KfK 2761
EUR 5756e
Februar 1979

Fault Tree Analysis with Multistate Components

L. Caldarola
Institut für Reaktorentwicklung
Projekt Nukleare Sicherheit

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Reaktorentwicklung
Projekt Nukleare Sicherheit

KfK 2761
EUR 5756e

Fault Tree Analysis with Multistate Components

L. Caldarola

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

FAULT TREE ANALYSIS WITH MULTISTATE COMPONENTS

ABSTRACT

A general analytical theory has been developed which allows one to calculate the occurrence probability of the top event of a fault tree with multistate (more than two states) components.

It is shown that, in order to correctly describe a system with multistate components, a special type of Boolean algebra is required. This is called "Boolean algebra with restrictions on variables" and its basic rules are the same as those of the traditional Boolean algebra with some additional restrictions on the variables. These restrictions are extensively discussed in the paper.

Important features of the method are the identification of the complete base and of the smallest irredundant base of a Boolean function which does not necessarily need to be coherent. It is shown that the identification of the complete base of a Boolean function requires the application of some algorithms which are not used in today's computer programmes for fault tree analysis.

The problem of statistical dependence among primary components is discussed. The paper includes a small demonstrative example to illustrate the method. The example includes also statistical dependent components.

Zusammenfassung

Fehlerbaumanalyse mit Komponenten mit mehreren Zuständen

Es wurde eine allgemeine analytische Theorie entwickelt, mit der die Eintrittswahrscheinlichkeit des TOP-Ereignisses eines Fehlerbaumes mit Komponenten mit mehreren Zuständen (mehr als zwei) berechnet werden kann.

Es wird gezeigt, daß eine spezielle Boolesche Algebra benötigt wird, um ein System mit Komponenten mit mehreren Zuständen richtig zu beschreiben. Es ist die sogenannte "Boolesche Algebra mit Einschränkungen bezüglich der Variablen". Ihre Grundregeln sind die gleichen wie bei der traditionellen Booleschen Algebra mit einigen zusätzlichen Einschränkungen bezüglich der Variablen. Diese Einschränkungen werden ausführlich diskutiert.

Wichtige Merkmale der Methode sind die Identifizierung der vollständigen Basis sowie der kleinsten nicht redundanten Basis einer Booleschen Funktion, die nicht unbedingt kohärent sein muß. Es wird gezeigt, daß zur Identifizierung der vollständigen Basis einer Booleschen Funktion einige Algorithmen angewandt werden müssen, die in den derzeitigen Rechenprogrammen für die Fehlerbaumanalyse noch nicht benutzt werden.

Das Problem der statistischen Abhängigkeit primärer Komponenten voneinander wird diskutiert.

Im Bericht wird auch ein Beispiel mit statistisch abhängigen primären Komponenten angegeben.

INTRODUCTION

The evaluation of the occurrence probability of the top event of a fault tree can be carried out by means of simulation methods (Monte Carlo-type methods) or by means of analytical methods. Numerical 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. In addition, since the failure probability of a system is usually very low, precise results can be achieved only at the expense of very long computational times.

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. In 1970 Vesely[1] gave the foundations of the analytical method for fault tree analysis. However, Vesely's method can be applied only to coherent systems with binary (two states) components. Another important limitation of the method is that the boolean function which describes the top variable of the fault tree must not contain negated variables. Since there are components (like a switch) which have more than two states, and since the technique of multistate super-components can be used to remove statistical dependencies from the fault tree[3], a theory has been developed at the nuclear research center of Karlsruhe[2,3] to handle systems with multistate components. One interesting feature of the method is that the boolean function which describes the top variable of the fault tree does not necessarily need to be coherent. In addition boolean functions containing negated variables can be treated.

In this paper a formalization of the theory by means of the so called "boolean algebra with restriction on variables" has been developed. In addition the basic and important boolean operations of this special type of boolean algebra are described. The paper also includes a small demonstrative example to illustrate the method.

1. BOOLEAN ALGEBRA WITH RESTRICTIONS ON VARIABLES. DEFINITION OF FAULT TREE

Let us consider a variable which can take discrete values (states) only. The set of all possible values, which the variable can take, constitute the state space associated to that variable. Each value of the variable is called member or element of the state space. The variable can take only one value of its state space at a time. The value taken by the variable at a given time is called event.

Variables can be classified in two large categories: primary variables and non-primary variables.

A variable is called primary variable if and only if the occurrence probabilities of all members of its state space are as such already available. A primary variable can be the variable associated to a primary component of a complex system, such as a pump, a relay etc. Probability data associated to the occurrence of the states (failed, not failed etc.) of these components are in fact in general available from data banks. The values which a primary variable can take are called primary events. A non-primary variable can be, for instance, the variable associated to a complex system such as the emergency core cooling system of a nuclear power plant. Probability data related to the events "system failed", "system intact", are in fact as such in general not available.

We consider a system at a fixed moment in time. The state of the system at a given time obviously depends upon the state at that time of each individual component belonging to the system. We now select a special set of states of the system (i.e. the set of all failed states) and call it "top" (with small letters). We associate to it a boolean variable which we call TOP (with capital letters). The variable TOP will take the value 1 (true) if the system occupies one of the states belonging to the selected set and the value 0 (false) otherwise.

Any non-primary variable can be chosen as TOP variable. The chosen set of states "top" is called partition of the state space of the system. If we want now to calculate the occurrence probability of the event

$$\text{top} \equiv \left\{ \text{TOP} = 1 \right\}$$

we must first dissect the TOP variable into combinations of primary variables, that is to express the TOP variable as a proper function of the primary variables. The occurrence probability of the event $\{\text{TOP} = 1\}$ can then be calculated as a function of the occurrence probabilities of the primary events.

Due to the complexity of the systems, the operation of dissection of the TOP variable into combinations of primary variables is in general carried out in steps. The TOP variable is first dissected into combinations of simpler non-primary variables (intermediate variables). These intermediate variables are in turn dissected into combinations of even simpler intermediate variables and so on. The process of dissection comes to an end when all combinations are combinations of primary variables only.

The process of dissection can be carried out in a graphic form by constructing a fault tree of the chosen TOP variable.

A fault tree is a logic model which shows in diagrammatic form the connections between the TOP variable and the primary variables.

A more precise definition of a fault tree can be given by making use of the graph theory.

"A fault tree is a finite directed graph without loops. Each vertex may be in one of several states. For each vertex a function is given which specifies its states in terms of the states of its predecessors. Those vertices without predecessors are considered the independent variables of the fault tree." 4

We are following the graphical terminology of Berge[5] here. In the technical literature a vertex with predecessors is currently called gate. The output variable of a gate is called (improperly) output event of the gate. An input variable to a gate is called predecessor of (again improperly) input event to the gate. In the technical literature the improper terms TOP event, primary event are also currently used. One should instead use the more correct terms TOP variable and primary variable. In fact the word event is used (in the set theory and in the propositional calculus) to indicate a value or a set of values of a variable. We shall use the correct mathematical terminology here.

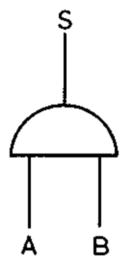
Note that in the above definition of fault tree the term "independent variable" is used and not "primary variable". The word independent in this context means "logically independent", that is each input variable to the tree can take any value of its sample space independently from the values taken by the other input variables. The truth table of the fault tree containes all possible combinations among the values of the input variables. Each row of the truth table represents a state of the system. If all primary components of the system are characterized by only two states (inact and failed), we assign to each primary component a boolean variable which takes the value 1 if the component is failed and the value 0 if the component is intact. These are the primary variables which are pairwise mutually logically independent. The primary variables in this case are also independent variables.

If the fault tree has m binary primary components, that is m input variables, the truth table of the fault tree has 2^m rows.

The function which links the output to the inputs of a gate are boolean functions. The basic gates are the AND (conjunction), OR (disjunction) and the NOT (negative) gates.

Let us first consider an AND gate with two inputs, namely A and B (Fig. 1-1)

AND Gate



Truth Table

Inputs		Output
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 1-1. AND Gate $(S = A \wedge B)$

The truth table of Fig. 1-1 gives the value of the output S for each pair of values of the two predecessors A and B. This truth table can be expressed in words as follows

"Output takes the value 1 if and only if all predecessors take the value 1, and the value 0 if at least one of its predecessors takes the value 0."

We now order the values 1 and 0 in that we say, for instance the 1 is larger than 0

$$1 > 0$$

We can synthetize the AND operation as follows

$$S = \min (A; B)$$

which means that S takes the smallest between the values of A and B.

Fig. 1-2 shows the OR gate with associated truth table.

OR Gate

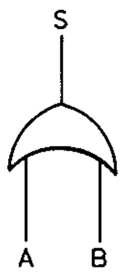


Fig. 1-2. OR Gate $(S = A \vee B)$

Truth Table

Inputs		Output
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Also in this case the truth table of Fig. 1-2 can be expressed in words as follows

" 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."

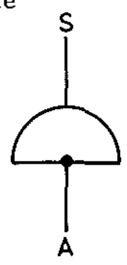
If we put $1 > 0$, we can write in the case of the OR gate

$$S = \max (A; B)$$

which means that S takes the largest between the values of A and B.

Fig. 1-3 shows the NOT gate with associated truth table.

NOT Gate



Truth Table

Inputs	Output
A	S
0	1
1	0

Fig. 1-3. NOT Gate $(S = \bar{A})$

In words

"Output takes the value 1 if predecessor takes the value 0 and viceversa."

In a fault tree the truth tables of each gate are properly combined to get the truth table of the TOP. We show this by means of an example.

We consider the simple fault tree of Fig. 1-4 (Example No.1). Each one of the two OR gates will be characterized by a truth table of the type of Fig. 1-2. The outputs of the two OR gates will be the inputs to the AND gate, which has a truth table of the type shown in Fig. 1-1. By properly combining the three truth tables one finally gets the overall truth table of the fault tree. This truth table has 16 rows (Fig. 1-5).

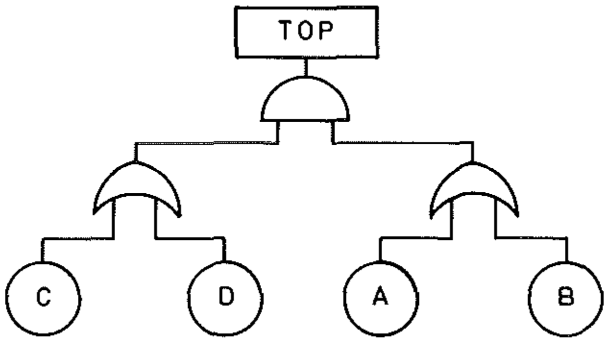


Fig. 1-4. Fault Tree - TOP = $(C \vee D) \wedge (A \vee B)$

	Row Number	Inputs				Output
		A	B	C	D	TOP
+	1	0	0	0	0	0
	2	0	0	0	1	0
+	3	0	0	1	0	0
	4	0	0	1	1	0
+	5	0	1	0	0	0
	6	0	1	0	1	1
+	7	0	1	1	0	1
	8	0	1	1	1	1
	9	1	0	0	0	0
—	10	1	0	0	1	1
	11	1	0	1	0	1
—	12	1	0	1	1	1
	13	1	1	0	0	0
—	14	1	1	0	1	1
	15	1	1	1	0	1
—	16	1	1	1	1	1

Fig. 1-5.
Complete truth table of the fault tree of Fig. 1-4 (Example No.1)

In the previous example we have assumed that all primary components are binary. There are however primary components which are characterized by more than two states. For instance an electrical switch is characterized by at least three states, namely (1) intact, (2) failed in closed position and (3) failed in open position.

One could in this case assign to each primary component a multivalued variable characterized by a number of values equal to the number of states of the primary component. Each value of the variable corresponds to a specific state of the primary component. These multivalued variables are the primary variables. They are pairwise mutually logically independent. Primary variables and independent variables are also in this case identical. The function which links the output to the input of a gate is a logic function which is in general not boolean. This way of thinking is consistent with the definition of fault tree given above. There is however, a considerable drawback, namely that a more complicated multivalued logic must be developed. The basic gates are not any more simply the AND, OR and NOT gates as in the case of the boolean algebra. New basic gates must be found. Some authors[6] are following this way of thinking. We want to follow another path instead. We want to have primary variables which are binary.

Let us consider the state space of a primary component. A state belonging to the state space of a primary component is called primary state. The event of the primary component occupying a given state of its state space at a given time is called primary event.

