

To Not or Not to Not!!

Dr J. D. Andrews; Department of Mathematical Sciences, Loughborough University; Loughborough, Leicestershire, UK

Keywords: Fault Tree Analysis, Non-coherent, Binary Decision Diagram

Abstract

Fault Tree Analysis is a technique commonly used in safety system assessment. Causes of the system failure mode, termed the Top Event, are expressed in an inverted tree structure using logic gates to connect events which increase in detail as they progress down the branches of the tree structure. The branches terminate in basic events which represent component failures. The gates are usually either explicitly of the AND or OR type or can be expressed in terms of these two logic operators (such as the VOTE gate). The use of the NOT gate is generally discouraged by exponents of the method. Its inclusion indicates that components in their working state can contribute to the failure of the system. This is counter intuitive to the way we perceive the functionality of systems. From an analysis point of view it also creates difficulties. The logic expressions become much larger and at times unmanageable, conventional methods of deducing the combinations of events (success and failure) which result in system failure do not work, approximations used to calculate the top event probability are no longer appropriate.

This paper examines the potential uses of NOT logic within the fault tree method. It indicates situations where the inclusion of this logic operator within the tree structure can add to the analysis. It is shown that the Binary Decision Diagram (BDD) method overcomes some of the difficulties encountered during system quantification when conventional Kinetic Tree Theory based approaches are employed.

Introduction

The majority of works on fault tree construction recommend that failure logic represented in the diagram is restricted to the use of AND and OR (refs. 1-3). This makes the structure coherent. Non-coherent fault tree structures can be produced if the third logic operator NOT is used directly or implied (exclusive OR, XOR). When the NOT gate is used in a fault tree it means that components NOT failing i.e. working, contribute to the system failing.

The grounds for using AND and OR gates alone and no NOT logic is firstly philosophical in that it can be considered a bad design of system if a failed component being repaired (i.e. being restored to the working condition) makes the system state deteriorate. Secondly, that the inclusion of working states in the system failure diagram yields little information and thirdly it results in a considerable increase in complexity for the quantitative analysis.

This paper investigates the objections raised for using NOT gates, the difficulties in the analysis of failure logic diagrams when it is used and if there are circumstances when it can help in the understanding and analysis of systems.

Non-coherent Structure Function

A fault tree is non-coherent if its structure function $\phi(\underline{x})$ does not conform to the requirements of coherency which are:

- i) each component i must be relevant so

$$\phi(1_i, \underline{x}) \neq \phi(0_i, \underline{x})$$

for some \underline{x} and

- ii) $\phi(\underline{x})$ is increasing (non-decreasing) in each x_i where:

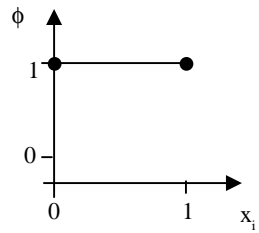
$$\phi(\underline{x}) = \phi(x_1, \dots, x_n)$$

$$\phi(1_i, \underline{x}) = \phi(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

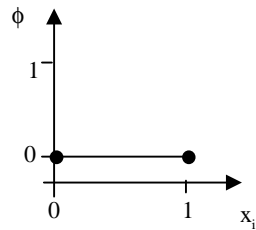
$$\phi(0_i, \underline{x}) = \phi(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

and x_i is the indicator variable for component i .

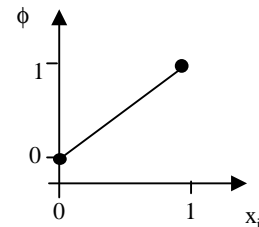
The second of these conditions, an increasing (non-decreasing) structure function, means that as the component deteriorates (fails, so $x_i = 1$) the system condition either does not change or also deteriorates. This gives three possibilities (see figure 1).



