Chapter 6 Quantitative Systems Risk Analysis Methods

6.1 Introduction

The methods presented in this chapter, particularly the Fault Tree Analysis and Event Tree Analysis, are the most widely used techniques for analyzing in a qualitative *and* quantitative way the safety and reliability of complex industrial systems. Such systems are characterized by the fact that the number and linking of the different subsystems are too complex to permit to apply only and directly the elementary calculation techniques of Chapter 3.

6.2 Success Path Analysis (SPA) - Reliability Block Diagram (RBD)

General presentation of the method

The SPA method is a graphical representation and calculation tool used to model relatively complex systems. To this end, the different components of a system are symbolized as individual graphic and functional elements, called "*reliability blocks*"; for this reason, in its industrial applications this method is also known under the name of "*Reliability Block Diagram*", or RBD, method. These blocks are reliability-wise arranged and related, often, but not necessarily, in the same way that the corresponding components are physically connected. Such a diagram can be viewed as representing how a "system operation signal" would be successfully transmitted from the input to the output of the system. Once the blocks are properly configured, and reliability data for these blocks is provided, calculations can be performed in order to calculate the failure rate, the "mean time to failure" (MTTF), reliability and availability of the system. Obviously, as the configuration of the system, and thus of the block diagram changes, the calculation results also change. A reliability block diagram provides therefore a simple way to compare various possible configurations in an attempt to find the best overall system design

Historically, the SPA method was the first to be developed for the analysis of industrial systems in view of calculating their reliability. This method is applicable when a detailed analysis of the causes of failures is not required and when the component failures are *independent*. Moreover, whereas the SPA can straightforwardly be applied to *irreparable systems*, it can only be used for the reliability, availability or maintainability evaluation of repairable systems under quite restrictive conditions.

Series and active parallel configurations

The simplest and most elementary types of reliability blocks configurations are the *series* and *active-parallel* configurations. Items connected in series must all work for the system to fulfill its function ("*success path*"). In the example of figure 6.1, the system will fail if either C_1, C_2 or C_3 fails.

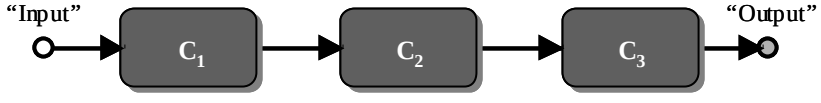


Figure 6.1 ***Reliability Block Diagram for a series configuration***

The reliability R_s of a system made of *N* independent components, all in series, can be calculated from the following expression (R_i: reliability of component *i*, assumed to be known):

$$R_s(t) = \prod_{i=1}^{N} R_i(t) \qquad [6.1]$$

In the case of constant failure rates λ_i it comes (from Eq. [3.11]):

$$R_s(t) = \exp\left(-\left\{ \sum_{i=1}^{n} \lambda_i \right\} \cdot t \right) \qquad [6.2]$$

The corresponding "Mean Time To Failure" $MTTF_s$ is thus given by (see Eq. [3.15]):

$$MTTF_s = \int_0^{\infty} \exp\left(-\left\{ \sum_{i=1}^{n} \lambda_i \right\} \cdot t' \, dt' \right) = \frac{1}{\sum_{i=1}^{n} \lambda_i} \qquad [6.3]$$

Items placed in parallel are considered to be redundant, because the good working of only one of them is enough for the system to function. In the example of figure 6.2, either C_1 or C_2 (but not C_1 and C_2 simultaneously) can fail and the system will continue to function.

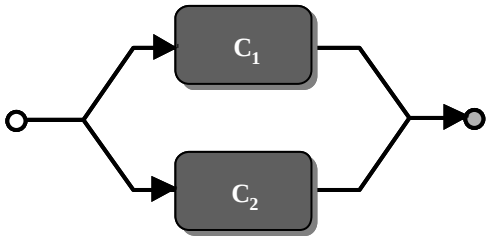


Figure 6.2 ***Reliability Block Diagram for a parallel configuration***

The reliability R_p of a system of *N* independent components, all in active-parallel, is given by the following mathematical expression:

$$1 - R_p(t) = \prod_{i=1}^{N} \left[1 - R_i(t) \right] \qquad [6.4]$$

In the case of constant failure rates, R_p takes the form:

$$R_p(t) = \sum_{i=1}^{N} \exp\left(-\lambda_i \cdot t \right) - \sum_{i=1}^{N} \sum_{j \neq i} \exp\left(-\left\{ \lambda_i + \lambda_j \right\} \cdot t \right) +$$
$$\sum_{i=1}^{N} \sum_{j \neq i} \sum_{k \neq i,\, j} \exp\left(-\left\{ \lambda_i + \lambda_j + \lambda_k \right\} \cdot t \right) - \ldots + (-1)^{(N+1)} \cdot \exp\left(-\left\{ \sum_{i=1}^{N} \lambda_i \right\} \cdot t \right) \qquad [6.5]$$

and the "Mean Time To Failure" $MTTF_p$ becomes:

$$MTTF_p = \sum_{i=1}^{N} \frac{1}{\lambda_i} - \sum_{i=1}^{N} \sum_{j \neq i} \frac{1}{\lambda_i + \lambda_j} + \ldots + (-1)^{(N+1)} \cdot \frac{1}{\sum_{i=1}^{N} \lambda_i} \qquad [6.6]$$

The two elementary configurations described above form the basis of the reliability block diagram construct and success path analysis.

This concept can be straightforwardly extended further to combinations of series and parallel configurations in the same diagram (see example in Fig. 6.3).

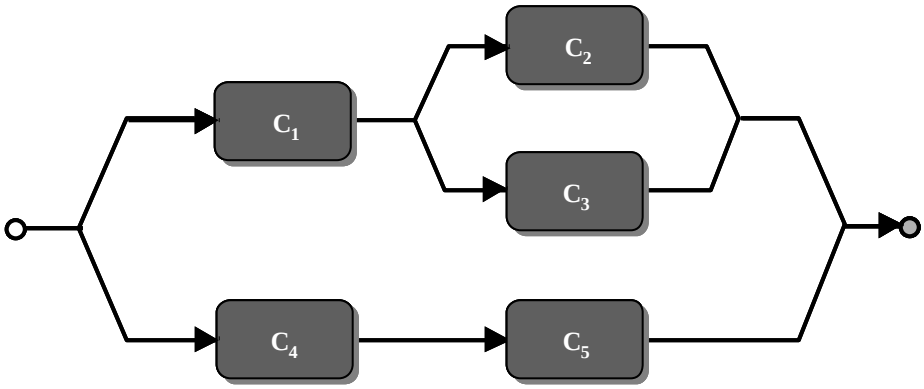


Figure 6.3 ***Reliability Block Diagram for a combination of series and parallel configuration***

Considering first the upper branch of the diagram, we have (C_1 in series with (C_2 and C_3 in parallel)):

$$R_{upp} = R_1 \cdot (R_2 + R_3 - R_2 \cdot R_3) \qquad [6.7]$$

The reliability of the lower branch (C_4 an C_5 in series) is simply given by:

$$R_{low} = R_4 \cdot R_5 \qquad [6.8]$$

Finally, combining this two branches in parallel leads to:

$$\overline{R}_{sys} = \overline{R}_{sup} \cdot \overline{R}_{inf} = \{1 - R_1 \cdot (R_2 + R_3 - R_2 \cdot R_3)\} \cdot (1 - R_4 \cdot R_5) \qquad [6.9]$$

and therefore:

$$R_{sys} = 1 - \overline{R}_{sys} = R_4 \cdot R_5 - (R_1 \cdot (R_2 + R_3 - R_2 \cdot R_3)) \cdot (1 - R_4 \cdot R_5) \qquad [6.10]$$

Complex block diagrams

If one takes the approach a step further, "complex block diagrams" can be analyzed. A *complex block diagram* is a diagram that cannot be expressed as a simple combination of series and parallel blocks (see example in Fig. 6.4).

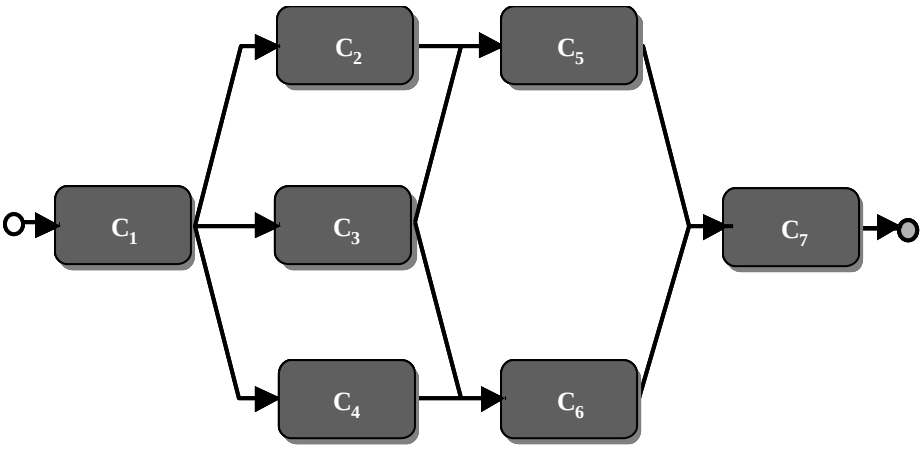


Figure 6.4 ***Reliability Block Diagram for a complex system configuration***

Several methods exist for obtaining the reliability of complex system, e.g.:

- the decomposition method,
- the event space method,
- the path-tracing method,
- the minimal cut set method.

To show how these different methods work, they will first be applied below to a very simple case, namely (see Fig. 6.5):

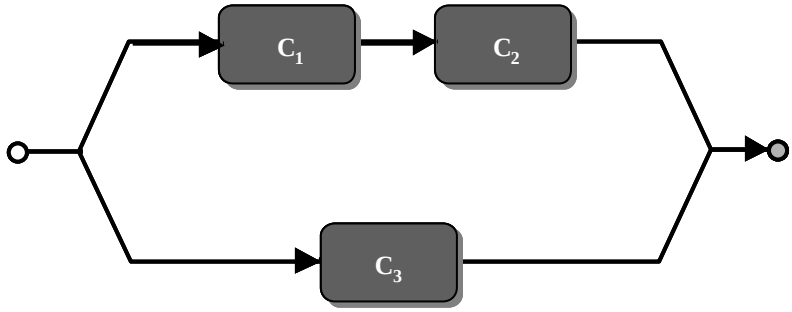


Figure 6.5 ***Demonstration case for the resolution methods that can be used to treat complex system configurations***

Decomposition method

The decomposition method is an application of the law of total probability (see section 2.4). It involves choosing a "*key component*" and then calculating the reliability of the system in two steps: first considering that the key component failed ($R_{key} = 0$) and secondly that it succeeded ($R_{key} = 1$). These two "complementary" probabilities are then combined to obtain the reliability of the system. Using probability theory, the equation is:

$$\begin{aligned} R_{sys} &= P\left(S \cap E_{key}\right) + P\left(S \cap \overline{E}_{key}\right) \\ &= P\left(S \middle| E_{key}\right) \cdot P\left(E_{key}\right) + P\left(S \middle| \overline{E}_{key}\right) \cdot P\left(\overline{E}_{key}\right) \end{aligned} \qquad [6.11]$$

where S and E_{key} represent respectively the events: "the system operates" and the "component C_{key}" operates ($\overline{E}_{key}$: "the component C_{key} fails").

In the example of the figure 6.5, selecting the component 2 as the "key component", the system reliability can be written as follows:

$$R_{sys} = P\left(S \middle| E_2\right) \cdot P\left(E_2\right) + P\left(S \middle| \overline{E}_2\right) \cdot P\left(\overline{E}_2\right) = P\left(S \middle| E_2\right) \cdot R_2 + P\left(S \middle| \overline{E}_2\right) \cdot \left(1 - R_2\right) \qquad [6.12]$$

If component 2 is known to operate, the probability that the system operates is given by the probability that component 1 *or* component 3 also survives, that is:

$$P\left(S \middle| E_2\right) = R_1 + R_3 - R_1 \cdot R_3 \qquad [6.13]$$

If, on the contrary this key component fails, the probability that the system still operates is simply equals to the probability that the component 3 operates:

$$P\left(S \middle| \overline{E}_2\right) = R_3 \qquad [6.14]$$

Introducing these two conditional probabilities in Eq. [6.12] leads to:

$$R_{sys} = (R_1 + R_3 - R_1 \cdot R_3) \cdot R_2 + R_3 \cdot (1 - R_2) = R_3 + R_1 \cdot R_2 - R_1 \cdot R_2 \cdot R_3 \qquad [6.15]$$

Event space method

The event space method is an application of the mutually exclusive events axiom. All mutually exclusive events are determined and only those that result in system success are retained. The reliability of the system is simply equal to the probability of the union of all mutually exclusive events that yield a system success. Reciprocally, the unreliability of the system is the probability of the union of mutually exclusive events that yield a system failure.

Applied to the example of figure 6.5, this gives:

- all components succeed → $E_{123} = E_1 \cap E_2 \cap E_3$
- only component 1 fails → $E_{\bar{1}23} = \overline{E}_1 \cap E_2 \cap E_3$
- only component 2 fails → $E_{1\bar{2}3} = E_1 \cap \overline{E}_2 \cap E_3$
- only component 3 fails → $E_{12\bar{3}} = E_1 \cap E_2 \cap \overline{E}_3$
- components 1 and 2 fail → $E_{\bar{1}\bar{2}3} = \overline{E}_1 \cap \overline{E}_2 \cap E_3$
- components 1 and 3 fail → $E_{\bar{1}2\bar{3}} = \overline{E}_1 \cap E_2 \cap \overline{E}_3$
- components 2 and 3 fail → $E_{1\bar{2}\bar{3}} = E_1 \cap \overline{E}_2 \cap \overline{E}_3$
- all components fail → $E_{\bar{1}\bar{2}\bar{3}} = \overline{E}_1 \cap \overline{E}_2 \cap \overline{E}_3$

The five first events result in system success. Thus the total probability of success of the system is:

$$R_{sys} = P\left(E_{123} \cup E_{\bar{1}23} \cup E_{1\bar{2}3} \cup E_{12\bar{3}} \cup E_{\bar{1}\bar{2}3}\right) \qquad [6.16]$$

Since these five events are mutually exclusive, then:

$$R_{sys} = P\left(E_{123}\right) + P\left(E_{\bar{1}23}\right) + P\left(E_{1\bar{2}3}\right) + P\left(E_{12\bar{3}}\right) + P\left(E_{\bar{1}\bar{2}3}\right) \qquad [6.17]$$

with:

$$P\left(E_{123}\right) = P\left(E_1 \cap E_2 \cap E_3\right) = R_1 \cdot R_2 \cdot R_3$$
$$P\left(E_{\bar{1}23}\right) = P\left(\overline{E}_1 \cap E_2 \cap E_3\right) = \left(1 - R_1\right) \cdot R_2 \cdot R_3$$
$$P\left(E_{1\bar{2}3}\right) = P\left(E_1 \cap \overline{E}_2 \cap E_3\right) = R_1 \cdot \left(1 - R_2\right) \cdot R_3$$
$$P\left(E_{12\bar{3}}\right) = P\left(E_1 \cap E_2 \cap \overline{E}_3\right) = R_1 \cdot R_2 \cdot \left(1 - R_3\right)$$
$$P\left(E_{\bar{1}\bar{2}3}\right) = P\left(\overline{E}_1 \cap \overline{E}_2 \cap E_3\right) = \left(1 - R_1\right) \cdot \left(1 - R_2\right) \cdot R_3$$

Adding the above five relations gives after simplification:

$$R_{sys} = R_3 + R_1 \cdot R_2 - R_1 \cdot R_2 \cdot R_3$$

This is of course the same result as the one obtained previously using the decomposition method.

Path tracing method

This method is based on the observation that as long as at least one success path exist from the input to the output of the RBD, then the system has not failed. It thus consists in identifying all of the success paths the “signal” could take and calculating the reliability of these paths based on the components that lie along each of them. The reliability of the system is simply the probability of the union of these paths.

In the example of the figure 6.5, the success paths are:

$$SP_1 = \{E_1, E_2\} \quad \text{and} \quad SP_2 = E_3$$

Therefore, the probability of success of the system is given by:

$$R_{sys} = P(SP_1 \cup SP_2) = P(SP_1) + P(SP_2) - P(SP_1 \cap SP_2) \qquad [6.18]$$

Replacing the three probabilities on the left right side of Eq. [6.18] by their calculated values, we get:

$$R_{sys} = R_1 \cdot R_2 + R_3 - R_1 \cdot R_2 \cdot R_3$$

Again, we find the same results as with the previous methods.

Minimal cut set method

As its name implies, this method is based on the search for the "minimal cut sets" of the system under study, i.e. a set of system events that, if they all occur, will cause system failure (this concept will be developed further in the next section). The unreliability of the system can then be calculated as the probability of the union of these events and its reliability by taking the complement of the result thus obtained.

The procedure is developed below for the example of figure 6.5:

a) Create a table ("*Incidence Matrix*") with one line for each of the identified success paths, and columns representing the different components of the system. If a component "belongs" to the considered success path a "1" is entered in the corresponding column of the concerned line, and if not, a "0". In the case of the very simple example of figure 6.5 this table takes the form:

Table 6.1 ***Incidence matrix for the example of figure 6.5***

Success paths / components	C_1	C_2	C_3
$\{E_1, E_2\}$	1	1	0
E_3	0	0	1

b) Now, examine each column. If a column contains nothing but "1" numbers, this means that the failure of this unique component will induce in any case the failure of the whole system (because this component lies along each of the success paths). This is called a *minimal cut set of order one* (one component only involved in the success path). There are no minimal cut sets of order one in the considered example.

c) Go forward in the analysis of the table, considering this time two by two the different columns, with the following rules for combining the scores:

$$0 + 0 = 0 \qquad 1 + 0 = 1 \qquad 0 + 1 = 1 \qquad 1 + 1 = 1$$

A combined column full of "1" means that the *simultaneous* failures of the two components involved induce the failure of the system. This defines consequently a *minimal cut set of order two*. We have two such cut sets in the considered example, corresponding to the combination of events: $\overline{E}_1 \cap \overline{E}_3$ and $\overline{E}_2 \cap \overline{E}_3$.

d) Continue the search for possible higher order cut sets (3, 4, …) that do not include lower order cut sets already considered. There are no such cut sets in our illustration example.

The unreliability of the system is given by the probability of the union of the above-defined minimal cut sets MCS_i:

$$\overline{R}_{sys} = P\left(\bigcup_i MCS_i\right) \qquad [6.19]$$

In our example, this leads to:

$$\overline{R}_{sys} = P\left\{\left(\overline{E}_1 \cap \overline{E}_3\right) \cup \left(\overline{E}_2 \cap \overline{E}_3\right)\right\} \qquad [6.20]$$

which, using Morgan's theorem and other Boolean Algebra rules (see section 2.3), can be written as follows:

$$\overline{R}_{sys} = P\left\{\overline{E}_3 \cap \left(\overline{E}_1 \cup \overline{E}_2\right)\right\} = P\left\{\overline{E}_3 \cap \left(\overline{E_1 \cap E_2}\right)\right\} \qquad [6.21]$$

Therefore:

$$\overline{R}_{sys} = \left(1 - R_3\right) \cdot \left(1 - R_1 \cdot R_2\right) = 1 - R_1 \cdot R_2 - R_3 + R_1 \cdot R_2 \cdot R_3 \qquad [6.22]$$

and $R_{sys} = 1 - \overline{R}_{sys}$ takes here again the same value as before.

To show that the above approaches can indeed be used to calculate the reliability of complex systems, let us apply the decomposition and minimal cut set methods to the following example:

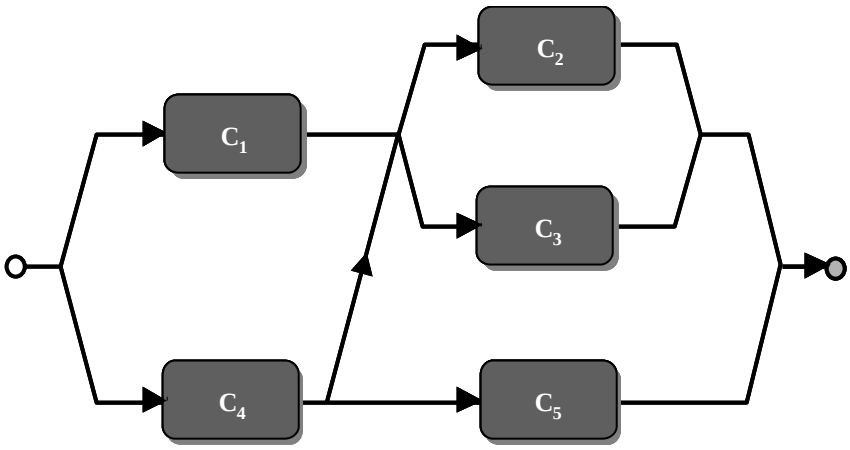


Figure 6.6 ***Application example (complex configuration) for the decomposition and minimal cut set methods***

Decomposition method

The best choice of key component would here be C4. This will give the results we are looking for in just one step. Let us however take another key component first, C_3, to demonstrate that the decomposition method can be used repetitively to solve a given problem.

