probability techniques in which the probability of the combination event depends on the logic gate as well as the input probabilities. The second method, Boolean algebra, can be used only with AND and OR gates. Boolean algebra is useful for eliminating events which appear more than once on the lowest level of the tree.

2. Probability Techniques

Consider the AND gate. If A and B are independent[1] input events and event A occurs with probability P(A) and B occurs with probability P(B), the probability of output from the AND gate is P(A)*P(B). This is the probability of both inputs

[1] Independence means the occurence of event A will not alter P(B) and the occurence of B will not alter P(A).

occuring. The situation is illustrated with a Venn diagram[2] (top of Figure 6) of unit area containing overlapping circles A and B. The area of circle A is P(A), likewise circle B has area P(B). The independence condition specifies the area of overlap (cross hatched area in the Venn diagram) is P(A)*P(B).

An OR gate with independent inputs A and B (again with probabilities P(A) and P(B)) produces an output when one or the other or both inputs occur. The probability of output is P(A) + P(B) - P(A)*P(B). Referring to the same Venn diagram used above, the area within the heavy outer line is the output probability. The cross hatched area of overlap is subtracted in the output expression because it is counted twice, once in P(A) and once in P(B).

An approximation is used for the output of an OR gate when the input failure probabilities are small. For small P(A) and P(B) the pro ıct P(A)*P(B) is a corresponding order of magnitude smaller than either input, and the product can be ignored with small error. Table 1 shows the approximation for probabilities of 1% and 1/10%. Clearly the approximation is better for small probabilities. In any event, the approximation is larger than the actual value, hence the simpler form will not give false optimistic values.

[2] Consult an algebra or statistics text for a description of Venn diagrams, for example appendix B of Concepts of Probability Theory by Paul E. Pfeiffer, McGraw-Hill Book Company, New York, 1965.

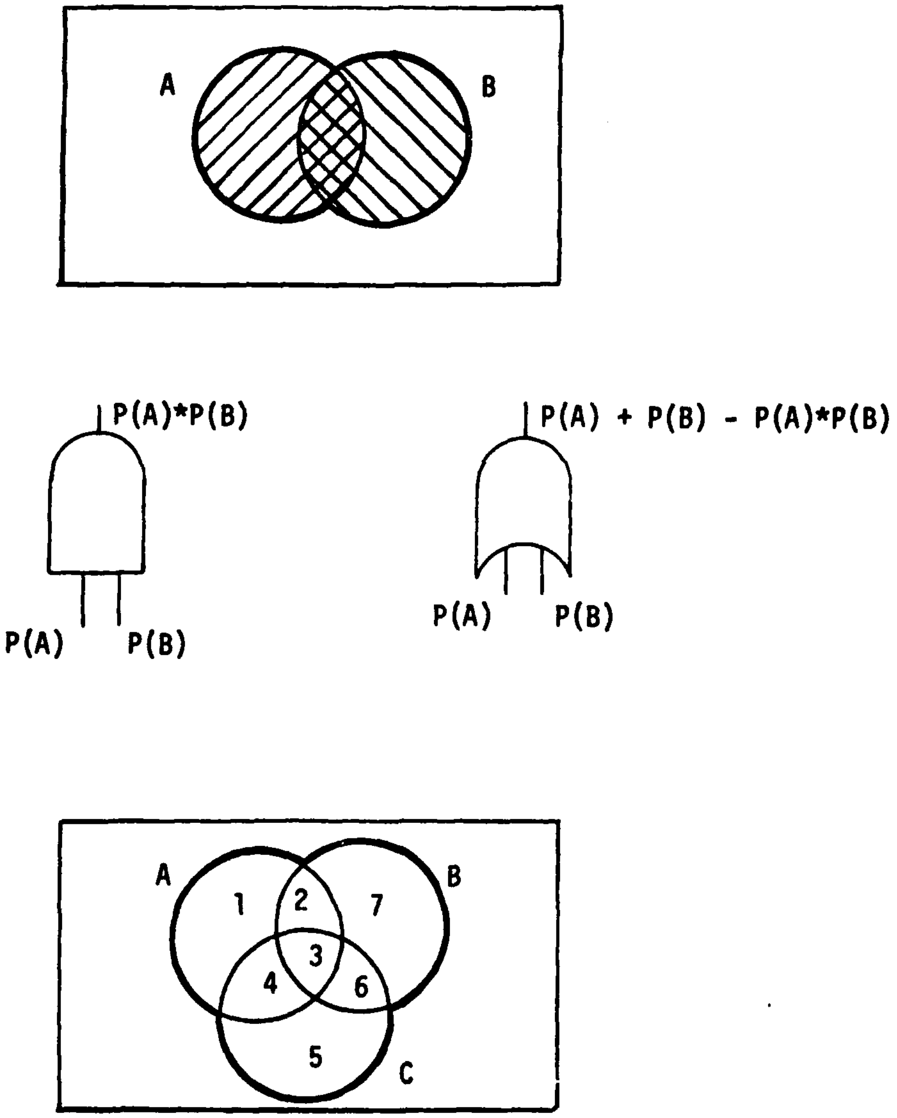


Figure 6. Venn Diagrams

TABLE 1

OR Gate Approximations

A	B	Actual A + B - A*B	Approximation A + B	Error A*B
0.01	0.10	0.109	0.11	0.001
0.01	0.01	0.0199	0.02	0.0001
0.001	0.01	0.01099	0.011	0.00001

For an OR gate with three independent inputs (A, B, and C with probabilities P(A), (B), and P(C)) the approximation is P(A) + P(B) + P(C) whereas the actual output probability is P(A) + P(B) + P(C) - P(A)*P(B) - P(A)*P(C) - P(B)*P(C) + P(A)*P(B)*P(C). The actual output probability can be visualized by the Venn diagram at the bottom of Figure 6. The output probability is the sum of areas one through seven. The approximation counts areas two, four, and six twice, and area three three times. Subtracting the three products leaves areas two, four, and six counted once, but area three not at all. Adding the product of the three probabilities completes the expression for the area enclosed by the heavy outer lines. That area is the actual output probability.

The three input OR gate approximation is used in the first tree in example one to approximate the probability of system failure. The second tree in example one uses no approximations to find the probability of system success. The sum of the probability for success and failure is greater than one because of the inflated failure figure. The excess above one is shown to be the approximation error for the OR gate. In this example, the error is about four percent.