A primary component will be indicated by the small letter c followed by an integer positive number (c1; c2; c3 etc.). In general we shall have cj with j=1;2...; m, where "m" is the total number of primary components contained in the system.

A state of a primary component will be indicated by the same notation of the primary component to which it belongs followed by a positive integer number as an index. (cj_1; cj_2; cj_3 etc.) In general we shall have cj_q with $q=1;2;...nj$, where nj is the total number of states belonging to primary component cj. We can now associate to each state cj_q a boolean variable Cj_q which takes the value 1 (true) if primary component cj occupies state cj_q and the value 0 (false) if cj does not occupy cj_q.

The event

$$\left\{ Cj_q = 1 \right\} \equiv cj_q$$

indicates that primary component cj occupies state cj_q.

Conversely, the event

$$\left\{ Cj_q = 0 \right\} \equiv \bigcup_{k=1}^{nj} cj_k \qquad k \neq q$$

indicates that primary component cj does not occupy state cj_q and therefore occupies one of its other possible states.

Note the one to one correspondance between state cj_q (small c) and boolean variable Cj_q (capital C) associated to it. We obviously have

$$cj_q = \left\{ Cj_q = 1 \right\} \quad \text{and} \quad \overline{cj}_q \equiv \left\{ Cj_q = 0 \right\}$$

We shall say that the primary state cj_q belongs to component cj ($cj_q \in cj$). The word "primary component" (with small c) is here intended as the set of all possible states which the component can occupy.

We shall also say that the variable Cj_q belongs to Component Cj ($Cj_q \in Cj$). The word "primary Component" (with capital C) means here the complete set of variables associated to its states.

The binary variables Cj_q are the primary variables. <u>They are however not any more pairwise mutually independent.</u>

Since a primary component <u>must</u> occupy one of its states and <u>can</u> occupy only one state at a time, the variables Cj_q must obviously satisfy the following two types of restrictions.

<u>Restriction Type 1</u> The disjunction of all binary variables associated to the same primary Component is always equal to 1

$$\bigvee_{q=1}^{nj} Cj_q = 1 \qquad (1\text{-}1)$$

The notation "1" in Eq. 1-1 means "true". Eq.1 must be read as follows. The proposition "at least one of the variables Cj_q (q=1; ;...nj) takes the value 1" is true.

<u>Restrictions Type 2</u> The conjunction of two different binary variables associated to the same primary Component is always equal to 0.

$$Cj_q \wedge Cj_k = 0 \qquad q \neq k \qquad (1\text{-}2)$$

The notation "0" in Eq. 1-2 means "false". Eq. 2 must be read as follows. The proposition "both variables Cj_q and Cj_k ($q \neq k$) take the value 1" is false.

Note that there is only one restriction type 1 and $\frac{nj(nj-1)}{2}$ restrictions type 2.

Note also that Eqs. 1-1 and 1-2 can be translated straightforward in the equivalent equations among states. We have obviously

Restriction Type 1

$$\bigcup_{q=1}^{nj} cj_q = 1 \qquad (1\text{-}1a)$$

and

Restrictions Type 2

$$cj_q \cap cj_k = 0 \qquad q \neq k \qquad (1\text{-}2a)$$

Eqs. 1-1a and 1-2a have been obtained respectively from Eqs. 1-1 and 1-2 by carrying out the following simple operations.

Capital C		is replaced by	small c
Disjunction operator	$\vee$	" " "	Union operator $\cup$
Conjunction operator	$\wedge$	" " "	Intersection operator $\cap$

Note that the notation "1" and "0" in Eqs. 1-1a and 1-2a have a different meaning. They indicate respectively the "universal set" and the "empty set". Eq. 1-1a means therefore that the union of all states of a primary component constitutes an universal set, that is its complete state space. Eq. 1-1b means that the intersection of two different states of a primary component constitutes an empty set.

Since we have introduced primary variables which are not any more pairwise mutually independent, we have to slightly modify the definition of a fault tree.

"A fault tree is a finite directed graph without loops. Each vertex may be in one of several states. For each vertex a function is given which specifies its states in terms of the states of its predecessors. Those vertices without predecessors are the primary variables of the fault tree. The primary variables may satisfy some conditions (called restrictions) which are associated to the fault tree."

Since a fault tree does not contain loops, it follows that the restrictions contain primary variables only. We shall limit ourselves to consider fault trees characterized by a boolean TOP variabls and boolean primary variables which satisfy restrictions of the types given respectively by the Eqs. 1-1 and 1-2.

We consider now the truth table of the TOP variable.

Restriction type 1 means that the primary events

$$\left\{Cj_q = 0\right\} \quad ; \quad \left\{Cj_2 = 0\right\} \quad ; \quad \left\{Cj_{nj} = 0\right\}$$

cannot co-exist all together at the same time. This is equivalent saying that all the rows of the truth table in which the variables Cj_1; Cj_2; Cj_3... take simultaneously the value 0 are prohibited and must be deleted.

The restrictions type 2 mean that the primary events

$$\left\{Cj_q = 1\right\} \quad \text{and} \quad \left\{Cj_k = 1\right\} \quad , \quad q \neq k$$

cannot co-exist at the same time. This is equivalent saying that all the rows of the truth table in which both the two input variables Cj_q and Cj_k take the value 1 are prohibited and must be deleted. The following two examples will make this point clearer.

Let us consider the fault tree of Fig. 1-4 and let us assume that the primary variables A and D belong both to the same primary Component which is characterized by two states (Example No. 2). Eqs. 1-1 and 1-2 become respectively

$$A \vee D = 1 \qquad (1\text{-}3)$$
$$A \wedge D = 0 \qquad (1\text{-}4)$$

Eq. 1-3 tells us that the events $\left\{A = 0\right\}$ and $\left\{D = 0\right\}$ cannot co-exist. If we now look at the complete truth table of the fault tree (Fig. 1-5) we notice that the rows 1; 3; 5 and 7 are prohibited because in these rows A and D have both the value 0. These rows must therefore be deleted.

Eq. 1-4 tells us that the events $\left\{A = 1\right\}$ and $\left\{D = 1\right\}$ cannot co-exist. This is equivalent saying that the rows No. 10; 12; 14 and 16 (Fig. 1-5) are also prohibited and must be deleted. The truth table of the fault tree of Fig. 1-4 with the additional conditions 1-3 and 1-4 will be reduced to that of Fig. 1-6 which contains eight rows only.

The input (primary) variables of the truth table of Fig. 1-6 are not all pairwise mutually independent. In fact the eight rows containing the combinations of values (0;0) or (1;1) for the variables A and D do not appear in the truth table of Fig. 1-6.

Row Number	Inputs				Output
	A	B	C	D	TOP
1	0	0	0	1	0
2	0	0	1	1	0
3	0	1	0	1	1
4	0	1	1	1	1
5	1	0	0	0	0
6	1	0	1	0	1
7	1	1	0	0	0
8	1	1	1	0	1

Fig. 1-6. Truth Table of Example No.2

It is sometimes possible however to reduce the number of the primary variables and to get the independent variables only. In the case of example No. 2 this is possible.

We notice that Eqs. 1-3 and 1-4 can be reduced to the following equation.

$$D = \overline{A} \tag{1-5}$$

Eq. 1-5 means that, once a value has been assigned to the variable A, the variable D takes a defined value according to the truth table of Fig. 1-3 (NOT Gate). For this reason the column corresponding to the variable D in the truth table of Fig. 1-6 is redundant and can be deleted. The value of the TOP is in fact completely determined if the values of the primary variables A; B; C have been previously chosen. The truth table of Fig. 1-6 can be further reduced by deleting the column of the primary variable D (Fig. 1-7).

Row Number	Inputs			Output
	A	B	C	TOP
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1

Fig. 1-7. Truth Table of Example No. 2 (Final)

Conversely one could keep the variable D as independent variable and delete in Fig. 1-6 the column corresponding to the variable A which would now be redundant.

Let us consider again the fault tree of Fig. 1-4 and let us assume that the primary variables A and D belong both to the same primary Component (as in example No. 2) but that this component is characterized now by three states and that the primary variable associated to the third state (call it E) is not present in the fault tree (example No. 3). In this case Eqs. 1-1 and 1-2 become respectively

$$A \vee D \vee E = 1 \qquad (1\text{-}6)$$

and

$$A \wedge D = 0 \ (1\text{-}7a) \qquad A \wedge E = 0 \ (1\text{-}7b) \qquad D \wedge E = 0 \qquad (1\text{-}7c)$$

The rows 10; 12; 14; 16 of the truth table of Fig. 1-5 are <u>prohibited</u> because the events $\{A=1\}$ and $\{D=1\}$ cannot co-exist at the same time (Eq. 1-7a). By deleting these rows one obtains the truth table of Fig. 1-8 which contains 12 rows only.

Row Number	Inputs				Output
	A	B	C	D	TOP
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	1	1	0
5	0	1	0	0	0
6	0	1	0	1	1
7	0	1	1	0	1
8	0	1	1	1	1
9	1	0	0	0	0
10	1	0	1	0	1
11	1	1	0	0	0
12	1	1	1	0	1

Fig. 1-8. Truth Table of Example No. 3.

Note that in this case we don't make any use of the restrictions given by Eqs. 1-6; 1-7a and 1-7c because the primary variable E is not explicitly contained in the fault tree.

The input variables of the truth table of Fig. 1-8 are not all pairwise mutually independent. In fact the four rows which contain the combination of values (1; 1) for the variables A and D do not appear in the truth table of Fig. 1-8. In this case however it is not possible to reduce the number of primary variables as in the case of Example No. 2. In fact no column in the truth table of Fig. 1-8 is redundant.

In conclusion the following rule can be stated (Rule No. 1)

"The truth table of the TOP variable of a fault tree can be obtained from the complete truth table (in which all primary variables present in the fault tree are assumed to be pairwise mutually independent) by deleting the <u>prohibited</u> rows and the <u>redundant</u> columns. The restrictions allow one to identify these prohibited rows and redundant columns. Each survived row corresponds to a specific state of the system. The survived primary variables may or may not be pairwise mutually independent."

We notice that we have defined primary variables which are binary as in the classical boolean algebra, but not necessarily pairwise mutually independent. We shall therefore introduce the term "boolean algebra with restrictions on variables" to indicate an algebra in which the basic (primary) variables are boolean but not necessarily pairwise mutually independent. The classical boolean algebra can be considered as a particular case of this boolean algebra with restrictions on variables in that the basic variables are all pairwise mutually independent.

We now consider the states of the partition top. Each state of the partition top can be expressed by the smallest cartesian product of primary states. This expression of the state is called smallest form of the state. Consider, for instance, the row 7 of the truth table of Fig. 1-8 (Example No. 3).

$$\text{System event No. 7} = \left\{B=1\right\} \times \left\{C=1\right\} \times \left[\left\{A=0\right\} \cap \left\{D=0\right\}\right] \qquad (1\text{-}8)$$

Note that Equation 1-8 is obtained (1) by grouping all events which belong to the same component and linking them with the intersection operator $\cap$ and (2) by linking all groups with the cartesian product operator x. In fact the events $\{A=0\}$ and $\{D=0\}$ belong to the same component and must therefore be grouped together.

We now want to eliminate the groups.

From Eq. 1-6 we get

$$E = \overline{A \vee D} = \bar{A} \wedge \bar{D} \qquad (1\text{-}9)$$

From Eq. 1-9 we get

$$\left\{E=1\right\} \equiv \left\{\bar{A} \wedge \bar{D}=1\right\} \equiv \left\{\bar{A}=1\right\} \cap \left\{\bar{D}=1\right\} \qquad (1\text{-}10)$$

We have the following identities

$$\left\{ \bar{A}=1 \right\} \equiv \left\{ A=0 \right\} \qquad (1\text{-}11)$$

and

$$\left\{ \bar{D}=1 \right\} \equiv \left\{ D=0 \right\} \qquad (1\text{-}12)$$

Taking into account Eqs. 1-11 and 1-12, Eq. 1-10 becomes

$$\left\{ A=0 \right\} \cap \left\{ D=0 \right\} \equiv \left\{ E=1 \right\} \qquad (1\text{-}13)$$

Taking into account Eq. 1-13, Eq. 1-8 becomes

$$\text{System event No. 7} = \left\{ B=1 \right\} \times \left\{ C=1 \right\} \times \left\{ E=1 \right\} \qquad (1\text{-}14)$$

Note that Eq. 1-14 does not contain any more the intersection operator $\cap$ and all events contain the symbol 1.