The probability that the system operates knowing that C_3 does not fail corresponds to the probability of success of the system represented in figure 6.7, i.e. C1 and C_4 in parallel. The corresponding conditional probability is therefore equal to:

$$P\left(S \middle| E_3\right) = R_1 + R_4 - R_1 \cdot R_4 \qquad [6.23]$$

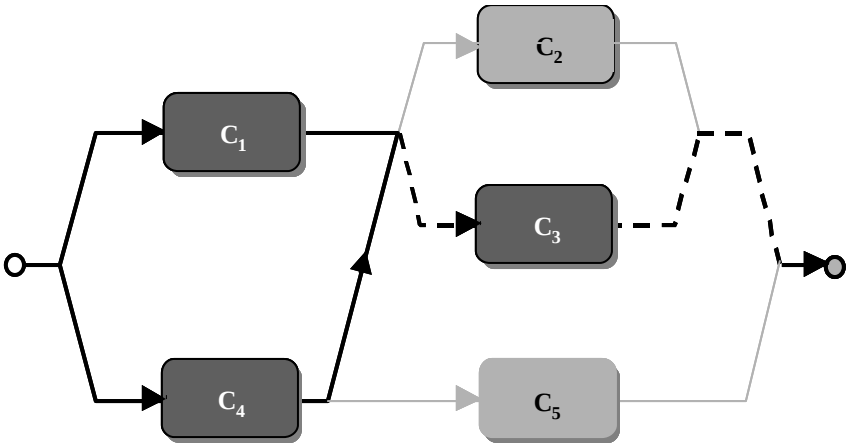


Figure 6.7 ***Configuration of the example of Fig. 6.6 when the key component C_3 is known to succeed***

In the "complementary" case, where C_3 is known to fail, we are however still left with a system that cannot be broken down in a combination of series and parallel configurations (see Fig. 6.7).

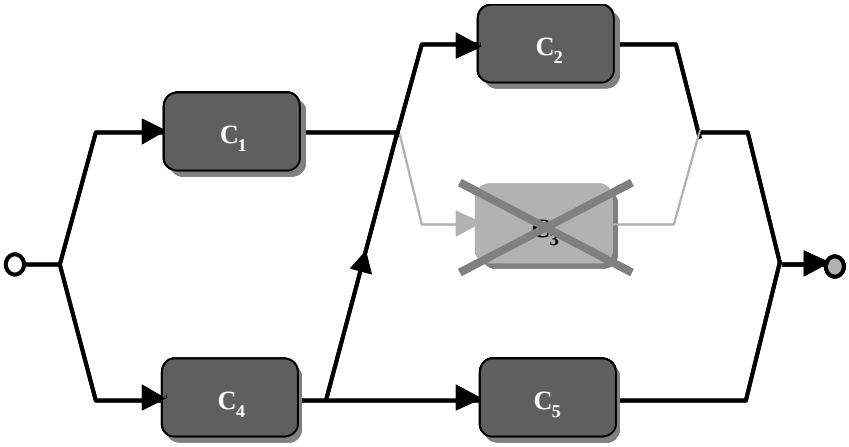


Figure 6.8 ***Configuration of the example of Fig. 6.6 when the key component C_3 is known to fail***

We have therefore to use a second time the decomposition principle, choosing this time C_2 as new key component.

The probability that the system operates knowing that C_2 succeed is again given by the probability of having a successful parallel configuration C_1 and C_4, i.e.:

$$P\left(S \middle| E_2 \cap \overline{E}_3\right) = R_1 + R_4 - R_1 \cdot R_4 \qquad [6.24]$$

In the case where C_2 equally fails, the system to consider is the one represented in Fig. 6.9 (C_4 and C_5 in series). Its reliability is given by:

$$P\left(S \middle| \overline{E}_2 \cap \overline{E}_3\right) = R_4 \cdot R_5 \qquad [6.25]$$

Multiplying the three conditional probabilities (Eqs [6.23] to [6.25]) by the corresponding probabilities of occurrence of the concerned events allows us to calculate the reliability of the whole system.

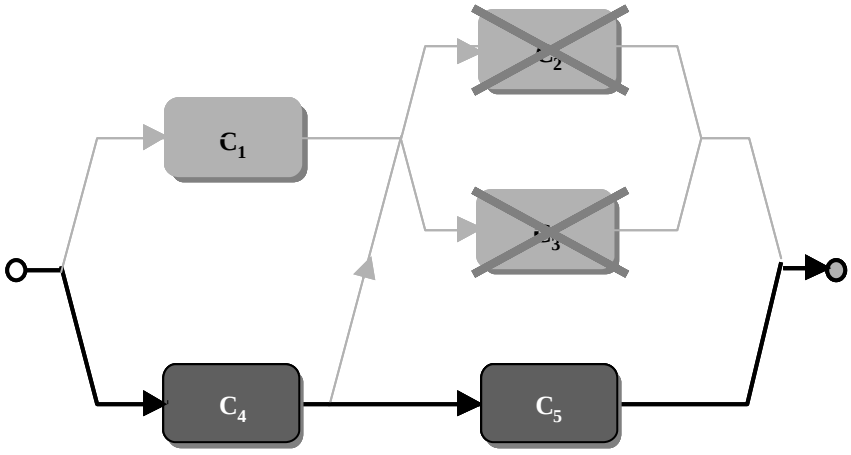


Figure 6.9 ***Configuration of the example of Fig. 6.6 when the key components C_2 and C_3 are known to fail***

The reliability of the complex system selected as example is thus given by:

$$R_{sys} = (R_1 + R_4 - R_1 \cdot R_4) \cdot R_3 + (R_1 + R_4 - R_1 \cdot R_4) \cdot R_2 \cdot (1-R_3) + R_4 \cdot R_5 \cdot (1-R_2) \cdot (1-R_3)$$
$$= (R_1 + R_4 - R_1 \cdot R_4) \cdot (R_3 + R_2 - R_2 \cdot R_3) + R_4 \cdot R_5 \cdot (1-R_2-R_3+ R_2 \cdot R_3) \qquad [6.26]$$

Minimal cut set method

The incidence matrix for the selected example takes the form:

Table 6.2 ***Incidence matrix for the example of figure 6.6***

Success paths / components	C_1	C_2	C_3	C_4	C_5
$\{E_1, E_2\}$	1	1	0	0	0
$\{E_1, E_3\}$	1	0	1	0	0
$\{E_2, E_4\}$	0	1	0	1	0
$\{E_3, E_4\}$	0	0	1	1	0
$\{E_4, E_5\}$	0	0	0	1	1

There are here no minimal cut sets of order 1, one minimal cut sets of order 2, and two minimal cut sets of order 3:

- MCS of order 2 $\rightarrow$ $\overline{E}_1 \cap \overline{E}_4$
- MCS of order 3 $\rightarrow$ $\overline{E}_2 \cap \overline{E}_3 \cap \overline{E}_4$ and $\overline{E}_2 \cap \overline{E}_3 \cap \overline{E}_5$

Thus, the unreliability of the whole system can be calculated from the expression:

$$\overline{R}_{sys} = P\left\{\left(\overline{E}_1 \cap \overline{E}_4\right) \cup \left(\overline{E}_2 \cap \overline{E}_3 \cap \overline{E}_4\right) \cup \left(\overline{E}_2 \cap \overline{E}_3 \cap \overline{E}_5\right)\right\} \qquad [6.27]$$

which, with the help of the Poincaré's theorem (see section 2.3), can be transformed in an algebraic sum of probabilities of the form:

$$P\left(\overline{E}_i\right) \cdot P\left(\overline{E}_j\right) \cdot \ldots P\left(\overline{E}_k\right)$$

Replacing each $P\left(\overline{E}_\alpha\right)$ by $(1-R_\alpha)$ allows us finally to calculate the system reliability.

Alternative forms of redundancy

Until now we have considered only the simplest form of redundancy, i.e. the active one. It is also possible to introduce alternative forms of redundancy in the RBD "toolkit", such as the redundancy known as *k-out-of-n redundancy* for example. A k-out-of-n configuration is a special form of parallel redundancy; it requires that at least k out of the n possible parallel paths leading to a given node (see Fig. 6.10) must function for the system to operate.

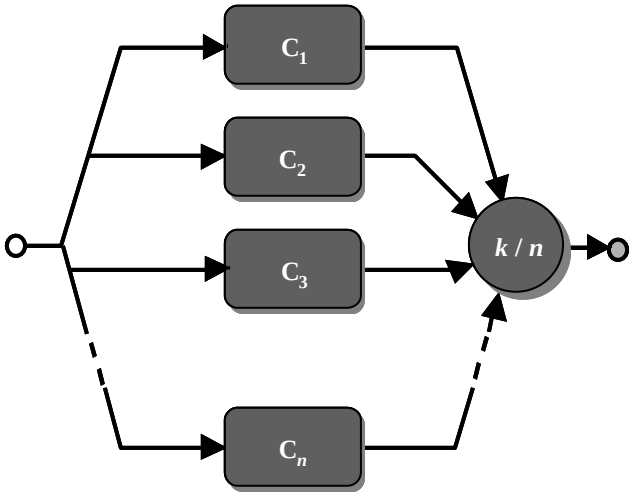
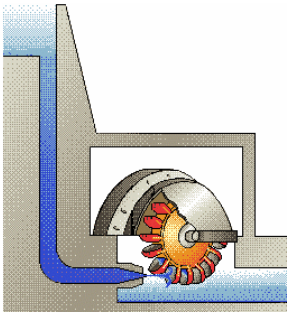


Figure 6.10 ***"k-out-of-n" configuration***



For example, suppose that a hydropower plant is equipped with six turbines and that at least five of them are required to function for the plant to remain operational. This means that the turbines are reliability-wise in a k-out-of-n configuration where k=5 and n=6.

This particular example is a good illustration of the fact that although a k-out-of-n configuration is in principle classified as a special case of parallel redundancy, the system behavior tends towards that of a series system when the number of units required to keep the system operating approaches the total number of units in the system (see Fig. 6.11, calculated from Eq. [6.28] with R = 0.85).

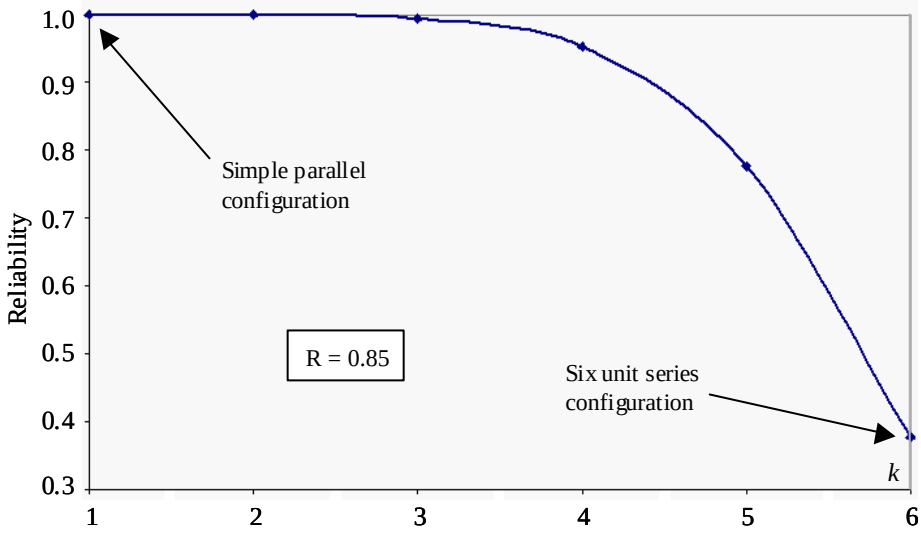


Figure 6.11 ***Reliability vs. k (k-out-of-n, for n=6)***

The simplest case of a k-out-of-n configuration is when the components are independent and identical. In such a case, the reliability of the system can be evaluated using the binomial distribution (see section 2.1):

$$R_{k/n}(t) = \sum_{r=k}^{n} \frac{n!}{r!(n-r)!} \cdot R(t)^r \cdot (1 - R(t))^{(n-r)} \qquad [6.28]$$

where $R(t)$ is the reliability of each unit. Assuming moreover that $R(t) = \exp(-\lambda t)$, with λ constant, we can write:

$$MTTF_{k/n} = \sum_{r=k}^{n} \int_0^{\infty} \frac{n!}{r!(n-r)!} \cdot \left(e^{-\lambda t'}\right)^r \cdot \left(1 - e^{-\lambda t'}\right)^{(n-r)} dt' = \sum_{r=k}^{n} I_r \qquad [6.29]$$

The integrals I_r can be calculated using integration by parts; it comes out that a recurrence relationship exists between I_r and I_{r+1} - of the form $I_r = \{(r+1)/r\} \cdot I_{r+1}$ - and that $I_n = 1/(n\lambda)$. From this, we deduce that:

$$MTTF_{k/n} = \sum_{r=k}^{n} \frac{1}{r \cdot \lambda} \qquad [6.30]$$

Another type of redundancy is the so-called *passive redundancy*. It defines a class of systems that are load-sharing or sequential in operations in such a way that only one unit of the system is in operation at a time. Other units are on stand-by, ready to take the load at their turn should a previously operating unit fail. In order to minimize the complexity of the equations, we will assume that a unit on stand-by never fails and, furthermore, that the switching from one unit to another one is instantaneous and not subject to failure. Such system can be depicted as in Fig. 6.12.

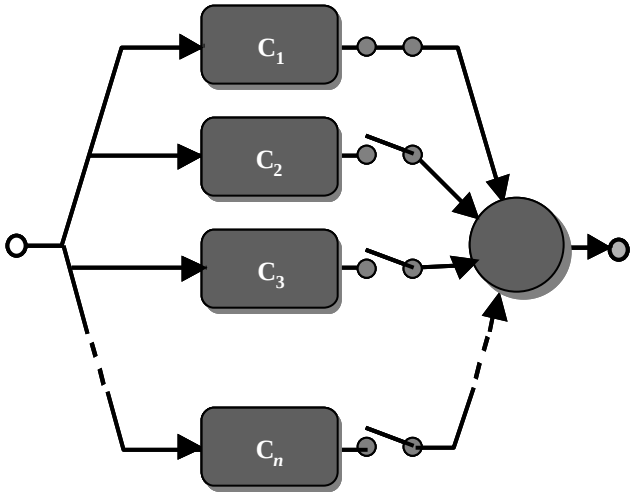


Figure 6.12 ***Passive redundancy configuration***

The probability density function of such a system can be written:

$$f_T(t) = \int_{x_{n-1}=0}^{t} \int_{x_{n-2}=0}^{x_{n-1}} \cdots \int_{x_1=0}^{x_2} f_1(x_1) \cdot f_2(x_2 - x_1) \ldots f_n(t - x_{n-1}) dx_1 dx_2 \ldots dx_{n-1} \qquad [6.31]$$

where the f_i are the probability density functions of the different units i.

The reliability of the whole system is, by virtue of Eq. [3.4], given by:

$$R_{sys}(t) = 1 - \int_0^t f_T(t')\,dt' \qquad [6.32]$$

The Laplace transform of the convolution integral on the right-hand side of Eq. [6.32] is (see Appendix 3.1):

$$\mathcal{L}\{\overline{R}_{sys}(t)\} = \frac{1}{s}\cdot\mathcal{L}\{f_T(t)\} = \frac{1}{s}\prod_{i=1}^{n}\mathcal{L}\{f_i(t)\} \qquad [6.33]$$

When all the λ_is are constant, we have (see Table [3.1]):

$$f_i(t) = \lambda_i \cdot e^{-\lambda_i \cdot t} \;\Rightarrow\; \mathcal{L}\{\overline{R}_{sys}(t)\} = \frac{1}{s}\prod_{i=1}^{n}\frac{\lambda_i}{s+\lambda_i} \qquad [6.34]$$

If moreover the λ_is are all different, taking the inverse transform of the expression on the right-hand side of Eq. [6.34] leads to:

$$\overline{R}_{sys}(t) = \sum_{i=0}^{n} B_i \cdot e^{-\lambda_i \cdot t} \text{ with: } B_i = \frac{\prod_{j=1}^{n}\lambda_j}{\prod_{j=0,\,j\neq i}^{n}(\lambda_j - \lambda_i)} \text{ and } \lambda_0 = 0 \qquad [6.35]$$

Thus, finally:

$$R_{sys}(t) = \lambda_1 \cdot \lambda_2 \ldots \lambda_n \cdot \sum_{i=1}^{n}\frac{e^{-\lambda_i \cdot t}}{\prod_{j=0,\,j\neq i}^{n}(\lambda_j - \lambda_i)} \qquad [6.36]$$

If, on the contrary, all the λ_is take the same constant value λ, Eq.[6.34] becomes:

$$\mathcal{L}\{\overline{R}_{sys}(t)\} = \frac{1}{s}\cdot\mathcal{L}\{f_T(t)\} = \frac{1}{s}\cdot\frac{\lambda^n}{(s+\lambda)^n} \qquad [6.37]$$

$f_T(t)$ is in this case an Erlang distribution:

$$f_T(t) = \frac{\lambda^n \cdot t^{(n-1)} \cdot e^{-\lambda \cdot t}}{(n-1)!} \qquad [6.38]$$

From Eq.[6.32], and after successive integrations by parts, it comes:

$$R_{sys}(t) = 1 - \int_0^t \frac{\lambda^n \cdot t'^{(n-1)} \cdot e^{-\lambda \cdot t'}}{(n-1)!}\,dt' = \sum_{i=1}^{n}\frac{(\lambda t)^{(i-1)} \cdot e^{-\lambda \cdot t}}{(i-1)!} \qquad [6.39]$$

The MTTF is here given by (see Eq.[3.14] and Appendix 3.1):

$$MTTF = \int_0^\infty t \cdot f_T(t)\,dt = \lim_{S\to 0}\mathcal{L}\{t \cdot f_T(t)\} = \lim_{S\to 0} -\frac{d}{ds}\,\mathcal{L}\{f_T(t)\} \qquad [6.40]$$

Therefore, for constant and equal λ_is (from Eq. [6.37]):

$$MTTF = \frac{n}{\lambda} \qquad [6.41]$$

Application to repairable systems

Because of the underlying assumption regarding the independence of the components, the SPA/RBD approach can only be applied under quite restrictive conditions to the evaluation of the availability of repairable systems, in particular:

- there should be no repeated components in the reliability block diagram;
- component repairs should be conducted in a totally independent way (therefore, in principle, by different repairmen);
- there should be only active redundancies in the system (passive redundancies imply a direct link between the operating times of the different components).

Series configurations

The availability $A_s(t)$ of a series system is simply equal to the product of the availability $a_i(t)$ of its components at the time *t*:

$$A_s(t) = \prod_{i=1}^{n} a_i(t) \qquad [6.42]$$

If the failure and reparation rates are constant and, moreover A(0) = 1, we have from Eq. [3.28]:

$$A_s(t) = \prod_{i=1}^{n} \left[\frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i} \cdot e^{-(\lambda_i + \mu_i)\cdot t} \right] \qquad [6.43]$$

The asymptotic value of the availability takes thus the form:

$$A_s(\infty) = \lim_{t \to \infty} A_s(t) = \prod_{i=1}^{n} \frac{\mu_i}{\lambda_i + \mu_i} \qquad [6.44]$$

In the usual situation where $\lambda_i/\mu_i << 1$, the asymptotic unavailability becomes:

$$\overline{A}_s(\infty) = 1 - A_s(\infty) = \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i} \qquad [6.45]$$

Parallel configurations

The unavailability $\overline{A}_p(t)$ of a parallel system is equal to the product of the unavailability $\overline{a}_i(t)$ of its components at the time *t*, therefore:

$$\overline{A}_p(t) = \prod_{i=1}^{n} \overline{a}_i(t) \quad \Rightarrow \quad A_p(t) = 1 - \prod_{i=1}^{n} (1 - a_i(t)) \qquad [6.46]$$

If the failure and reparation rates are constant and, moreover A(0) = 1, it comes:

$$A_p(t) = 1 - \prod_{i=1}^{n} \frac{\lambda_i}{\lambda_i + \mu_i} \cdot \left[1 - e^{-(\lambda_i + \mu_i)\cdot t} \right] \qquad [6.47]$$

The asymptotic unreliability is thus here given by:

$$\overline{A}_p(\infty) = 1 - A_p(\infty) = \prod_{i=1}^{n} \frac{\lambda_i}{\lambda_i + \mu_i} \approx \prod_{i=1}^{n} \frac{\lambda_i}{\mu_i} \qquad [6.48]$$

Strengths and limitations of SPA/RBD

As with any approach or methodology, reliability block diagrams have their advantages as well as disadvantages compared to competing methods. The main advantage of the SPA/RBD method is rooted in the fact that it is easy to implement and remains close to the real configuration of the analyzed system. Because of the limitations in the practical use of SPA/RBD, essentially due to the hypothesis of independency of the events, methods such as the Failure Modes and Effects Analysis or the Fault Tree Analysis are however today generally preferred to the former.

6.3 Fault Tree Analysis (FTA)

General presentation of the method

Fault Tree Analysis is a deductive, top-down logical and structured process to systematic failure analysis. FTA provides a graphic "model" of the pathways within a system that can lead to a foreseeable undesirable damaging event. It helps identifying potential causes of system failures before the failures actually occur. FTA is acknowledged as one of the best and most popular techniques in complex system design, development, and operation for systematically identifying and graphically displaying the many ways something can go wrong.

The FTA method was originally developed in the U.S.A. at Bell Telephone Laboratories in 1962 to evaluate and improve the reliability of the "Minuteman" missile launch control system. It has since this time largely been used for reliability and/or safety studies in the aerospace, nuclear (in particular, in the beginning of the 1970s, for the Probabilistic Safety Assessment, or PSA, of light water nuclear reactors described in the "Rasmussen Report", WASH-1400), automotive and weapons industries.