The output probability for a two input PRIORITY AND gate is somewhat less than the product of the input probabilities. The size of the reduction depends on the chance of the specified failure happening first. A mathematical discussion is presented in the article by Gilmore, Woodcock, and Jewell (1964).

When the INHIBIT gate serves the function of a PRIORITY AND gate, its output is calculated the same way. In a RANDOM INHIBIT gate, the input probability is multiplied by the probability the input will not be inhibited.

The RESTRICTED OR gate and the MATRIX gate are treated as combinations of AND and OR gates (see example three).

When n elements function in parallel such that all elements must fail before the parallel combination fails, the AND gate is used. The output of the AND gate is the product of the inputs (see Figure 7). When the n elements are identical, the ellipse symbol, shown in Figure 5, is used to indicate redundancy and only one input to the AND gate is shown. The output probability is the input probability to the nth power, that is, the product of n identical inputs.

With n identical elements in series, failure of at least one element causes failure of the series. An ellipse may be used above an OR gate to represent the situation. The approximate output probability would be the failure probability of one element (p) times n. This approximation becomes unsatisfactory as n and p become large. If $p = .3$ and $n = 5$ the approximation says the combination fails 150% of the time. The exact output probability is $1-(1-p)^n$ fraction of the time. Using $p = .3$ and $n = 5$, the actual failure probability is about .17,

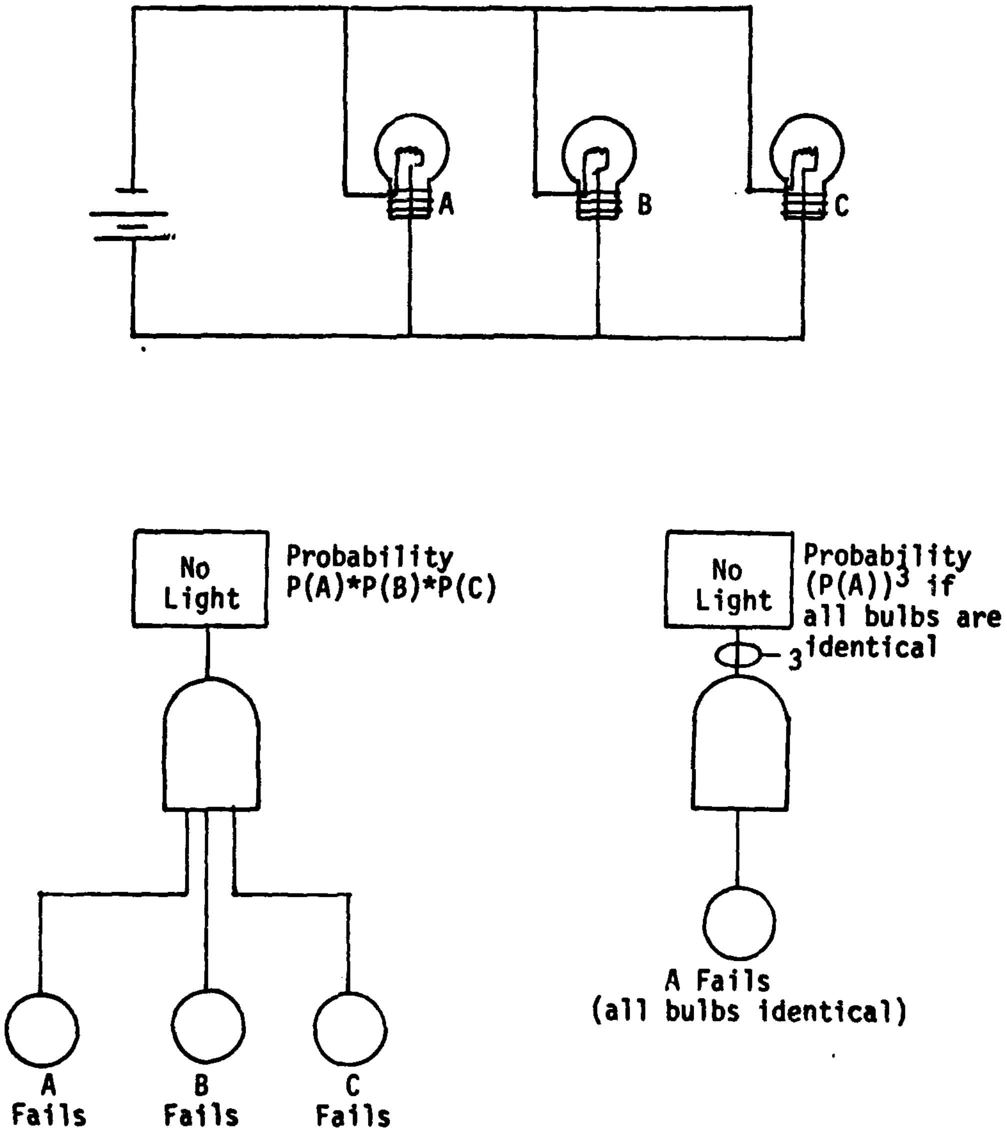


Figure 7. Example of Ellipse Symbol

not 1.5 as the approximation predicts.

Several simple approximations are based on relative size of inputs. A very small input (in relation to the others) to an OR gate can be ignored. An input probability close to one in an AND gate can be ignored. An extremely small input to an AND gate means the whole gate and its output can be ignored.

3. Boolean Algebra

The above simplifications are adequate when independent events appear at the lowest level of the tree; however, a different approach is needed when an event appears more than once at the lowest level. One effective approach is Boolean algebra which is briefly described here.

The key to understanding the simplest form of Boolean algebra is that variables may assume one of two values, zero or one. Three symbols are used, an equivalence relation[3] as well as two operations "+" and "*", which correspond to OR and AND respectively. In Table 2, the two operations combine zero's, one's and a variable A (which may take on either value). The last concept needed is that each element a has a unique inverse a' such that if $a = 0$, $a' = 1$ and if $a = 1$ then $a' = 0$. Also $a'' = a$.

Understanding the equivalence relation, both operations, and

[3] An equivalence relation is reflexive ($a=a$), symmetric (if $a=b$ then $b=a$), and transitive (if $a=b$ and $b=c$, then $a=c$).