(a)



(b)



(c)

Figure 1 - Non-decreasing Structure Function

In figure 1(a) the condition of other components in the system have resulted in its failure ($\phi = 1$) when component i is working ($x_i = 0$). The failure of component i ($x_i = 1$) makes no difference to the system state. Figure 1(b) represents the situation where the system is not in a critical state for component i . When i fails the system continues to function ($\phi = 0$). Figure 1(c) illustrates the situation where the system is critical for component i and when i fails it causes system failure.

Figure 2 illustrates a structure function which is non-coherent for component i . The system is failed when the component i works and then when i fails the system is restored to the functioning condition.

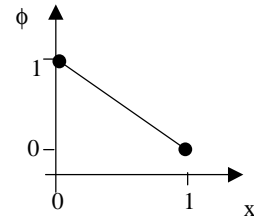


Figure 2 - Non-coherent Structure Function

Analysis of Non-coherent Fault Trees

As with coherent fault trees, the analysis of a non-coherent fault tree structure is performed in two stages: qualitative and quantitative:

Qualitative Analysis: The objective of the qualitative analysis is to determine the combinations of component conditions (working or failed) which are necessary and sufficient to cause system failure. These are termed PRIME IMPLICANTS.

As will be described in the next section a qualitative analysis to produce Prime Implicants requires more work than for the minimal cut sets of a coherent structure. The order of the prime implicants (number of components, working or failed) tend to be larger than minimal cut sets due to the inclusion of the working states.

Quantitative Analysis: The probability of components working is usually numerically close to 1, component failure has a probability which is relatively small, they are often rare events. When component failure is small the inclusion - exclusion expansion for the top event probability in terms of minimal cut set (C_i) probabilities can be accurately truncated after a small number of terms

$$P(\text{TOP}) = \sum_{i=1} P(C_i) - \sum_{\text{all combinations}} \sum_{i,j} P(C_i \wedge C_j) +$$

all combinations

i, j (1)

$$\sum_{\text{all combinations}} \sum_{i,j,k} P(C_i \wedge C_j \wedge C_k) \dots$$

all combinations

i, j, k

When equation (1) is expressed in terms of prime implicants the convergence is slow and many more terms need to be evaluated to obtain an accurate result.

Performing both qualitative and quantitative analysis of a non-coherent fault tree requires increased memory resources and processing time when compared to a coherent alternative.

Determining Prime Implicants

Consider as an example the traffic light system shown in figure 3 (ref.4).

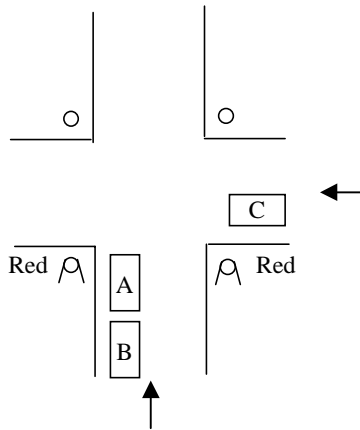


Figure 3 - Traffic Light System

Cars A and B are approaching the lights on red and should stop. Car C is approaching the junction and has the right of way and should proceed. With basic (failure) events:

- A - Car A fails to stop
- B - Car B fails to stop
- C - Car C fails to continue

The fault tree to represent causes of an accident situation at this crossroads is given in figure 4. Either Car A can stop and Car B drives into the back of it or A fails to stop and hits Car C.

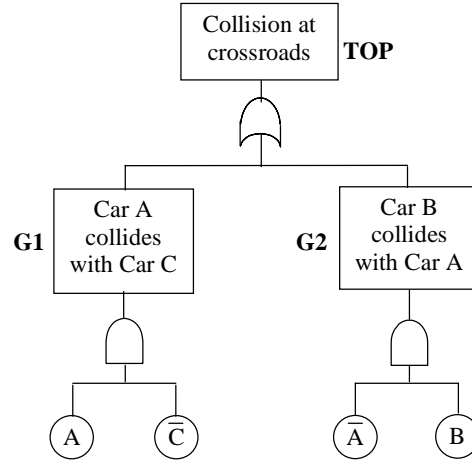


Figure 4 - Fault Tree for Traffic Light System

Boolean reduction on this fault tree produces the logic equation for the top event:

$$TOP = A\bar{C} + \bar{A}B \quad (2)$$

where '+' ≡ OR, '.' ≡ AND.

