Process Failure Analysis by Block Diagrams and Fault Trees

Sergio Caceres and Ernest J. Henley*

Department of Chemical Engineering, University of Houston, Houston, Texas 77004

A reliability or hazards analysis can be developed from either a block diagram or fault tree representation of the process. This paper emphasizes the equivalence of these two representations and develops a procedure for automatically transforming a block diagram into a fault tree. The technique is based on a computer-oriented algorithm developed to detect all the minimal paths leading to the success of a system represented by a block diagram. The fault tree thus developed is a true, sequential representation of possible modes of process failure.

Introduction

A fault tree is a particular type of event logic diagram used to represent the various combinations of faults (failure events) that will lead to a major undesired event in a system. Graphically, the main undesired event is shown at the top of the tree and its immediate causes are indicated below and are connected to it by standard logic gates. The causes of these faults are then further developed until basic (independent) events are reached.

Because of the relatively recent development of fault tree analysis, many researchers are only familiar with reliability diagrams and are hesitant to use the fault tree approach. Even though both techniques yield the same results (Barlow, 1973; Inoue and Henley, 1975; Shooman, 1970), each has its own advantages and disadvantages. Guidelines and methodology for the construction of reliability block diagrams are well established, but the quantitative analysis as well as a clear definition of what is flowing between the blocks represent problem areas (Barlow et al., 1965; Bazovsky, 1961; Green and Bourne, 1972; Henley and Williams, 1973; Messinger and Shooman, 1967). On the other hand, the lack of organized methods for fault tree construction and analysis often leads to ambiguities (Fussel et al., 1974c; Powers and Tompkins, 1974).

A fault tree can be represented in many different ways. This, however, does not represent a limitation on fault tree analysis; in fact, it is an advantage because different fault trees are used for different applications. The quantitative analysis of fault trees is quite simple because the relations between component failures are expressed in terms of simple Boolean logic operations.

This paper describes a procedure for automatic generation of a fault tree from the minimal paths leading from the initial input node (cause) to the final output node (effect) of a block diagram. The output event on a block diagram represents a variable or combination of variables chosen to characterize the successful operation of the system. This technique is dependent on the order in which the paths are arranged and on a method that is capable of detecting all the minimal paths, no matter how complicated a block diagram may be.

Equivalence of Fault Trees and Reliability Diagrams

The system shown in Figure 1 is designed to: (i) decrease the temperature of a hot gas stream (i.e., tail gas from an industrial boiler); (ii) saturate the gas with the water vapor; and (iii) remove solid particles entrained in the gas flow.

A simplified reliability block diagram for this system is given in Figure 2. The booster fan (A), either quench pump (B or C), the feed water pump (D), either circulation pump (E or F), and the filter system (G) must be successful for the system to work. The probability of success, $Pr\{S\}$, for this system can then be written as:

$$Pr\{S\} = Pr\{a\cdot(b + c)\cdot d\cdot(e + f)\cdot g\} \tag{1}$$

where the expression in brackets is the Boolean expression for system success. The letter "a" is used to represent the successful state of A, etc.

If a horizontal bar over an expression is used to indicate the negation of that expression, the probability of system failure, $Pr\{F\}$, is given by:

$$Pr\{F\} = Pr\{\bar{S}\} = Pr\{\overline{a\cdot(b + c)\cdot d\cdot(e + f)\cdot g}\} \tag{2}$$

Applying De Morgan's rules

$$Pr\{F\} = Pr\{\bar{a} + (\bar{b}\cdot\bar{c}) + \bar{d} + (\bar{e}\cdot\bar{f}) + \bar{g}\} \tag{3}$$

If the input events for a time independent fault tree are independent and nonrepairable the probability of the output event of an AND gate is the product of the probabilities of its input events. Under the same conditions the probabilities of the output event of an OR gate is approximately the sum of the probabilities of its input events. Using Boolean notation, this can be stated as $Prob\{a\cdot b\cdot c \ldots\} = Pr\{a\}\cdot Pr\{b\}\cdot Pr\{c\} \ldots$ and $Prob\{a + b + c + \ldots\} \simeq Pr\{a\} + Pr\{b\} + Pr\{c\} + \ldots$.

The approximation is good if the sum $Pr\{a\} + Pr\{b\} + Pr\{c\} + \ldots$ is a great deal less than 1.0 (preferably less than 0.1). The above equations are valid only if the input events are independent.

If, for example:

$$Pr\{\bar{a}\} = 0.010$$
$$Pr\{\bar{b}\} = Pr\{\bar{c}\} = Pr\{\bar{d}\} = 0.012$$
$$Pr\{\bar{e}\} = Pr\{\bar{f}\} = 0.008$$
$$Pr\{\bar{g}\} = 0.015$$

then eq 3 becomes

$$Pr\{F\} \simeq (0.010) + [(0.012)\cdot(0.012)] + (0.012) + [(0.008)\cdot(0.008)] + (0.015) = 0.037208$$

This example could equally be formulated in terms of a fault tree diagram. The failure of the system is then modeled in terms of the union of different failure modes using logic symbols. In terms of the example, there are five different ways in which the entire system could fail: booster fan failure (A), quench pump failures (B and C), feed water pump failure (D), circulation pump failures (E and F), or feed water pump failure (G). These five events are represented as inputs to an OR gate. Since a quench pump system failure or a circulation pump system failure will occur

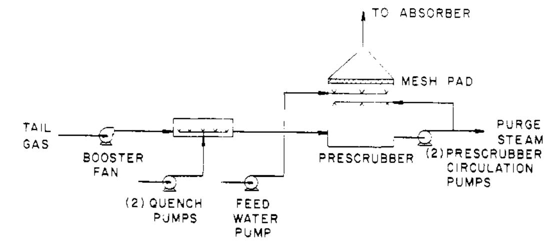

Figure 1. A model system.