TABLE 2

Boolean Operations

OR "+"

	0	1	A
0	0	1	A
1	1	1	1
A	A	1	A

AND "*"

	0	1	A
0	0	0	0
1	0	1	A
A	0	A	A

The above tables apply whether A takes on value of 0 or 1

			Same as a
b	a	a*b	a + a*b
0	0	0	0
1	0	0	0
0	1	0	1
1	1	1	1

the inverse, the reader should study the identities given in Figure 8, remembering that the variables a, b, and c may assume only values of zero or one. Also it may be helpful to refer back to Table 2 for the definition of "+" and "*".

Identity 10a of Figure 8 is so useful in simplifying fault trees that it will be explained here. If a = 1, then a + a*b = 1, no matter what value b assumes. On the other hand, if a = 0, then a + a*b will equal zero, so the value of a dictates the value of a + a*b. In Table 2 this identity is shown to be true for all possible values of a and b.

Using the identity a + a*b = a, the unusual form of the distributive law (identity 7b) is verified in Figure 8. The applicable identities used in each step are shown to the right.

An important principle called duality says that a valid Boolean identity is still valid if "+" is changed to "*", "*" changed to "+", zero changed to one, and one is changed to zero. Note that each of the identities of Figure 8 is presented with its dual on the same line. The duality principle is powerful because proving an identity also proves the dual of that identity. Although duality is important in Boolean algebra, it is not necessary for fault tree use. It is presented here only to give more insight into Boolean algebra.

An application of Boolean algebra to fault tree simplification given by Gilmore et al. (1964) is shown here in Figure 9. Since the event B is an input to two gates, the lowest events

1a	$a*a' = 0$	1b	$a + a' = 1$	
2a	$a*1 = a$	2b	$a + 0 = a$	
3a	$a*0 = 0$	3b	$a + 1 = 1$	
4a	$a*a = a$	4b	$a + a = a$	
5a	$a*b = b*a$	5b	$a + b = b + a$	commutative law
6a	$a*(b*c) = (a*b)*c$	6b	$a + (b + c) = (a + b) + c$	associative law
7a	$a*(b + c) = a*b + a*c$	7b	$a + (b*c) = (a + b)*(a + c)$	distributive law
8a	$(a + b)' = a'*b'$	8b	$(a*b)' = a' + b'$	DeMorgan's law
9a	$a + (a'*b) = a + b$	9b	$a*(a' + b) = a*b$	
10a	$a + a*b = a$	10b	$a*(a + b) = a$	

Proof that: $a + b*c = (a + b) * (a + c)$ Identity 7b

$(a + b)*(a + c) = a*a + a*c + b*a + b*c$

$= a + a*c + a*b + b*c$ 4a and 5a

$= a + a*b + b*c$ 10a

$= a + b*c$ 10a again

Figure 8. Boolean Identities

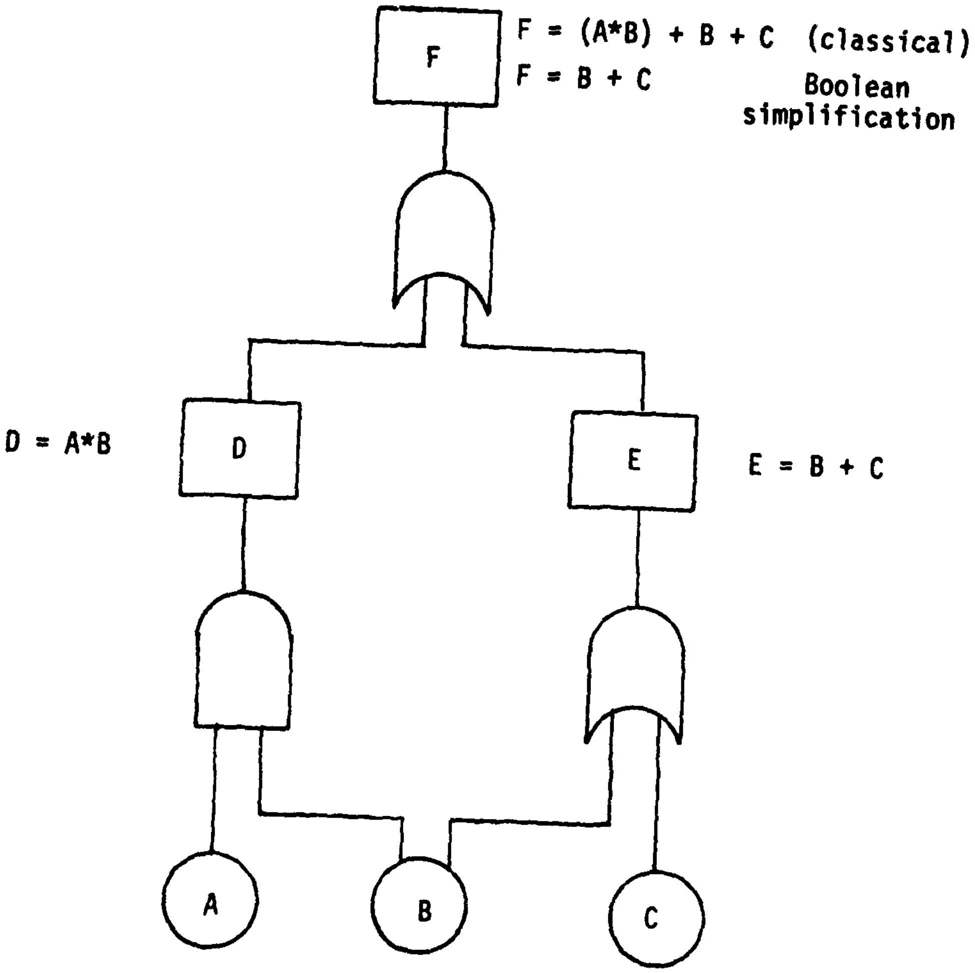


Figure 9. Boolean Simplification Example

(A, B, B, and C) are not independent. To simplify by the classical method (which is not correct here), the two equations $D = A*B$ and $E = B + C$ are ORed (using the approximation for both OR gates) to give $F = A*B + B + C$. Notice that a Boolean simplification ($B + B*A = B$) of F gives $F = B + C$.