We now introduce the notation for the states of primary components (small letters). We have

$$\left\{ B=1 \right\} \equiv b \qquad (1\text{-}15)$$

$$\left\{ C=1 \right\} \equiv c \qquad (1\text{-}16)$$

$$\left\{ E=1 \right\} \equiv e \qquad (1\text{-}17)$$

Taking into account Eqs. 1-15, 1-16, 1-17, Eq. 1-14 becomes

$$\text{System event No. 7} = bxcxe \qquad (1\text{-}18)$$

The expression on the right side of Eq. 1-18 is the smallest form of system state No. 7.

We can now state the following definition

" The smallest form of a state of a system is defined by the cartesian product of the states occupied by each single primary component belonging to the system."

We now go back to Eq. 1-14 which we can now write in a more compact form.

$$\begin{aligned} \text{System event No. 7} &= \left\{ B=1 \right\} \times \left\{ C=1 \right\} \times \left\{ E=1 \right\} \\ &= \left\{ B \wedge C \wedge E = 1 \right\} \end{aligned} \qquad (1\text{-}19)$$

From Eqs. 1-18 and 1-19, we get

$$bxcxe = \left\{ B \wedge C \wedge E = 1 \right\} \qquad (1\text{-}20)$$

Before discussing Eq.1-20, we want to introduce some new terms. A variable which results from the conjunction of primary variables is called monomial. A monomial containing two or more primary variables belonging to the same primary Component is obviously equal to zero (restrictions type 2). A non-zero monomial containing a number of primary variables equal to the number of primary components present in the system is called "complete monomial". For instance, the variable $B \wedge C \wedge E$ of Example 3 is a complete monomial". For a given system the number of complete monomials is equal to the number of its states.

Eq. 1-20 tells us that, given the complete monomial $B \wedge C \wedge E$, one obtains the minimal form of the corresponding state bxcxe by carrying out the following operations

B	is replaced by			b
C	"	"	"	c
E	"	"	"	e
conjunction operator $\wedge$	"	"	"	cartesian product operator x

We can now state the following rule (Rule No. 2).

"The smallest form of a state of a system is obtained from its corresponding complete monomial by replacing each primary variable by its associated primary state and each conjunction operator ($\wedge$) by the cartesian product operator (x).

Conversely we have

"A complete monomial of a system is obtained from the minimal form of its corresponding system state by replacing each primary state by its associated primary variable and each cartesian product operator (x) by the conjunction operator ($\wedge$)."

Going back to the truth table of Fig. 1-18 (Example No. 3), we select the rows for which TOP = 1. These are the rows No. 6, 7, 8, 10 and 12. Each selected row respresents a state of the system for which the equation TOP = 1 is satisfied. We now find the smallest form of each row. In order to do that we must introduce the states $\bar{b}$ and $\bar{c}$ which satisfy the restrictions respectively with b and c.

$b \cup \bar{b} = 1$ (1-21a) ; $b \cap \bar{b} = 0$ (1-21b)

and

$c \cup \bar{c} = 1$ (1-22a) $c \cap \bar{c} = 0$ (1-22b)

The smallest forms of the rows 6, 7, 8, 10 and 12 are given in the following table (Fig. 1.9).

System state	Smallest form
6	$b \times \bar{c} \times d$
7	$b \times c \times e$
8	$b \times c \times d$
10	$a \times \bar{b} \times c$
12	$a \times b \times c$

Fig. 1.9 Smallest form of system states (from the truth table of Fig. 1.8).

By making use of the above table we can now write

$$\text{top} = (b\times\bar{c}\times d) \cup (b\times c\times e) \cup (b\times c\times d) \cup (a\times\bar{b}\times c) \cup (a\times b\times c) \qquad (1\text{-}23)$$

Eq. 1-23 can be written as follows

$$\{\text{TOP} = 1\} = \{B\wedge\bar{C}\wedge D=1\} \cup \{B\wedge C\wedge E=1\} \cup \{B\wedge C\wedge D=1\} \cup$$
$$\cup \{A\wedge\bar{B}\wedge C=1\} \cup \{A\wedge B\wedge C=1\} \qquad (1\text{-}24)$$

Eq. 1-24 can be written in a more compact form

$$\{\text{TOP}=1\} = \{(B\wedge\bar{C}\wedge D) \vee (B\wedge C\wedge E) \vee (B\wedge C\wedge D) \vee (A\wedge\bar{B}\wedge C) \vee (A\wedge B\wedge C) = 1\} \qquad (1\text{-}25)$$

From Eq. 1-25 we also get

$$\text{TOP} = (B\wedge\bar{C}\wedge D) \vee (B\wedge C\wedge E) \vee (B\wedge C\wedge D) \vee (A\wedge\bar{B}\wedge C) \vee (A\wedge B\wedge C) \qquad (1\text{-}26)$$

Eqs. 1-23 and 1-26 tell us that given the variable TOP as a disjunction of complete monomials (Eq. 1-26) one obtains the expression of the partition top (Eq. 1-23) by carrying out the following operations

TOP	is	replaced	by	top
A	"	"	"	a
B	"	"	"	b
$\bar{B}$	"	"	"	$\bar{b}$
C	"	"	"	c
$\bar{C}$	"	"	"	$\bar{c}$
D	"	"	"	d
E	"	"	"	e
conjunction operator $\wedge$	"	"	"	cartesian product operator x
disjunction operator $\vee$	"	"	"	union operator $\cup$

The disjunction of complete monomials of a boolean function is called "disjunctive canonical form" of the function.

Now we can state the following rule (Rule No. 3)

"If the variable TOP is given in its disjunctive canonical form, the corresponding partition top is obtained by replacing each complete mononomial by the corresponding smallest form of system state and each disjunction operator ($\vee$) by the union operator ($\cup$)."

Converserly we have

"If the partition top is given in the form of union of smallest forms of states the corresponding disjunctive canonical form of the variable TOP is obtained by replacing each smallest form of state by the corresponding complete monomial and each union operator ($\cup$) by the disjunction operator ($\vee$)."

We notice that the disjunction operator $\vee$ is alsway replaced by the union operator $\cup$. The conjunction operator $\wedge$ instead is replaced by the intersection operator $\cup$ in the case of the restrictions type 2 (Eqs. 1-2 and 1-2a) and by the cartesian product operator x in the case of the complete monomials. This fact however does not cause any problem. In fact any complete monomial is a non-zero monomial which corresponds to a specific state of the system. A state is for definition a non-empty set. Since the restrictions are only used to identify the zero monomials of a boolean function that is the prohibited rows of the corresponding truth table and both are <u>always</u> deleted, it is impossible to get smallest forms of system states containing the intersection operator, and/or complete monomials which contain two or more primary variables belonging to the same component.

In conclusion the boolean algebra with restrictions on variables allows us to operate on boolean variables in a way similar to the classical boolean algebra, but with the additional complication of the restrictions. Once that the boolean expression of the TOP variable has been found, the rules No. 2 and 3 allow one to easily identify the smallest form of the states of the partition top.

The advantage of using boolean variables instead of states is obviously that of having a more flexible instrument ot operate. We show this point by developing Eq. 1-26. We notice that

$$(B \wedge \bar{C} \wedge D) \vee (B \wedge C \wedge D) = (B \wedge D) \qquad (1\text{-}27)$$

and

$$(A \wedge \bar{B} \wedge C) \vee (A \wedge B \wedge C) = (A \wedge C) \qquad (1\text{-}28)$$

Taking into account Eqs. 1-27 and 1-28, Eq. 1-26 becomes

$$\mathrm{TOP} = (B\wedge D)\vee(A\wedge C)\vee(B\wedge C\wedge E) \qquad (1\text{-}29)$$

We also notice that

$$E = \bar{A}\wedge\bar{D} \qquad (1\text{-}30)$$

and therefore

$$(B\wedge D)\vee(B\wedge C\wedge E) = B\wedge\left[D\vee(C\wedge\bar{A}\wedge\bar{D})\right] = (B\wedge D)\vee(B\wedge C\wedge\bar{A}) \qquad (1\text{-}31)$$

Taking into account Eq. 1-31, Eq. 1-29 becomes

$$\mathrm{TOP} = (B\wedge D)\vee(A\wedge C)\vee(B\wedge C\wedge\bar{A}) \qquad (1\text{-}32)$$

We have

$$(A\wedge C)\vee(B\wedge C\wedge\bar{A})=C\wedge\left[A\vee(B\wedge\bar{A})\right]= (C\wedge A)\vee(C\wedge B) \qquad (1\text{-}33)$$

Taking into account Eq. 1-33, Eq. 1-32 becomes finally

$$\mathrm{TOP} = (B\wedge D)\vee(A\wedge C)\vee(C\wedge B) \qquad (1\text{-}34)$$

The partition top is simply given by

$$\mathrm{top} = \left\{(B\wedge D)\vee(A\wedge C)\vee(C\wedge B) = 1\right\} \qquad (1\text{-}35)$$

Note that the expression of the partition top given by Eq. 1-35 (i.e. by using the boolean variables) is much simpler and much more compact than the equivalent expression given by Eq. 1-23 (i.e. by using the set theory).

2. FAULT TREE SYMBOLOGY

The graphical symbology of a fault tree which is being used here is derived from that proposed by Fussell[7] with some modifications and some additional symbols.

The symbols have been organized in two tables, namely

A. Table of Variables (Fig. 2-1)
B. Table of Basic Gates (Fig. 2-2)

The two tables are selfexplanatory so that only few additional comments are needed for a correct use of the symbols contained in them.

1. The House (Table of Variables) is used to modify the structure of the fault tree. If the House is given the value 0, the whole branch of the fault tree under the AND gate (to which the House is input) is cancelled out. If the House is given

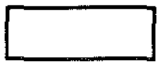
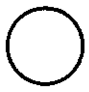
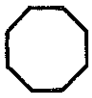
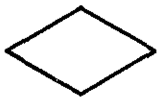
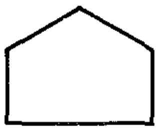
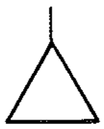
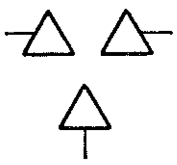
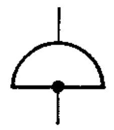
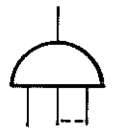
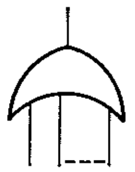
No.	Symbol	Denomination	Meaning
1		Rectangle	Variable Description
2		Circle	A primary variable belonging to an independent component.
3		Octagon	A primary variable belonging to a dependent component.
4		Diamond	A non-primary variable which would require dissection in more basic variables, but that for 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

No.	Symbol	Denomination	Boolean Notation	Output/Inputs Relationship	Rules for the Generation of the Truth Table
1		NOT	$B=\overline{A}$	$B=1-A$	Output takes the value 1 if predecessor takes the value 0 and vice versa.
2		AND	$B=\bigwedge_{i=1}^{n} A_i$	$B=\min(A_1;A_2..;A_n)$	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	$B=\bigvee_{i=1}^{n} A_i$	$B=\max(A_1;A_2..;A_n)$	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.

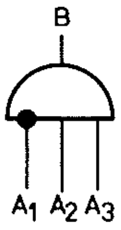
Fig. 2-2. Table of Basic Gates.

the value 1 no modification of the structure of the fault tree occurs.

2. Transfer IN and Transfer OUT (Table of Variables) are used in the case in which a variable is at the same time an output (Transfer OUT) from a gate and input (Transfer IN) to some other gates which are located (in the drawing of the fault tree) far away one from the other.

3. If an input to a gate (Tables of Basic Gates) is marked with a point, it means that the input variable is complemented (negated) before entering the gate.

For instance we have



$$B = \bar{A}_1 \wedge A_2 \wedge A_3$$

3. CONSTRUCTION OF A FAULT TREE. AN EXAMPLE

Fig. 3-1 shows a very simplified electric power supply system (EPSS) consisting of the bus bars C which are supplied either by the external network B or by the electric generator A. Network and electric generator are connected in parallel to the bus bars respectively through the electrically operated circuit breakers F and L. The dotted lines (with arrows) indicate that the position (open or closed) of each circuit breaker depends upon the state (failed or intact) of the component to which the circuit breaker is associated.