Fault Tree Analysis is best applied when:

- there are concerns regarding human safety, or perceived threats of important material losses, i.e. high risks;
- the system in question is complex and made of multiple elements;
- there are numerous potential contributors to a mishap and the causes of this one are not directly discernible.

This powerful technique has long been a staple of safety engineering and the safety profession and is often used as a design tool, which can help ensure that product performance safety objectives are met. Fault trees are particularly adept at representing and analyzing redundancy arrangements. In addition common cause events are easily handled in this type of approach (a common cause is an event or a phenomenon which, if it happens, will induce the concomitant failure of two or more other system elements).

Fault tree construction

FTA is a deductive analysis process that begins with the consideration of a critical undesirable event. This undesirable event at the system level is referred to as the *top event*. It generally represents a system failure mode or hazard for which the occurrence probability is not directly available, but required. The principle is that if there is such a critical failure mode, then all possible ways that mode could occur must be discovered and analyzed in a systematic way.

Top events represent potential high-penalty losses. Top events must not be too broad in scope; narrowing the scope reduces the effort spent in the analysis by confining it to relevant considerations. Typical top events might be:

- loss of power supply;
- loss of compressed air supply;
- loss of minimum flow to heat exchanger;
- reactor loss of coolant;
- tank rupture;
- uncommanded ignition;
- circuit breaker does not open;
- fire, explosion, etc.

The identification of relevant top events, as well as the further fault tree construction and structuring (see below), can often greatly profit from prior Preliminary Hazard, Failure Mode and Effect, or Reliability Block Diagram analyses.

Based on a set of rules and logic symbols from probability theory and Boolean algebra, FTA then uses a top-down approach to generate a logic and graphical model that provides for both a qualitative description of the failure paths (combinations of equipment failures, dependent failures and human failures) and a quantitative evaluation of the top event occurrence probability. The approach consists in defining successive levels of subordinate failure events (*intermediate events*), each level going a step deeper in the explanation of the possible causes of the failures identified at the preceding level. The intermediate events at a given level are linked to the events at the immediately superior level by logical connective functions. This let us construct in a very systematic way a complete tree structure representing the various possible failure paths leading to the occurrence of the top event. When a contributing failure event does not need to be divided further, because its failure rate is known or readily available, or it is decided to limit further analysis of a subsystem for practical reasons, the corresponding branch of the tree structure is terminated with a *basic event*.

The basic event for a branch is termed a *primary fault event* if the corresponding subsystem failed because of a basic "internal" mode such as a structural fault for example. The basic event is considered to be a primary one if the failed subsystem has not been exposed to environmental or service stresses exceeding its design limits (e.g. leakage of a valve seal within its pressure rating).

If the subsystem is out of tolerance so that it fails because of operational, or environmental stresses exceeding its intended ratings, placed on it, the basic event is said to be a *secondary fault event*. It is in particular the case every time the failed element has been improperly designed, or selected, or installed, or calibrated for the application.

The standard *logic symbols* used in the construction of the fault tree are described below. Note that events and gates are *not* component parts of the system being analyzed, but symbols representing the logic of the analysis.

Table 6.3 ***Logical symbols most commonly used in the graphical representation of fault trees***

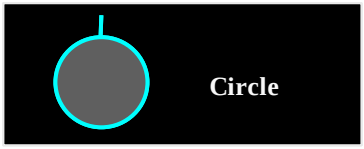
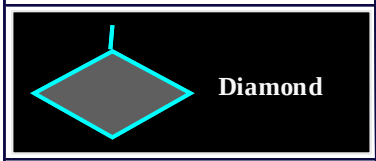
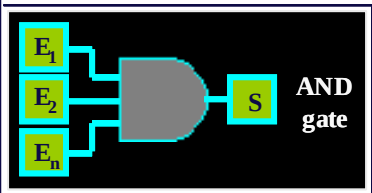
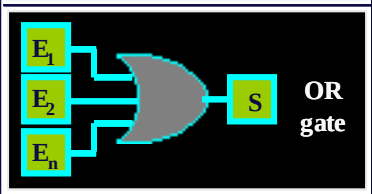
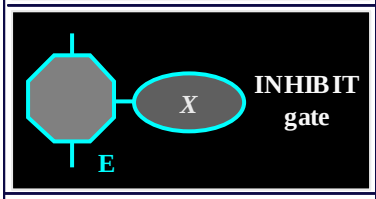
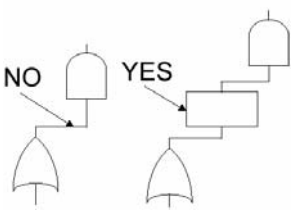
Event Symbols	
 **Rectangle**	***Top event*** : foreseeable, undesirable event, for which the occurrence probability is not directly available ***Intermediate event*** : describe a system state produced by "lower level" fault events
 **Circle**	***Basic "terminal" event*** : the basic event (assumed to be independent) marks the limit of resolution of the analysis
 **Diamond**	***Fault event not fully developed as its causes*** : it is only an assumed basic event
 **House**	***Event normally occurring in the operation of the system*** : it is not a fault event

Table 6.3 ***Logical symbols most commonly used in the graphical representation of fault trees (cont'd.)***

Logical Gate Symbols	
	The output event occurs if and only if all the inputs occur (Boolean intersection operation "∩" of the input events); all inputs, individually, must be necessary and sufficient to cause the input event
	The output event occurs if one or more of the inputs occur (Boolean union operation "∪" of the input events); any input, individually, must be necessary and sufficient to cause the input event
	Output exists when the input event E exists and the condition *X* is satisfied; this gate functions somewhat like an AND gate and is used for a secondary fault event E
Subtree Symbols	
	Triangle symbols provide a way to avoid repeating sections of a fault tree, or to transfer a subtree construction from one sheet to the next; the triangle-in appears at the bottom of a tree structure and represents that branch (subtree) of the tree (in the example: "α") shown someplace else
	Triangle-out appears at the top of a subtree and denotes that the corresponding tree structure ("α" in the example) is a subtree of a tree shown someplace else

Actual construction of fault trees (see summary in Fig. 6.13) requires thorough knowledge of how the system works and of the logic relationships in the system (interlocks, control interfaces, power supply feeds …). Fault tree construction remains however an art as well as a science and comes only through experience. It is nevertheless possible to give some conventions and rules that prove helpful in this construction process (adapted from [Lambert, 1973] and [McCormick, 1981])::

- Specify in the description of the top event the specific mission phase or portion of the mission to which it applies; this often helps to generate a very concise fault tree. Do not consider top events too broad in scope (see p. 108).
- At each level of the tree, the input events to a gate must be "*immediate*, *necessary* and *sufficient*" (INS) contributors to the output (upper level) event.
- Throughout the fault tree construction, systematically apply a consistent nomenclature to events. This is critical to identifying the same event in multiple fault tree branches. If, for example, a given event is named differently in another branch of the fault tree, cutset analysis (see below) identifies multiple events leading to different failures, rather than the same event leading to different failures. Such a nomenclature error can hide the fact that the event in question is a major contributor to the top event and thereby improvements or controls for it will fail to be recommended by the analyst.



Similarly, when two identical components are installed in different locations within a system, they must be identified as physically different components by using distinct designators in the nomenclature. Otherwise, cutset analysis identifies how the same component-type failure contributes to several scenarios when the failures are actually caused by different components.

- Test the type of the fault event. If it is a "state-of-component" statement, always use an OR gate. If it is a "state-of-system" statement, either AND, OR or INHIBIT gates may be used.
- Do not let gates feed gates (no "gate-to-gate" relationships), i.e., put an event statement between any two gates.
- Complete the gates first, i.e. identify *all* the input events of a logical gate before to start the detailed analysis of one of them.
- Consider that "causes are always anterior to consequences", this allows to eliminate certain causes and branches in order to eliminate from the tree any so-called "looped-systems".
- Do not expect miracles to "save" the system". Those things that would normally occur as the result of a fault will happen, and only those things!

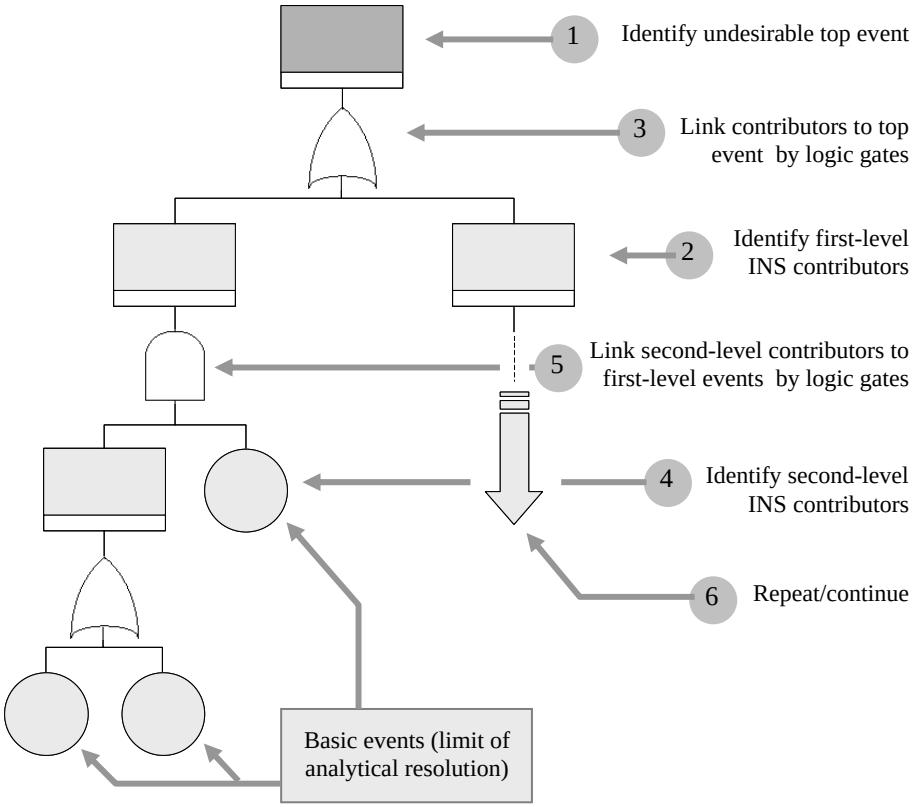


Figure 6.13 ***Step-by-step fault tree construction***

At a given level, under a given gate, each fault event must be independent of all others. However, the same fault event may appear at other points on the tree.

Fault tree evaluation

Once all failures, events, and conditions that can lead to the occurrence of the top event have been properly identified, the resulting fault tree can be "translated" into a *Boolean algebraic expression*. For each gate, the input events (intermediate or basic) are the independent variables, and the output event (intermediate or top) is the dependent variable. Using the rules of Boolean algebra (see section 2.3), these equations can then be solved so that the top event is expressed in terms of *minimal cut sets* (see p. 100) that involve only basic events.

A cut set is *any* group of basic events that, if *all* occur, will cause the top event to occur. A minimal cut set is a *least* group of basic events that, if all occur, will cause the top event to occur. A cutset can be a single-point failure or event, or can be a set of many events. Different cutsets can include different combinations of the same event. In large trees, the events that cause the top event to occur are often buried deep within the system and are not easily discovered without performing a cutset analysis. Generally (but not necessarily), the cutsets that have the highest probability of occurrence are the ones that are made of the fewest number of events.

The traditional cut set analysis process is to obtain a reduced expression made of the logical union of groups of events linked by AND logical connectors. By definition, these groups are the minimal cutsets looked for (because the simultaneous realization of each of the events of anyone of these group is a necessary and sufficient condition to cause the top event to occur).

Example (from [Villemeur, 1988]): find the minimal cut sets of the fault tree shown on the left part of figure 6.14 and represent it in its reduced form.

The Boolean expression of the top event in the original fault tree takes the form:

$$T = E_1 \cap E_2 = (A \cup E_3) \cap (C \cup E_4) = (A \cup B \cup C) \cap \{C \cup (A \cap B)\}$$

This expression can be simplified using the rules of the Boolean Algebra to give:

$$\begin{aligned} T &= (A \cup B \cup C) \cap \{C \cup (A \cap B)\} \\ &= (A \cap C) \cup (B \cap C) \cup C \cup \underbrace{(A \cap B) \cup (A \cap B)}_{A \cap B} \cup (A \cap B \cap C) \\ &= C \cap \{\underbrace{\Omega \cup A \cup B \cup (A \cap B)}_{\Omega}\} \cup (A \cap B) = C \cup (A \cap B) \end{aligned}$$

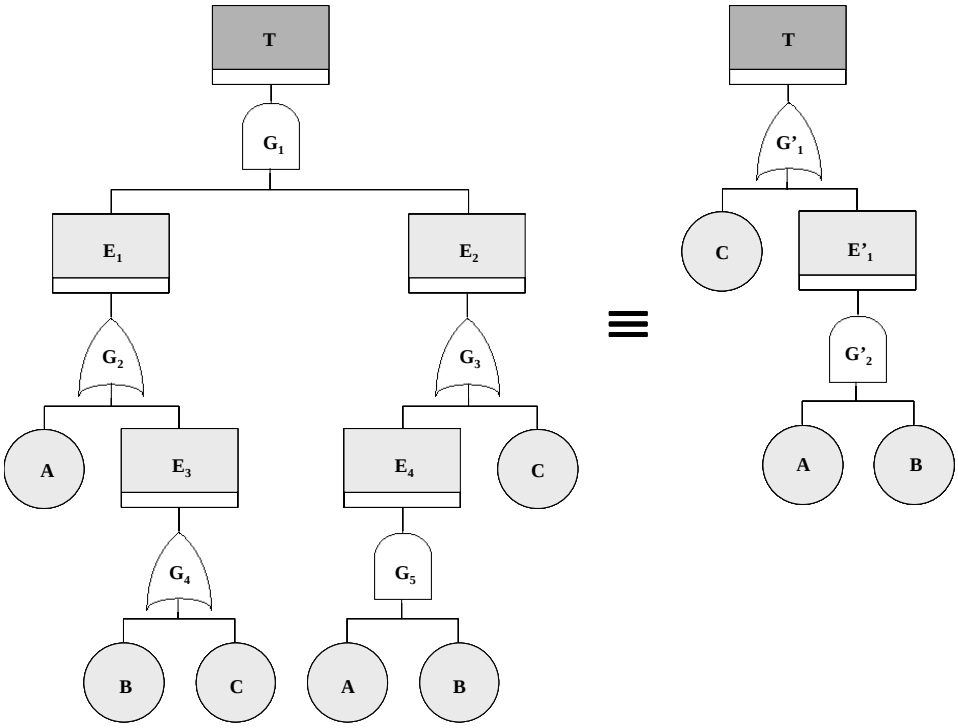


Figure 6.14 ***Example of fault tree reduction process***

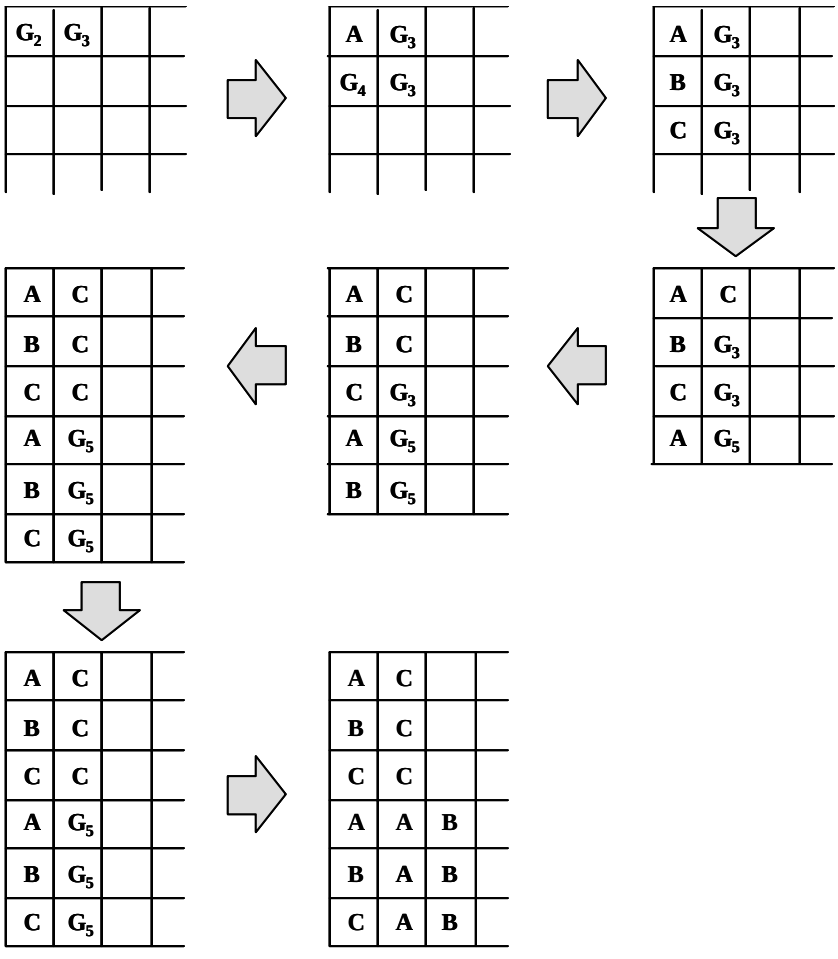
There are thus two minimal cutsets in this example: C (order 1) and A ∩ B (order 2). This means that the original fault tree can be reduced to the much simpler tree structure given in the right part of figure 6.14.

There is a more "mechanical" way to find the minimal cutsets, one that does not make *explicit* reference to the rules of Boolean Algebra [Clemens, 2002]. This process is explained step-by-step below, with the case study of figure 6.14 as practical application.

1. Ignore all tree elements except the gates and the basic events.
2. Proceeding stepwise from top event downward, construct a matrix using the gates and basic events names. The name of the top event gate becomes the initial matrix entry.

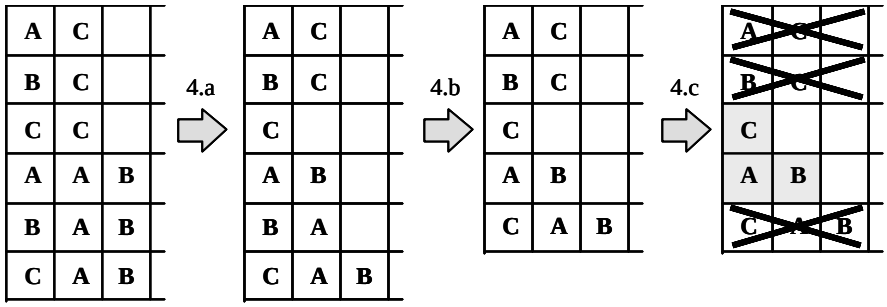
G_1			

3. As the construction progresses:
   - replace the name of each AND gate by the name of all gates/basic events that are its inputs; display these horizontally, in matrix row;
   - replace the name of each OR gate by the names of all gates/basic events that are its inputs; display these vertically, in matrix column; each newly formed OR gate replacement row must also contain all other entries already found in the original parent row;



4. A final matrix results, containing only names representing the basic events. Each row of this matrix is a Boolean indicated cut set. By inspection, then:
   a) eliminate redundant elements within rows;
   b) eliminate rows that duplicate other rows
   c) eliminate any row that contains all elements already found in a lesser (inferior cut set order) row.

   The rows that remain correspond to the minimal cut sets.



Quantitative analysis

The minimal cutset information obtained during qualitative analysis, together with information about the probability of occurrence of the basic events, can finally be used during quantitative analysis for computing the unavailability or unreliability values of the system.

Assuming that the probability of occurrence of the events A, B, C in the above example are respectively: 0.01, 0.02 and 0.0001, the probability of occurrence of the top event T becomes:

$$P[T] = 0.0001 + (0.01*0.02) - [0.0001*(0.01*0.02)] \cong 0.0003$$

More generally, if we note MCS_i ($1 \le i \le m$) the different minimal cut sets of the analyzed system, the probability of occurrence of the top event (T) is by definition given by:

$$P(T) = P(MCS_1 \cup MCS_2 \cup \ldots \cup MCS_m) \qquad [6.49]$$

Using Poincaré's theorem (Eq. [2.79]), this expression becomes:

$$\begin{aligned} P(T) = & \sum_{i=1}^{m} P(MCS_i) - \sum_{j=2}^{m}\sum_{i=1}^{j-1} P(MCS_i \cap MCS_j) \\ & + \sum_{k=3}^{m}\sum_{j=2}^{k-1}\sum_{i=1}^{j-1} P(MCS_i \cap MCS_j \cap MCS_k) \\ & - \ldots + (-1)^m \cdot P(MCS_1 \cap MCS_2 \cap \ldots \cap MCS_m) \end{aligned} \qquad [6.50]$$

In practice, because the failure probabilities, and thus the minimal cut set probabilities, are normally (very) small, it is generally suitable to approximate P(T) by the first term of the right-hand side of Eq. [6.50] only:

$$P(T) \cong \sum_{i=1}^{m} P(MCS_i) \qquad [6.51]$$

We know moreover that the following inequality is verified in this case and gives the bounds of the error margin on the above result:

$$\sum_{i=1}^{m} P(MCS_i) - \sum_{j=2}^{m}\sum_{i=1}^{j-1} P(MCS_i \cap MCS_j) \le P(T) \le \sum_{i=1}^{m} P(MCS_i) \qquad [6.52]$$

The evaluation of the probability of each minimal cutest depends on the characteristics of the basic events. Generally, the probabilities of these events correspond to unavailabilities of components ($\overline{A}_l(t) = 1 - A_l(t)$). It is moreover in principle assumed that the basic events intervening in a given minimal cut set are independent. In these conditions:

$$P(MCS_i) = P\left(\overline{A}_1^i \cap \overline{A}_2^i \cap ... \cap \overline{A}_{m_i}^i\right) = \prod_{l=1}^{m_i} \overline{A}_l^i(t) \qquad [6.53]$$

The calculation of the unavailability of irreparable as well as repairable components has been tackled in section 3.2.