Therefore {A \bar{C} } and { $\bar{A}B$ } are prime implicants. However, unlike the reduction of the logic expression for a coherent fault tree it does not produce a complete list of all prime implicants, { $\bar{B}\bar{C}$ } is missing i.e. if B fails to stop and C continues across the lights there will be a collision whatever A does. Therefore the full logic expression for the TOP event is:

$$TOP = A\bar{C} + \bar{A}B + \bar{B}\bar{C} \quad (3)$$

which can be obtained by applying the consensus law:

$$AX + \bar{A}Y = AX + \bar{A}Y + XY \quad (4)$$

By including the success states it can be argued that prime implicants $\bar{B}\bar{A}$ and $\bar{B}\bar{C}$ are giving little extra information than a single minimal cut set \bar{B} . The increase in length of the logic equation can cause difficulties for large fault trees.

Quantification of this fault tree can be obtained from:

$$P(TOP) = P(P_1) + P(P_2) + P(P_3) - P(P_1P_2) - P(P_2P_3) - P(P_3P_1) + P(P_1P_2P_3)$$

where P_i are the prime implicants $P_1 = \{\overline{AC}\}$,
 $P_2 = \{\overline{AB}\}$, $P_3 = \{\overline{BC}\}$

$$\begin{aligned} P(\text{TOP}) &= P(\overline{AC}) + P(\overline{AB}) + P(\overline{BC}) \\ &\quad - P(\overline{ABC}) - P(AB\overline{C}) \\ &= q_A(1 - q_C) + (1 - q_A)q_B + q_B(1 - q_C) \\ &\quad - (1 - q_A)(1 - q_C)q_B - q_A q_B(1 - q_C) \\ &= q_A(1 - q_C) + (1 - q_A)q_B \\ &= q_A + q_B - q_A q_C - q_A q_B \end{aligned}$$

For larger, realistic, fault trees, the fact that this expression cannot be truncated can make the quantification intractable. An alternative solution process is to use a coherent approximation. This involves reducing the prime implicants to minimal cut sets by deleting all negated literals from each prime implicant (on the assumption that $P(\overline{X}) \approx 1.0$) and then re-minimising the results. If this process is applied to the traffic light system the coherent approximation is:

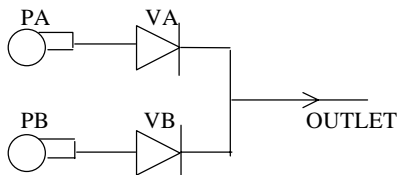
$$\text{TOP} \approx A + B$$

$$\text{and } P(\text{TOP}) = q_A + q_B - q_A q_B$$

[an overestimate to the exact probability by $q_A q_C$]

NOT Logic in Coherent Fault Trees

The inclusion of NOT gates in the development of a top event does not guarantee that the resulting structure will be non-coherent. For example (ref. 4) consider the pumping system shown in figure 5. There are two parallel pumping streams consisting of a pump (P) and non-return valve (V) supplying fluid to the outlet. Sufficient flow can be maintained by the complete capacity of one stream alone.



Failure Modes

- PA/PB - Pump A/B fails
- VA/VB - Non-return valve A/B fails

Figure 5 - Pumping System

The fault tree for top event "insufficient flow at outlet" is given in figure 6. This fault tree includes \overline{PA} , \overline{PB} conditions for the pump to function on one stream in order to produce a reversed flow in the second.