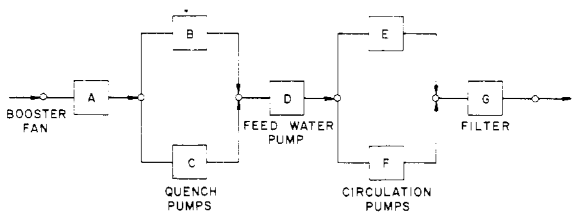

Figure 2. Block diagram for Figure 1.

if both quench pumps fail or if both circulation pumps fail, these failure modes are connected to an AND gate. The fault tree for this system is shown in Figure 3, and its corresponding Boolean expression is:

$$Pr\{F\} = Pr\{\bar{a} + (\bar{b} \cdot \bar{c}) + \bar{d} + (\bar{e} \cdot \bar{f}) + \bar{g}\} \tag{4}$$

which is identical with eq 3.

This simple example serves the purpose of demonstrating the equivalence of fault trees and reliability diagrams. However, most systems are much more complicated. Block diagrams that contain loops or recycles (or in general, non-series-parallel systems) often pose severe difficulties for quantitive analysis and complicated techniques are required. This is not the case with fault trees, for which Boolean expressions for system failure are always easily obtained (Semanderes, 1970; Shooman, 1970, for example).

Characterization of Block Diagrams

A block diagram is composed of blocks and nodes (which shall be denoted by capital letters and numbers, respectively) interconnected by directed lines. This suggests a notation that is capable of representing the diagram by other means than a graph.

Let a diagram "element" be described by three components: an input node, a block, and an output node. Each block can be used to characterize an element since there will be as many elements as there are blocks in the diagram. If E_B is the element corresponding to block B, then

$$E_B = (x, B, y) \tag{5}$$

where x is an input node to B and y is an output node from B. Therefore, if n is the total number of blocks in a diagram, its corresponding system (S) is completely defined by the set of its n elements as:

$$S = \{E_{B_i} = (x_i, B_i, y_i); i = 1, 2, \ldots, n\} \tag{6}$$

This type of notation is particularly useful for computer applications.

Parallel Sets. A parallel set, S_p, will correspond to a diagram having a parallel configuration. As such, all of its elements will have the same input and output nodes.

$$S_p = \{E_{Bi} = (x_i, B_i, y_i); i = 1, 2, \ldots, n/x_i = x_j, y_i = y_j; \forall i, j\} \tag{7}$$

If the term "element failure" is used to indicate the failure of the block corresponding to that element, the failure of a system represented by a parallel set will occur only when all of its elements fail. The Boolean expression corresponding to this condition is:

$$\bar{S}_p = \bar{b}_1 \cdot \bar{b}_2 \cdot \cdots \cdot \bar{b}_i \cdot \cdots \cdot \bar{b}_n \tag{8}$$

where $\bar{S}_p$ denotes the failure of S_p, $\bar{b}_i$ the failure of B_i (or E_{Bi}), and "$\cdot$" is the familiar "AND" operation of Boolean algebra.

Series Sets. A series set, S_s, will be defined as a set corresponding to a diagram having a series configuration. Except for the last element, the output node of an element in a series set will be equal to the input node of the element that follows

$$S_s = \{E_{Bi} = (x_i, B_i, y_i); i = 1, 2, \ldots, n/y_i = x_{i+1}; i = 1, 2, \ldots, n-1\} \tag{9}$$

The failure of this type of system will occur when any of its elements fail. Its Boolean expression is:

$$\bar{S}_s = \bar{b}_1 + \bar{b}_2 + \cdots + \bar{b}_i + \cdots + \bar{b}_n \tag{10}$$

where the symbol "+" stands for the "OR" operation of Boolean algebra.

Class Subsets. If all the elements in a diagram are classified according to their output nodes, there will be a class of one or more elements associated with each node in the diagram (except for the initial input node of the system, since it does not appear as the output node of any elements). A class subset (or simply, class), C_i, can then be identified as the set of all the elements in a system that have node i as an output node.