Common cause failures

Contrary to the assumption made above, within industrial systems it is often the case that the failure of one component directly affects the operation of other components. Oversight of such *common cause failures* is a frequently found fault tree flaw. It should be noted however that it is not always straightforward to identify dependent failure events. This leads to major uncertainties in qualitative fault tree analysis, which of course also affect quantitative evaluations; generally, there is a lack of data of sufficiently good quality.

Generally speaking, dependent failures can have various causes:

- *Common cause initiating events*: supply outages (electricity, steam, cooling water, pneumatic pressure), natural disastrous events (fire, flood, earthquake), man-made disruptive events (explosion, electromagnetic disturbance).
- *Intersystem or intercomponent dependencies*: failure of a system/component that induces failure of another system/component.
- *Functional dependencies*: dependencies due to process coupling, either direct (output of one device constitutes an input to another), or indirect (functional requirements of one device on the state of another).
- *Shared-equipment dependencies*: common components or supply for several subsystems.
- *Physical interactions*: common causes of failures that are not events but e.g. unfavorable environmental conditions (freezing, overheat, humidity).
- *Human-interaction dependencies*: anthropogenic actions or behavior that happen during maintenance/operation.

Ignoring such dependencies can lead to highly optimistic results in safety analysis.

Example: the probability of simultaneous failures of two *independent* components A and B, each having a failure probability of 10^{-3}, is:

$$P(A \cap B) = 10^{-3} \cdot 10^{-3} = 10^{-6}$$

But if the failure of B is forced by the failure of A (100% dependency), then this probability rises to:

$$P(A \cap B) = P(A) \cdot P(B|A) = 10^{-3} \cdot 1 = 10^{-3}$$

One way of tackling the common cause failure problem is to use the so-called *implicit β-Factor Model*. The β-factor is defined as follows:

$$\beta = \frac{\text{Number of dependent failures}}{\text{Total number of failures}} = \frac{Q_n}{Q_1 + Q_n} = \frac{Q_n}{Q_t} \qquad [6.54]$$

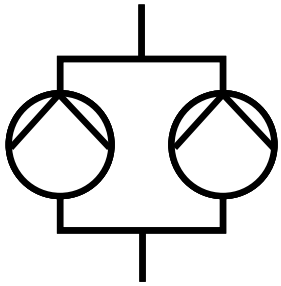
where Q_1 represents the probability of independent failure of a component *i*, and Q_n represents the probability that *n* components failed coincidentally.

If β=0, the events are totally independent; if β=1, they are fully dependent.

From equation [6.54]. we deduce that:

$$Q_n = \beta \cdot Q_t = Q_t - Q_1 \Rightarrow Q_1 = (1-\beta) \cdot Q_t \qquad [6.55]$$

Example: consider the case of a system made of two identical pumps (#1 and #2) having a total failure probability per pump $Q_t = Q_1 + Q_n = 0.01$. If during a given time interval 96 independent failures and 4 dependent failures have been observed, then:



$$\beta = \frac{4}{96 + 4} = 0.04$$

The system failure probability is given by:

$$\begin{aligned} \overline{R}_{sys} &= \text{P(independent failures of the two pumps)} + \text{P(dependent failures)} \\ &= (Q_1)^2 + Q_{n=2} = [(1-\beta) \cdot Q_t]^2 + \beta \cdot Q_t = 9.216\ 10^{-5} + 4\ 10^{-4} \cong 5\ 10^{-5} \end{aligned}$$

The exclusive use of such implicit models is however not recommended. It can result in a poorly detailed fault tree analysis of a technical system. The implicit approach should not be considered as a substitute of a detailed and explicit modeling of the dependencies when feasible.

Let us see on an example how such an explicit modeling of the dependencies within a system can be carried out. The example is that of a detector/alarm system intended to prevent hostile intrusion in a nuclear power plant. To this end, four wholly independent alarm systems are provided to detect and alarm about intrusion. A first analysis of the possible causes of the top event would thus lead to the following simple fault tree (Fig. 6.15).

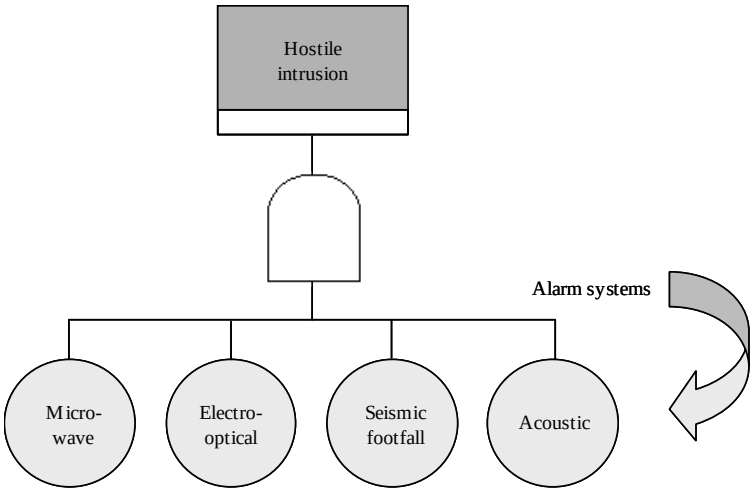


Figure 6.15 ***Initial fault tree for an intrusion detector/alarm system*** **(from [Clemens, 2002])**

As no two of the alarm systems share a common operating principle, redundancy appears at first sight to be absolute and the single AND gate to the top event seems appropriate.

But let us suppose now that these four alarm systems share a single source of power, with some emergency backup power system ready to take over should the former fails.

Because failures of both power sources are events that will disable *all* four alarm systems, they should not be considered as independent INS contributors to each of the four (provisory) basic events considered in figure 6.15.Power failure should rather be recognized as a new potential INS contributor to the considered top event, at the same level (and not at a lower level) as the primary failures of the four alarm systems, as shown in figure 6.16.

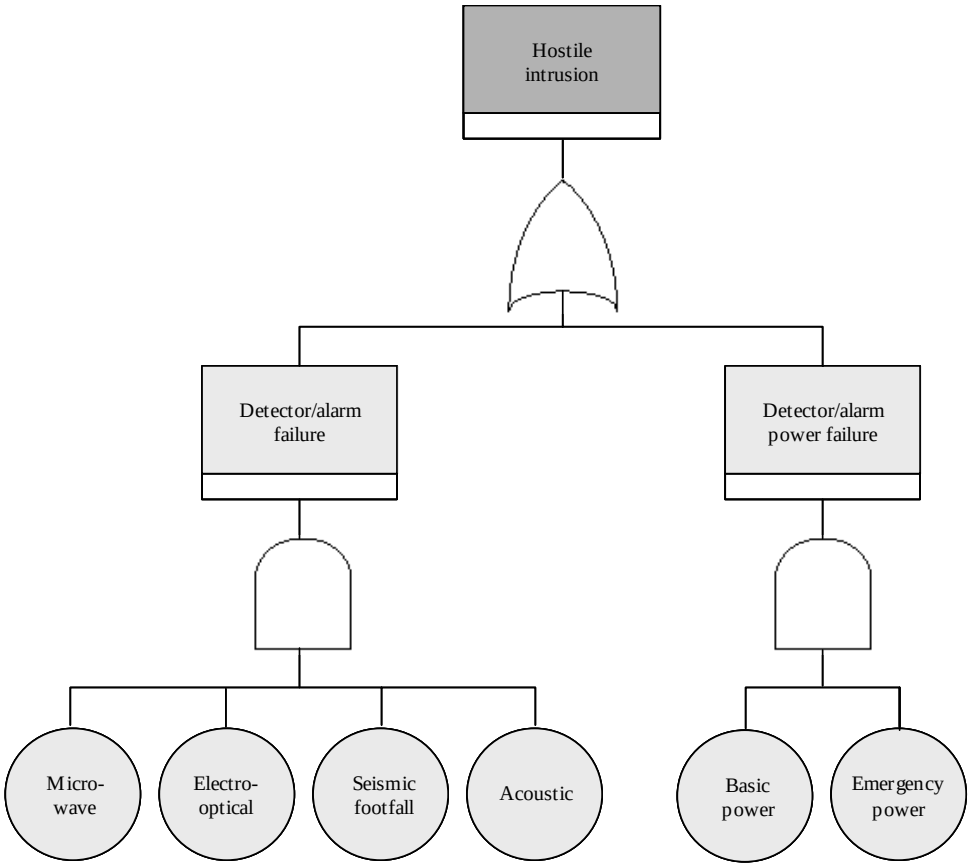


Figure 6.16 ***Intrusion detector/alarm system fault tree with the common cause event "power failure" accounted for***
(from [Clemens, 2002])

The search for common cause failures should be conducted systematically in the framework of a FTA and such failures as far as possible explicitly accounted for in the fault tree structure.

Example of FTA practical application

As an example of practical application of the Fault Tree Analysis, we will consider the case of the reliability analysis of transmission line protective systems [Schweitzer and Anderson, 1998]. Transmission line protective systems can be very complex, incorporating many different equipment groups, often at widely separated places and often requiring high-speed communications for proper operation.

Figure 6.17 shows a transmission line corresponding to a POTT (Permissive Over-reaching Transfer Trip) scheme. This transmission line is equipped with a single circuit breaker and redundant relays at each end. The relays communicate through tone equipment and analog microwave gear. The protection subsystems operate from 125Vdc batteries, whereas the communication subsystems operate from 48Vdc batteries.

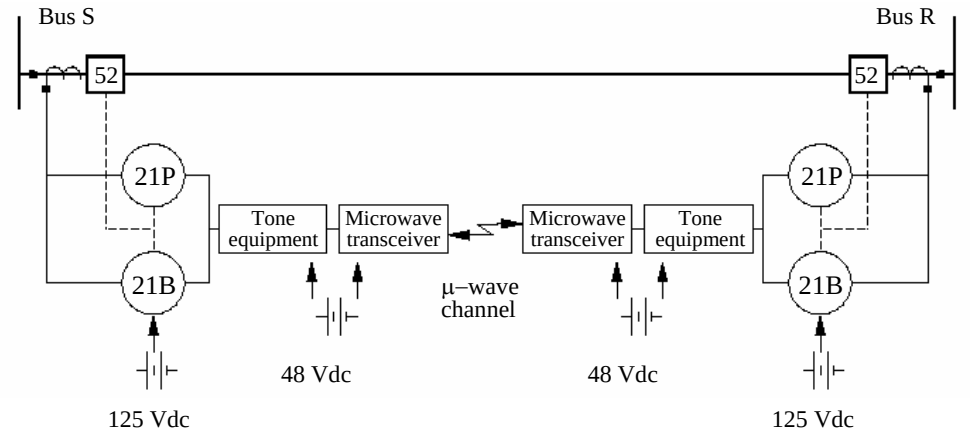


Figure 6.17 ***POTT scheme with redundant relays and single channel microwave*** **(from [Schweitzer and Andersons, 1998])**

To construct a fault tree for this system, the first step is to choose a top event of interest. Let us take here as top event: "Protection fails to clear in-section fault in the prescribed time".

Proceeding level-by-level, we then identify on the basis of he above scheme all the events that may directly or indirectly contribute to the occurrence of this undesirable top event and link them with appropriate logical connectors (gates). The resulting tree is given in figure 6.18.

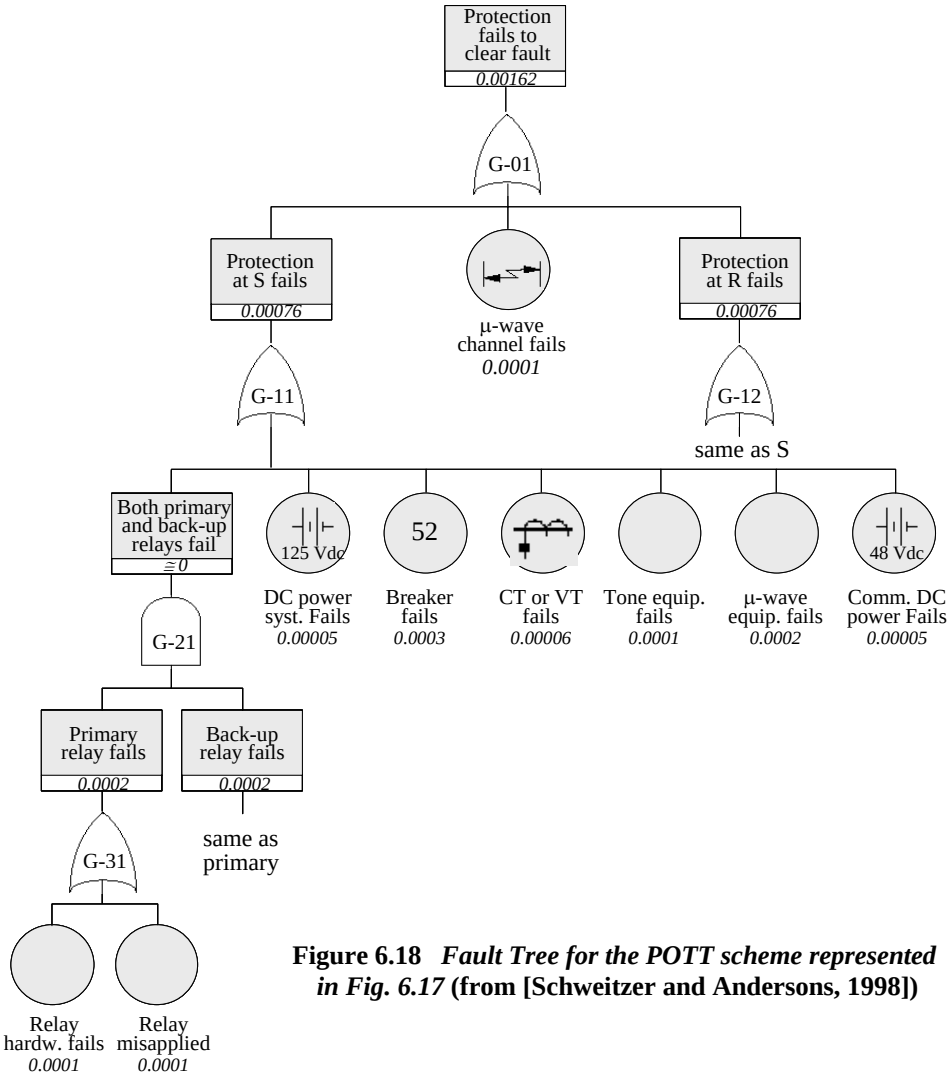


Figure 6.18 ***Fault Tree for the POTT scheme represented in Fig. 6.17*** **(from [Schweitzer and Andersons, 1998])**

In the above fault tree construction, it has been assumed that the failures of the primary and back-up relays are independent (a failure in one relay does not affect the other relay). On the other hand, other possible common cause failures (e.g common DC supplies) have been explicitly accounted for under the G-11 gate.

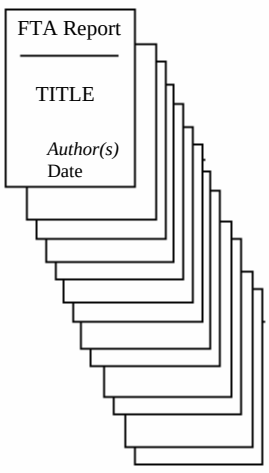
The devices unavailability data required to carry out the quantitative analysis of the investigated system are given in italic in figure 6.18, under the corresponding basic event symbols. Note that it is not very useful in this particular example to search beforehand for the minimal cutsets, because the resulting "reduced" tree would here not be really simpler that the original fault tree. Because the device unavailabilities are small, the global unavailability of the considered protection system can easily be calculated manually using the rare event approximation (i.e. calculating the unavailability associated with the output of an OR gate as simply the sum of the unavailability for each input of this gate). The unavailability of this protection system to clear transmission line faults is thus $1.62\ 10^{-3}$. The reader can easily verify that the fact that the system has redundant relays indeed reduces its unavailability by over 20% compared to a system with a single relay ($1.62\ 10^{-3}$, instead of $2.02\ 10^{-3}$).

For systems that are more complex than the above example, computer programs are available to assist in developing and analyzing fault trees.

FTA final report

Generally speaking, the main results produced by a Fault Tree Analysis are the following:

- graphic display of chains of events/conditions leading to the undesirable event(s);
- improved understanding of the system and of the system behavior;
- identification of those potential contributors to failures that are critical; the corresponding components may need testing or more rigorous quality assurance;
- identification of the root causes of equipment failures;
- qualitative/quantitative insight into probability of the undesirable event selected for the analysis;
- identification of resources committed to preventing failure; this can provide guidance for redeploying resources to optimize control of risk.



These and other pertinent information should be thoroughly documented in the FTA final report. This one should typically include the following headings:

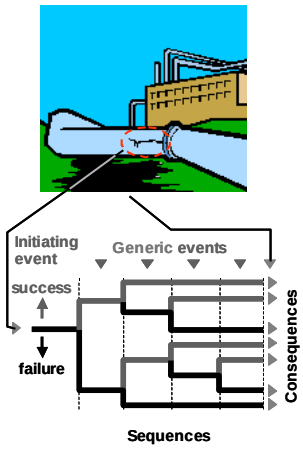
- *Executive summary*: an abstract of the content of the complete report.
- *Scope of the analysis* (say what is analyzed and what is not analyzed): a brief description of the system, as well as of the system and analysis boundaries (physical boundaries, operational boundaries, operational phases, interfaces treated, resolution limit, exposure interval, etc.).
- *The analysis* (show trees as figures, include data sources, cut sets, etc. as tables): discussion of method and software used, sources of probability data, common cause search, sensitivity test(s), cutsets and minimal cutsets.
- *Findings*: occurrence probability, reliability or availability of the top event, comments on system vulnerability, candidate reduction measures and/or actions.
- *Conclusions and recommendations*: risk comparisons; is further analysis needed? by what method?

Strengths and limitations of FTA

FTAs greatly help identifying possible system reliability or safety problems at design time, or assessing system reliability or safety during operation. FTAs identify the causes of *single point* failures. FTAs can be used in diagnostic work for a system failure. FTAs complement FMEAs keying in on the worst identified failure modes. The limitations of FTA are related to the fact that undesirable events must be foreseen and are only analyzed singly, that each fault/failure must be constrained to two conditional modes only when modeled in the tree, and finally that there is a risk of overlooking common causes.

6.4 Event Tree Analysis (ETA)

General presentation of the method



ETA is an inductive, of forward logic, technique that is meant to trace the development or escalation of a potentially hazardous accident, failure or other unwanted event (*initiating event*) and assess its foreseeable consequences. Such events disrupt normal system operation or condition. The method is based on a graphical representation that provides a convenient way to systematically explore all the sequences of subsequent events (*generic events*) resulting from the success or failure of controlling or mitigating systems and procedures. These include both specific accident mitigating and normal operating actions; they can be either system actions or operator actions. ETA is very useful to identify possible *accident scenarios.*

As the graphical representation develops by taking into account successive generic events, the ETA picture fans out like the branches of a tree. Generally, different event trees (corresponding to different initiating events) must be constructed and evaluated in the framework of the risk analysis of a given system Event trees are similar to fault trees but difference is that they are used to examine the possible *consequences* of the initiating events and not the *causes* of a top event. For this reason, Event Tree Analysis is also sometimes referred to as *Consequence Analysis.* It is moreover nothing but an adaptation of the more general *Decision Tree Method* that is widely used in business and economic analyzes.

The ETA methodology has changed very little since the conception of the technique back in the 1960's when it was successfully used in the WASH 1400 study [WASH 1400, 1975]. It has since then been used in risk analyses of a wide range of technological systems and is now a natural part of most probabilistic risk assessment studies.

The ETA approach can be used in the detailed design phase of a proposed system or plant, during its operational phases or prior to decommissioning. It is often used to evaluate the effectiveness of safeguards to prevent a failure from becoming an undesired event and to allow decisions on the necessity for existing or additional safeguards. A risk analyst may apply this technique in particular when a structure can partially fail and function (although at a reduced level) at the same time. Example: a pumping-engine with two pumps can still produce flow when one of the pumps does not work. However, it does not produce to its rated capacity. It fails, but remains at least partially operating. The risk analyst cannot use fault tree analysis in this situation, since FTA only considers total failure of a structure.

ETA is applicable to systems in which all components are continuously operating, or to systems in which some or all of the components are in standby mode – those that involve sequential operational logic and switching. In the case of continuously operated systems, the events to consider can occur (i.e., components success or failure) in any arbitrary order. In the event tree analysis, such components can be introduced in any order since they do not operate chronologically with respect to each other.

Event tree construction

A general block diagram of the different steps to be followed in constructing an event tree is given in figure 6.19.

An Event Tree Analysis starts by identifying possible initiating events, i.e. events that may give rise to unwanted damaging consequences. The identification of the initiating events can be based on experience, on a technical or scientific preliminary analysis of the system under scrutiny, or on the construction of a fault tree having as top event some general undesirable event considered at the level of the whole system.

Initiating events are “anticipated”. They are events that designers have put in physical barriers, systems, procedures alarms, etc., which are meant to respond to the upset, to terminate the sequence or to mitigate the consequences of the accident.

To be of interest for further analysis, an initiating event must give rise to a number of consequence sequences. If the initiating event induces only one consequence sequence, FTA is “a priori” a more suitable technique to analyze the problem.