The circuit breakers in Fig. 3-1 are shown in the position open (coil deenergized). In normal operating conditions both circuit breakers F and L are closed (coil energized)and the generator A supplies electric power to the bus bars C as well as to the external network B. If the generator A fails the circuit breaker L opens and the external network feeds the bus bars C. If the network B fails the circuit breaker F opens and the generator A feeds the bus bars C only. The function of each circuit breaker is that of disconnecting its associated component (conditioning component) when this fails. If the circuit breaker fails to open, no electric voltage will be available at the bus bars C. In addition B may cause by failing the failure of A and vice versa (a failure of A may cause B to fail). Components A and B are said to be correlated.

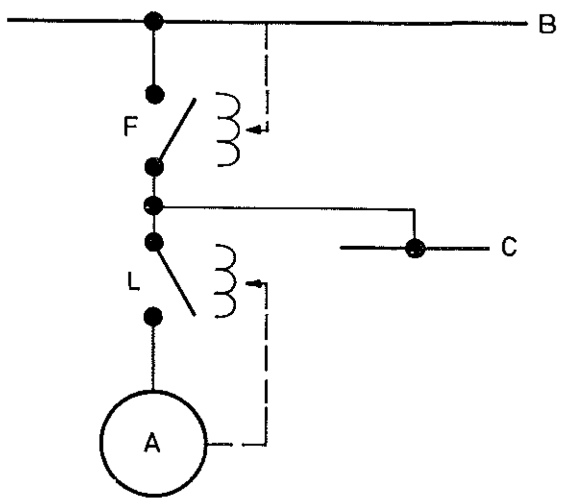


Fig. 3-1. Schematic diagram of a simplified electric power supply system (EPSS)

The primary components with associated states are shown in the table of Fig. 3-2. Here for each primary component the conditioning components are listed in the homonymous column. The correlated components,which are each other statistically dependent (in our example A and B), are also shown.

Note that in our example the conditioning components of F and L are also primary components. However, in general the conditioning components may be not primary (i.e. the variables belonging to them are not primary). In this case additional information must be given to identify these conditioning components.

We can now proceed to define the TOP variable. The EPPS is failed if no electric voltage is available at the bus bars C. We have therefore

TOP = No voltage at bus bars C

We observe that the absence of voltage at the bus bars C is caused either by the failure of the bus bars C or by the fact that no voltage arrives at C. In this way we have dissected the TOP variable into the disjunction of two other variables namely "bus bars C failed" and "no voltage at the input of bus bars C". This dissection is graphically shown in Fig. 3-3, where the OR gate G01 has the TOP as output and the other two above defined variables as input.

Primary Component		Conditioning Components	Correlated Components	State	
Denomination	Symbol			Denomination	Symbol of associated primary variable
Generator	A		B	Failed	A_1
				Intact	A_2
Network	B		A	Failed	B_1
				Intact	B_2
Bus bars	C			Failed	C_1
				Intact	C_2
Circuit Breaker F	F	B		Failed open	F_1
				Failed closed	F_2
				Intact	F_3
Circuit Breaker L	L	A		Failed open	L_1
				Failed closed	L_2
				Intact	L_3

Fig. 3-2. Table of the primary components of the EPSS,

We point out that the probability data associated to the variable "bus bars C failed" are available from reliability data banks. This variable is therefore a primary variable. We call it C_1 and we draw a circle in Fig. 3-3 because C is statistically independent (see table of Fig. 3-2). We now dissect the variable "No voltage at the input of bus bars C".

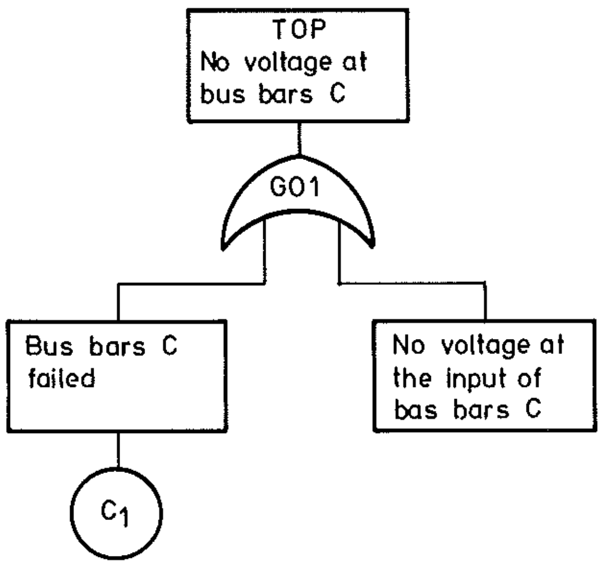


Fig. 3-3. Partial fault tree of the EPPS (1st step)

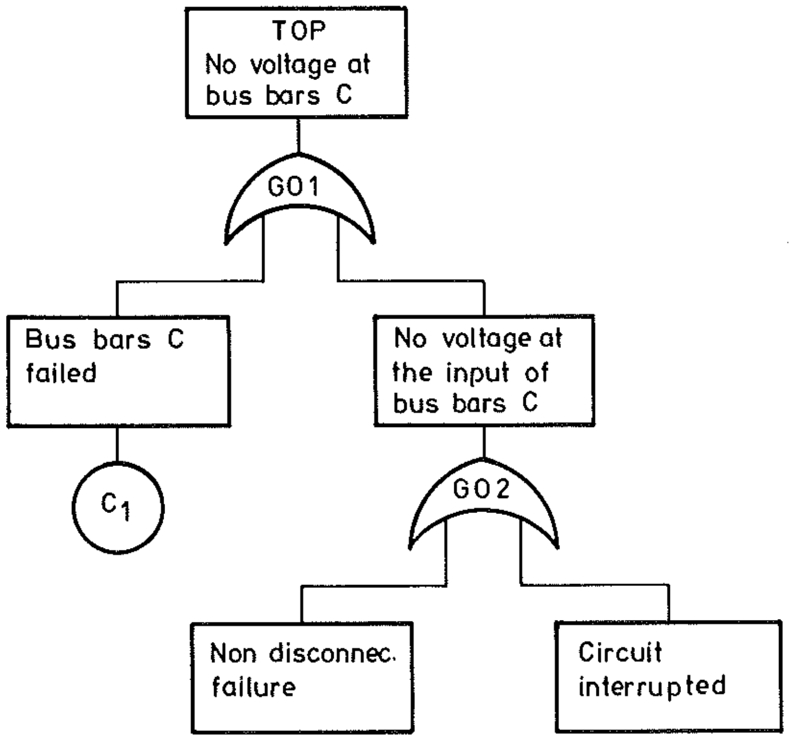


Fig. 3-4. Partial fault tree of the EPPS (2nd step)

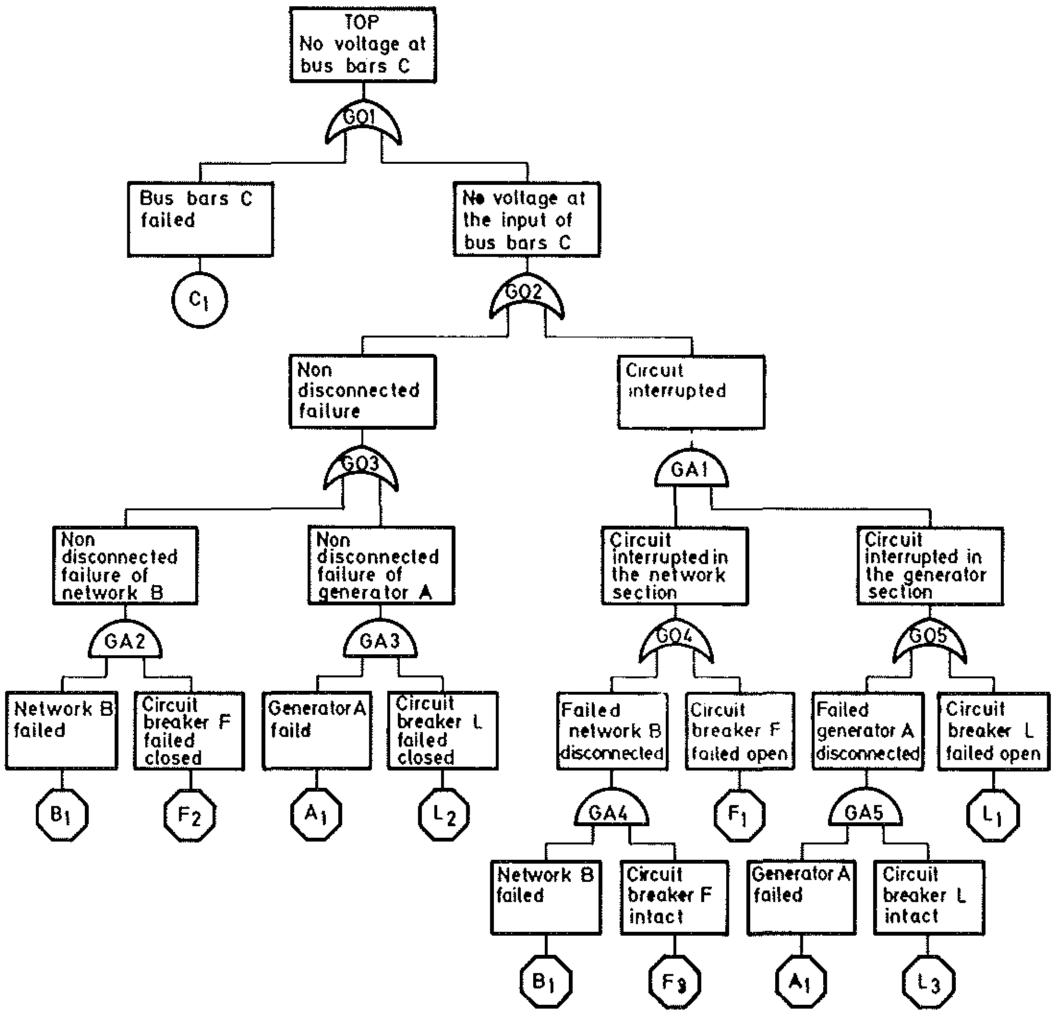
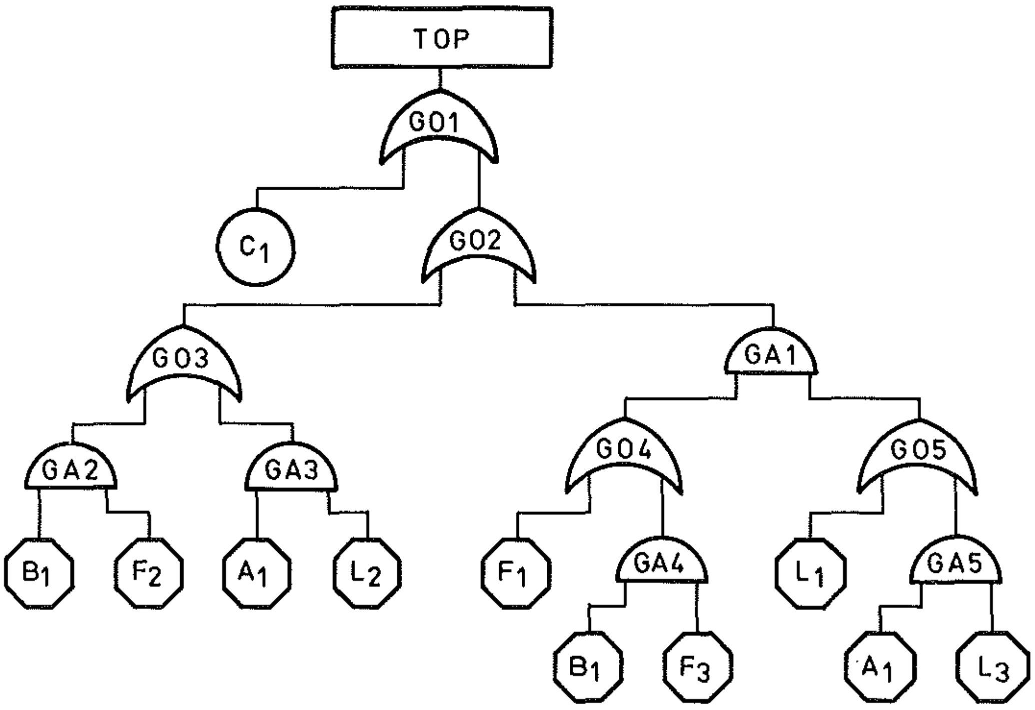

TOP
No voltage at bus bars C
GO1
Bus bars C failed
C_1
No voltage at the input of bus bars C
GO2
Non disconnected failure
Circuit interrupted
GO3
GA1
Non disconnected failure of network B
Non disconnected failure of generator A
Circuit interrupted in the network section
Circuit interrupted in the generator section
GA2
GA3
GO4
GO5
Network B failed
Circuit breaker F failed closed
Generator A faild
Circuit breaker L failed closed
Failed network B disconnected
Circuit breaker F failed open
Failed generator A disconnected
Circuit breaker L failed open
B_1
F_2
A_1
L_2
GA4
F_1
GA5
L_1
Network B failed
Circuit breaker F intact
Generator A failed
Circuit breaker L intact
B_1
F_3
A_1
L_3

Fig. 3-5. Fault Tree of the EPSS.