As an example, the diagram shown in Figure 4 has the following classes:

$$C_1 = \phi \text{ (the empty set)} \tag{11}$$

$$C_2 = \{(1, A, 2), (1, B, 2)\} \tag{12}$$

$$C_3 = \{(2, D, 3)\} \tag{13}$$

$$C_4 = \{(1, C, 4)\} \tag{14}$$

$$C_5 = \{(3, F, 5), (3, G, 5), (4, E, 5)\} \tag{15}$$

As for parallel sets, a subsystem represented by a class will fail when all of its elements fail. Therefore, a node can

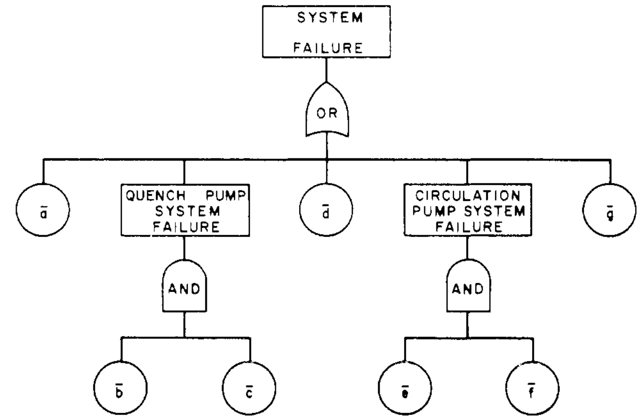

Figure 3. Fault tree for Figure 1.

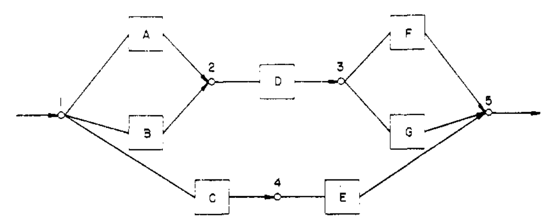

Figure 4. System block diagram.

be identified with an "AND" operator if its corresponding class contains more than one element (necessary and sufficient condition). Observe that a sufficient (but not necessary) condition is given if the node in consideration appears as the output node of a parallel set.

Path Sets. A path set, P, can be defined as a series of elements leading from the initial input node to the final output node of a system. Its failure expression is then equal to that given in eq 10. Furthermore, a path, p, will be defined as the sequence of nodes and blocks appearing in a path set P.

As an example, a path set for the diagram shown in Figure 4 is:

$$P = \{(1, A, 2), (2, D, 3), (3, F, 5)\} \tag{16}$$

Its corresponding path is:

$$p = 1\text{–}A\text{–}2\text{–}D\text{–}3\text{–}F\text{–}5 \tag{17}$$

At this point it is important to distinguish between minimal and nonminimal path sets. A minimal path set will contain a minimum number of elements leading from the initial to the final node of a system, that is, each element in the set must belong to a different class (which is equivalent to saying that a node may not be traversed more than once within a path). The system represented by a block diagram will fail when all of its paths fail simultaneously. The condition for system failure, F, is therefore given by:

$$F = \bar{S} = \bar{P}_1 \cdot \bar{P}_2 \cdot \cdots \cdot \bar{P}_j \cdot \cdots \cdot \bar{P}_m \tag{18}$$

where m is the number of paths in the system, and $\bar{P}_j$ is obtained from eq 10:

$$\bar{P}_j = \bar{b}_{1j} + \bar{b}_{2j} + \cdots + \bar{b}_{ij} + \cdots + \bar{b}_{nj} \tag{19}$$

where $\bar{b}_{ij}$ represents the failure of element E_{Bi} (block B_i) in path set P_j, and n_j is the number of blocks in P_j.

Nodal Subsets. A "nodal subset" will be defined as the set of all the elements that form the paths leading to a given node. In this way, a different nodal subset is associated with each node in the diagram and will be identified by using the corresponding node number as a subscript. Furthermore, the set corresponding to the final node of the system will be the system itself, and the set associated with the initial node of the system will not contain any elements.

In general, the nodal subset for node i can be obtained as a recurrence relation from the union of class C_i and the nodal subsets corresponding to the input nodes of the elements in C_i.

$$S_i = C_i \; U \; (S_{j1} \; U \; S_{j2} \; U \cdots U \; S_{jn}) \tag{20}$$

where $j_1, j_2, \cdots, j_n$ are the input nodes that appear in C_i.

As an example, the nodal subsets for the diagram in Figure 4 will be obtained from its classes given in eq 11–15:

$$S_1 = C_1 = \phi \tag{21}$$

Only node 1 appears as an input node in C_2; therefore

$$S_2 = C_2 \; U \; S_1 = C_2 \; U \; \phi = C_2 = \{(1, A, 2), (1, B, 2)\} \tag{22}$$

Similarly, only nodes 2 and 1 appear as input nodes in C_3 and C_4, respectively; then

$$S_3 = C_3 \; U \; S_2 = \{(1, A, 2), (1, B, 2), (2, D, 3)\} \tag{23}$$

$$S_4 = C_4 \; U \; S_1 = C_4 = \{(1, C, 4)\} \tag{24}$$

Nodes 3 and 4 are input nodes in C_5; thus

$$S_5 = S = C_5 \; U \; S_3 \; U \; S_4 = \{(1, A, 2), (1, B, 2), (1, C, 4), (2, D, 3), (4, E, 5), (3, F, 5), (3, G, 5)\} \tag{25}$$

Block Subsets. As mentioned previously, each block in the diagram defines a different element. If $E_B = (x, B, y)$, then the "block" subset S_B will be defined as

$$S_B = \{E_B = (x, B, y)\} \; U \; S_x \tag{26}$$

Therefore, there will be as many block subsets as there are blocks (or elements) in the diagram.