The selection of relevant initiating events is very important for the analysis, but this can be done in different ways; various analysts may define slightly different initiating events.

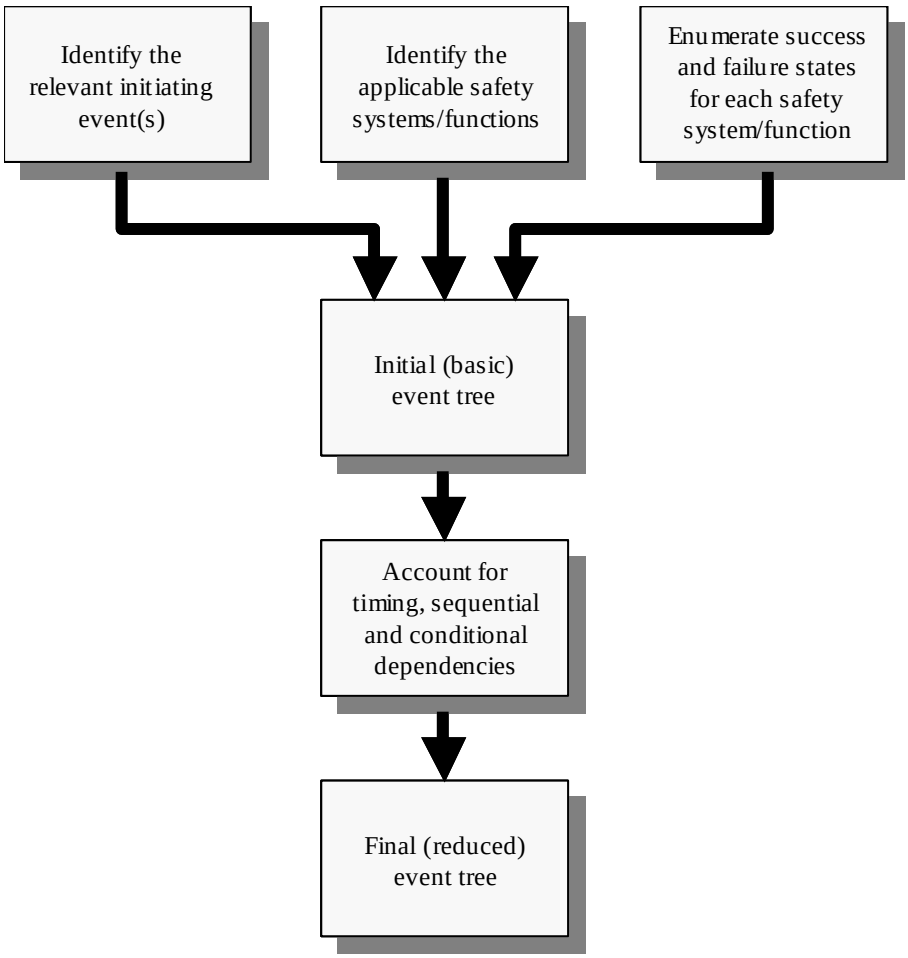


Figure 6.19 ***Step-by-step process for constructing an event tree*** **(from [Erdmann, 1979])**

After an initiating event has been selected, all the safety subsystems/functions (specific safeguard systems or conditions) that can possibly intervene following the occurrence of the selected initiating event and prevent an undesirable outcome must in their turn be identified.

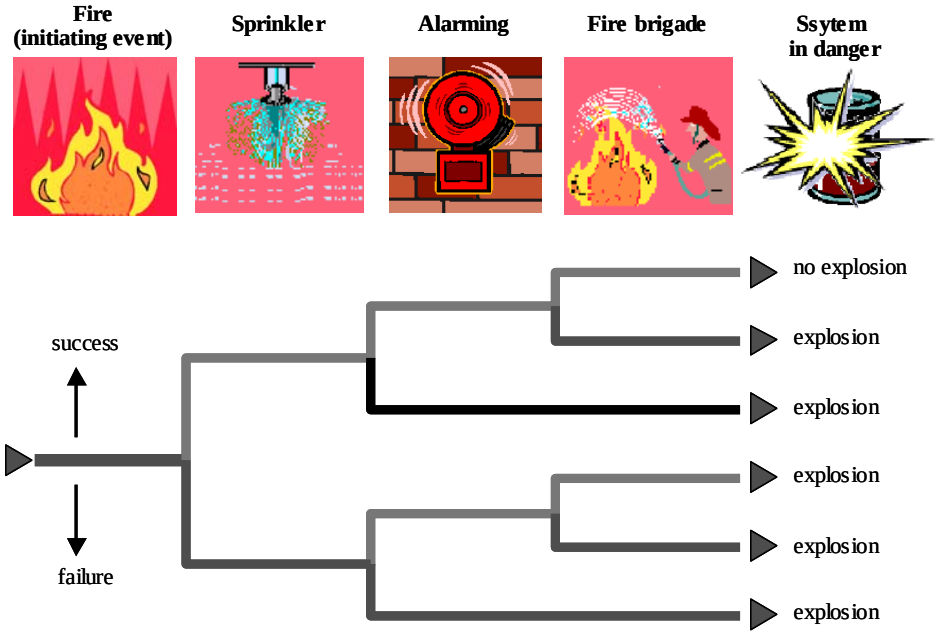
The safety subsystems/functions that respond to a given initiating event may be thought of as the system’s defense against the potential unacceptable consequences of the initiating event. These safety subsystems/functions can be classified in various specific categories.

These categories may for example be [Rausand, 1999]:

- Safety systems that automatically respond to the initiating event (e.g. automatic shutdown systems).
- Alarms that alert the operator(s) when the initiating event occurs (e.g. fire alarm systems).
- Operator procedures following an alarm.
- Barriers or containment methods that are intended to limit the effects of he initiating event.

The fate (success or failure, which could be total or partial see below) of these subsystems/functions is then examined to determine the sequences of generic events that can lead to unacceptable consequences.

The diagram is usually drawn from left to right, starting from the initiating event (see Fig. 6.20). The branch points in the event tree are called *nodes*, and are formulated either as an event description or as a question regarding actions that may be taken. The development is continued to the resulting consequences.



Fire in the storehouse may cause the explosion of fuel containers. For the time gap until the fire brigade arrives, the containers must be cooled down; this is normally achieve by a sprinkler system. Alarm is triggered automatically to alert the fire brigade.

Figure 6.20 ***A simple example of event tree for a fire in a fuel containers storehouse***

Usually, only a two-state modeling (binary branching logic: complete success or complete failure, “yes” or “no”, “go” or “no go”) is considered. In some cases, it could be necessary to introduce a greater number of discrete states (e.g. partial failure states); a separate branch must then be included for each additional state.

The last step in the qualitative part of the ETA is to describe the different event sequences arising from the initiating event. One or more of the sequences may represent a safe recovery or an orderly shutdown of the system (see Fig. 6.20).

From a safety viewpoint, the sequences of importance are however those that result in accidents. When these have been defined, the analyst may rank them according to their criticality. This information, together with the structure of the diagram clearly showing the progression of the accident, helps specifying where additional procedures or safety systems will be most effective in protecting against unacceptable consequences.

It could prove beneficial in some cases to split the outcomes (end consequences) of the event tree into various consequence categories as illustrated in figure 6.21.

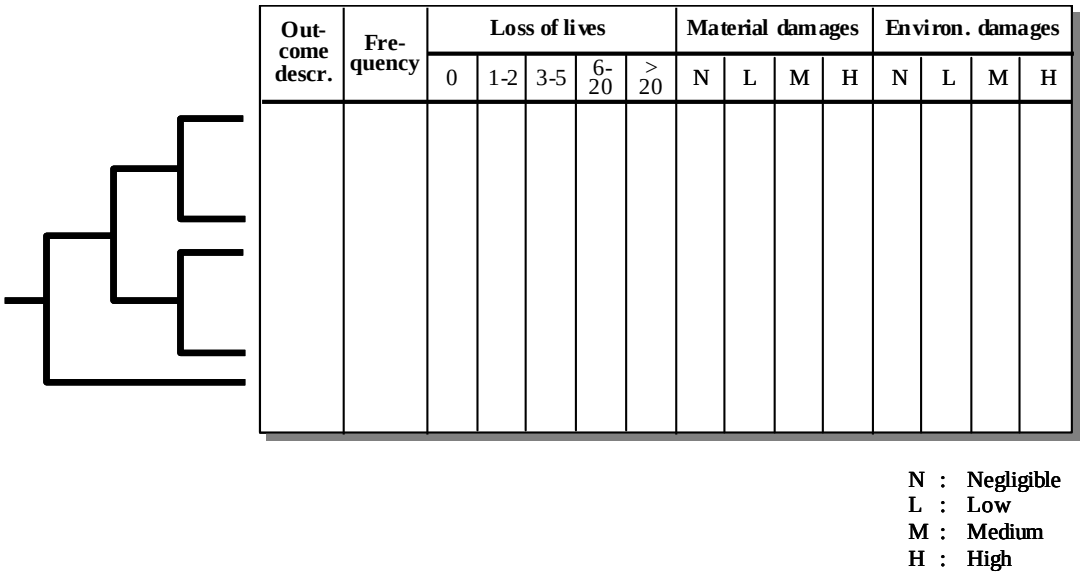


Figure 6.21 ***Split presentation of the event tree outcomes (probability distribution over the subcategories)*** **[Rausand, 1999]**

The tree evaluation has normally for final goal the quantification of the sequences in order to be able to predict the frequency, or probability of occurrence, of each of the resulting consequences (or at least of those leading to undesirable consequences).

Event tree reduction process

Prior to this operation, the initial (basic) tree must be reduced to its most elementary form. The reduction process in fact already takes place throughout the construction phase of the event tree. Two factors assist in simplifying the tree structure: timing, and functional interactions. Taking the time into account allows considering only one well-defined arrangement of the generic events, which greatly reduces the number of sequences. For example, there are 2^4 = 16 sequences to study for a well-ordered binary tree of 4 generic events, but 4! times more, i.e. 384 sequences, if the generic events can “a priori” be arranged in every possible orders (see Fig. 6.22).

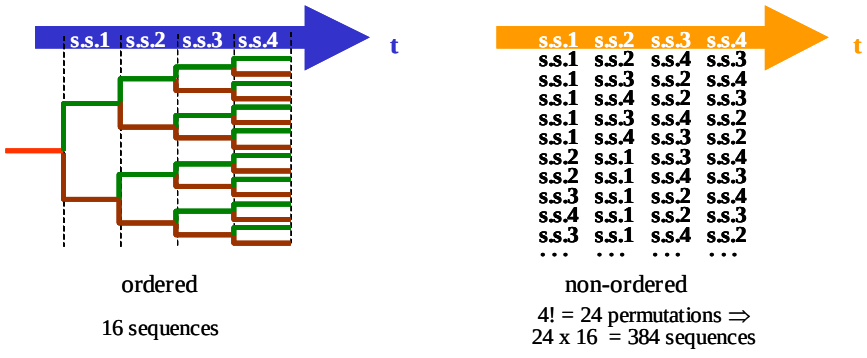


Figure 6.22 ***Event tree reduction by ordered timing of generic events***

The functional dependencies between events allow to "prune" an event tree by eliminating the branches that have a zero conditional probability. This has been done in the example of figure 6.20; if the alarm does not sound, coming after either a success or a failure of the sprinkler system, the consideration of a possible success of the following event "fire brigade" becomes pointless.

Event tree quantification

When the branch point (generic) events are independent of each other, quantification of the diagram is trivial and is simply achieved by calculating the product of the frequency of the initiating event with the probabilities of passing along each branch leading to a given outcome consequence (see example in Fig. 6.23).

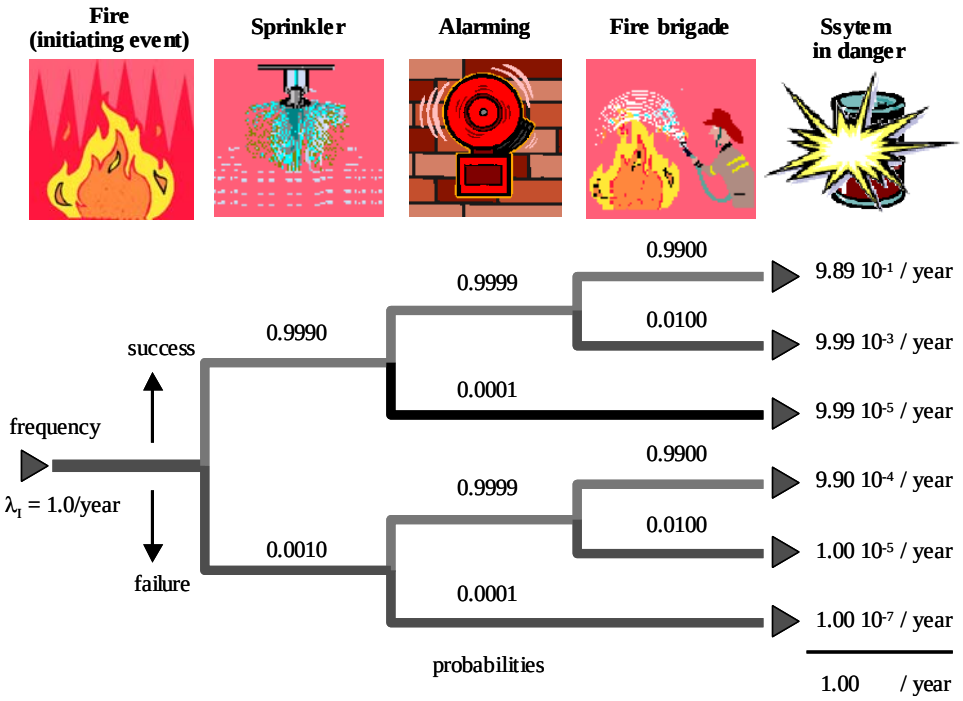


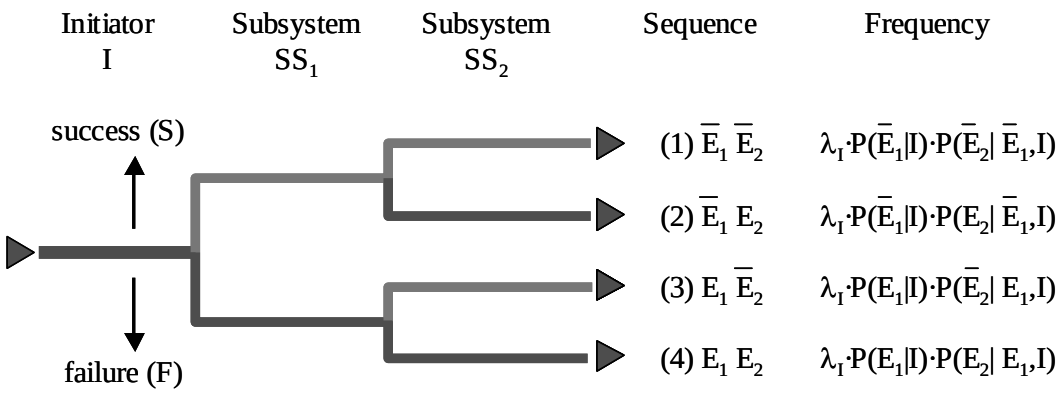
Figure 6.23 ***Quantification of the sequences of Fig. 6.20 assuming a total independency of the generic events***

In principle, however, the subsystem states on a given branch of the event tree are conditional on the previous states having already occurred (dependencies between the branch events). For example, in figure 6.23, the success or failure of the generic event "fire brigade" must be defined under the condition that the preceding events - "alarming", "sprinkler" and, of course, the initiating event I itself - had previously occurred (*conditional* probability, which in this particular case leads to a zero success probability should the alarm subsystem have previously failed). In the general case, we have therefore for a given sequence of genetic events $\{E_1 , E_2 , \ldots E_n\}$:

$$P(\text{event sequence}) = P(E_1 \mid I)\cdot P(E_2 \mid E_1, I)\cdot \ldots P(E_n \mid E_{n-1} \ldots E_1, I) \qquad [6.56]$$

The quantification of the probability of passing along different branch points of an event tree becomes more complex in the non-trivial situations when there are dependencies between the subsystems. The quantification is then performed by quantifying fault trees whose top events are defined as combination (through an AND gate) of occurrence and non-occurrence of the branch point events that have in turn been developed as fault tree structures.

To illustrate this use of fault trees in the event tree approach, we will consider the simple example given in figure 6.24. The failure mechanisms of the subsystems SS_1 and SS_2 are shown in figures 6.25a and 6.25b respectively.



E_i: "failure of subsystem SS_i" event

$\bar{E}_i$: "success of subsystem SS_i" event

Figure 6.24 ***Simple event tree example*** **(from [Andrews & Dunnett, 2000])**

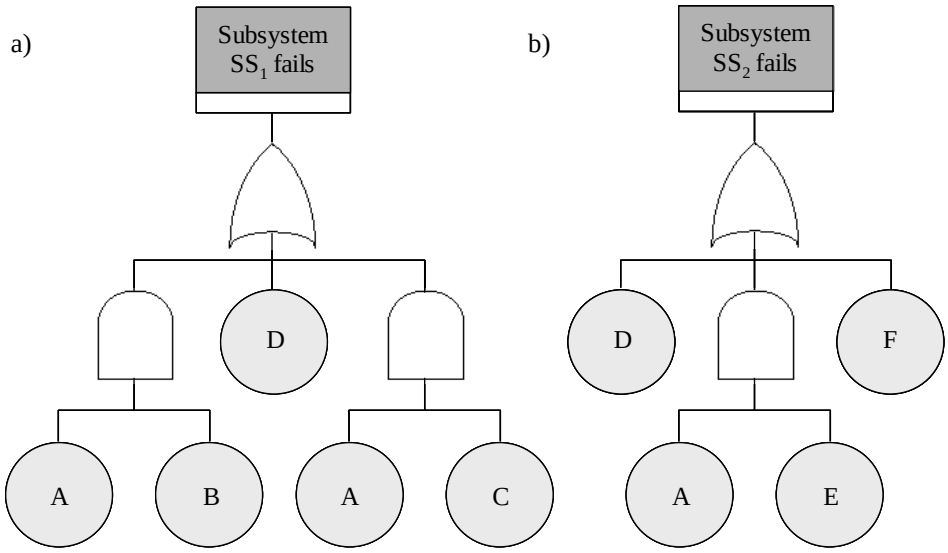
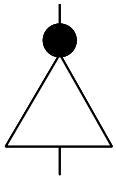


Figure 6.25 ***Subsystems SS_1 (a) and SS_2 (b) fault tress (see Fig. 6.24)***
(from [Andrews & Dunnett, 2000])

The basic events "A" and "D" occur in both subsystem fault trees; the subsystem failure events are thus not totally independent. Taking into account this "weak" dependency on some common basic events, the four possible outcomes corresponding to the sequences (1) to (4) described in figure 6.24 are represented by the fault tree structures given in figure 6.26. In trees (1) to (3), a new gate, the "NOT gate", has been introduced, which symbol is:



As it name implies, the "NOT gate" is used to indicate that the output occurs when the input event does not occur.

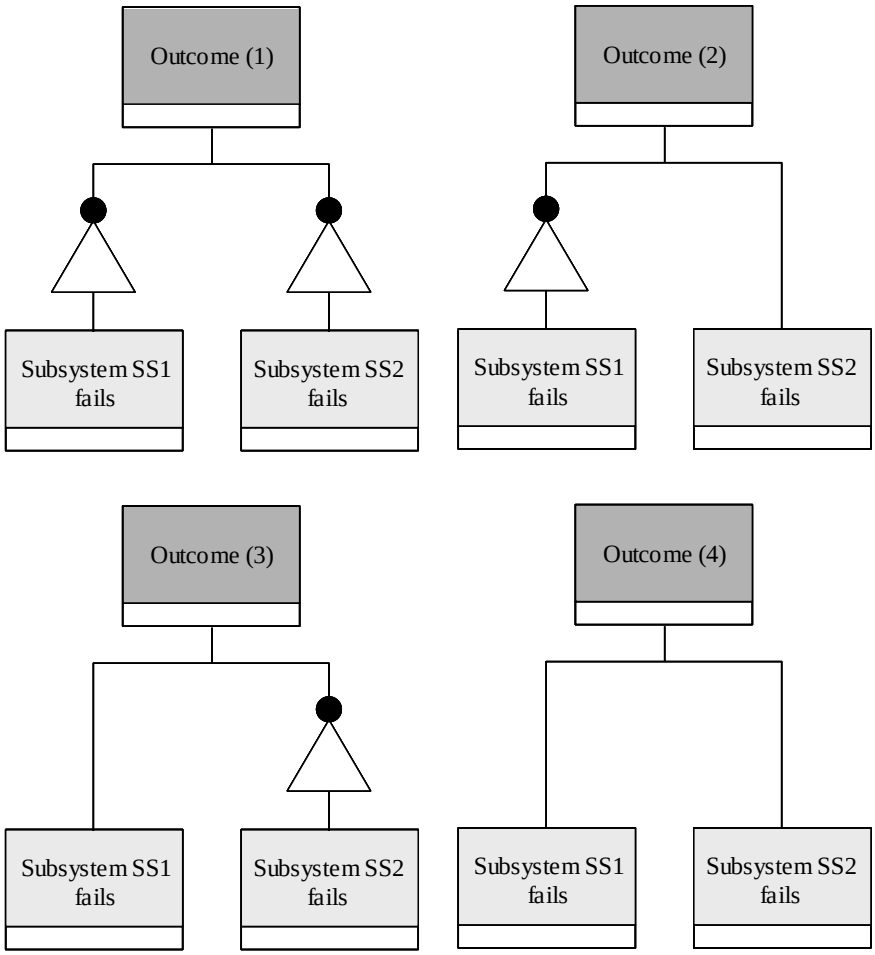


Figure 6.26 ***Fault trees representing the different outcomes described in Fig. 6.24***

The presence of "NOT gates" in the above diagrams has an important consequence. It may give rise to "*non-coherent*" fault trees, i.e. trees in which the non-occurrence of an event causes the top event to occur. Whereas the Boolean reduction of the logic function representing the top event leads to the identification of the *minimal cutsets* in the case of a coherent fault tree, the equivalent of these logic expressions are called *prime implicants* in the case of a non-coherent fault-tree:

- a *minimal cutest* is a combination of component *failure events* that are necessary and sufficient to cause the top event;
- a *prime implicant* is a combination of basic events – corresponding to *failures* or *successes* – that are necessary and sufficient to cause the top event.