·Fig. 3-6. Fault tree of the EPPS (without variable descriptions)

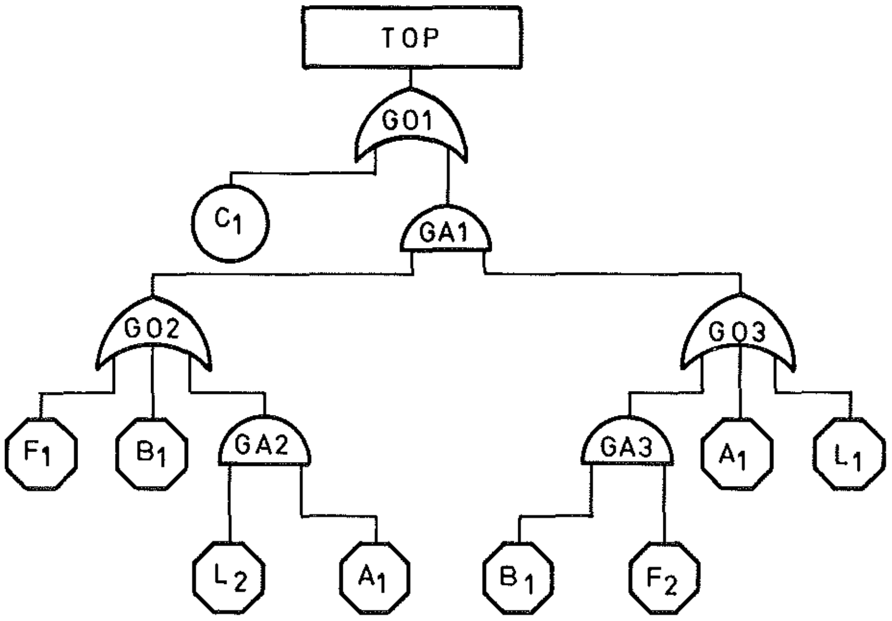


Fig. 3-7. Fault tree of the EPPS (Alternative)

We notice that the absence of voltage at the bus bars C can be caused either by a "non-disconnected failure" or by an "interruption of the continuity of the electric circuit". This dissection is whown graphically in Fig. 3-4.

The process of dissection can be carried further on until all variables are primary variables. The complete fault tree is shown in Fig. 3-5. Note that the variables A_1; B_1; L_1; L_2; L_3; F_1; F_2 and F_3 are all represented by octagons because they belong to dependent components.

The fault tree of Fig. 3-5 has been redrawn in simplified form in Fig. 3-6 without rectangles (i.e. variable descriptions).

Since there are in general, different possible ways of dissecting the variables, different fault trees of the same TOP can be drawn. The fault tree of Fig. 3-7 has exactly the same TOP variable of that of Fig. 3-6. In general different people generate different fault trees for the same TOP variable.

4. MODIFIED FAULT TREE. OCCURRENCE PROBABILITY OF THE PRIMARY EVENTS

Given a boolean variable A, we define as expectation of A ($E\left\{A\right\}$) the occurrence probability of the event $\left\{A=1\right\}$, that is

$$E\left\{A\right\} = P\left\{A=1\right\} \tag{4-1}$$

where $P\left\{\ldots\right\}$ means occurrence probability of the event under brackets.

The probability data related to the primary variables of the system described in the previous section are given in the table of Fig. 4-1. Here we assume that all failure rates of the primary components are constant. The transition rates are identified as follows. The primary variable of the row refers to the state before the transition (state of departure). The number of the column identifies the state after the transition (state of arrival).

The components A and B have transitions which are correlated. If B fails (transition $B_2 \rightarrow B_1$) there is a constant probability K_A that A fails too (transition $A_2 \rightarrow A_1$). In this case the transition $B_2 \rightarrow B_1$ is the conditioning transition and the transition $A_2 \rightarrow A_1$ is the conditioned transition. This is shown in the table of Fig. 4-1. The table shows also that if A fails (conditioning transition $A_2 \rightarrow A_1$) there is a constant probability K_B that B fails too (conditioned transition $B_2 \rightarrow B_1$).

Fig. 4-1. Table of the input probability data of the primary variables (System of Fig. 3-1)

Primary Component	Conditioning Variable	Primary Variable	Transition rates (hours^{-1})			Correlated transitions		
			1	2	3	Conditioning Transition	Conditioned Transition	Conditional Probability
A		A_1		μ_A				
		A_2	λ_A			$B_2 \rightarrow B_1$	$A_2 \rightarrow A_1$	K_A
B		B_1		μ_B				
		B_2	λ_B			$A_2 \rightarrow A_1$	$B_2 \rightarrow B_1$	K_B
C		C_1		μ_c				
		C_2	λ_c					
F	B_1	F_1			ρ_1			
		F_2			ω_1			
		F_3	ν_1	σ_1				
	B_2	F_1			ρ_2			
		F_2			ω_2			
		F_3	ν_2	σ_2				
L	A_1	L_1			η_1			
		L_2			α_1			
		L_3	ε_1	ζ_1				
	A_2	L_1			η_2			
		L_2			α_2			
		L_3	ε_2	ζ_2				

We introduce the following symbol

λ_A	=	transition rate of	$A_2 \rightarrow A_1$
μ_A	=	" " "	$A_1 \rightarrow A_2$
λ_B	=	" " "	$B_2 \rightarrow B_1$
μ_B	=	" " "	$B_1 \rightarrow B_2$

We can now draw the state diagram of the super-component G characterized by the four states which one obtains by intersecting the state of A and B in all possible ways. The state diagram of super-component G is shown in Fig. 4-2.

With reference to the state diagram of Fig. 4-2, we can now express the primary variables of components A and B as functions of the primary variables of G. We have

$$A_1 = G_1 V G_3 \tag{4-2}$$

$$A_2 = G_2 V G_4 \tag{4-3}$$

$$B_1 = G_1 V G_2 \tag{4-4}$$

$$B_2 = G_3 V G_4 \tag{4-5}$$

We now replace in fault tree of Fig. 3-6 the primary variables A_1 and B_1 with the new primary variables G_1; G_2; G_3 and G_4 by making use of Eqs. 4-2 and 4-4. The new fault tree is shown in Fig. 4-3.

In the fault tree of Fig. 4-3 the primary variables A_1 and B_1 have been replaced respectively by the OR Gates G07 (inputs G_1 and G_3) and G06 (inputs G_1 and G_2). Note that the primary variables G_1; G_2 and G_3 are represented by circles because they belong to an independent super-component. In fact their expectations can be calculated by solving the state diagram of Fig. 4-2. The new primary veriables have been introduced also in the fault tree of Fig. 3-7 (See Fig. 4-4).

The expectations of the primary variables G_1; G_2; G_3 and G_4 can easily be calculated by means of the very well known methods of state analysis.

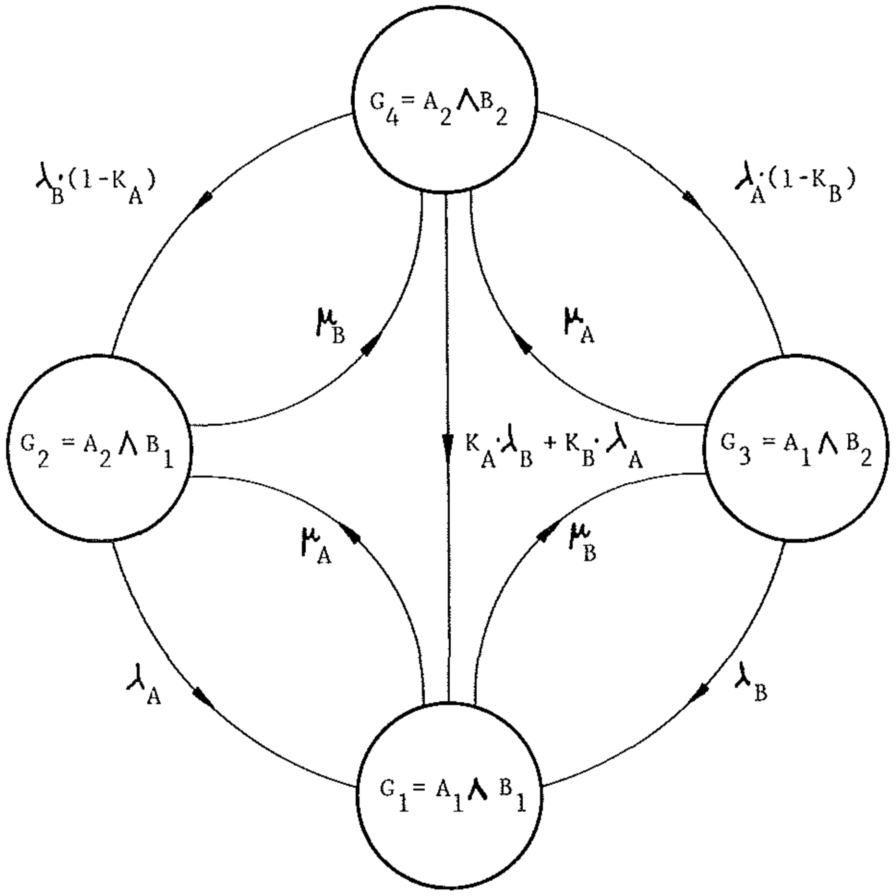


Fig. 4-2. Statee diagram of super-component G

We go back to the table of Fig. 4-1 and we consider the circuit breaker F. The circuit breaker F is a bipolar switch with conditioning variables B_1 and B_2. The theory of the bipolar switch has been fully developed by the author in[11]. Here only some important points of the model are recalled. Since a bipolar switch has three states, there will be three primary variables, namely (in the case of F)

F_1	associated	to	state	f_1	(failed open)
F_2	"	"	"	f_2	(failed closed)
F_3	"	"	"	f_3	(intact)

We shall assume that the two failed states of the switch don't communicate directly with each other. This means that the switch must be repaired before failing again. This is exactly what happens in practice. Failure and repair rates (i.e. transitions rates) of