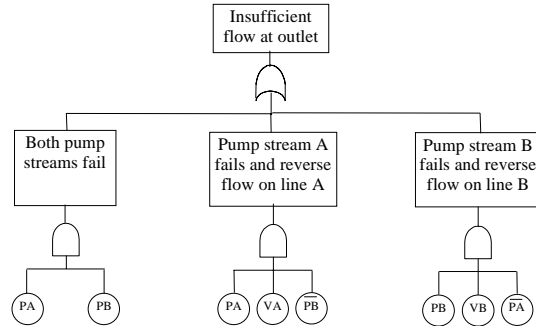


Figure 6 - Pumping System Fault Tree

This fault tree represents the logic equation:

$$\text{TOP} = PA.PB + PA.VA.\overline{PB} + PB.VB.\overline{PA} \quad (5)$$

Taking the first two terms and applying the consensus rule (4) gives:

$$\begin{aligned} PA.PB + PA.VA.\overline{PB} &= PA.PB + PA.VA.\overline{PB} \\ &\quad + PA.VA \\ &= PA.PB + PA.VA \end{aligned}$$

Considering the first and third terms in the same way produces:

$$\text{TOP} = PA.PB + PA.VA. + PB.VB \quad (6)$$

which is coherent.

Multitask Systems

Having outlined some of the difficulties with the use of NOT gates in fault trees it has been demonstrated that there needs to be a convincing reason for their inclusion. An application which will benefit from the incorporation of component success states in the system failure logic is when a system which is being analysed performs more than one task. In this case the outcomes of the system performance can produce combinations of some tasks being performed whilst others have failed. The causes of each system outcome cannot be identified correctly without accounting for the parts of the system which have worked.

The best way of illustrating the use of NOT logic and the advantage it offers for a multitasking system is by an example. Consider the simplified gas detection system shown in figure 7.

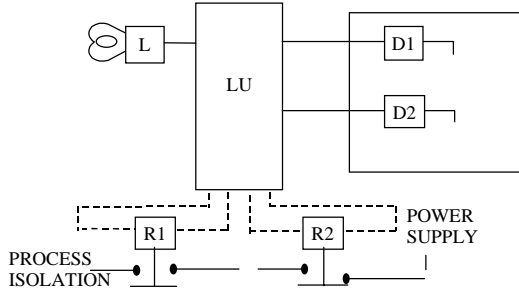


Figure 7 - Gas Detection System

Two gas sensors (D1 and D2) are used to detect a leakage of gas in a confined space. The signals from these detectors are fed along individual cables back to the computer logic control unit (LU). On receiving a signal which represents a gas leak from either detector the system has three functions:

- (a) Process shut-down (isolation) - by de-energising relay R1
- (b) Inform the operator of the leak by lamp/siren labelled L
- (c) Remove the power supply (potential ignition sources) to the affected area by de-energising relay R2.

The system could be considered to fail if it does not perform any of the tasks for which it was designed given a leak condition. The fault tree for top event "Gas detection system failure" is shown in figure 8.

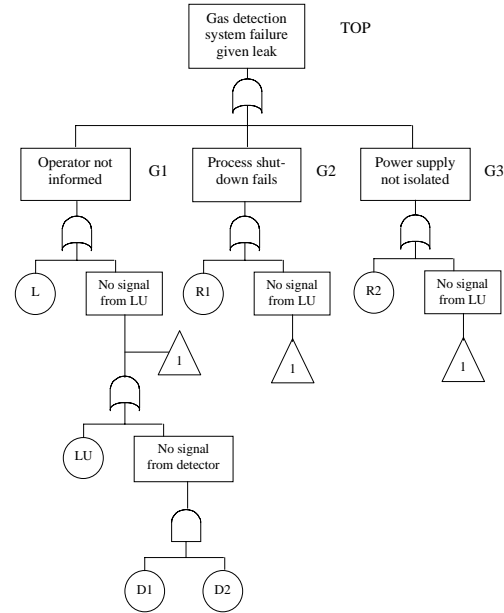


Figure 8 - Gas Detection System Failure Fault Tree

Boolean analysis of this fault tree gives minimal cut sets:

- L
- R1
- R2
- LU
- D1 D2

However there are different degrees by which this system can be considered to fail. The fault tree shown in figure 8 represents the most severe option - any malfunction is considered as system failure. There are eight (2^3) outcomes of this system which performs three tasks, shown in Table 1 each of differing severity.