Consider the diagram to see why A*B is unnecessary. If input B occurs (the variable B takes on the value one) the event E will occur producing F. The event B travels through both OR gates regardless of A and C. What if B does not occur? The AND gate cannot produce an output without input B, but C is sufficient to produce events E and F. Because of the system's nature, A is unnecessary. Event A may occur or not, without affecting F.

The example above illustrates simplification of repeated inputs. The procedure is to write events in symbolic form and apply Boolean simplification whenever possible. This simplification procedure will be used in the third and fourth examples of the following chapter. Bowran (1965) and Arnold (1962) are two sources of more detailed Boolean algebra discussions.

CHAPTER V

EXAMPLES

Introduction

The four examples in this chapter are designed to assist the reader in digesting information from Chapters III and IV. The first example demonstrates setting up a tree from knowledge of the system. First the components are discussed, followed by a list of successful events necessary for system success. In addition to a fault tree describing system failure, a similar logic tree describing system success is added to illustrate the OR gate approximation. Probabilities are given for the bottom events and calculated for the top event.

The second example illustrates an undeveloped use of the fault tree. This tree points out possible errors made by a photographer whose pictures come back from the developer over or under exposed. The tree examines only possible operator mistakes, no probabilities are used.

The third example uses more complicated gates and some Boolean algebra simplification. Events are symbolized by letters without probabilities. Only the tree is presented, no attempt is made to relate the tree to a physical system.

The last example is also a tree only. It uses AND and OR gates exclusively, and Boolean simplification.

Example One

This example is a hypothetical information processing system illustrated in Figure 10. The fault tree analysis is concerned with accurate transmission of answers, which affect the safety of another system. The system receives data and transmits answers calculated by man or computer. Normally the computer performs calculations with one hundred percent accuracy, but a buzzer is provided to ring when the computer is not working and cannot supply an answer. Each time the buzzer rings the man overrides the computer and calculates the answer himself, but the buzzer can fail in two ways. When the computer is working, sometimes the buzzer will sound, causing the man to override the computer unnecessarily; when the computer is not working, sometimes the buzzer will fail to sound, resulting in no answer being transmitted. The probabilities of the components working as designed are shown in Figure 10.

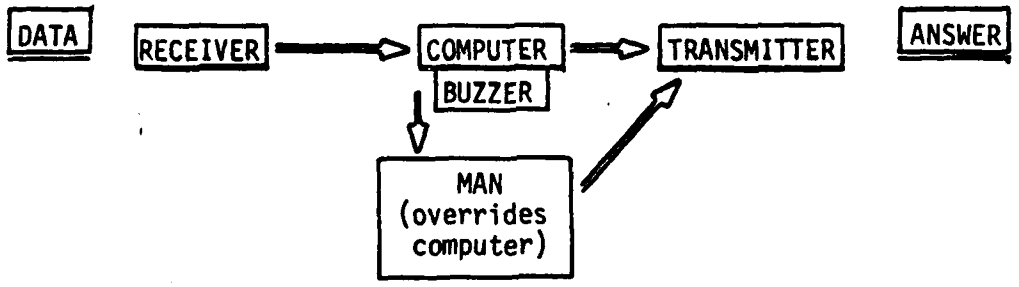
For correct output, the data must be correctly received. In addition, the computer must perform correctly, without the buzzer ringing. If the buzzer rings, the man must calculate accurately, and finally, the transmitter must work correctly.

The event investigated by the fault tree is failure to transmit the correct answer when data is received under normal conditions (the table of probabilities apply).

A preliminary calculation is required to find what fraction of the time the buzzer rings. This fraction is the probability that the man will override the computer and calculate the answer himself. The calculations are shown in Figure 10.

Figure 11 is a fault tree of the system. Notice that the probability for the failure is conservative (slightly high) because of the OR gate approximation. Figure 12 is a logic tree computing the probability of system success. It gives an accurate answer because no regular OR gates are used. Below the logic tree of Figure 12, the probability for success and failure are added and the excess above one is shown to be the OR gate approximation for the fault tree of Figure 10. Notice that the error is due only to the top OR gate. The other OR gate (RESTRICTED OR) gives the correct answer by adding both inputs because it is impossible for the computer and the man to both give an answer (incorrect or otherwise).

Inspection of the fault tree's probabilities show the transmitter is responsible for nearly two-thirds of all system failures, and the receiver is responsible for a majority of the remaining failures. Improvement of the system would probably start with one of these two components, taking cost into consideration.



Component	Task	% of Time Tast Completed
receiver	receives correctly	95
buzzer	buzz when computer "down"	90
buzzer	no buzz when computer works	99
computer .	works	99
man	calculates correctly	60
transmitter	transmits accurately	90