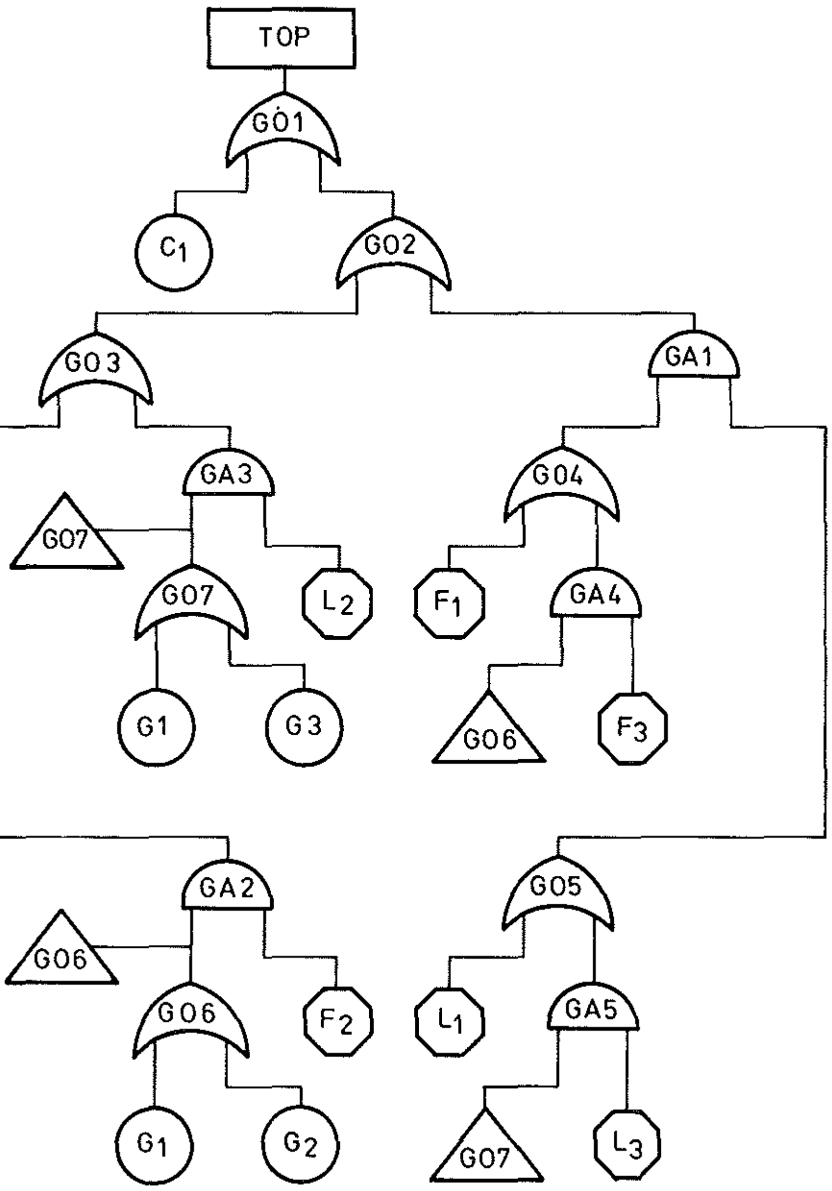


Fig. 4-3. Modified fault tree of the EPPS

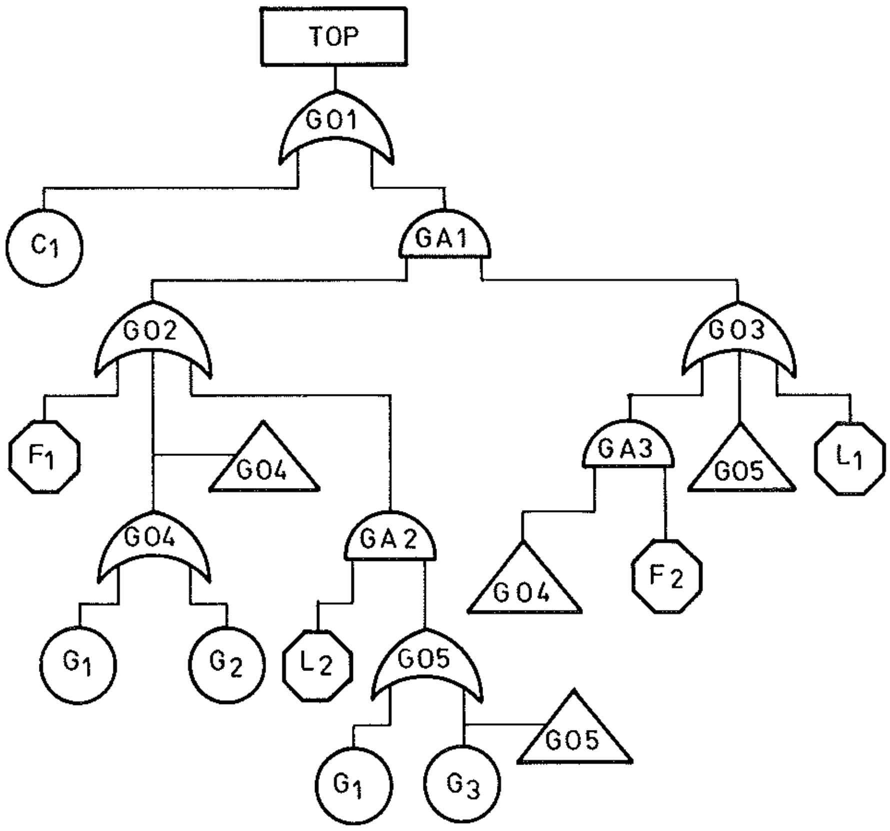


Fig. 4-4. Modified fault tree of the EPPS (Alternative)

the switch will be in general dependent upon its position i.e. upon the state of the network B (intact or failed). They are conditional transition rates.

In the case of a bipolar switch a procedure has been used[11] which is quite different from that used in the case of supercomponent G.

In this case in fact it is possible to identify a conditioning component B and a dependent component F. For this reason we define first the conditional expectation of a dependent variable (say F_k) given a conditioning variable (say B_q) as the occurrence probability of the event $\{F_k = 1\}$ given the event $\{B_q = 1\}$

$$E\left\{F_k \mid B_q\right\} = P\left\{F_k = 1 \mid B_q = 1\right\} \qquad (4\text{-}12)$$

One can easily calculate the conditional expectations of the primary variables F_1; F_2; and F_3 by means of the analysis developed in[11].

In[11] it has been demonstrated that the following relationships hold under some conditions which are always satisfied in the practical cases (for instance, repair rates must be order of magnitudes larger than failure rates). The relationships are

$$E\left\{F_k \mid B_q \wedge Y\right\} \cong E\left\{F_k \mid B_q\right\} \qquad (4\text{-}13)$$

(k=1; 2;3) (q=1; 2) where Y is an arbitrary boolean function which does not contain any literal of F.

$$E\left\{F_k \mid G_1\right\} \cong E\left\{F_k \mid G_2\right\} \cong E\left\{F_k \mid B_1\right\} \qquad (4\text{-}14)$$

$$E\left\{F_k \mid G_3\right\} \cong E\left\{F_k \mid G_4\right\} \cong E\left\{F_k \mid B_2\right\} \qquad (4\text{-}15)$$

(k=1; 2; 3)

In the case of circuit breaker L, one can also write expressions similar to (4-13) to (4-15). We have

$$E\left\{L_k \mid A_q \wedge Y\right\} \cong E\left\{L_k \mid A_q\right\} \quad (k=1;\ 2;3)\ (q=1;\ 2) \qquad (4\text{-}16)$$

where Y is an arbitrary boolean function which does not contain any literal of L.

$$E\left\{L_k \mid G_1\right\} \cong E\left\{L_k \mid G_3\right\} \cong E\left\{L_k \mid A_1\right\} \qquad (4\text{-}17)$$

$$E\left\{L_k \mid G_2\right\} \cong E\left\{L_k \mid G_4\right\} \cong E\left\{L_k \mid A_2\right\} \qquad (4\text{-}18)$$

5. BOOLEAN OPERATIONS

5.1 Generalities

The reader must become acquainted with some terms which are currently used throughout this paper.

In the following primary variables will be also called literals. A boolean function can be expressed in the form of a disjunction of conjunctions of literals (disjunctive form). A conjunction of literals belonging to a disjunctive form of a boolean function will be called shortly "monomial". A monomial X of the disjunctive form of a boolean function (TOP) is said to be an implicant of the TOP. It must satisfy the following boolean identy.

$$\text{TOP} \wedge X = X \qquad (5\text{-}1)$$

Let X_j and X_k be two monomials. We say that X_k subsumes X_j if every literal of X_j is contained in X_k. This is the same as saying that X_k is an implicant of X_j, that is

$$X_j \wedge X_k = X_k \qquad (5\text{-}1)$$

A disjunctive form of a boolean function will be called "normal disjunctive form" if its monomials satisfy the following four properties.

1. Each monomial (X) must be a non-zero monomial ($X \neq 0$, i.e. no pair of mutually exclusive literals must be contained in it)

2. Each monomial must not contain any literal (primary variable) more than once (no repeated literals).

3. Monomials must not subsume pairwise each other.
   ($X_j \neq X_j \wedge X_k \neq X_k$)

4. Monomials must not contain negated literals .

Each negated literal must have been previously replaced by the corresponding disjunction of all remaining literals belonging to the same primary component, that is

$$\bar{A}_i = \bigvee_{k=1}^{n} A_k \quad k \neq i \quad (i=1;\ 2;\ \ldots;n) \qquad (5\text{-}2)$$

A boolean function can have in general many normal disjunctive forms. For a given fault tree, there is a particular normal disjunctive form of its TOP variable which is associated to that fault tree. We shall call it "associated normal disjunctive form".

We say that a monomial X_j is a "prime implicant" (minimal cut set) of the boolean function TOP if (1) X_j implies the TOP ($X_j \wedge TOP = X_j$) and (2) any other monomial Y subsumed by X_j (i.e. obtained from X_j be deleting one of its literals) does not imply the TOP ($Y \wedge TOP \neq Y$).

We shall call any disjunction of prime implicants, which is equivalent to the function TOP , a "base of the function TOP". The disjunction of all prime implicants has this property. We shall call it the "complete base". We shall describe as an "irredundant base" a base which ceases to be a base if one of the prime implicants occuring in it is removed (deleted). Boolean functions may have many irredundant bases. We shall call "smallest irredundant base" the irredundant base having the smallest number of prime implicants. There may be more than one base with the smallest number of prime implicants.

<u>If a boolean function has only one base, which is at the same time complete and irredundant, the boolean function is said to be coherent.</u> The identification of an irredundant base (or one of the smallest irredundant bases) of the boolean function TOP of a fault tree is carried out in three steps:

<u>Step No. 1</u> Identification of the associated normal disjunctive form.

<u>Step No. 2</u> Identification of the complete base starting from the associated normal disjunctive form.

<u>Step No. 3</u> Extraction of an irredundant base (or one of the smallest irredundant bases) from the complete base.

After having identified an irredundant base of the TOP variable, some other transformations are carried out to get the boolean function in a form suitable for probability calculations. We have

<u>Step No. 4</u> Expression of the TOP as a disjunction of pairwise mutually exclusive boolean functions (keystone functions).

<u>Step No. 5</u> Identification of the conditioning variables to be associated to each keystone function.

The purpose of step No. 4 is that of getting an expression of the TOP which facilitates the operation of expectation. This will become clear in section 6 of this paper.

5.2 Step No. 1 - Identification of the Associated Normal Disjunctive Form

The Variables of the fault tree are first ordered in a list (table of variables). The literals are first listed. The acceptance criterion of a variable (gate) in the list is the following: the variable is accepted only and only if the input variables to the gate have already been accepted. If the gate satisfies the acceptance criterion is written in the list. The ordering process comes to an end when all variables have been written in the list.

By simple inspection of the fault tree of Fig. 3-6 we get the table of variables of Fig. 5-1.

The algorithm to identify the monomials of the associated normal disjunctive form is the so called "downward algorithm" which is based on the principle already described in[7] by Fussell and in[8]. Some additional features have been incorporated in the original downward algorithm so that the NOT gate and multistate components can be handled. The algorithm begins with the TOP and systematically goes down through the tree from the highest to the lowest variable, that if from the bottom to the top of the ordered list of variables. The fault tree is developed in a table (table of monomials). The elements of the table are variables. Each row of the table is a monomial. The numbers of the elements contained in a row is called length of the row. Each time an OR gate is encountered new rows will be produced (so many as the number of input variables to the gate). Each time an AND gate will be encountered the length of the rows (in which the gate appears) will be increased. Each time a NOT gate is encountered the input variable to the gate receives a negation mark. If a negated non primary variable will be dissected, the gate type will be replaced by its dual type (AND will be changed into OR and viceversa) and the negation mark is transmitted to all input variables of the gate. If a primary variable is negated, it is replaced by an OR gate which has as input variables all the remaining primary variables belonging to the same primary component.

The process of dissection comes to an end when all the elements of the table of monomials are primary variables (literals).

In addition the three following simplification rules are applied:

1. Delete zero monomials, that is rows which contain at least one pair of mutually exclusive literals.
   $Cj_q \wedge Cj_k = 0$ for $q \neq k$ (exclusion law).

2. Delete the repeated literals of a monomial (row).
   $Cj_q \wedge Cj_q = Cj_q$ (idempower law).

Ordering Numbers	Variable	Boolean Relationship	Predecessors	Successors
1	C_1	-	-	GO1
2	G_1	-	-	GO6;GO7
3	G_2	-	-	GO6
4	G_3	-	-	GO7
5	L_1	-	-	GO5
6	L_2	-	-	GA3
7	L_3	-	-	GA5
8	F_1	-	-	GO4
9	F_2	-	-	GA2
10	F_3	-	-	GA4
11	GO6	OR	G_1;G_2	GA2;GA4
12	GO7	OR	G1;G3	GA3;GA5
13	GA5	AND	GO7;L_3	GO5
14	GA4	AND	GO6;F_3	GO4
15	GA3	AND	L_2;GO7	GO3
16	GA2	AND	GO6;F_2	GO3
17	GO4	OR	F_1;GA4	GA1
18	GO5	OR	L_1;GA5	GA1
19	GA1	AND	GO4;GO5	GO2
20	GO3	OR	GA2;GA3	GO2
21	GO2	OR	G O3;GA1	GO1
22	GO1(TOP)	OR	C_1;GO2	-

Fig. 5-1. Table of variables of the fault tree of Fig. 4-3.