Outcome	Operator informed	Process Shut-down	Power Isolation
1	W	W	W
2	W	W	F
3	W	F	W
4	W	F	F
5	F	W	W
6	F	W	F
7	F	F	W
8	F	F	F

W - Subsystem functions
F - Subsystem fails

Table 1 - Gas Detection System Outcome

The figure 8 fault tree represents the occurrence of any task failure. In the analysis of the system, one outcome which we would wish to have a very small probability, or even design out is outcome 4, where the operator thinks everything is fine but the process has not shut-down, nor the power been isolated. The operator could instigate the safety tasks manually if aware of the true situation. A fault tree analysis performed to determine the likelihood of this would have the top event structure shown in figure 9 for a coherent assessment.

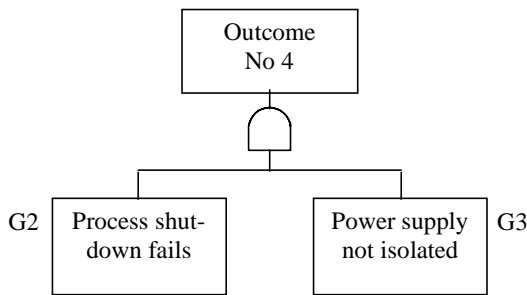


Figure 9 - Outcome 4 Coherent Fault Tree

The minimal cut sets obtained from this will be:

1. R1.R2.
2. D1 D2
3. LU

Quantification of the fault tree will substantially overestimate the probability of the outcome 4 unless the part of the system which functioned is taken into account. If the operator was informed of the leak Min Cut Sets 2 and 3 could not have occurred. For a correct assessment we need to use a NOT gate as shown in figure 10.

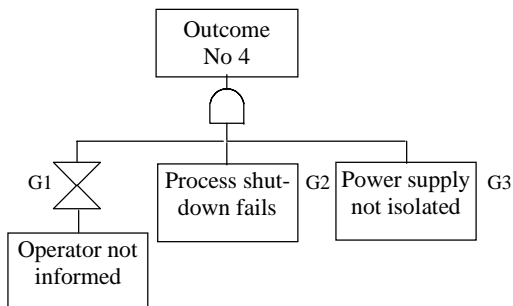


Figure 10 - Outcome 4 Non-coherent Fault Tree

Giving Logic equation

$$\begin{aligned}
 G1 &= \overline{L + LU + D1.D2} \\
 &= \overline{L.LU.(D1 + D2)} \\
 &= \overline{L.LU.D1} + \overline{L.LU.D2}
 \end{aligned}$$

$$G2 = R1 + LU + D1.D2$$

$$G3 = R2 + LU + D1.D2$$

$$TOP = G1.G2.G3$$

$$\begin{aligned}
 &= \overline{L.LU.(D1 + D2)}.(R1 + LU + D1.D2). \\
 &\quad (R2 + LU + D1.D2) \\
 &= \overline{L.LU.(D1 + D2)}. \\
 &\quad (R1.R2 + LU + D1.D2) \\
 &= \overline{L.LU.R1.R2.(D1 + D2)}
 \end{aligned}$$

The coherent approximation is R1.R2 which agrees with minimal cut set 1 produced by the Boolean reduction of the coherent fault tree.

The use of Not logic has successfully removed the inappropriate failure combinations.

Analysis of Non-coherent Fault Trees

If component success states are to be used in the fault trees an alternative means of analysis is required which overcomes the difficulties encountered with conventional Fault Tree theory outlined in section 4. Recent advances in methodology have resulted in the Binary Decision Diagram (BDD) method which has distinct advantages for quantifying Non-coherent Fault trees.