A shown by the figures 6.24 and 6.26, evaluating the frequencies of the different event tree outcomes implies to know the probabilities of the events E_i ("subsystem SS_i fails") as well as $\overline{E}_i$ ("subsystem SS_i succeeds")

The Boolean expressions of the branch point events E_1 and E_2 (failure top events of coherent fault trees, see Fig. 6.25) are:

$$E_1 = (A \cap B) \cup (A \cap C) \cup D \quad \text{(minimal cutsets: } A \cap B,\ A \cap C,\ D) \qquad [6.57]$$

$$E_2 = (A \cap E) \cup D \cup F \quad \text{(minimal cutsets: } A \cap E,\ D,\ F) \qquad [6.58]$$

To deduce the expressions corresponding to the success of the concerned sub-systems, we have first to establish the *dual formulation* of the two fault trees presented in figure 6.25. Compared to the original formulation, in the dual formulation all AND gates are replaced by OR gates and vice-versa, moreover all failed component states become working component states (in pursuance of the Morgan's theorem, see section 2.3). The dual formulations of the fault trees of figure 6.25 therefore become:

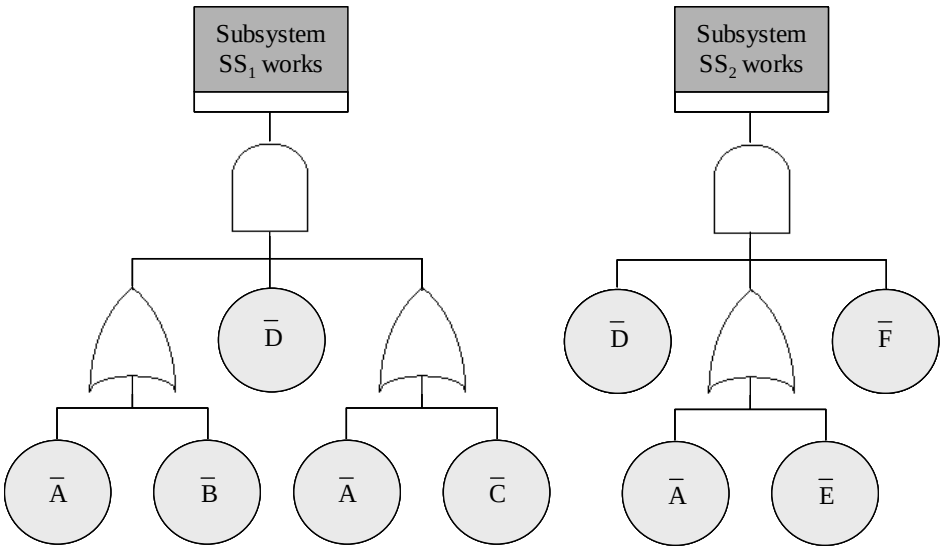


Figure 6.27 ***Dual formulations of the fault trees of Fig. 6.25 ("success trees")***

The Boolean expressions of the branch point events $\overline{E}_1$ and $\overline{E}_2$ (success top events of non-coherent trees) take respectively the forms:

$$\overline{E}_1 = \left(\overline{A} \cup \overline{B}\right) \cap \left(\overline{A} \cup \overline{C}\right) \cap \overline{D} = \left[\overline{A} \cup \left(\overline{B} \cap \overline{C}\right)\right] \cap \overline{D}$$
$$= \left(\overline{A} \cap \overline{D}\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D}\right)$$
(prime implicants: $\overline{A} \cap \overline{D}$, $\overline{B} \cap \overline{C} \cap \overline{D}$) [6.59]

$$\overline{E}_2 = \left(\overline{A} \cup \overline{E}\right) \cap \overline{D} \cap \overline{F} = \left(\overline{A} \cap \overline{D} \cap \overline{F}\right) \cup \left(\overline{D} \cap \overline{E} \cap \overline{F}\right)$$
(prime implicants: $\overline{A} \cap \overline{D} \cap \overline{F}$, $\overline{D} \cap \overline{E} \cap \overline{F}$) [6.60]

Using Eqs. [6.57] to [6.50], the Boolean expressions of the four sequences of our example event tree are thus given by:

$$T_{(1)} = \overline{E}_1 \cap \overline{E}_2$$
$$= \left[\left(\overline{A} \cap \overline{D}\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D}\right)\right] \cap \left[\left(\overline{A} \cap \overline{D} \cap \overline{F}\right) \cup \left(\overline{D} \cap \overline{E} \cap \overline{F}\right)\right]$$
$$= \left(\overline{A} \cap \overline{D} \cap \overline{F}\right) \cup \left(\overline{A} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \cup \left(\overline{A} \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{F}\right)$$
$$\cup \left(\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right)$$
$$= \left(\overline{A} \cap \overline{D} \cap \overline{F}\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \quad [6.61]$$

$$T_{(2)} = \overline{E}_1 \cap E_2$$
$$= \left[\left(\overline{A} \cap \overline{D}\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D}\right)\right] \cap \left[\left(A \cap E\right) \cup D \cup F\right]$$
$$= \left(\overline{A} \cap \overline{D} \cap F\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D} \cap F\right) \cup \left(A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E\right) \quad [6.62]$$

$$T_{(3)} = E_1 \cap \overline{E}_2$$
$$= [(A \cap B) \cup (A \cap C) \cup D] \cap [(\overline{A} \cap \overline{D} \cap \overline{F}) \cup (\overline{D} \cap \overline{E} \cap \overline{F})]$$
$$= (A \cap B \cap \overline{D} \cap \overline{E} \cap \overline{F}) \cup (A \cap C \cap \overline{D} \cap \overline{E} \cap \overline{F}) \quad [6.63]$$

$$T_{(4)} = E_1 \cap E_2$$
$$= [(A \cap B) \cup (A \cap C) \cup D] \cap [(A \cap E) \cup D \cup F]$$
$$= (A \cap B \cap F) \cup (A \cap B \cap E) \cup (A \cap C \cap F) \cup (A \cap C \cap E) \cup D \quad [6.64]$$

We are now in a position, using Poincaré's theorem ("inclusion-exclusion" expansion technique, see section 2.3), to calculate the frequencies of each of the possible outcomes of the example. To make possible a numerical comparison of these exactly computed frequencies with the result of some commonly used approximation we will assign a failure probability of 0.1 (not very realistic of course for usual technical systems) to each component and a frequency of 1.0 per year to the initiating event. The results of the exact calculations are then the following:

$$\lambda_{(1)} = \lambda_I \cdot P(T_{(1)})$$
$$= \lambda_I \cdot P\{(\overline{A} \cap \overline{D} \cap \overline{F}) \cup (\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F})\}$$
$$= \lambda_I \cdot \{P(\overline{A} \cap \overline{D} \cap \overline{F}) + P(\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F})$$
$$- P(\overline{A} \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F})\}$$
$$= 1.0/yr \cdot \{0.729 + 0.59049 - 0.531441\} = 0.788049/yr \quad [6.65]$$

$$\lambda_{(2)} = \lambda_I \cdot P(T_{(2)})$$
$$= \lambda_I \cdot P\{(\overline{A} \cap \overline{D} \cap F) \cup (\overline{B} \cap \overline{C} \cap \overline{D} \cap F) \cup (A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E)\}$$
$$= \lambda_I \cdot \{P(\overline{A} \cap \overline{D} \cap F) + P(\overline{B} \cap \overline{C} \cap \overline{D} \cap F) + P(A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E)$$
$$- P(\overline{A} \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap F) - P(\overline{A} \cap A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E \cap F)$$
$$- P(A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E \cap F) + P(\overline{A} \cap A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E \cap F)\}$$
$$= 1.0/yr \cdot \{0.081+0.0729+0.00729-0.06561-0-0.000729+0\}$$
$$= 0.094851/yr \quad [6.66]$$

$$\lambda_{(3)} = \lambda_I \cdot P(T_{(3)})$$
$$= \lambda_I \cdot P\{(A \cap B \cap \overline{D} \cap \overline{E} \cap \overline{F}) \cup (A \cap C \cap \overline{D} \cap \overline{E} \cap \overline{F})\}$$
$$= \lambda_I \cdot \{P(A \cap B \cap \overline{D} \cap \overline{E} \cap \overline{F}) + P(A \cap C \cap \overline{D} \cap \overline{E} \cap \overline{F})$$
$$- P(A \cap B \cap C \cap \overline{D} \cap \overline{E} \cap \overline{F})\}$$
$$= 1.0/yr \cdot \{0.00729+0.00729-0.000729\} = 0.013851/yr \quad [6.67]$$

$$\lambda_{(4)} = \lambda_I \cdot P(T_{(4)})$$
$$= \lambda_I \cdot P\{(A \cap B \cap F) \cup (A \cap B \cap E) \cup (A \cap C \cap F) \cup (A \cap C \cap E) \cup D\}$$
$$= \lambda_I \cdot \{P(A \cap B \cap F) + P(A \cap B \cap E) + P(A \cap C \cap F)$$
$$+ P(A \cap C \cap E) + P(D) - \ldots + P(A \cap B \cap C \cap D \cap E \cap F)\}$$
$$= 1.0/yr \cdot \{0.001+0.001+0.001+0.001+0.1 - \ldots +0.000001\}$$
$$= 0.103249/yr \quad [6.68]$$

Although a little tedious (specially when carried out "by hand") the process of exact calculations just described in the preceding pages is relatively easy to perform for a system as simple as the one considered above. When the system to be studied is much more complex however, leading to several thousands, perhaps hundreds of thousands, of minimal cut sets and/or prime implicants and very large fault trees, it is beyond the capability of even modern day computers to evaluate the full expansion in any reasonable time. Approximate ways of performing the sequence frequency calculations are therefore required in ad hoc computer programs.

When the probabilities of the operands linked by OR gates are small, as in the case of the minimal cutsets of coherent fault trees (the probability of each minimal cutest being the product of basic event probabilities much smaller than one in principle), the series expansion resulting from the Poincaré's theorem can be validly truncated after the first one or two terms ("rare events approximation"). Alternatively, the following upper bound inequality (this inequality becoming equality when the minimal cutsets are independent, see Eq.[2.80]) can also be used to approximate the results of the expansion:

$$P(T) \leq 1 - \prod_{i=1}^{n} (1 - C_i) \qquad [6.69]$$

where the C_i's represent the minimal cutset probabilities.

For non-coherent fault trees however, the convergence of the inclusion-exclusion expansion can be very slow, requiring the evaluation of a large number of terms. This can prove excessively time-consuming for large fault trees and thus unfeasible in practice. For this reason, many commercial computer programs make use of the so-called *coherent approximation* to shorten the required computing time. In the coherent approximation, any working states for the components in the expression to be calculated are set to TRUE and it is assumed that P(component works) ≈ 1. The calculation of the minimal cutsets/prime implicants can then be minimized and approximations such as the one given in Eq. [6.69] used.

For the simple example, this leads to the following results:

$$\begin{aligned}
&\lambda_{(1)} = \lambda_I \cdot P(T_{(1)}) \\
&= \lambda_I \cdot P\{ \left(\overline{A} \cap \overline{D} \cap \overline{F}\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \} \\
&= \lambda_I \cdot \{ P\left(\overline{A} \cap \overline{D} \cap \overline{F}\right) + P\left(\overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \\
&\quad - P\left(\overline{A} \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \} \cong 1.0/\text{yr} \cdot (1 + 1 - 1) = 1.0/\text{yr}
\end{aligned} \qquad [6.70]$$

$$\begin{aligned}
&\lambda_{(2)} = \lambda_I \cdot P(T_{(2)}) \\
&\quad = \lambda_I \cdot P\{ \left(\overline{A} \cap \overline{D} \cap F\right) \cup \left(\overline{B} \cap \overline{C} \cap \overline{D} \cap F\right) \cup \left(A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cap E\right) \} \\
&\quad = \lambda_I \cdot \{ P\left(F\right) + P\left(A \cap E\right) - P\left(A \cap E \cap F\right) \} \\
&\quad = 0.1/\text{yr} \cdot \{0.1 + 0.01 - 0.001\} = 0.109/\text{yr}
\end{aligned} \qquad [6.71]$$

$$\begin{aligned}
&\lambda_{(3)} = \lambda_I \cdot P(T_{(3)}) \\
&\quad = \lambda_I \cdot P\{ \left(A \cap B \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \cup \left(A \cap C \cap \overline{D} \cap \overline{E} \cap \overline{F}\right) \} \\
&\quad = \lambda_I \cdot \{ P\left(A \cap B\right) + P\left(A \cap C\right) - P\left(A \cap B \cap C\right) \} \\
&\quad = 0.1/\text{yr} \cdot \{0.01 + 0.01 - 0.001\} = 0.019/\text{yr}
\end{aligned} \qquad [6.72]$$

$$\lambda_{(4)} = \lambda_I \cdot P(T_{(4)}) = 0.103249/\text{yr} \qquad [6.73]$$

A direct comparison of the exact and approximate (coherent approximation) results is given in Table 6.3.

Table 6.3 ***Comparison of exact and approximate sequence frequency results***

Event tree sequence	Exact frequency [yr^{-1}]	Approximate frequency [yr^{-1}]	Relative error [%]
(1)	0.788049	1.000000	26.8
(2)	0.094851	0.109000	14.9
(3)	0.013851	0.019000	37.2
(4)	0.103249	0.103249	0.0

It can be seen that the use of the coherent approximation in this particular example leads to relatively large percentage errors for at least two of the sequences.

The use of Binary Decision Diagrams

Recent developments in digital logic provide an alternative analysis procedure for fault trees. This alternative approach, based on the use of *Binary Decision Diagrams* (BDD), works directly with the logical expressions instead of the cutsets/prime implicants. A BDD can be thought of as a graphical representation of a data structure for a logical function. With this approach, the exact system failure probability can be deduced without the need to resort to any approximations. This results in both accuracy and efficiency improvements compared with the traditional minimal cutest analysis. Different investigations have shown that orders of magnitude reduction in processing time for large fault trees can be achieved. These improvements would be expected to be even more significant for non-coherent fault trees, which tend to produce a great number of system failure modes that include component success states (prime implicants) There is however a cost to pay, which is the effort required to convert the fault tree structure to the BDD.

The BDD that is used for fault tree analysis is more exactly referred to as *Reduced Ordered BDD*. “Reduced” means that the BDD is in minimal form. “Ordered” means that the variables appear in the same order (to be defined initially) on each path.

Rules for the detailed fault tree to BDD conversion process are given in the appendix 6.1. Essentially, the diagram features a series of vertices or nodes, representing the basic events of the fault tree, linked by logical “0” or “1” branches. A “0” branch indicates the non-occurrence (success) of the basic event, whereas a “1” branch indicates the occurrence (failure) of this same event. Starting from a *root node* - the first of the ordered basic events - placed at the top of the tree structure, each successive (in the ordered list) basic events is connected to the preceding one in a way that respects the AND or OR (according to the gate case) “truth table” outputs respectively (see Appendix 6.1).

Paths through the BDD terminate at one of two types of terminal node, labeled “0” and “1”. Paths terminating in a “0” node represent the top event non-occurrence. Conversely, paths that lead to a terminal “1” node specify the conditions for the fault tree top event to occur. Listing just the failure events on such a path is equivalent to producing the cutsets for the fault tree. Unless the basic event ordering selected has produced a minimal form BDD, this will have to be processed to remove redundant cutsets and produce the minimal cutsets. Whilst this is an important source of information to the analyst, it is however not required to evaluate the event tree sequence frequencies.

The equivalent BDD structures for the fault trees of figure 6.25 are given in figure 6.28 for the basic events ordered according to the alphabetical order (A→B→C→D / A→D→E→F).

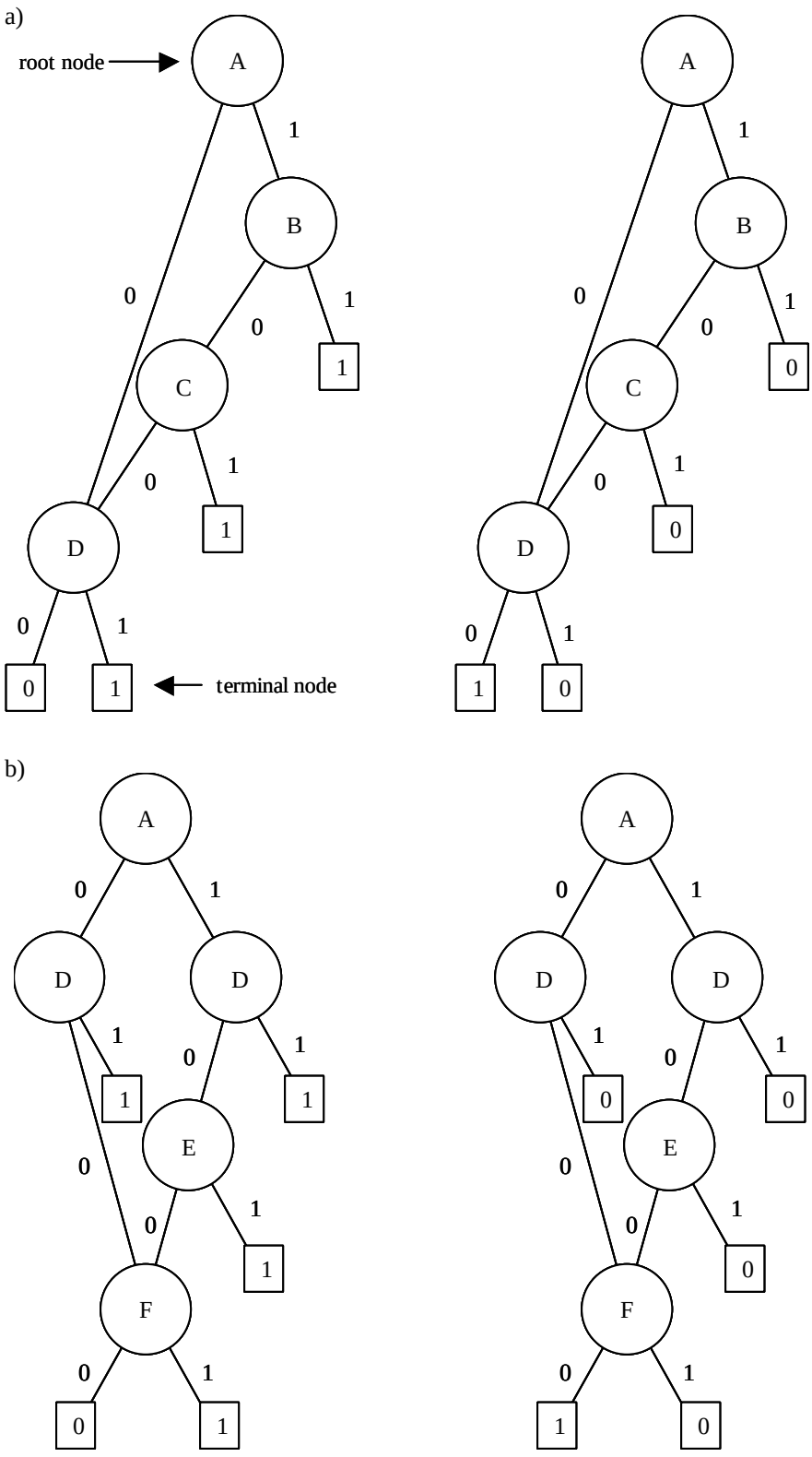


Figure 6.28 ***BDD (left) and dual BDD (right) structures for the fault trees of Fig. 6.25*** **(from [Andrews & Dunnett, 2000])**

Using the BDD approach to analyze the sequences of event trees such as the one shown in figure 6.24 requires both the BDD and dual BDD formulation of fault trees. The dual BDD structures of the two subsystems SS1 (a) and SS2 (b) are also given in figure 6.28. As shown, the dual of a BDD is created by simply changing the terminal "1"s to terminal "0"s and vice-versa. Note that the dual nodes on the BDD still represent in this formulation component failure states.

The path through the dual BDD to a terminal node "1", which includes each node passed through on the "0" branch (working component) represents the *path sets* of the fault tree. A path set is a list of working components that result in the system working if they all occur at the same time.

Table 6.4 resumes the cut and path sets for the two subsystems SS_1 and SS_2.

Table 6.4 ***Cut and path sets for the two subsystems SS_1 and SS_2 (see Fig. 6.28)***

Subsystem	Cut sets	Minimal cut sets	Path sets (minimal)
SS_1	AB, AC, AD, D	AB, AC, D	BCD, AD
SS_2	AD, AE, AF, D, F	AE, D, F	DEF, ADF

Applying the rules to manipulate BDD structures (see Appendix 6.1), the diagrams shown in figure 6.28 can be combined to obtain the BDD's representing the four possible outcomes (sequences $\overline{E}_1\overline{E}_2$, $\overline{E}_1E_2$, $E_1\overline{E}_2$, E_1E_2) of the example event tree (see Fig. 6.24). These are given in figure 6.29a and 6.29b.