Again, for the example in Figure 4 (refer to eq 21–25 for the nodal subsets):

$$S_A = \{(1, A, 2)\} \; U \; S_1 = \{(1, A, 2)\} \tag{27}$$

$$S_B = \{(1, B, 2)\} \; U \; S_1 = \{(1, B, 2)\} \tag{28}$$

$$S_C = \{(1, C, 4)\} \; U \; S_1 = \{(1, C, 4)\} \tag{29}$$

$$S_D = \{(2, D, 3)\} \; U \; S_2 = \{(1, A, 2), (1, B, 2), (2, D, 3)\} \tag{30}$$

$$S_E = \{(4, E, 5)\} \; U \; S_4 = \{(1, C, 4), (4, E, 5)\} \tag{31}$$

$$S_F = \{(3, F, 5)\} \; U \; S_3 = \{(1, A, 2), (1, B, 2), (2, D, 3), (3, F, 5)\} \tag{32}$$

$$S_G = \{(3, G, 5)\} \; U \; S_3 = \{(1, A, 2), (1, B, 2), (2, D, 3), (3, G, 5)\} \tag{33}$$

A particular situation arises when a class contains only one element. In this case, the nodal subset corresponding to the output node of the element will be equal to the block subset associated with its block. For the previous example

$$C_3 = \{(2, D, 3)\}; \text{ therefore, } S_3 = S_{\mathrm{D}} \tag{34}$$

Path Finding Algorithm

General Procedure. Starting in reverse order (from output to input), the first element in a path set will be an element of the class associated with the final node of the system. Since a path set is also a series set, the input node of the first element will correspond to the output node of the second element in the path set. Therefore, this second element can be any one of the elements belonging to the class associated with the input node of the first element. Similarly, the $(i + 1)$th element in the path will belong to the class associated with the input node of the ith element. In this fashion, subsequent elements can be added to the series until the initial node of the diagram is reached.

In order to obtain all the different paths, a counter associated with each class can be used to indicate which of the elements in that class should be added to the path in consideration.

Initially, all the counters are set equal to 1, meaning that the first element in a class will be added to the path whenever that class is encountered. After a path is completed, the counter associated with the class of the last element in the path is increased by 1. If that counter does not exceed the number of elements in that class, the path finding procedure can be repeated and a new path obtained. On the other hand, if the counter becomes greater than the number of elements in its class, the counter is set equal to 1 and the counter adjustment procedure is repeated for the previous classes encountered in the path until one of the counters can be increased. The procedure is terminated when, after a cycle, all the counters are again equal to 1.

Each element in the path must belong to a different class; otherwise the path would not be minimal. A path must then be terminated when the input node of the last element that has been added corresponds to a previous class encountered in the path. The counters are then adjusted as indicated before, and, after deleting the wrong path, the procedure can be repeated to obtain a new path.

The path finding procedure can be visualized in terms of an example. The non-empty classes for the system representing the diagram shown in Figure 4 are given in eq 12 to

Table I

Element no.	C_2	C_3	C_4	C_5
	Classes			
1	(1, A, 2)	(2, D, 3)	(1, C, 4)	(3, F, 5)
2	(1, B, 2)			(3, G, 5)
3				(4, E, 5)
	$N_2 = 2$	$N_3 = 1$	$N_4 = 1$	$N_5 = 3$

15. The elements in each class are numbered and listed as columns in Table I. The number of elements in each class is given in the last row, where N_i will represent the number of elements in C_i.

There will be four counters, each one corresponding to one of the four classes shown in the table. The procedure is started by initially setting each counter equal to 1. Let the counters be denoted by $K_{i,j}$, where the subscripts i and j will indicate the class and path number, respectively. Then, $K_{5,1} = K_{4,1} = K_{3,1} = K_{2,1} = 1$. The final node of the system is node 5. Therefore, the first element in each path set will belong to C_5.

First Path. Since $K_{5,1} = 1$, the first element in path 1 will be (3, F, 5). The input node of this element is node 3. Therefore, the second element will belong to C_3.

Since $K_{3,1} = 1$, the second element in the path will be (2, D, 3), which indicates that the next element belongs to C_2.

Since $K_{2,1} = 1$, the third element is (1, A, 2). The first path set is now complete since node 1 is the input node for the system

$$P_1 = \{(3, F, 5), (2, D, 3), (1, A, 2)\} \tag{35}$$

and the corresponding path (written in reverse order) is:

$$p_1 = 5\text{–}F\text{–}3\text{–}D\text{–}2\text{–}A\text{–}1 \tag{36}$$

The next step in the procedure is to reset the counters in order to obtain the second path. The last element in P_1, (1, A, 2), belongs to C_2. Therefore

$$K_{2,2} = K_{2,1} + 1 = 1 + 1 = 2 = N_2 \tag{37}$$

Since the number of elements in C_2 has not been exceeded, the remaining counters are not changed. Thus

$$\begin{aligned} K_{5,2} &= K_{5,1} = 1 \\ K_{4,2} &= K_{4,1} = 1 \\ K_{3,2} &= K_{3,1} = 1 \end{aligned} \tag{38}$$