3. Delete any subsuming monomial, that is any row which contains all elements or another row.
$X_a \vee X_b = X_a$ if $X_a \wedge X_b = X_b$ (absorption law).

At the end of the process each row of the table of monomials is a monomial and the disjunction of all monomials is the normal disjunctive form of the TOP associated to the fault tree under considerations.

We now apply the above described procedure to the table of variables of Fig. 5-1. The example is self explanatory. We have

Ordering Number	Boolean Identity	Table of Monomials	
	TOP = GO1	GO1	
22	GO1 = C_1VGO2	C_1	
		GO2	
21	GO2 = GO3 VGA1	C_1	
		GO3	
		GA1	
20	GO3 = GA2 VGA3	C_1	
		GA2	
		GA3	
		GA1	
19	GA1 = GO4 $\wedge$ GO5	C_1	
		GA2	
		GA3	
		GO4	GO5
18	GO5 = L_1VGA5	C_1	
		GA2	
		GA3	
		GO4	L_1
		GO4	GA5

and so on.

At the end of the process the table of monomials will look as follows (Fig. 5-2).

We can therefore write the following boolean identity for the TOP (we indicate from now on the conjunction by means of the simpler multiplication symbol ".").

$$TOP = C_1 \vee F_2 \cdot G_1 \vee F_2 \cdot G_2 \vee L_2 \cdot G_1 \vee L_2 \cdot G_3 \vee G_1 \cdot F_3 \cdot L_1 \vee G_2 \cdot F_3 \cdot L_1 \vee$$
$$\vee F_1 \cdot G_1 \cdot L_3 \vee F_1 \cdot G_3 \cdot L_3 \vee F_1 \cdot L_1 \vee G_1 \cdot F_3 \cdot L_3 \qquad (5\text{-}3)$$

If we now apply the same above procedure to the fault tree of Fig. 4-4, we get

$$TOP = C_1 \vee L_1 \cdot F_1 \vee F_1 \cdot G_3 \vee G_3 \cdot L_2 \vee L_1 \cdot G_2 \vee G_1 \vee F_2 \cdot G_2 \qquad (5\text{-}4)$$

C_1		
F_2	G_1	
F_2	G_2	
L_2	G_1	
L_2	G_3	
G_1	F_3	L_1
G_2	F_3	L_1
F_1	G_1	L_3
F_1	G_3	L_3
F_1	L_1	
G_1	F_3	L_3

Fig. 5-2. Table of monomials of the fault tree of Fig. 3-6.

The two expressions 5-3 and 5-4 look very different. However they are the same boolean function. This will be shown in the next section. Here we can say that it is not possible to prove whether or not two boolean functions are equal by making use only of algorithms which calculate normal disjunctive forms of boolean functions.

5.3 Step No. 2 - Identification of the complete base

Various algorithms for the identification of the complete base of a boolean function (step No. 2) are available from the literature[9]. An algorithm due to Nelson[10] is particularly convenient. This algorithm consists simply in complementing (negating) a normal disjunctive form of a boolean function TOP (which from now on we also call Φ) and then in complementing its complement $\bar{\Phi}$. After each of the two complement operations, the three simplification rules (section 5.2) are applied to the result.

Nelson's algorithm can be described as follows

1. Complement ϕ, expand $\overline{\phi}$ into disjunctive form, apply simplification rules and call the result $\overline{F}$.

2. Complement $\overline{F}$, expand F into disjunctive form, apply simplification rules and call the result K.

The disjunction of the monomials of K is the complete base of the boolean function ϕ.

We now apply the Nelson algorithm to our case, that is to Eq. 5-3. By complementing Eq. 5-3, we can write

$$\begin{aligned}\overline{TOP} = {} & \overline{C}\cdot(\overline{F}_2 \vee \overline{G}_1)\cdot(\overline{F}_2 \vee \overline{G}_2)\cdot(\overline{L}_2 \vee \overline{G}_1)\cdot(\overline{L}_2 \vee \overline{G}_3)\cdot \\ & \cdot(\overline{G}_1 \vee \overline{F}_3 \vee \overline{L}_1)\cdot(\overline{G}_2 \vee \overline{F}_3 \vee \overline{L}_1)\cdot(\overline{F}_1 \vee \overline{G}_1 \vee \overline{L}_3)\cdot \\ & \cdot(\overline{F}_1 \vee \overline{G}_3 \vee \overline{L}_3)\cdot(\overline{G}_1 \vee \overline{L}_1)\cdot(\overline{G}_1 \vee \overline{F}_3 \vee \overline{L}_3) \qquad (5\text{-}5)\end{aligned}$$

Now we have

$$\overline{C}_1 = C_2 \qquad (5\text{-}6)$$

$$\overline{G}_k = \bigvee_{q=1}^{4} Gq \qquad k \neq q \qquad (k=1;\ 2;\ 3;\ 4) \qquad (5\text{-}7)$$

$$\overline{F}_k = \bigvee_{q=1}^{3} F_q \qquad k \neq q \qquad (k=1;\ 2;\ 3) \qquad (5\text{-}8)$$

and

$$\overline{L}_k = \bigvee_{q=1}^{3} Lq \qquad k \neq q \qquad (k=1;\ 2;\ 3) \qquad (5\text{-}9)$$

By taking into account Eqs. 5-6 to 5-9, Eq. 5-5 becomes

$$\begin{aligned}\overline{TOP} = {} & C_2\cdot(F_1 \vee F_3 \vee G_2 \vee G_3 \vee G_4)\cdot(F_1 \vee F_3 \vee G_1 \vee G_3 \vee G_4)\cdot \\ & \cdot(L_1 \vee L_3 \vee G_2 \vee G_3 \vee G_4)\cdot(L_1 \vee L_3 \vee G_1 \vee G_2 \vee G_4)\cdot \\ & \cdot(G_2 \vee G_3 \vee G_4 \vee F_1 \vee F_2 \vee L_2 \vee L_3)\cdot(G_1 \vee G_3 \vee G_4 \vee F_1 \vee F_2 \vee L_2 \vee L_3)\cdot \\ & \cdot(F_2 \vee F_3 \vee G_2 \vee G_3 \vee G_4 \vee L_1 \vee L_2)\cdot(F_2 \vee F_3 \vee G_1 \vee G_2 \vee G_4 \vee L_1 \vee L_2)\cdot \\ & \cdot(F_2 \vee F_3 \vee L_2 \vee L_3)\cdot(G_2 \vee G_3 \vee G_4 \vee F_1 \vee F_2 \vee L_1 \vee L_2) \qquad (5\text{-}10)\end{aligned}$$

We execute the operations of Eq. 5-10 and we apply the three simplification rules. We get

$$\begin{aligned}\overline{TOP} = {} & C_2 \cdot G_2 \cdot F_1 \cdot L_2 \vee C_2 \cdot G_2 \cdot F_1 \cdot L_3 \vee C_2 \cdot G_2 \cdot F_3 \cdot L_2 \vee C_2 \cdot G_2 \cdot F_3 \cdot L_3 \vee \\ & \vee C_2 \cdot G_3 \cdot F_2 \cdot L_1 \vee C_2 \cdot G_3 \cdot F_2 \cdot L_3 \vee C_2 \cdot G_3 \cdot F_3 \cdot L_1 \vee C_2 \cdot G_3 \cdot F_2 \cdot L_3 \vee \\ & \vee C_2 \cdot G_4 \cdot F_2 \vee C_2 \cdot G_4 \cdot F_3 \vee C_2 \cdot G_4 \cdot L_2 \vee C_2 \cdot G_4 \cdot L_3 \qquad (5\text{-}11)\end{aligned}$$

We now complement $\overline{TOP}$ and we execute all operations including the application of the three simplification rules. We get finally

$$TOP = C_1 \vee L_1 \cdot F_1 \vee F_1 \cdot G_3 \vee G_3 \cdot L_2 \vee L_1 \cdot G_2 \vee G_1 \vee F_2 \cdot G_2 \qquad (5\text{-}12)$$

Eq. 5-12 is the complete base of the TOP.

We notice that Eq. 5-12 and 5-4 (that is the fault trees of Figs. 3-6 adn 3-7) have the same TOP. The knowledge of the complete base of a boolean function is important also because offers the possibility to find out if two or more fault trees have the same TOP.

We can state the following criterion

"<u>If two boolean functions have the same complete base they are identical</u>".

Nelson's algorithm was improved by Hulme and Worrell[11] to reduce the computing time. A modified Nelson's algorithm has been developed at Karlsruhe[3]. The execution times of the three algorithms are compared in the table of Fig. 5-3. The examples have been taken from[11].

5.4 Step No. 3 - Extraction of an Irredundant Base (or One of the Smallest Irredundant Bases) from the Complete Base.

Various algorithms for the extraction of the smallest irredundant base of a boolean function from its complete base are available from the literature[9].

We consider a method, which may be called the method of the expansion coefficients. The basic principles of this method have been described in[3].

A fast algorithm based on this principle has been developed at Karlsruhe[3] which allows one to identify the smallest irredundant base of a boolean function. The table of Fig. 5-4 gives the required execution times for the examples 3 to 7 of the table 5-3.

Example	Number of prime implicants in complete base	CPU time (sec)		
		Nelson algorithm (CDC6600)	Sandia algorithm (CDC 6600)	Karlsruhe algorithm (IBM370/168)
1	4	0.158	0.156	0.11
2	3	0.367	0.182	not performed
3	15	221.418	0.391	0.26
4	15	1413.580	0.388	0.26
5	32	5300(1)	3.868	0.42
6	61	4600(1)	303.657	1.03
7	87	6000(1)	417.371	1.12

(1) These entries indicate times at which execution was terminated without completing the algorithm.

Fig. 5-3. Computational times of different types of Nelson Algorithms.

Example	Number of prime implicants in complete base	Number of prime implicants in smallest irredundant base	CPU time needed to identify smallest irredundant base (secs)
3	15	7	0.24
4	15	8	0.23
5	32	12	0.49
6	61	17	6.07
7	87	19	19.51

Fig. 5-4. Computational times of the algorithm for the extraction of the smallest irredundant base.

An even faster algorithm for the extraction of an irredundant base (which is not necessarily the smallest) has been developed at Karlsruhe. This algorithm will be described elsewhere.

Since the boolean function of our example is coherent, the complete base is already irredundant and the algorithm for the extraction of an irredundant base does not need to be applied.

5.5 Step No. 4 - Expression of the TOP as a Disjunction of Pairwise Mutually Exclusive Boolean Functions

We have the TOP as disjunction of the prime implicants "X_j" (irredundant base).

$$TOP = \bigvee_{j=1}^{N} X_j \qquad (5\text{-}13)$$

where

N = total number of prime implicants belonging to the irredundant base.

We now want to transform Eq. 5-13 in an expression of the type

$$TOP = \bigvee_{i=1}^{Q} Y_i \qquad (5\text{-}14)$$

where Y_i are boolean functions (called keystone functions) which are pairwise mutually exclusive, that is satisfy the conditions

$$Y_i \cdot Y_k = 0 \quad i \neq k \quad (i;\ k = 1;\ 2 \ldots ;Q) \qquad (5\text{-}15)$$

In addition each Y_i results to be of the form

$$Y_i = M_i \cdot \bigvee_{s=1}^{n_i} P_{is} \quad (i = 1;\ 2 \ldots ;Q) \qquad (5\text{-}16)$$

where the M_i and the P_{is} are non-zero boolean monomials satisfying the following conditions

$$M_i \cdot M_k = 0 \quad i \neq k \quad (i;\ k = 1;\ 2 \ldots ;Q) \qquad (5\text{-}17)$$

$$\bigvee_{i=1}^{Q} M_i = 1 \qquad (5\text{-}18)$$

- the monomials P_{is} are pairwise logically independent, that is if a literal A_q Appears in a monomial P_{is}, no other literal belonging to the same component will appear in any other monomial P_{ir} ($r \neq s \quad r;s = 1;2 \ldots ;n_i$).

- eych monomial P_{is} is logically independent with M_i.