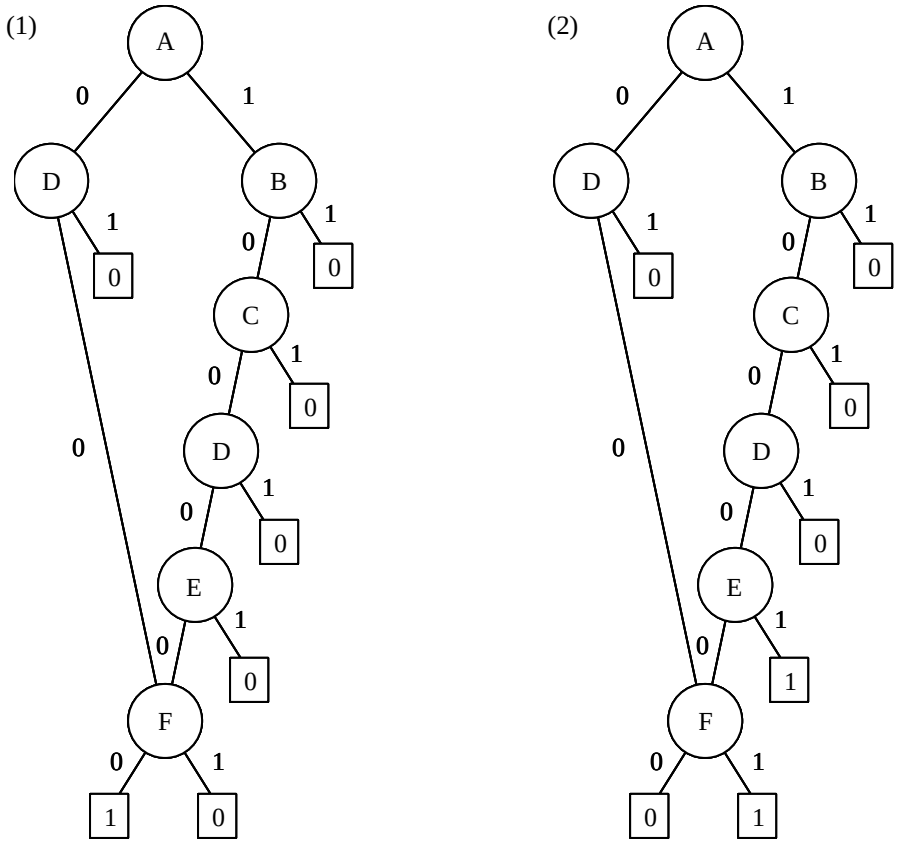


Figure 6.29a ***BDD structures for the*** $\overline{E}_1\overline{E}_2$ *and* $\overline{E}_1E_2$ sequences of Fig. 6.24
(from [Andrews & Dunnett, 2000])

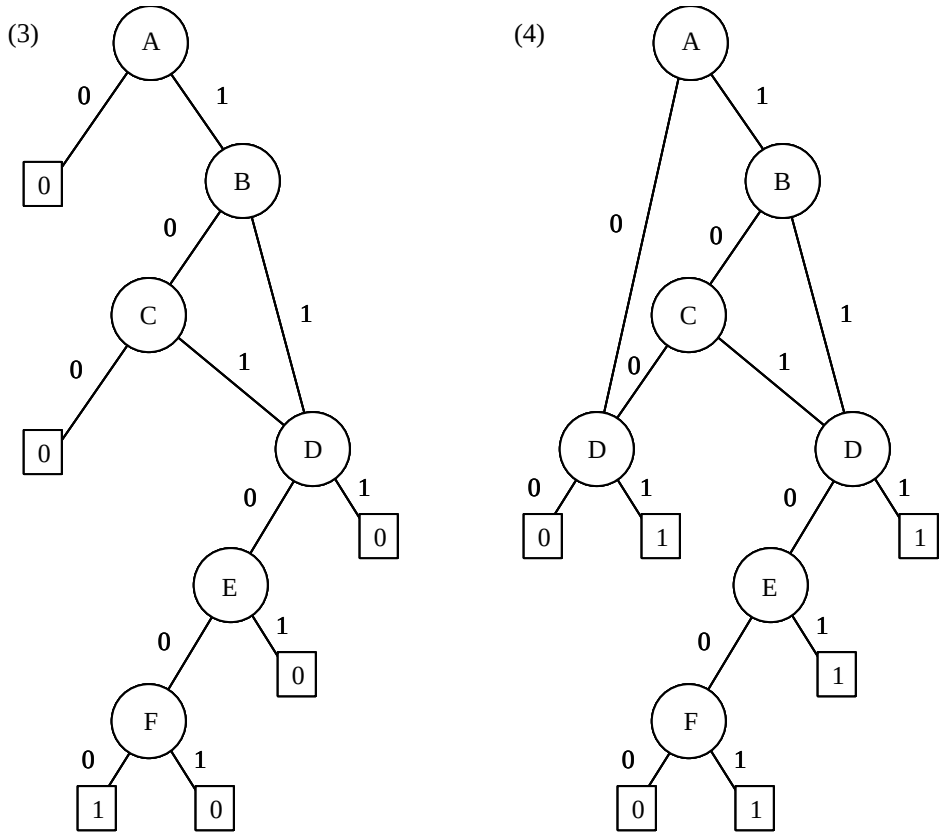


Figure 6.29b ***BDD structures for the*** $E_1\overline{E}_2$ *and* E_1E_2 sequences of Fig. 6.24 **(from [Andrews & Dunnett, 2000])**

By construction, the different paths in a BDD are mutually exclusive (binary branching). The probability of system failure is thus obtained by simply summing the probabilities of all the disjoint implicant paths leading to a terminal "1" node. Applied to the above diagram, this leads to the following top event probabilities:

Table 6.5 ***Top event probabilities from BDD structures of Fig. 6.29a and 6.29b***

(1) Sequence $\overline{E}_1\overline{E}_2$ (see Table 6.3)		(3) Sequence $E_1\overline{E}_2$ (see Table 6.3)	
I mplicant paths	Probabilities	I mplicant paths	Probabilities
$A\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}$	0.059049	$A\,B\,\overline{D}\,\overline{E}\,\overline{F}$	0.007290
$\overline{A}\,\overline{D}\,\overline{F}$	0.729000	$A\,\overline{B}\,C\,\overline{D}\,\overline{E}\,\overline{F}$	0.006561
	0.788049		0.013851
(2) Sequence $\overline{E}_1E_2$ (see Table 6.3)		(4) Sequence E_1E_2 (see Table 6.3)	
$A\,\overline{B}\,\overline{C}\,\overline{D}\,E$	0.007290	$A\,B\,D$	0.001000
$A\,\overline{B}\,\overline{C}\,D\,\overline{E}\,F$	0.006561	$A\,B\,\overline{D}\,E$	0.000900
$\overline{A}\,\overline{D}\,F$	0.081000	$A\,B\,\overline{D}\,\overline{E}\,F$	0.000810
	0.094851	$A\,\overline{B}\,C\,D$	0.000900
		$A\,\overline{B}\,C\,\overline{D}\,E$	0.000810
		$A\,\overline{B}\,C\,\overline{D}\,\overline{E}\,F$	0.000729
		$A\,\overline{B}\,\overline{C}\,D$	0.008100
		$\overline{A}\,D$	0.090000
			0.103249

6.5 Cause-Consequence Analysis (CCA)

General presentation of the method

The *Cause-Consequence Analysis* or *Cause-Consequence Diagram Method* is a well-structured technique that combines cause analysis (described by fault trees) and consequence analysis (described by event trees). This way, both inductive and deductive analyses are used in this approach. The consequences evaluated include those that characterize the system functioning as well as those that describe an undesirable failure sequence of events.

Compared with the FTA method, the CCA technique, which also documents the failure logic, has the extra capability of enabling the analysis of systems subject to sequential failures. Contrary to FTA, CCA is moreover capable of identifying both the possible causes of an undesirable event *and* all the possible consequences resulting from it The CCA method is also superior to ETA, which can similarly identify all consequences of a given critical event, as it models at component level and therefore is functionality driven and not subsystem driven. In addition to this, CCA can account for time delays, which is not a feature available in the ETA technique. CCA is thus a method to explore time-sequenced system responses to initiating "challenges" and to enable probability assessments of success/failure outcomes at staged increments.

The CCA technique was initially invented by RIS∅ Laboratories in Denmark to assist in the risk analysis of nuclear power plants [Nielsen, 1971]. It was then adopted (and adapted) by other industries in the estimation of the safety of protective or other types of systems

The basis of the CCA technique is the consideration of a *critical event*, i.e. an event that disturbs the normal (and safe) behavior of the system under study. Once such critical event has been identified, all relevant causes of this event and potential consequences are developed using FTA (see section 6.3) and ETA (see section 6.4) conventional analysis methods. The FTA method is as a matter of fact used in two independent situations in the CCA process. Firstly, this approach is used to precise the causes of the critical event. The second function for the FTA method is to clarify the causes of the possible failures of the accident-limiting subsystems. The ETA method is for its part used as a link between the causes of the critical event and the various consequences that could result (see Fig. 6.30).

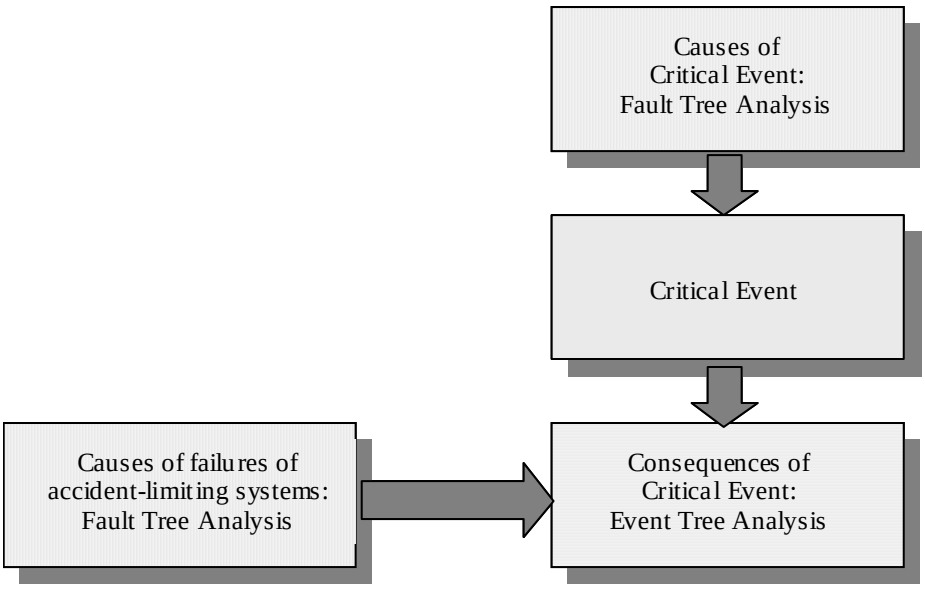


Figure 6.30 ***Basic structure of the Cause-Consequence Analysis method***

Construction of the Cause-Consequence Diagram

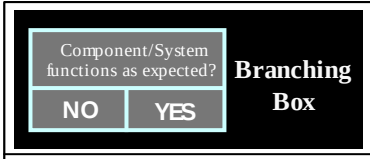
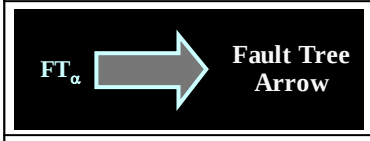
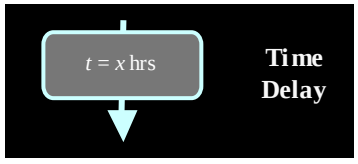
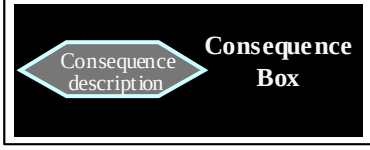
Rules for the construction of a cause-consequence diagram can be classified in two separate classes: those for the cause part of the diagram and those for the consequence part of the diagram [Ridley and Andrews, 2001].

The cause-consequence diagram construction starts with the identification of the critical event, which is problem dependent. Choosing the right place to start is important as there are many possible initiating events, but not all of them have serious consequences. This choice can prove particularly difficult in complex situations; in theory at least, such "central" event could be anywhere along the chain of linked events. A useful solution to get round this difficulty is for the analyst to use an energy damage model and to say that the critical event is the point at which control of the potentially damaging energy is lost. This is however sometimes not totally obvious. For example, in an analysis for an electrical authority with high voltage transmission lines, the point of loss of control of energy was defined as when someone or something penetrated the flashover envelope of the high voltage conductor [Robinson. 2000]. That is, despite having entered this region with a fishing pole on the back of a vehicle, the flashover may not occur with fatal results to the occupants. They might be insulated from the road or it may be a very dray day and the envelope was a little smaller than usual, with the result that the event in such a case cannot be considered as "critical" anymore.

After the critical event has been identified, the causes of this undesired event are discovered and connected by means of logical gates, using the same rules and symbols developed in the FTA section (which will therefore not be repeated here).

Starting from the initiating component, the functionality of each component/sub-system is then investigated and the consequences of the corresponding sequences determined. For the construction of this consequence part of the diagram some new symbols are introduced, which are presented in Table 6.6.

Table 6.6 ***Specific logical symbols used in Cause-Consequence diagrams***

	The ***branching box*** represents a functionality condition to be fulfilled by a component/subsystem; output is "yes" if the condition is met, "no" if it is not met (note that the branching operator may be written in either fault or success domain
	The ***fault tree arrow*** indicates under which designation (FT_g here) the fault tree corresponding to the undesirable fulfillment of the condition given in the branching box it points to can be found
	The ***time delay*** is used to indicate that the following event in the diagram cannot occur before the time interval given in the symbol is elapsed
	The ***consequence box*** represents the event/condition to which analysis of a particular sequence leads, with, usually, severity level stated

If the branching box is governed by a sub-system, then the probability of failure of this one is obtained via a fault tree diagram. If any branching box is found irrelevant, e.g. the boxes attached to the "No" and "Yes" branches are identical and their outcomes and consequences are the same, then these should be removed to reduce the CCA diagram to a minimal form (this has no effect on the end result).

Application example of the CCA method

To illustrate the application of the CCA approach, we will consider the example of a pressure tank system [Hassl et al., 1981]. In addition to its operational phase, the system includes a start-up, shutdown sequence. The system configuration is given in figure 6.31.

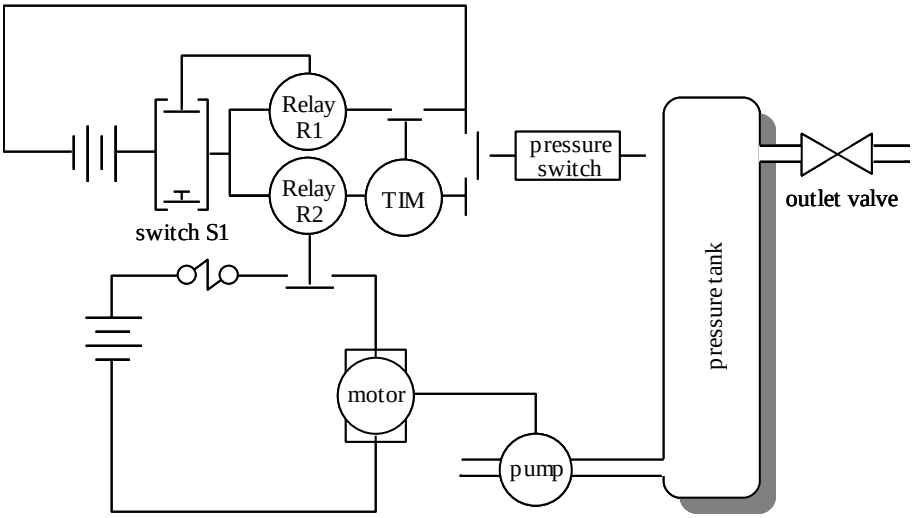


Figure 6.31 ***Pressure tank example for CCA application***

The components individual functions and failures modes are described in Table 6.7.

Table 6.7 ***Component functions and failure modes of the pressure tank example***

Component	Function	Failure Modes	Effect on system
Switch S1	To apply power to coil of R1 relay	S1C: switch fails closed	Cct remains energized but can be broken by R2
		S1O: switch fails open	No power to energize cct
Relay R1	Electrically self-latched applying power to relay R2	R1D: relay fails de-energized	No power to cct
		R1CC: contact fails closed	Cct remains energized but can be broken by R2
		R1CO:contact fails open	No power to cct
Relay R2	Deliver power to the motor	R2D: relay fails de-energized	No power to motor
		R2CC: contact fails closed	Continuous power to motor
		R2CO:contact fails open	No power to motor
Timer Relay (TIM)	Provides emergency shut-down in event of pressure switch failing	TIMCC: timer contact fails closed	Cct energized but PRSW can open
		TIMCO: timer contact fails open	No power to motor

Table 6.7 ***Component functions and failure modes of the pressure tank example (cont'd)***

Pressure Switch (PRSW)	De-energizes coil of R2 when tank is full	PSWC: switch fails closed PSWO: switch fails open	Continuous power to motor No power to motor
Fuse	To prevent power surge	F: fuse fails broken	No power to motor
Power supplies 1 & 2	Supplies power to relays and motor	PS1, PS2: no power	No power to motor
Motor	Pumps fluid into tank	M: pump fails broken	No power to pump

Initially, the system is considered to be in dormant state and therefore de-energized. In this state, the switch S1, the relay contacts R1 and R2 are all open and the timer and pressure switch are closed. Depressing the switch S1 provides power to the coil of R1, which results in the closure of the R1 contacts. R1 self latches when S1 being released opens and power is also supplied to R2 resulting in R2 contacts closing. This starts the pump motor. It is assumed that the tank takes 30 minutes to fill and once the pressure threshold is reached the pressure switch contacts open, de-energizing R2, which results in the removal of power from the pump motor. After a period of time, the tank becomes empty and the pressure switch closes, which energizes R2. The pump restarts and the filling process commences again. The tank is filled twice daily and the system is inspected at 6 monthly intervals for latent failures. In the event of the pressure switch failing to open, a safety feature is included in the form of a timer relay. Power is applied to the timer relay following the closure the R1 contacts, which initiates a stopwatch. If the stopwatch registers 30 minutes of continuous pumping the timer relay contacts are open; this results in a break in the circuit to R1 and system shutdown.

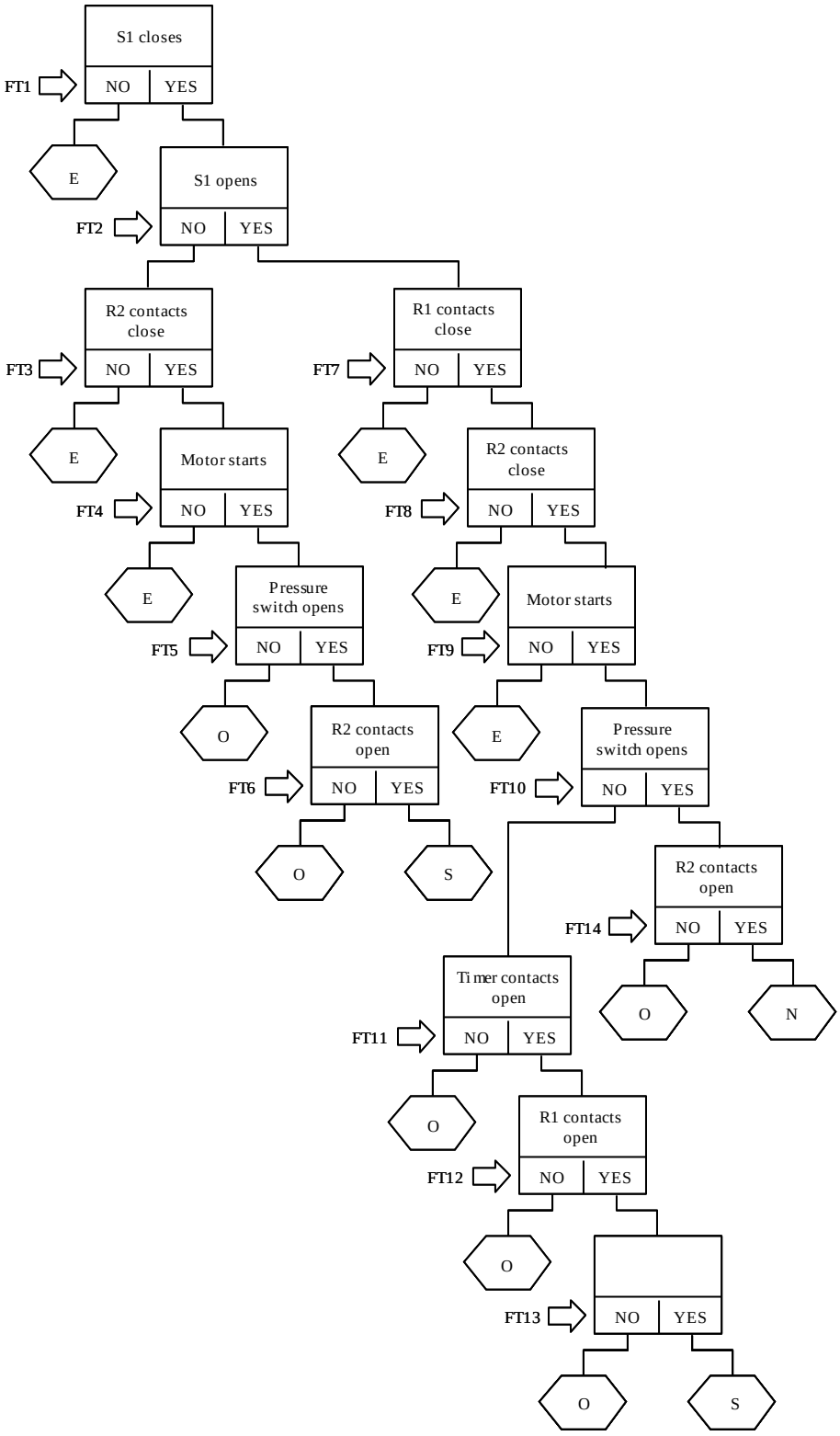
The first step in constructing the cause-consequence diagram of this system is to order the component failure events. This is done by considering the temporal patterns of the system and leads to the order:

S1, R1, R2, PRSW, TIM, R1, R2

It comes out that the components R1 and R2 both occur twice in the ordering sequence. This results from the system containing two different phases and hence some components perform different actions in each different phases. Components R1 and R2 are both required to close in the start-up sequence and open in the shutdown sequence.