Second Path. By starting again from node 5, the procedure is repeated and the elements in the second path set are obtained as follows:

$$\begin{aligned} K_{5,2} &= 1 \rightarrow (3, F, 5) \\ K_{3,2} &= 1 \rightarrow (2, D, 3) \\ K_{2,2} &= 2 \rightarrow (1, B, 2) \end{aligned}$$

thus,

$$P_2 = \{(3, F, 5), (2, D, 3), (1, B, 2)\} \tag{39}$$

and

$$P_2 = 5\text{–}F\text{–}3\text{–}D\text{–}2\text{–}B\text{–}1 \tag{40}$$

As before, the last element, (1, B, 2), belongs to C_2, but this time the number of elements in C_2 will be exceeded if its corresponding counter is incremented: $K_{2,2} + 1 = 2 + 1 = 3 > N_2$. Therefore, the counter must be reset to 1: $K_{2,3} = 1$.

The procedure must then be repeated for the previous class represented in the path. The previous element, (2, D, 3), belongs to C_3. But, as before: $K_{3,2} + 1 = 1 + 1 = 2 > N_3$. Therefore, $K_{3,3} = 1$. The remaining element in P_2, (3, F, 5), belongs to C_5. This time, $K_{5,2} + 1 = 1 + 1 = 2 < N_5$. Then, $K_{5,3} = 2$. Since C_4 is not represented in P_2, its counter still remains unchanged, $K_{4,3} = K_{4,2} = 1$. This procedure is continued until all five paths are identified. This process is summarized in Table II.

Table II. S = {(1, A, 2), (1, B, 2), (1, C, 4), (2, D, 3), (4, E, 5), (3, F, 5), (3, G, 5)}

	C_2	C_3	C_4	C_5
	Classes			
1	(1, A, 2)	(2, D, 3)	(1, C, 4)	(3, F, 5)
2	(1, B, 2)			(3, G, 5)
3				(4, E, 5)
	$N_2 = 2$	$N_3 = 1$	$N_4 = 1$	$N_5 = 3$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	
	Counters						
$K_{2,j}$	1 →	2 →	1 →	2 →	1	1	
$K_{3,j}$	1	1 →	1	1 →	1	1	
$K_{4,j}$	1	1	1	1	1 →	1	C-
$K_{5,j}$	1	1 →	2 →	2 →	3 →	1	
	↓	↓	↓	↓	↓		
	p_1	p_2	p_3	p_4	p_5		
	Paths						
	5	5	5	5	5		
	F	F	G	G	E		
	3	3	3	3	4		
	D	D	D	D	C		
	2	2	2	2	1		
	A	B	A	B			
	1	1	1	1			

Athough the diagram for the example above does not contain any loops, these are easily handled and examples involving loops are available (Caceres, 1974).

Applications. A computer program was developed for application of the path finding algorithm. One of the main features of this program is its input simplicity since only the list of elements needs to be supplied. The position in which an element appears in the list is irrelevant, and the nodes do not have to be numbered in any particular order. The input node for the system is easily detected because it does not appear as the output node of any elements. Similarly, the output node of the system does not appear as an input node. The classes are obtained by grouping together all those elements that have the same output nodes. As such, the algorithm is also applicable to systems that have multiple inputs and outputs, a necessary requirement for the analysis of chemical process networks.

No examples have been found which cannot be solved by the routine application of the algorithm. It compares favorably with other methods that have been developed. A computer program developed by Batts (1971), and Nelson et al. (1970), cannot handle systems containing loops and has been structured so as to limit its application to small systems. The user needs to supply a list of predecessors for each element in the diagram (in other words, the class sets must be provided). These elements have to be numbered and must be listed in sequential order. Special coding is required to identify input and output elements. Similarly, the algorithm of Misra and Rao (1970) fails to apply to diagrams with loops. In addition, the method requires the nodes to be numbered in ascending order and the connection of elements must be supplied in the form of a connection matrix.

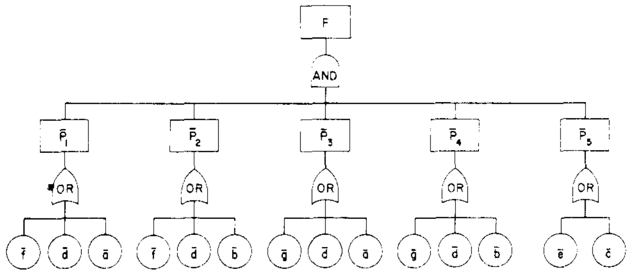

Figure 5. Fault tree for Figure 4.

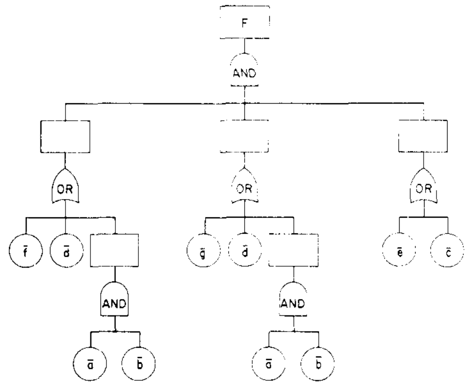

Figure 6. Alternate fault tree for Figure 4.

Fault Tree Generation

A fault tree is just a graphical representation of a Boolean expression for system failure. Therefore, there is no unique way of representing a fault tree since the number of equivalent Boolean expressions is infinite.