A BDD is a directed acyclic graph. All paths through the BDD terminate in one of two states, either a 1 state, which corresponds to system failure, or a 0 state which corresponds to system success. All the paths terminating in a 1 state give the cut sets/prime implicants of the fault tree. A BDD is composed of terminal and non-terminal vertices, which are connected by branches. Terminal vertices have the value 0 or 1 and non-terminal vertices correspond to the basic events of the fault tree. Each non-terminal vertex has a 0 branch which represents the basic event non-occurrence (works) and a 1 branch which represents basic event occurrence (fails). As an example BDD consider figure 11.

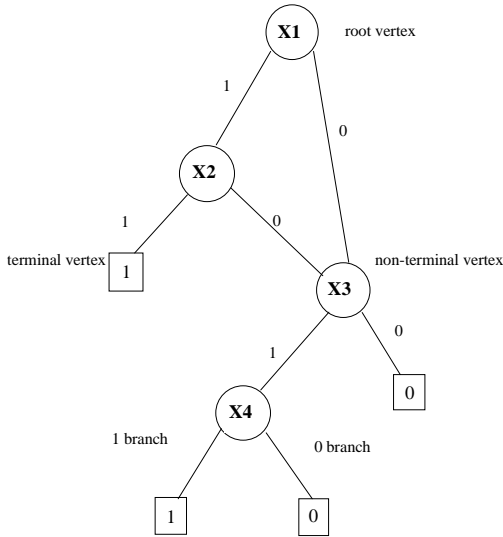


Figure 11 - Example BDD

All the left hand branches leaving each vertex are the 1 branches and all the right hand branches are the 0 branches.

Every path starts from the root vertex, and proceeds down through the diagram to the terminal vertices. Only the vertices that lie on a 1 branch on the way to a terminal 1 vertex are included in a path which represents the cut sets. For example, the paths, or cut sets, of figure 11 are:

- 1) X1.X2
- 2) X1.X3.X4
- 3) X3.X4

Each vertex in a BDD has an if-then-else (**ite**) structure. To illustrate this consider the if-then-else structure **-ite** (X1,f1,f2), which means if X1 fails then consider function f1 else consider function f2. Also f1 lies on the 1 branch of X1 and f2 on the 0 branch. The BDD for this is the one given in figure 12.

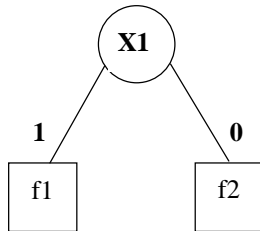


Figure 12 - **ite** Structure

Constructing the BDD from a fault tree initially requires the basic events in the fault tree to be given an ordering, such as X1<X2 (normally a Top-Down ordering is used, i.e., the basic events which are placed higher up the tree are listed first and are regarded as being 'less than' those further down the tree). It also involves using **ite** structures and the following procedures:

- 1) Taking X < Y;
Let J = **ite**(X, F1, F2) and H = **ite**(Y, G1, G2)
then;

$$J < op > H = \mathbf{ite}(X, F1 < op > H, F2 < op > H) \quad (7)$$

- 2) Taking X = Y;
i.e. J = **ite**(X, F1, F2) and H = **ite**(X, G1, G2)
then;

$$J < op > H = \mathbf{ite}(X, F1 < op > G1, F2 < op > G2) \quad (8)$$

where <op> corresponds to the Boolean operation of the logic gates in the fault tree.

For example consider the fault tree drawn in figure 4. The basic events can be represented as **ite** structures:

$$A = \mathbf{ite}(A, 1, 0), \bar{A} = \mathbf{ite}(A, 0, 1)$$

$$B = \mathbf{ite}(B, 1, 0) \text{ and } \bar{C} = \mathbf{ite}(C, 0, 1)$$

Using **ite** structures given in equation 7 and 8 and basic event ordering A<B<C.