Buzzer rings

computer works and buzzer fails or computer "down" and buzzer works

.99 * .01 + .01 * .90

.0099 + .0090 = .0189

Buzzer will not ring

computer works and buzzer works or computer "down" and buzzer fails

.99 * .99 + .01 * .10

.9801 + .001 = .9811

Figure 10. Example One Data

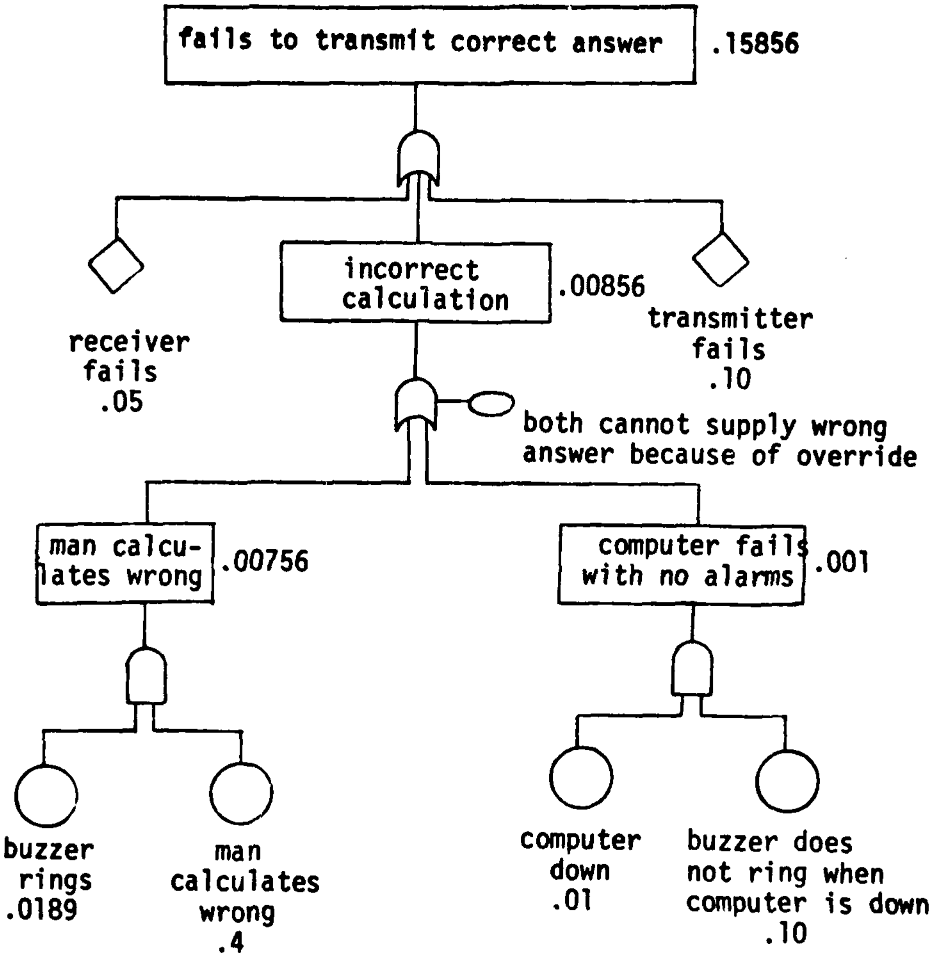
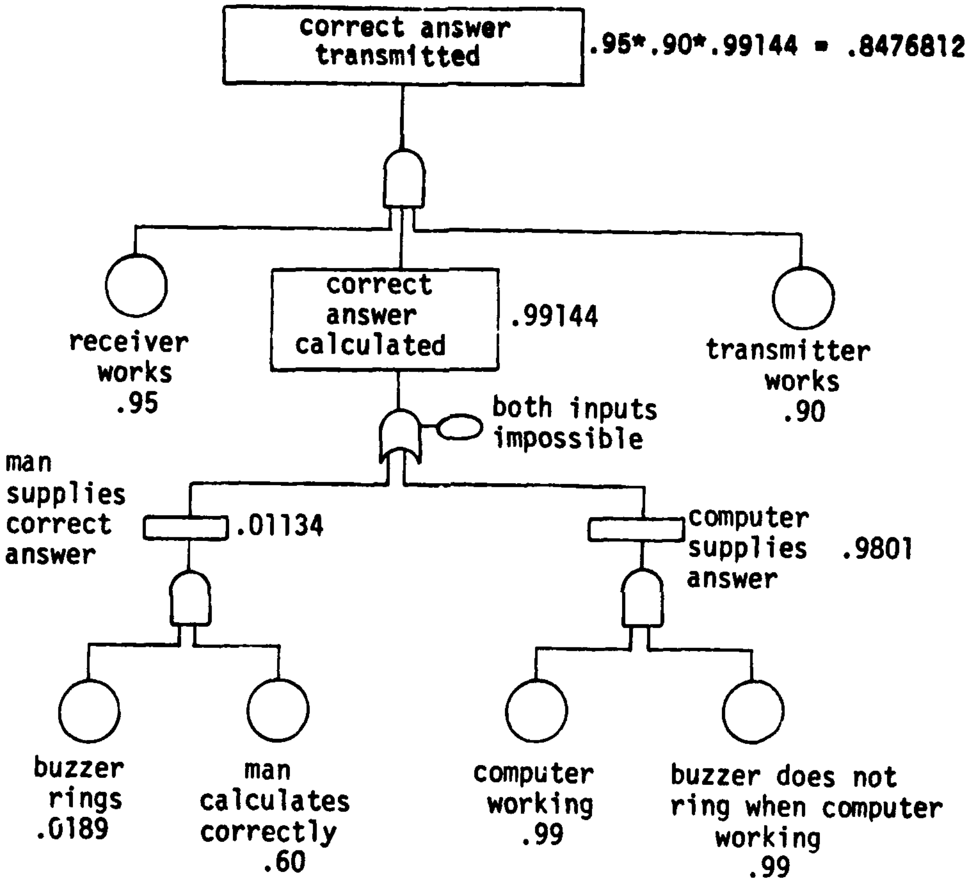


Figure 11. Example One Fault Tree



.8476812
.15856 (Fig. 10)
1.0062412

Correction from Figure 10

-.005	receiver and transmitter
-.000856	answer and transmitter
-.000428	answer and receiver
-.006284	second order correction
-.0000428	third order correction
-.0062412	TOTAL ERROR

Figure 12. Example One Logic Tree

Example Two

The second example illustrates fault tree analysis in an undeveloped application; that is instructions for operators. This tree describes conditions which may cause a poor picture for an amateur photographer, who has his film sent away for automatic developing. The event under consideration is the return of poor pictures, assuming the camera had film and was operative. The three operator responsibilities considered are film speed, shutter speed and aperture opening. These are factors built into the camera but controlled by the operator.

Causes for underexposure and overexposure are described in terms of improper operator settings. No probabilities are given, the tree in this application shows only causes.

Fault trees have much undeveloped potential in the general area of operator instructions. One tree can present an overview of maintenance, troubleshooting, or operating instructions. The value of a graphical presentation (such as a tree) is in the speed and clarity of understanding. More detailed instructions are necessary, but the tree can present each alternative in perspective, much like a chain of command block diagram.