The path sets obtained for Figure 4 are given in Table II. A Boolean expression for the failure of this system can be written as:

$$F = (\bar{f} + \bar{d} + \bar{a}) \cdot (\bar{f} + \bar{d} + \bar{b}) \cdot (\bar{g} + \bar{d} + \bar{a}) \cdot (\bar{g} + \bar{d} + \bar{b}) \cdot (\bar{e} + \bar{c}) \quad (41)$$

where each of the expressions in parenthesis corresponds to a path set failure. Its corresponding fault tree is shown in Figure 5.

Equation 4 can be reduced to give:

$$F = \bar{f} + \bar{d} + (\bar{a} \cdot \bar{b}) \cdot \bar{g} + \bar{d} + (\bar{a} \cdot \bar{b}) \cdot (\bar{e} + \bar{c}) \quad (42)$$

corresponding to the fault tree in Figure 6.

Similarly, eq 42 can be further reduced to

$$F = (\bar{f} \cdot \bar{g}) + \bar{d} + (\bar{a} \cdot \bar{b}) \cdot (\bar{e} + \bar{c}) \quad (43)$$

which yields the fault tree shown in Figure 7.

Although equivalent, each of these fault trees has its own advantages and disadvantages. In particular, the greatest advantage of the fault tree shown in Figure 7 is that it is free of redundancies and all of its events are independent. It is therefore ideal for quantitive analysis. Several papers have been published indicating different procedures for reducing a fault tree that will yield an expression such as eq 43, and, at present, this aspect of quantitive analysis does not pose any major problems (Barlow and Chatterjee, 1973; Bass et al., 1974; Caceres, 1974; Crosetti, 1971; Crosetti and Bruce, 1970; Gandhi, 1973; Fussel, 1974a; Haasl, 1965). However, from a qualitative point of view, the fault tree obtained from eq 43 is very poor. One of the major advantages of fault tree analysis is in subsystem trade off studies to improve the reliability of a system. This cannot be done with a reduced fault tree of this type since the subsystems of interest are not represented.

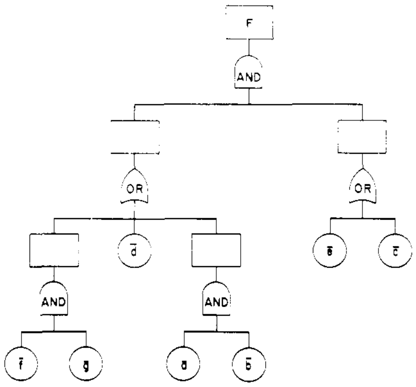

Figure 7. Alternate fault tree for Figure 4.

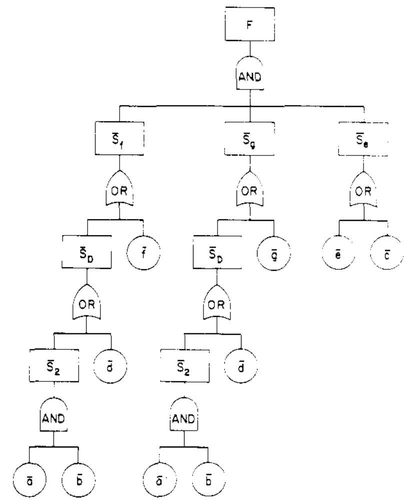

Figure 8. Fault tree corresponding to eq 44–50.

The last point indicated in the above paragraph is what constitutes one of the major potentials of the fault tree corresponding to eq 42. By simple Boolean manipulations it can be decomposed into:

$$F = \bar{S}_F \cdot \bar{S}_G \cdot \bar{S}_E \quad (44)$$

where

$$\bar{S}_F = \bar{f} + \bar{S}_3 \quad (45)$$

$$\bar{S}_G = \bar{g} + \bar{S}_3 \quad (46)$$

$$\bar{S}_E = \bar{e} + \bar{S}_4 \quad (47)$$

$$\bar{S}_3 = \bar{S}_D = \bar{d} + \bar{S}_2 \quad (48)$$

$$\bar{S}_4 = \bar{S}_c = \bar{c} \quad (49)$$

$$\bar{S}_2 = \bar{a} \cdot \bar{b} = \bar{S}_A \cdot \bar{S}_B \quad (50)$$

The fault tree representing all these relations is shown in Figure 8. Note that each and every one of the subsystem failures is represented. This type of fault tree is qualitatively very descriptive and can be directly applied to the analysis of sequential failures (as is usually the case with chemical process plants) unless the situation is complicated by feedforward or feedback loops.

This last idea has been used as the basis for developing an automatic procedure for constructing a sequential fault tree from all minimal paths of a system.

If each different path (including all of its nodes, except for the initial node) obtained from the path finding algorithm is listed as a separate column (with the final node appearing at the top), a table (or matrix for computer purposes) is obtained, such as those appearing at the bottom of

Table III

p_1	p_2	p_3	p_4	p_5
5				
F		*G*		*E*
3		3		4
D		*D*		*C*
2		2		
A	*B*	*A*	*B*	

Table IV

p_1	p_2	p_3	p_4	p_5
5				
F		*G*		*E*
D		*D*		*C*
2		2		
A	*B*	*A*	*B*	

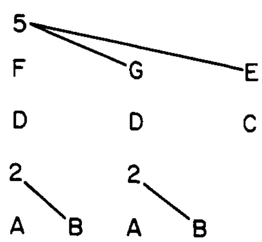

Figure 9. Fault tree structure.