The cause-consequence diagram is then constructed by considering the effect of each component in the chosen order on the system operation. The resulting cause-consequence diagram is given in figure 6.32 and the corresponding fault trees illustrated in figure 6.33.

Note that this is not the final form of the cause-consequence diagram. Prior to multiplying the probabilities associated with each decision box to quantify each of the sequences, the diagram must be checked for any dependent failure events and then appropriately modified. For example, a common failure event (PS1) is present on FT7 and FT8. PS1 should thus be extracted and placed in a new decision box preceding decision box 7. Other modifications are required to take into account inconsistent failure events.



E : “Empty”
O : “Overpressurized”
S : “Safe”
N : “Normal”

Figure 6.32 ***Primary cause-consequence diagram for the pressure tank example***

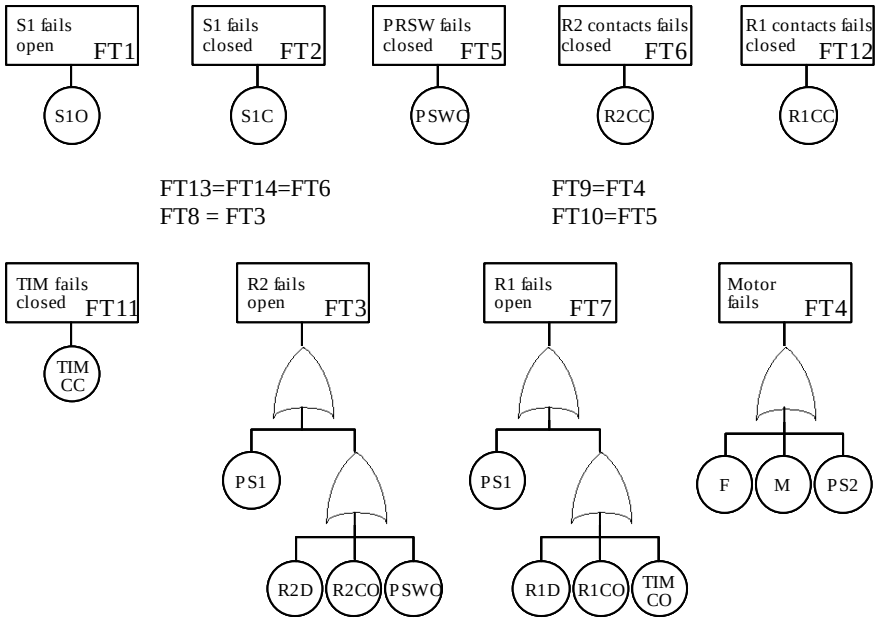


Figure 6.33 ***Fault trees associated to the cause-consequence diagram of Fig. 6.32***

Inconsistent failure events are encountered when, in certain systems, components are required to perform different functions, which if successfully accomplished results in the components residing in different states at different times of operation. For example, initially a valve may be required to be closed and later in the sequence of operation to be open. The identification of such events and the resulting modified cause-consequence diagram and associated fault trees for the above example can be found in the paper of Ridley and Andrews [Ridley and Andrews, 2001].

6.6 Other Methods

The methods presented in the preceding pages are only a sample of the existing methods that can be used for safety/reliability assessments. Without pretending to be exhaustive, some additional methods are presented very briefly below.

Markov Modeling is a classical modeling technique for assessing the time-dependent behavior of dynamic systems. The state probabilities of the system P(t) in a continuous Markov system analysis are obtained by the solution of a coupled set of first order, constant coefficient differential equations: $dP/dt = M \cdot P(t)$, where M is the matrix of coefficients whose off-diagonal elements are the transition rate and whose diagonal elements are such that the matrix columns sum to zero.

The *GO Method* can be used to compute the probability that a system exists in each of a few states. The system being studied is modeled in the form of a "GO chart", which consists in selecting functional operators (or "building blocks") to represent each component and logical junction, and connecting them with arrows to represent the flow of information. The GO method can be considered a competitor of FTA.

Dynamic Event Tree Analysis Method is an approach that treats time-dependent evolution of systems states, process variable values, and operator states over the course of a scenario. In general, a dynamic tree is an event tree in which branchings are allowed at different points in time.

Monte Carlo Simulation can also be a useful general technique for risk analyses. First, the random numbers are sampled for each of the uncertain assumptions. Secondly, the random numbers obtained are used together with the other assumption values to perform the basic analysis.

Appendix 6.1 Binary Decision Diagrams (from [NASA, 2002])

The construction of a Binary Decision Diagram (BDD) from a fault tree is a recursive, "bottom-up" process. Each basic event has an associated single-node, with "0" and "1" children, BDD. For example, the BDD for a basic event A is shown in figure 6.34.

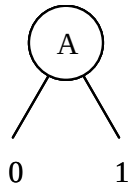


Figure 6.34 ***BDD for a basic event A***

Starting at the bottom of the tree, a BDD is constructed for each basic event and these (in an order pre-defined) are then combined according to the type of logic ("truth table") represented by the gate to which the events in question are connected.

The first step in the construction of the BDD for the OR relation A ∪ B is schematized in figure 6.35. Since A is first in the OR relation, it becomes the "root" node; the B BDD is then combined with each "child" node of A.

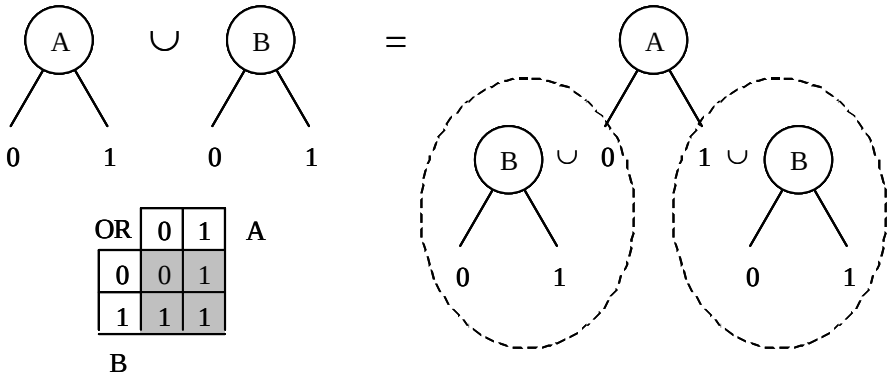


Figure 6.35 ***BDD for A ∪B (first step)***

To this end, consider respectively the left and right "children" of A (terminal nodes "0" and "1"). According to the OR "truth table", 0 ∪ X = X and 1 ∪ X = 1. Thus, the left child reduces to B and the right child reduces to 1 as represented in figure 6.36.

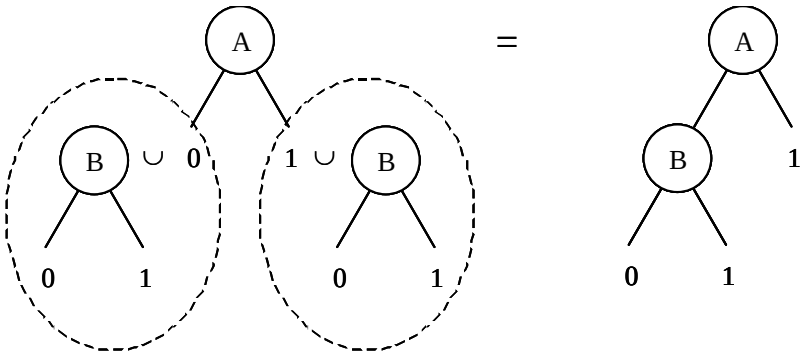


Figure 6.36 ***BDD for A ∪B (final step)***

Applying the same combinatorial process to the case of the AND gate leads to the result presented in figure 6.37, which takes into account that here $0 \cap X = 0$ and $1 \cap X = X$.

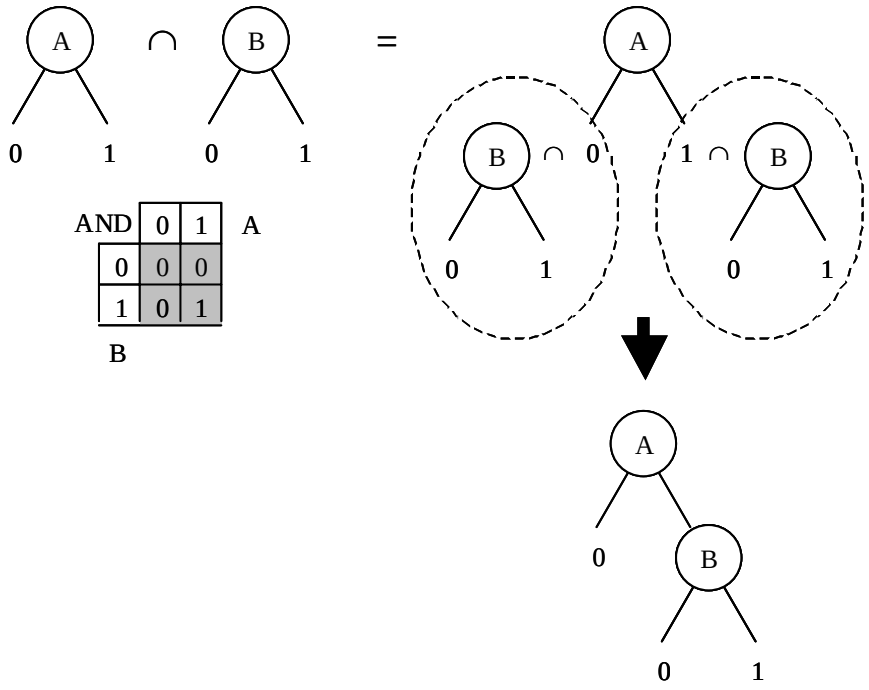


Figure 6.37 ***BDD for A ∩ B***

Consider now the case of a basic event C that is OR'ed with the AND gate of A and B, i.e. $C \cup (A \cap B)$. Since A comes first, A remains the root node of the combined diagram and the OR operation is applied to A's children. The left child reduces by Boolean algebra to C and the right child to the result of figure 6.36, as shown in figure 6.38.

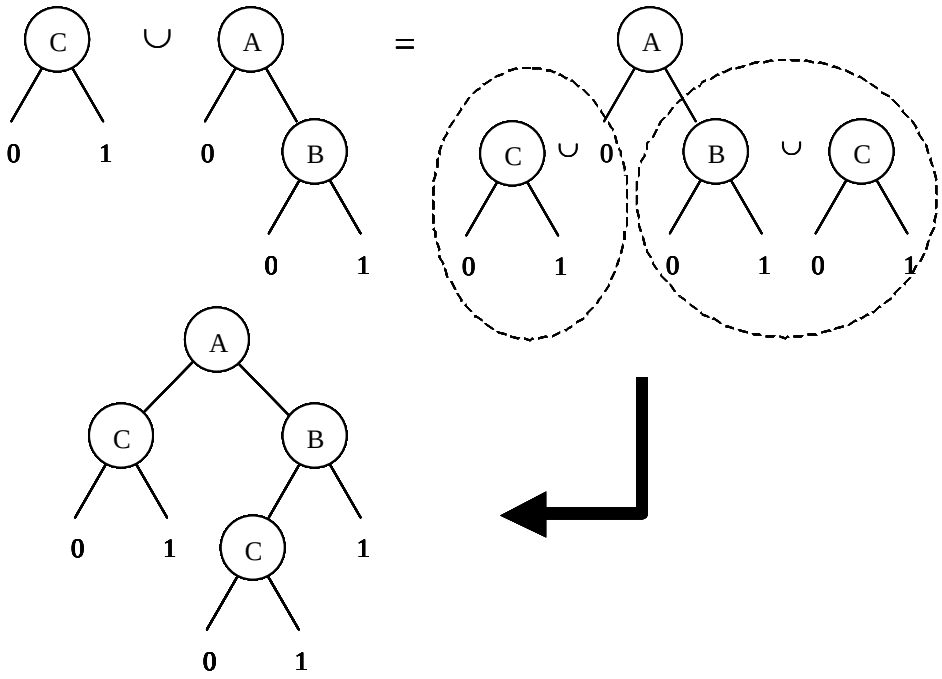


Figure 6.38 ***BDD for (A ∩ B) ∪C (initial steps)***

The resulting BDD can be reduced further, noticing that there are two identical instances of the node representing C in the diagram. One is redundant and can thus be removed, which leads to the final diagram represented in figure 6.39.

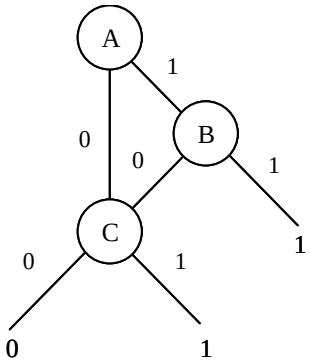
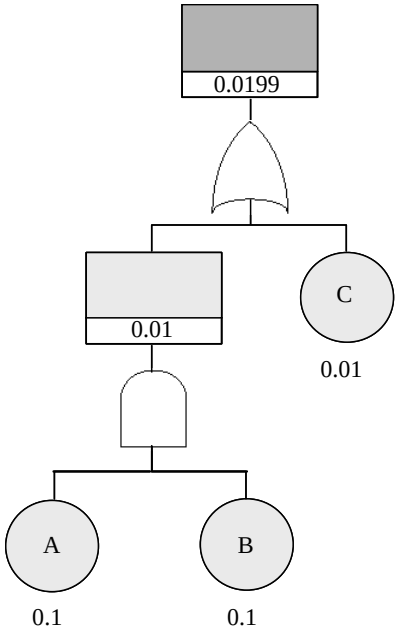


Figure 6.39 ***BDD for (A ∩ B) ∪C (final steps)***

Each path from the root node to a terminal node with value "1" represents a disjoint combination of events that, by definition, causes system failure. Thus, for the system represented by the diagram of figure 6.39, the failure paths are; $\overline{A}\,C$, $A\,B$, $A\,\overline{B}\,C$. Since the paths are disjoint, the calculation of the system probability of failure is straightforward (i.e. sum of the probabilities associated with the paths, see Fig. 6.40).



From Fig. 6.39 (disjoint failure paths):

$\overline{A}\,C$	$(1 - 0.1) \cdot 0.01$	=	0.0090
$A\,B$	$0.1 \cdot 0.1$	=	0.0100
$A\,\overline{B}\,C$	$0.1 \cdot (1-0.1) \cdot 0.01$	=	0.0009
			0.0199

Figure 6.40 ***Quantification of the BDD of Fig. 6.39***

The transformation and calculation of any fault tree would proceed in the above manner until all the gates have been linked in the BDD.

The BDD approach is a complementary approach to the minimal cutest (MCS) approach, each approach having its advantages and features. The MCS approach identifies the minimal sets (combinations) of basic events that could cause the top event. This approach thus highlights the most significant failure combinations and show where designs changes can eliminate or reduce undesirable combinations. Minimal cutsets also support fault tree validation in that specific minimal cutsets can be checked to determine if they indeed can cause the top event. They also support recovery actions by focusing the attention on the failures present in the dominant minimal cutsets. Minimal cutsets can furthermore be reviewed for dependencies and susceptibilities to common cause failure potentials.

For very large fault trees having many AND and OR gates, in which many minimal cutsets can be generated, the MCS approach must often truncate the lowest probability minimal cutsets to calculate the probability of the top event in reasonably short time. Results of such calculations are generally accurate to at least two significant figures, which is typically more accurate than the uncertainties on the basic event probabilities that are used. Present fault tree software packages have algorithms for bounding truncation error. When this error is out of pre-specified bounds, the truncation limit can be lowered and more minimal cutsets generated and calculated. Because of the speed of present personal computers, evaluating a sufficient number of minimal cutsets is usually, but not always, not a problem.

Because the minimal paths generated in the BDD approach are disjoint, this approach provides an easy-to-perform exact calculation of the top event probability; it is the most efficient method for calculating failure probabilities. The exact probability is useful when many high-probability events appear in the model.

The BDD is thus more efficient and precise in quantifying probabilities and importances. The MCS approach provides, for its part, important qualitative information as well as quantitative information. The most information is provided by using both approaches. The use of binary decision diagrams therefore does not preclude the determination and evaluation of minimal cutsets. Most available software packages only use the MCS approach, which has been the standard method for fault tree evaluation for many decades. There are now however available packages that use the BDD approach, and a few that use both approaches. In the future, more software packages are expected to include both the MCS and BDD approaches.

Bibliography

American Chemical Society, *Understanding Risk Analysis, A Short Guide for health, Safety and Environmental Policy Making*, Internet Edition, 1998

Andrews J.D. and Dunnett S.J., *Event Tree Analysis using Binary Decision Diagrams*, IEEE Transactions on Reliability, Vol 49, No 2, June 2000

Brander R., *The Titanic: An Enduring Example of Money management vs. Risk management*, original essay, 1995

BUWAL (Bundesamt für Umwelt, Wald und Landschaft), *Risikoanalyse bei gravitativen Naturgefahren, Method*, Dokumentation, 107/I , 3003 Bern, Switzerland, 1999

Clemens P.L., *Fault Tree Analysis*, 4th Edition, web site: http://www.sverdrup.com/safety/fta.pdf, 2002

DMTP (Disaster Management Training Programme), *Vulnerability and Risk Assessment, 2nd Edition*, UNDP, DHA, Cambridge Architectural Research Limited, The Oast House, Malting Lane, Cambridge, United Kingdom, 1994

Engineering Statistics Handbook, High Performance Computing and Communications/Systems Integration for Manufacturing Applications, *NIST/SEMATECH e-Handbook of Statistical Methods*, web site: http://www.itl.nist.gov/div898/handbook/, 2003

Erdmann R.C., *A Risk methodology Presentation*, Electric Power Research Institute Informal Rep. NP-79-1-LD, 1979

Hendershot D.C., Vinson J.W., Lorenzo D.K., *Putting "OP" Back in "HAZOP"*, Rohm and Haas Company, PO Box 584, Bristol, PA 19007, USA, Paper prepared for presentation at the MAINTECH South '98 Conference and Exhibition, Houston, Texas, December 2-3, 1998

Hassl D.F., Roberts N.H., Vesely W.E., Goldberg F.F., *Fault Tree Handbook*, US Nuclear Regulatory Commission, NUREG-0492, 1981

Lambert H.E., *System Safety Analysis and Fault Tree Analysis*, Lawrence Livermore Laboratory, Rep. UCID-16238, 1973

McCormick N.J., *Reliability and Risk Analysis: Methods and Nuclear Power Applications*, Academic Press Inc., New York, 1981

Mohr R.R., *Failure Modes and Effects Analysis*, Sverdrup, web site: http://www.fmeainfocentre.com/download/fmeamanual.pdf, 1994

NASA, *Fault Tree Handbook with Aerospace Applications*, Prepared for NASA Office of Safety and Mission Assurance, NASA Headquarters, Washington, DC 20546, August 2002

Nielsen D.S, *The Cause/Consequence Diagram Method as a Basis for Quantitative Accident Analysis*, Danish Atomic Energy Commission, RISO-M-1374, 1971

Périlhon P., *Logiciel MADS-MOSAR II, CD Rom version2.09*, Ed.Fox Média, Grenoble, 1999

Rausand M., Supplement SIO3020 Safety and Reliability Engineering Event Tree Analysis, Dpt. Of Production and Quality Engineering, Norwegian University of Science and Technology, N-7491 Trondheim, Norway, 1999

Ridley L.M. and Andrews J.D.: *Reliability of Sequential Systems using the Cause-Consequence Diagram Method*, Dep. of Mathematical Sciences, Laughborough University, Laughborough, Leicestershire, LE11 3TU, Great Britain, 2001

Robinson R.M. et al., *Risk & Reliability - An Introductory Text*, fourth edition, 2000

Schweitzer E.O., Anderson P.M., *Reliability Analysis of Transmission Protection using Fault Tree Methods*, SEL, web site: http://www.selinc.com/techpprs/6060.pdf, 1998

UNDRO (Office of the united Nations Disaster Relief Coordinator), *Natural Disasters and Vulnerability Analysis*, Report of Expert Group Meeting, Geneva, 9-12 July 1979

Villemeur A., *Sûreté de fonctionnement des systèmes industriels* Collection de la Direction des Etudes et Recherches d'Electricité de France, Editions Eyrolles, Paris, 1997

WASH-1400, Reactor Safety Study, an Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants, Nuclear Regulatory Commission Rep. ("Rasmussen Report"), NUREG-75, October 1975