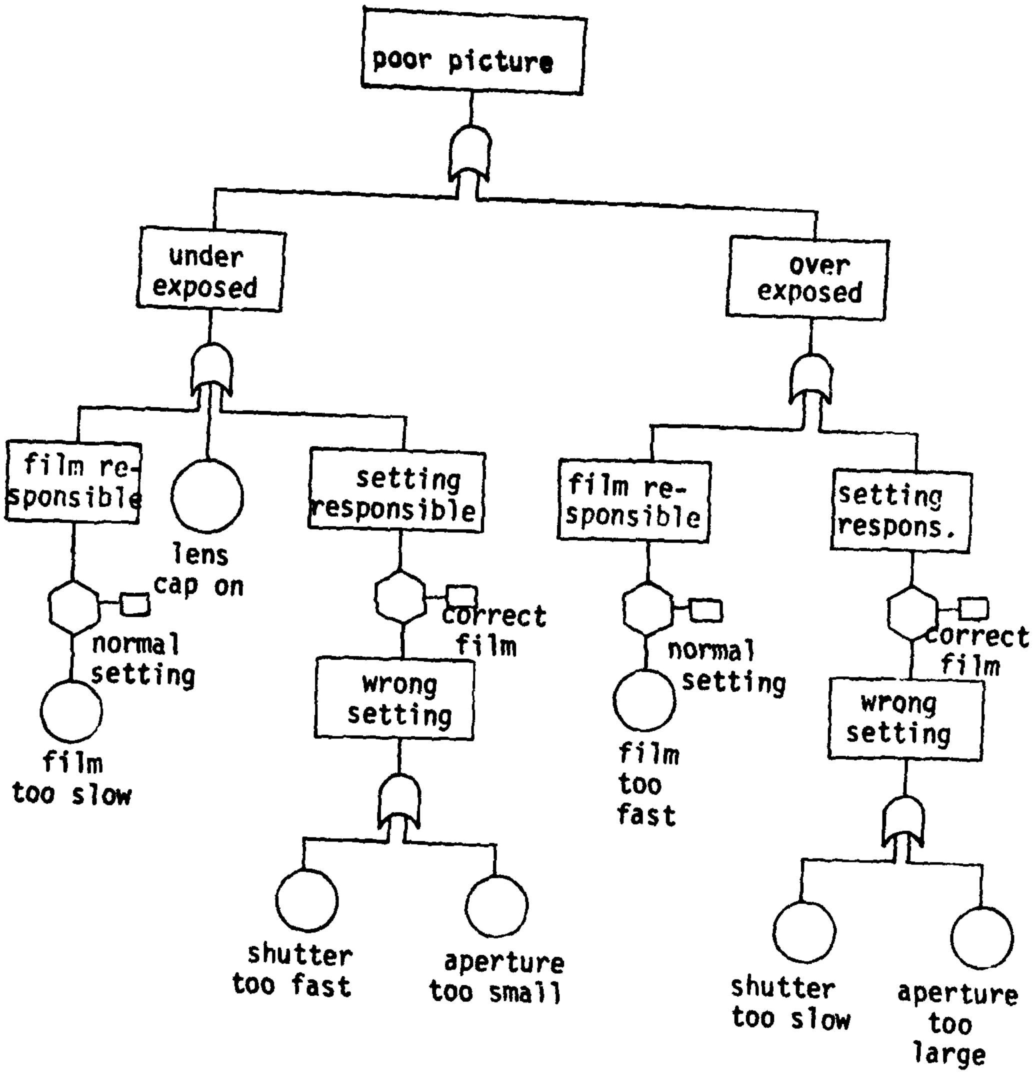


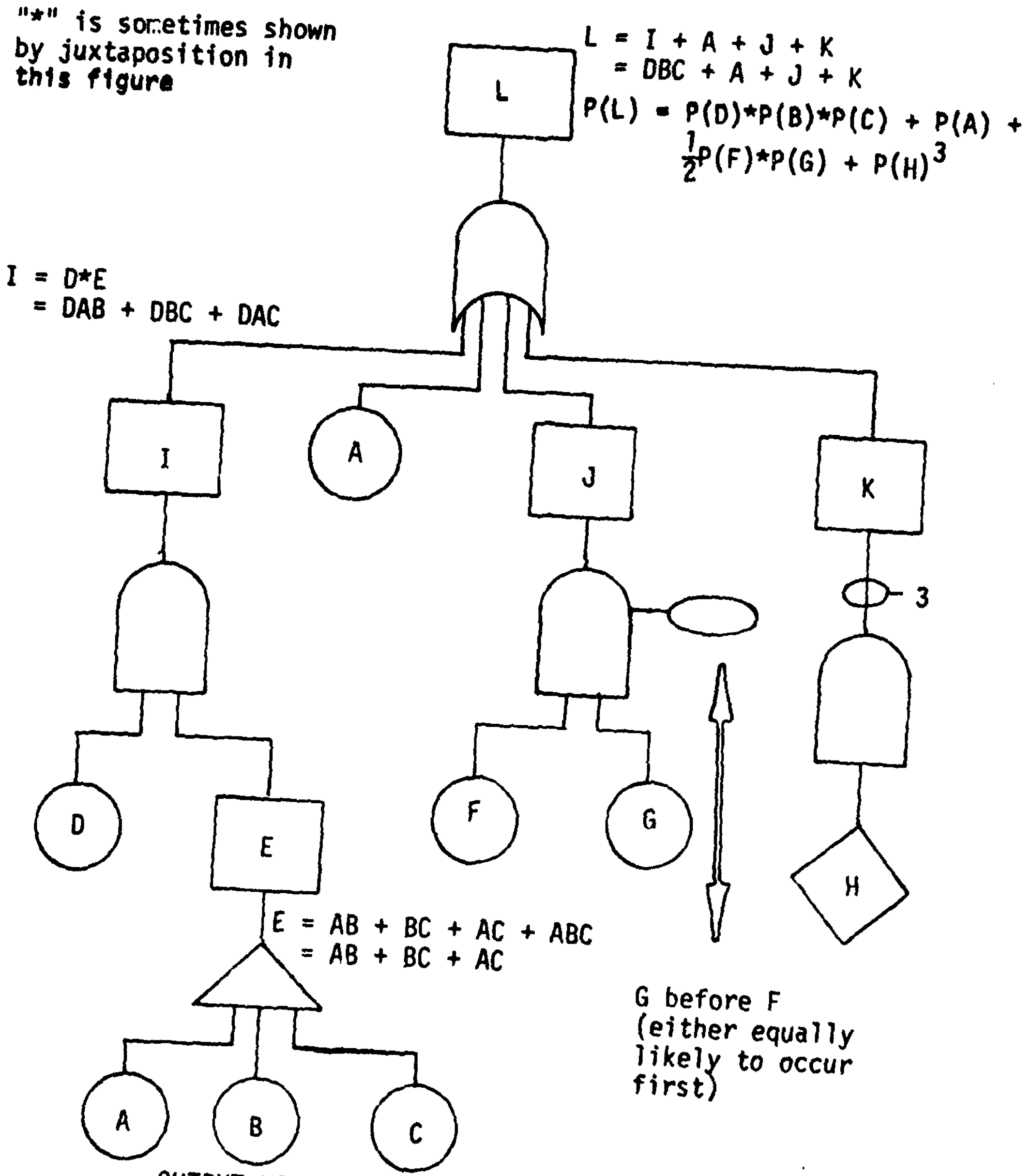
Figure 13. Example Two Fault Tree

Example Three

The third example is a tree only; no attempt is made to relate tne structure to a real system. The events are represented by by letters which are used in Boolean equations to evaluate the top event. Because Boolean algebra has only two operations, there is a difficulty with the ellipse symbol and the PRIORITY AND. These two gates are not involved in Boolean simplification in this example, however their probabilities occur in the expression for the top event.

The MATRIX gate is accompanied by a matrix that shows output is produced for any combination of two or more inputs, but no output is produced for any single input. The case of three simultaneous inputs is immediately eliminated by Boolean simplification.

The PRIORITY AND gate has inputs F and G. It is equally likely that event F will precede event G, or G will precede F, hence the probability of output is 1/2 P(G)*P(F). The rightmost AND gate with the ellipse around the output signifies three related inputs labeled H. The probability of output from the gate is P(H) cubed. This should not be simplified by Boolean techniques because the three events are similar to each other and occur with equal probabilities, but they are not the same event (one may happen without influencing the others). Two Boolean simplifications are done at the top event. These are not explicitly shown.