Table II, where the elements in odd numbered rows will correspond to nodes and those in even numbered rows will correspond to blocks in the diagram. At this point it is necessary to stress the importance of listing each consecutive path under a different consecutive column (i.e., the first path obtained should be listed in column 1, the second path in column 2, etc.).

If the top portions of each path that appear more than once in consecutive columns are eliminated starting from left to right, Table II (bottom) is reduced to the form seen in Table III. This reduction can be shown to be equivalent to the Boolean reduction used to obtain eq 42.

The next step in the procedure requires that a node should be eliminated from the path if its corresponding class contains only one element. This is consistent with the fact that, in this case, the subsystem represented by the node will be identical with the block subsystem of the block following it.

For the example being considered, only the classes corresponding to nodes 5 and 2 (from eq 15 and 12, respectively) contain more than one element. Therefore, nodes 4 and 3 are eliminated and Table III becomes Table IV.

At this point, the basic fault tree structure is inherent in the table, and can be visualized in Figure 9 where lines have been drawn from each node to the corresponding blocks in its class. In order to construct a fault tree from this structure, it is necessary to associate each of the remaining nodes and blocks with a portion of the fault tree.

Since each node left in the structure is associated with an AND gate and a nodal subsystem, it can be used to identify a portion of the fault tree given by Figure 10. Similarly, each block that is not a terminal block of a path can be related to the failure of its block subsystem and to the failure of the unit represented by that block, as seen in Figure 11. Finally, each terminal block (appearing at the bottom of the path) can be represented as observed in Figure 12. The

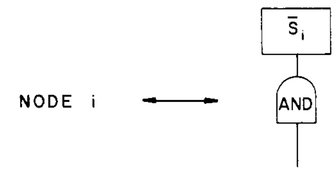

Figure 10. Portion of fault tree.

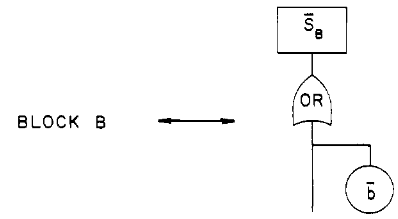

Figure 11. Unit block in a fault tree.

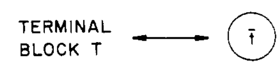

Figure 12. Terminal block in a fault tree.

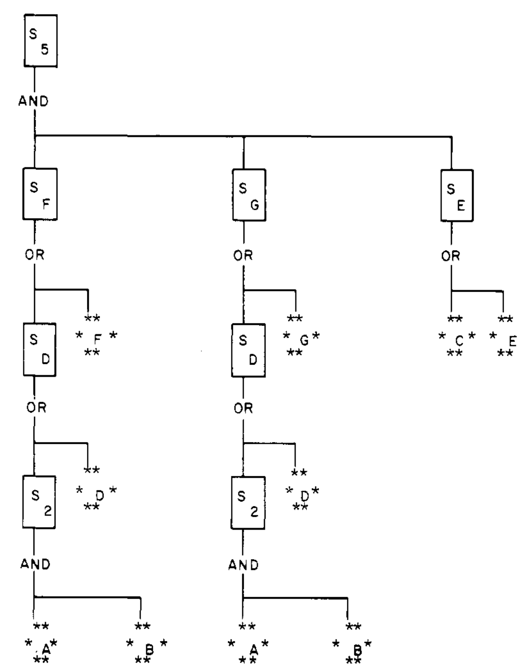

Figure 13. System fault tree.

fault tree for the system as given by the computer printout is shown in Figure 13.

Applications. The automatic fault tree construction procedure has been shown to be dependent on the path finding algorithm. Therefore, complex systems are not a limitation to its applicability. The only restriction arise from the amount of computer core that is required by a matrix where all the paths are stored. Calculation time does not seem to be a limiting factor since a diagram containing 55 paths can be processed on 0.78 min (CPU) time on an IBM 360/44. If cut sets and availabilities are required, the output from the fault tree construction algorithm can be used in conjunction with KITT, PREP, or MOCUS (Fussel, 1974b; Vesely, 1970).

Literature Cited

Barlow, R. E., Chatterjee, P., "Introduction to Fault Tree Analysis", Report ORC 73-30, Operations Research Center, University of California, Berkeley, Dec 1973.