The last two conditions can be expressed in the following way

If $\quad A_q \cdot P_{is} = P_{is}$

then $\quad 0 \neq A_q \cdot M_i \neq M_i$ and

$\quad 0 \neq A_q \cdot P_{ir} \neq P_{ir} \qquad r \neq s$

AND

If $\quad A_q \cdot M_i = M_i$

then $\quad 0 \neq A_q \cdot P_{is} \neq P_{is}$

In other words a component A can appear only once in a keystone function Y_i: either in the monomial M_i or in one of the monomials P_{is}.

A fast algorithm has been developed at Karlsruhe to identify the keystone functions Y_i (keystone algorithm). By applying the keystone algorithm to our example (Eq. 5-12) we get

$$TOP = \bigvee_{i=1}^{4} Y_i \tag{5-19}$$

where

$$Y_1 = G_1 \tag{5-20}$$

$$Y_2 = G_2 \cdot (C_q \vee L_1 \vee F_2) \tag{5-21}$$

$$Y_3 = G_3 \cdot (C_1 \vee F_1 \vee L_2) \tag{5-22}$$

$$Y_4 = G_4 \cdot (C_1 \vee L_1 \cdot F_1) \tag{5-23}$$

5.6 Step No. 5 - Identification of the Conditioning Variables to be associated to each Keystone Function.

Each keystone function receives marks for the identification of the conditioning variables associated to the statistically dependent variables which appear in it.

A fast algorithm (called "marking algorithm") has been developed at Karlsruhe for the identification of these conditioning variables. By applying this algorithm to our example we get the following table (Fig. 5-5)

Keystone Function	Statistically Dependent Primary Variable	Conditioning Variable
Y_1	-	-
Y_2	L_1	$A_2 = G_2 \vee G_4$
	F_2	$B_1 = G_1 \vee G_2$
Y_3	F_1	$B_2 = G_3 \vee G_4$
	L_2	$A_1 = G_1 \vee G_3$
Y_4	L_1	$A_2 = G_2 \vee G_4$
	F_1	$B_2 = G_3 \vee G_4$

Fig. 5-5. Table of the conditioning variables to be associated to each keystone function

6. CALCULATION OF THE OCCURRENCE PROBABILITY OF THE TOP EVENT

We now want to calculate the expectation of the TOP variable, that is the occurrence probability of the event $\{TOP = 1\}$.

$$E \left\{ TOP \right\} = P \left\{ TOP = 1 \right\} \tag{6-1}$$

Taking into account Eqs. 5-13 and 5-15 we can write

$$E \left\{ TOP \right\} = \sum_{i=1}^{Q} E \left\{ Y_i \right\} \tag{6-2}$$

In our example (Eqs. 5-20 to 5-23 and table of Fig. 5-5) we can write

$$E \left\{ Y_1 \right\} = E \left\{ G_1 \right\} \tag{6-3}$$

$$\begin{aligned} E\left\{ Y_2 \right\} &= E\left\{ G_2\cdot(C_1 \;\; L_1 \;\; F_2) \right\} = \\ &= E\left\{ G_2\cdot C_1 \right\} + E\left\{ G_2\cdot L_1 \right\} + E\left\{ G_2\cdot F_2 \right\} - \\ &\quad - E\left\{ G_2\cdot C_1\cdot L_1 \right\} + E\left\{ G_2\cdot C_1\cdot F_2 \right\} + E\left\{ G_2\cdot L_1\cdot F_2 \right\} + \\ &\quad + E\left\{ G_2\cdot C_1\cdot L_1\cdot F_2 \right\} \end{aligned} \qquad (6\text{-}4)$$

Taking into account Eqs. 4-13; 4-14; 4-16 and 4-18 and the table of Fig. 5-5, we can write

$$E\left\{ G_2\cdot C_1 \right\} = E\left\{ G_2 \right\} \cdot E\left\{ C_1 \right\} \qquad (6\text{-}5)$$

$$E\left\{ G_2\cdot L_1 \right\} = E\left\{ G_2 \right\}\cdot E\left\{ L_1 | G_2 \right\} = E\left\{ G_2 \right\}\cdot E\left\{ L_1 | A_2 \right\} \qquad (6\text{-}6)$$

$$E\left\{ G_2\cdot F_2 \right\} = E\left\{ G_2 \right\}\cdot E\left\{ F_2 | G_2 \right\} = E\left\{ G_2 \right\}\cdot E\left\{ F_2 | B_1 \right\} \qquad (6\text{-}7)$$

$$\begin{aligned} E\left\{ G_2\cdot C_1\cdot L_1 \right\} &= E\left\{ C_1 \right\}\cdot E\left\{ G_2\cdot L_1 \right\} = E\left\{ C_1 \right\}\cdot E\left\{ G_2 \right\}\cdot E\left\{ L_1 | G_2 \right\} = \\ &= E\left\{ C_1 \right\}\cdot E\left\{ G_2 \right\}\cdot E\left\{ L_1 | A_2 \right\} \end{aligned} \qquad (6\text{-}8)$$

$$E\left\{ G_2\cdot C_1\cdot F_1 \right\} = E\left\{ C_1 \right\}\cdot E\left\{ G_2 \right\} \; E\left\{ F_1 | B_1 \right\} \qquad (6\text{-}9)$$

$$\begin{aligned} E\left\{ G_2\cdot L_1\cdot F_2 \right\} &= E\left\{ G_2\cdot L_1 \right\}\cdot E\left\{ F_2 / G_2 L_1 \right\} = \\ &= E\left\{ G_2 \right\}\cdot E\left\{ L_1 | G_2 \right\}\cdot E\left\{ F_2 | G_2 \right\} = \\ &= E\left\{ G_2 \right\}\cdot E\left\{ L_1 | A_2 \right\}\cdot E\left\{ F_2 | B_1 \right\} \end{aligned} \qquad (6\text{-}10)$$

$$E\left\{ G_2\cdot C_1\cdot L_1\cdot F_2 \right\} = E\left\{ G_2 \right\}\cdot E\left\{ C_1 \right\}\cdot E\left\{ L_1 | A_2 \right\}\cdot E\left\{ F_2 | B_1 \right\} \qquad (6\text{-}11)$$

Taking into account Eqs. 6-5 to 6-11, Eq. 6-4 becomes finally

$$\begin{aligned} E\left\{ Y_2 \right\} &= E\left\{ G_2 \right\}\cdot\Big[E\left\{ C_1 \right\} + (1-E\left\{ C_1 \right\})\cdot E\left\{ L_1 / A_2 \right\} + \\ &\quad + (1-E\left\{ C_1 \right\})\cdot(1-E\left\{ L_1 | A_2 \right\})\cdot E\left\{ F_2 | B_1 \right\} \Big] \end{aligned} \qquad (6\text{-}12)$$

In a similar way one gets

$$\begin{aligned} E\left\{ Y_3 \right\} &= E\left\{ G_3 \right\}\cdot\Big[E\left\{ C_1 \right\} + (1-E\left\{ C_1 \right\})\cdot E\left\{ F_1 | B_2 \right\} + \\ &\quad + (1-E\left\{ C_1 \right\})\cdot(1-E\left\{ F_1 | B_2 \right\})\cdot E\left\{ L_2 | A_1 \right\} \Big] \end{aligned} \qquad (6\text{-}13)$$

and

$$E\left\{ Y_4 \right\} = E\left\{ G_4 \right\}\cdot\Big[E\left\{ C_1 \right\} + (1-E\left\{ C_1 \right\})\cdot E\left\{ L_1 / A_2 \right\}\cdot E\left\{ F_1 / B_2 \right\} \Big] \qquad (6\text{-}14)$$

By replacing Eqs. 6-3; 6-12; 6-13 and 6-14 into Eq. 6-2 (with Q=4) one finally gets the occurrence probability of the TOP event.

7. CONCLUSIONS

Following conclusions can be drawn:

1. The theory described in this paper is a powerful tool for the analysis of fault trees containing multistate (more than two states) primary components as well as statistically dependent primary components. This means that a very wide spectrum of problems which are met in practice can now be solved analytically by applying this theory.

2. A special type of boolean algebra has been developet to allow one to handle multistate primary components. This is the boolean algebra with restrictions on variables. Its basic rules have been described in this paper.

3. The problem of statistical dependence has been solved either (1) by removing it, that is by replacing in the fault tree the statistically dependent primary variables by means of ad hoc new defined primary variables or (2) by defining some conditioning variables and evaluating separately the associated necessary conditional probabilities.

   General criteria to establish which one of the two methods should be chosen have not been given in the paper. They are illustrated in Reference[11].

4. General criteria for the identification of the most convenient conditioning variables are given also in Reference[11]. The choice depends upon the type of statistical dependence and upon the way in which this statistical dependence enters in the fault tree.

5. The concept of expectation of a boolean variable and of conditional expectation of a boolean variable have been introduced in this paper in a rather intuitive way just pointing out the close relationship between conditional expectation and conditional probability. In Reference[11] a formalization of the concept is developed.

6. A computer programme based on the above theory has been developed at Karlsruhe and is now being tested. Two sample problems have been solved by using this programme.

   A system was given to three different people. Three different fault trees were generated for the same TOP variable.

The three associated disjunctive forms (output from the downward algorithm) were calculated and they looked each other remarkably different (large differences in the total number of monomials as well as in their composition). However, it was possible to verify that the three functions were identical by calculating the complete base (output of the Nelson algorithm), which resulted to be exactly the same for all three fault trees.

The second problem was chosen because it contained three different types of dependecies which are commonly met in practice, namely (1) common mode failure, (2) components characterized by failure rates which depend upon the occurrence of some non-primary events and (3) the case of a component whose repair affects the operation of another component. The computer programme solved the problem successfully.

7. A new definition of coherency has been given in this paper. We recall it again

> "A boolean function is said to be coherent if it is characterized by only one base which is at the same time complete and irredundant."

The question arises whether or not all technical systems are coherent. Some authors are convinced that there are examples of systems which they believe to be non-coherent. We have not yet deeply analysed this question also because we had no chance until now to study a non-coherent technical system. We can however not exclude at this stage that some non-coherent systems may exist.

Non-coherent boolean functions may instead be generated, when one analyses the problem of a transition from one partition of a system to another. These boolean functions are rather special because they describe the space at the boundary between the two partitions. The computer programme developed at Karlsruhe can handle coherent as well as non-coherent boolean functions.

8. ACKNOWLEDGEMENTS

The author wishes to thank Dr. Wenzelburger (IRE, Karlsruhe) for the fruitful discussions on the theory developed in this paper.

9. REFERENCES

1. W.E. Vesely, 1970, "A time dependent methodology for fault tree evaluation", Nucl. Eng. Des. 13, 337-360.

2. L. Caldarola, A. Wickenhäuser, 1977,"Recent Advancements in fault tree methodology at Karlsruhe", International Conf. on Nucl. Systems Reliability Engineering and Risk Assessment, Gatlinburg, SIAM, 518-542.

3. L. Caldarola, 1978, "Fault tree analysis of multistate systems with multistate components", ANS Topical Meeting on Probabilistic Analysis of Nuclear Reactor Safety, Los Angeles, California, Paper VIII.1.

4. J.D. Murchland, G. Weber, 1972, "A moment method for the calculation of a confidence interval for the failure probability of a system", IEEE Proceedings Annual Symposium on Reliability.

5. C. Berge, 1962, "The theory of graphs", Methuen and John Wiley

6. P. Mussio, S. Garriba, S. Fumagalli,1979, "Multiple valued logic in system representation", NATO ASI, Rel.Conf., Urbino, Italy

7. J.B. Fussell, 1973,"Fault tree analysis: Concept and techniques", NATO Conference on Reliability, Liverpool, England.

8. L. Caldarola, A. Wickenhäuser, 1977, "The Karlsruhe computer program for the evaluation of the availability and reliability of complex repairable systems", Nucl. Eng. Des. 43, 463-470.

9. J. Kuntzmann, 1967,"Fundamental Boolean Algebra," Blackie and Sons Ltd.

10. R.J. Nelson, 1954,"Simplest normal truth functions," the Journal of Symbolic Logic, vol. 20, Nr. 2, 105-108.

11. B.L. Hulme, R.B. Worrell, 1975,"A prime implicant algorithm with factoring," IEEE Transaction on computers, vol. C-24, Nr. 11, 1129-1131.

12. L. Caldarola, 1979, "Generalized fault tree analysis combined with state analysis", (being published).