"*" is sometimes shown by juxtaposition in this figure
L = I + A + J + K
= DBC + A + J + K
P(L) = P(D)*P(B)*P(C) + P(A) + ½P(F)*P(G) + P(H)³
L
I = D*E
= DAB + DBC + DAC
I
A
J
K
3
D
E
F
G
H
E = AB + BC + AC + ABC
= AB + BC + AC
G before F (either equally likely to occur first)
A
B
C

OUTPUT MATRIX

Not A

	Not B	B
Not C	-	B
C	C	X BC

A

	Not B	B
Not C	A	X AB
C	X AC	X ABC

Figure 14. Example Three Fault Tree

Example Four

Example four, like example three, is a tree only, with symbols for events and no probabilities assumed. It uses the Boolean identity of the form $a + a*b = a$ three times and the identity $a*a = a$ once. Only AND and OR gates are used to show the development of Boolean equations for each combination event represented by a rectangle. Boolean simplifications are made as soon as an opportunity appears.

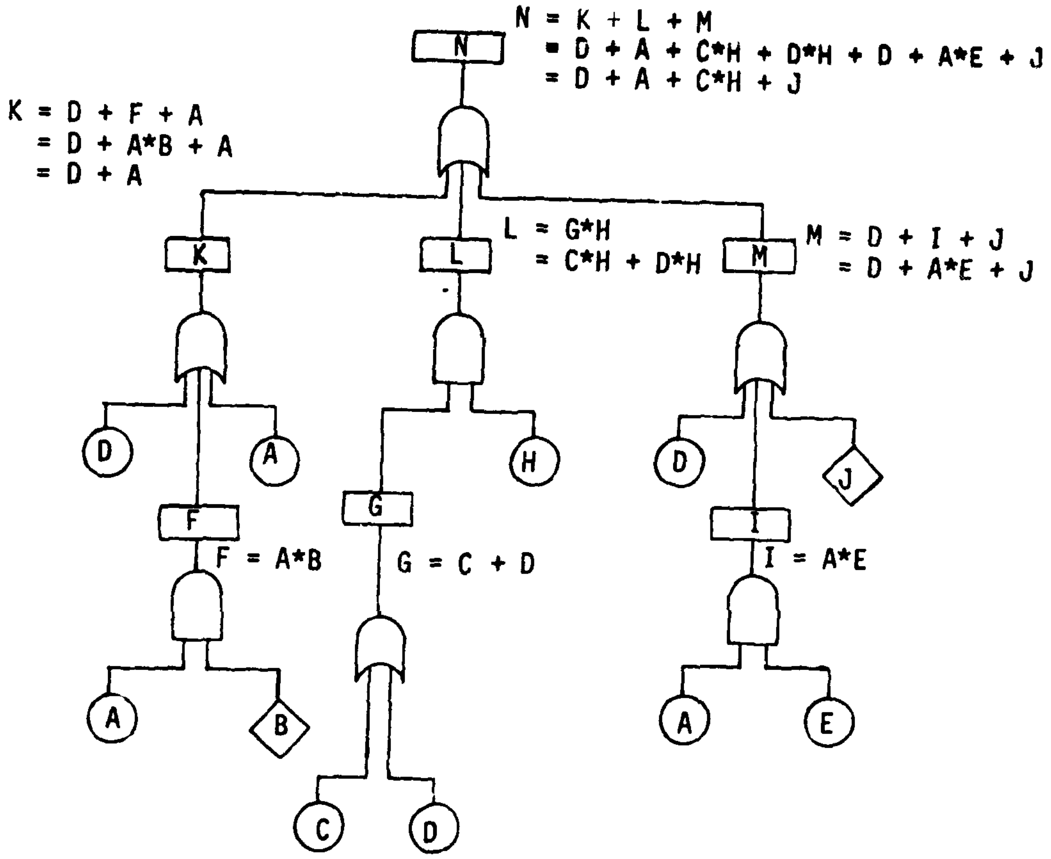


Figure 15. Example Four Fault Tree

CHAPTER VI

COMPUTER TECHNIQUES

The fault tree of example one clearly showed that the transmitter was responsible for a majority of the failures, but in systems costing millions of dollars (with fault trees done by dozens of men working many months), the most likely failure paths through the tree cannot be found by inspection. Instead, computer simulations are used to find the basic failures most likely to describe a path up to the top event. Two simulation techniques are described here.