Barlow, R. E., Proschan, F., Hunter, L. C., "Mathematical Theory of Reliability", Wiley, New York, N.Y., 1965.
Bass, L., Wynholds, H. W., Porterfield, W. R., "Fault Tree Graphics", presented at the Conference on Reliability and Fault Tree Analysis, University of California, Berkeley, Sept 1974.
Batts, J. R., *IEEE Trans. Reliab.*, **R-20**, 88 (1971).
Bazovsky, I., "Reliability Theory and Practice", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1961.
Caceres, S., M.S. Thesis, University of Houston, Houston, Texas, Dec 1974.
Crosetti, P. A., *IEEE Trans. Nucl., Sci.*, **NS-18**, 465 (1971).
Crosetti, P. A., Bruce, R. A., *Ann. Reliability Maintainability*, **9**, 230 (1970).
Fussel, J. B., "Fault Tree Analysis—Concepts and Techniques", in "Generic Techniques in Reliability Assessment", E. J. Henley, J. Lynn, Ed., Nordhoff Publishing Co., Leyden, The Netherlands, 1974a.
Fussel, J. B., "MOCUS—A Computer Program to Obtain Minimal Sets from Fault Trees", ANCR-1156, 1974b.
Fussel, J. B., Powers, G. J., Bennets, R. G., *IEEE Trans. Reliab.*, **R-23**, 51 (1974c).
Gandhi, S., Ph.D. Thesis, University of Houston, Houston, Texas, 1973.
Green, E. A., Bourne, A. J., "Reliability Technology", Wiley, New York, N.Y., 1972.
Haasl, D., "Advanced Concepts in Fault Tree Analysis", Systems Safety Symposium, sponsored by the University of Washington, Seattle, Wash., 1965.
Henley, E. J., Williams, R. A., "Graph Theory in Modern Engineering", Academic Press, New York, N.Y., 1973.
Inoue, K., Henley, E. J., "Computer Aided Reliability and Safety Analysis of Complex Systems", IFAC World Congress, Boston, Massachusetts, Parr. III, D, 1, IFAC. Secretariat, Pittsburgh, Pa., Sept 1975.
Messinger, M., Shooman, M. L., *Proc. IEEE Ann. Symposium on Reliability*, 292 (1967).
Misra, K. B., Rao, T. S. M., *IEEE Trans. Reliab.*, **R-19**, 19 (1970).
Nelson, A. C., Batts, J. R., Beadles, R. L., *IEEE Trans. Reliab.*, **R-19**, 61 (1970).
Powers, G. J., Tompkins, F. C., Jr., *AIChE J.*, **20**, 376 (1974).
Semanderes, S. N., *IEEE Trans. Nucl. Sci.*, 126 (Nov 1970).
Shooman, M. L., *IEEE Trans. Reliab.*, **R-19**, 74 (1970).
Vesely, W. E., "PREP and KITT, Computer Codes for the Automatic Evaluation of a Fault Tree", IN-1340, Aug 1970.

Received for review June 25, 1975
Accepted January 26, 1976

The Effect of Coalescence on the Average Drop Size in Liquid–Liquid Dispersions

Michael A. Delichatsios and Ronald F. Probstein*

Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

It is shown that the increase of drop sizes with the fraction by volume of all drops (holdup) in agitated liquid–liquid dispersions cannot be attributed entirely to turbulence damping caused by the dispersed phase. A new model, based on the role of coalescence of the dispersed phase, is suggested to account for the observed drop size behavior. The coalescence frequency resulting from binary drop collisions is equated to an effective breakup frequency to yield a semiempirical relation for the increase in drop sizes with holdup. The relation explains some differences among reported experimental results and correlates the data from systems with different degrees of chemical destabilization.

Introduction

It has recently been argued (Doulah, 1975) that the increase of the steady-state average drop size with dispersed phase volume fraction or holdup in an agitated liquid–liquid dispersion can be adequately explained by increased damping of the turbulence.

An alternative mechanism proposed by a number of authors (see, e.g., Mlynek and Resnick, 1972) is that the holdup behavior represents an effect of coalescence. In what follows we shall show that only coalescence could account for the magnitude of the observed holdup effect. We then derive an expression for the drop size for a model in which coalescense is a dominant mechanism.

Average drop sizes in agitated liquid–liquid dispersions can be correlated by the relation (see, e.g., Mlynek and Resnick, 1972)

$$d = C_1 \epsilon^{-2/5} \left(\frac{\sigma}{\rho}\right)^{3/5} f(\phi) \qquad (1)$$

where C_1 is a constant of order 1; ϵ is the rate of turbulent energy dissipation per unit mass; σ is the surface tension of the dispersed phase relative to the continuous phase; ρ is the density of the continuous phase, assumed not much different than the density of the dispersed phase; ϕ is the volume fraction of the dispersed phase, i.e., the holdup; and $f(\phi)$ is the holdup function, expressed empirically by the linear relation

$$f(\phi) = 1 + C_2\phi \qquad (2)$$

There does not appear to be any general agreement on the value of C_2, the reported range varying between 2.5 and 9 (Mlynek and Resnick, 1972).

Recent experimental results on two-phase jet flows, reported in the Russian literature (Laats and Frishman, 1974) have shown that the damping of turbulent intensities can be approximated by

$$\frac{u'}{u_0'} = \frac{1 + 0.2\phi}{1 + \phi} \qquad (3)$$

where u' is the root-mean-square turbulent fluctuating velocity in a dispersion of particles and u_0' is the respective value in a free particle fluid. Using eq 3 and the relation (Batchelor, 1960)

$$\epsilon \sim \frac{u'^3}{L} \qquad (4)$$

where the macroscale of turbulence L is essentially determined by the physical dimensions of the apparatus, it follows that the available turbulent energy is damped by the factor $(u'/u_0')^3$.