$$\begin{aligned} G1 &= \mathbf{ite}(A, 1, 0).\mathbf{ite}(C, 0, 1) \\ &= \mathbf{ite}(A, \mathbf{ite}(C, 0, 1), 0) \end{aligned} \quad (\text{using equation 7})$$

$$\begin{aligned} G2 &= \mathbf{ite}(A, 0, 1).\mathbf{ite}(B, 1, 0) \\ &= \mathbf{ite}(A, 0, \mathbf{ite}(B, 1, 0)) \end{aligned} \quad (\text{using equation 7})$$

$$\begin{aligned} \text{TOP} &= G1 + G2 \\ &= \mathbf{ite}(A, \mathbf{ite}(C, 0, 1), 0) + \mathbf{ite}(A, 0, \mathbf{ite}(B, 1, 0)) \\ &= \mathbf{ite}(A, \mathbf{ite}(C, 0, 1), \mathbf{ite}(B, 1, 0)) \end{aligned} \quad (\text{using equation 8}) \quad (9)$$

From equation 9 the BDD for the fault tree is constructed and shown in figure 13.

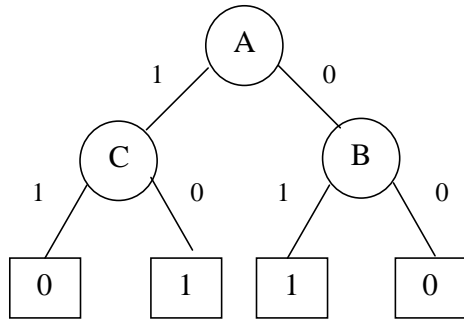


Figure 13 - BDD for the Traffic Light System

The Top event probability is obtained from a BDD by summing the probability of each disjoint path. For the BDD in figure 13 the disjoint paths resulting in system failure are:

Path	Probability
$\overline{A}\overline{C}$	$q_A(1 - q_C)$
$\overline{A}B$	$(1 - q_A)q_B$

ie

$$P(\text{TOP}) = q_A(1 - q_C) + (1 - q_A)q_B$$

agreeing with the exact equation derived earlier by the traditional fault tree method but does not require the intermediate stage of developing prime implicants or an inefficient full series expansion for the probability calculations.

Conclusions

1. Problems are encountered in both qualitative and quantitative analysis of fault trees whose structure function is non-coherent when using traditional analysis methods.
2. It has been demonstrated that non-coherent fault trees do have a practical use in assessing multitask systems.
3. Efficiency and accuracy problems encountered using traditional analysis methods can be overcome by employing Binary Decision Diagrams to quantify the

top event probability of non-coherent fault trees.

References

1. Hassl, D.F., Roberts, N.H., Vesely, W.E. and Goldberg, F.F. Fault Tree Handbook. US Nuclear Regulatory Commission, NUREG-0492.
2. Henley, E.J. and Kumamoto, H. Reliability Engineering and Risk Assessment. Prentice Hall, 1980.
3. Andrews, J.D. and Moss, R.R. Reliability and Risk Assessment. Longmans, 1993.
4. Johnson, B.D. and Matthews, R.H. Non-coherent Structure Theory: A Review and its Role in Fault Tree Analysis. UKAAE, SRD R245, Oct 1983.
5. Rauzy, A. New Algorithms for Fault Tree Analysis. Reliability Engineering and System Safety, Vol 40, 1993, pp 203-211.
6. Sinnamon, R.M. and Andrews, J.D. Quantitative Fault Tree Analysis Using Binary Decision Diagrams. European Journal of Automation, Vol 30, No 8, 1996.

Bibliography

Dr. J. D. Andrews., Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK. telephone - +44 (0)1509 222862, facsimile - +44 (0)1509 223969, e-mail - j.d.andrews@lboro.ac.uk.

John Andrews is a Senior Lecturer in the Department of Mathematical Sciences at Loughborough University. He joined this department in 1989 having previously gained nine years industrial experience at British Gas. His current research interests concern the assessment of the safety and risks of potentially hazardous industrial systems. This research has been heavily supported by funding from industry. Current research projects involve Mobil, Daimler Chrysler, and Rolls Royce Aero Engines.