A Monte Carlo procedure applied to a fault tree consists of picking random numbers between zero and one to simulate a component's operation. For example, if a component is expected to fail 5% of the time, a random number between zero and .05 would be a simulated failure and a random number between .05 and one would be a simulated success. The procedure is to pick a different random number for each bottom event of the tree. After the success or failure of each event is simulated, the failures are traced upward through the logic gates to find whether the top event is achieved.

When the top failure is simulated, the path up the tree (or the responsible primary failures at the bottom) is noted and a new set of random numbers is chosen. Examination of all the simulated system failures shows the path most likely to cause the failure. If in a million trials, system failures occur 100 times, perhaps half the failures are due to one set of causes and the other 50 failures due to different causes. The two sets can be redesigned for more reliable or safer operation.

The drawback to straightforward Monte Carlo simulation is time. Too many trials are necessary to simulate a meaningful number of failures because the components are very reliable and the probabilities are extremely low. Hixenbaugh (1968) gives the following figures showing size of trees and simulation time. Trees may have two or three hundred gates with perhaps two hundred bottom events and many possible paths leading to the top failure. These paths are so unlikely that it may take ten million trials to simulate a meaningful number of failures. If each trial takes one-tenth second, 275 hours of computer time is necessary. This is expensive and unwieldy.

To make a higher proportion of trials lead to the top failure, the probability of the bottom events are artificially increased for the purpose of simulation. In this technique, called importance sampling, (mentioned by Schroder [1968], and Hixenbaugh [1968]), the bottom failures are usually raised enough that one thousand trials will give a significant number of failures.

The last advanced topic mentioned here, the Lambda-Tau method, is used in the situation of faults which are detectable and reparable within the time limits defined for the top event. Consider a man on his way to work in the morning stopping at a newstand for the newspaper. When he returns to his car and turns the key, nothing happens. His horn does not blow either, so he opens the hood, cleans some corrosion from the battery terminals and drives to work on time. His car suffered a failure which was detected and repaired without interfering with anything more serious than leisure time.

In a system with detectable and reparable faults, the failures come and go periodically. The top event occurs only if the right combination of faults are present at one time. In the Lambda-Tau technique, which is programmed for the IBM 7094 computer, several assumptions are necessary. First, the primary failure rate (lambda) must be small, and secondly, the duration time of the hazard (tau) must be small compared to the time limits of operation of the system. When these two assumptions are met, the program is valuable for reparable systems of medium size.

The techniques described above complete this fault tree manual by indicating how the basic principles of the first four chapters are used in analyzing actual systems. In addition to safety applications in complex systems, the tree concept has potential in production control and operator's manuals (such as example two), two problems which have contributing factors in a cause and effect relationship contributing to an undesired event. Fault tree analysis is certainly not a cure-all, but it has made a significant contribution to the field of safety and perhaps will be extended to other areas.

BIBLIOGRAPHY

Arnold, B.H., Logic and Boolean Algebra, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962.

Bowran, A.P., Boolean Algebra, Macmillan and Company Ltd., London, England, 1965.

Driver, E.T., "Prediction of Accident Probability," paper presented at the Thirteenth Annual Western Safety Congress, Los Angeles, California, April 7, 1966.

Eckberg, C.R., "WS-133B Fault Tree Analysis Program Plan," The Boeing Co., Seattle, Washington, December 1963.

Feutz, R.J., "Application of Fault Tree Analysis to Dynamic Systems," presented at the System Safety Symposium, Seattle, Washington, June 1965.

Feutz, R.J. and Tracy, J.P., "Introduction to Fault Tree Analysis," The Boeing Co., Seattle, Washington, September 1965.

Gilmore, H.L.; Woodcock, R.W.; Jewell, R., "Minuteman Mark IIA Reentry Vehicle System Safety Engineering Plan," AVCO Corp., Wilmington, Mass., January 1964.

Gottfried, P.; Jettner, E.; Weiss, D, et al, "Evaluation of Reliability Prediction Techniques for Entire Flight Control Systems," Booz, Allen Applied Research Inc., Bethesda, Md., March 1968

Haasl, "Advanced Concepts in Fault Tree Analysis," presented at the System Safety Symposium, Seattle, Washington, June 1965.

Hiltz, P.A., "The Fundamental of Fault Tree Analysis," North American Aviation, Inc., Downey, California, November 1965.

Hixenbaugh, A.F., "Fault Tree for Safety," The Boeing Co., Seattle, Washington, November 1968.

Kanda, "Concept of System Safety Math," presented at the System Safety Symposium, Seattle, Washington, June 1965.

Larsen, W.F., "Fault Tree Analysis," Picatinny Arsenal, Dover, N.J., November 1968.

McDevitt, J.A.; Goldberg, M.I.; Cornwell, C.E., "The Systems Concept for a Hazard Analysis of Large Solid Propellant Rocket Motors," Naval Ordnance Station, Indian Head, Maryland, November 1968.

Mearns, A.B., "Fault Tree Analysis: The Study of Unlikely Events in Complex Systems," presented at the System Safety Symposium, Seattle, Washington, June 1965.

Michels, "Computer Evaluation of Fault Tree Model," presented at the System Safety Svmposium, Seattle, Washington, June 1965.

Nagel, P., "Monte Carlo Method to Compute Fault Tree Probabilities," presented at the System Safety Symposium, Seattle, Washington, June 1965.

Peller, J.B., "Fault Tree Analysis as a Tool for System Safety Engineering," North American Aviation, Autonetics Division, Anaheim, California, June 1965.

Pfeiffer, P.E., Concepts of Probability Theory, McGraw-Hill Book Co., New York, 1965.

Recht, J.L., "System Safety Analysis: The Fault Tree," National Safety News, Vol. 93 #4, April 1966, pp. 37-40.

Schroder, R.J., "Fault Tree Simulation with Importance Sampling," The Boeing Co., Seattle, Washington, March 1968.

Thorns, N.M., "A Fault Tree Model of the Man/Machine Interface in a Manned Space System," The Boeing Co., Seattle, Washington, August 1966.

Waldeck, T.A. and McMurdo, R.B., "System Safety Engineering Analysis Techniques," The Boeing Company, Seattle, Washington, February 1968.

Walker, R.Y., "Human Factors in Fault Tree Analysis," The Boeing Co., Seattle, Washington, March 1966.