Non-coherent Fault Tree Analysis

By

Sally Christian Beeson

A Doctoral Thesis
submitted in partial fulfilment of the requirements for the award of
Doctor of Philosophy of Loughborough University

September 2002

Abstract

The aim of this thesis is to extend the current techniques available for the analysis of non-coherent fault trees. At present importance analysis of non-coherent systems is extremely limited. The majority of measures of importance that have been developed can only be used to analyse coherent fault trees. If these measures are used to analyse non-coherent fault trees the results obtained are inaccurate and misleading. Extensions for seven of the most commonly used measures of importance have been proposed to enable accurate analysis of non-coherent systems.

The Binary Decision Diagram technique has been shown to provide an accurate and efficient means of analysing coherent fault trees. The application of this technique for the qualitative analysis of non-coherent fault trees has demonstrated the gains to be made in terms of efficiency and accuracy. Procedures for quantifying a non-coherent fault tree using this technique have been developed; these techniques enable significantly more efficient and accurate analysis than the conventional techniques for Fault Tree Analysis.

Although the Binary Decision Diagram technique provides an efficient and accurate means of analysing coherent and non-coherent fault trees, large trees with many repeated events cannot always be analysed exactly. In such circumstances partial analysis must be performed if any conclusions regarding system safety and reliability are to be drawn. Culling techniques employed in conjunction with the Binary Decision Diagram method have been developed for the partial analysis of both coherent and non-coherent fault trees.

Acknowledgements

Firstly I would like to thank my supervisor Professor John Andrews for his invaluable help, guidance, and friendship over the past three years.

Thanks are also extended to the staff in the mathematics department, who have helped me at various stages throughout my research. Particular thanks to Lisa for her efforts in proof-reading this thesis.

I am also very grateful to my family and friends for their support and encouragement throughout this work.

A huge thank-you to Karen who has kept me motivated and has always been around to discuss any problems or new ideas over a cup of tea.

My final thanks go to Robert for his love, support and understanding - thanks Babe!

Contents

Nomenclature

$A(t)$	Availability function
C	Consequence of an event
C_i	Minimal cut set i
C_i^n	Minimal cut set i, of order n
$f(t)$	Failure probability density function (p.d.f)
$F(t)$	System unreliability function
f_{MIN}	Frequency culling value
$g(t)$	Repair Function p.d.f
$G_i(\underline{q})$	Criticality function for event i (Birnbaum's measure of importance)
$G_i^*(\underline{q})$	Jackson's proposed extension for Birnbaum's measure of importance
$G_{C_i^n}(\underline{q})$	Probability that the components contained in cut set C_i^n are critical to the system failure
$G_{\{C_i^n\}}(\underline{q})$	Correction term for $G_{C_i^n}(\underline{q})$
$G_i^F(\underline{q})$	Component failure criticality
$G_i^R(\underline{q})$	Component repair criticality
$G_{i,j}(\underline{q})$	Probability system is in a working state such that components i and j are failure critical
$G_{i,\bar{j}}(\underline{q})$	Probability system is in a working state such that component i is failure critical and component j is repair critical
$G_{\bar{i},j}(\underline{q})$	Probability system is in a working state such that component i is repair critical and component j is failure critical
$G_{\bar{i},\bar{j}}(\underline{q})$	Probability system is in a working state such that components i and j are repair critical
$G_{M_{i,j}}(\underline{q})$	Probability system is in a working state such that components i and j are failure critical and the failure of either i or j is sufficient to cause system failure
$G_{M_{i,\bar{j}}}(\underline{q})$	Probability system is in a working state such that component i is failure critical and component j is repair critical and the failure of i or the repair of j alone is sufficient to cause system failure

$G_{M\bar{i},j}(q)$	Probability system is in a working state such that component i is repair critical and component j is failure critical and the repair of i or the failure of j alone is sufficient to cause system failure
$G_{M\bar{i},\bar{j}}(q)$	Probability system is in a working state such that components i and j is r repair critical and the repair of either i or j alone is sufficient to cause system failure
h(t)	Conditional failure rate
I_{C_i}	Component criticality measure
$I^F_{C_i}$	Component failure criticality measure
$I^R_{C_i}$	Component repair criticality measure
I_{FV_i}	Fussell-Vesely's measure of component importance
$I^F_{FV_i}$	Fussell-Vesely's measure of component failure importance
$I^R_{FV_i}$	Fussell-Vesely's measure of component repair importance
$I_{FV}(C_i)$	Fussell-Vesely's measure of cut set importance
$I_{FV}(\varepsilon_i)$	Fussell-Vesely's measure of prime implicant set importance
$I_F(C_i^n)$	Measure of cut set frequency importance
$I_F(\varepsilon_i^n)$	Measure of prime implicant set frequency importance
I_{IN_i}	Barlow and Proschan's measure of Initiator importance
$I^F_{IN_i}$	Component initiator failure importance
$I^R_{IN_i}$	Component initiator repair importance
I_{E_i}	Component enabler importance
$I^F_{E_i}$	Component failure enabler importance
$I^R_{E_i}$	Component repair enabler importance
$I_i(\varphi)$	Birnbaum's structure measure of importance
k_{MAX}	Culling order
L	Ordered list of basic events
n	Number of components in a system; all nodes encoding event i in a BDD
n_c	Number of minimal cut sets in a system
n_p	Number of prime implicant sets in a system

P2	Meta-products structure encoding prime implicant sets for which x is irrelevant
P1	Meta-products structure encoding prime implicant sets for which x is failure relevant
P0	Meta-products structure encoding prime implicant sets for which x is repair relevant
P	Probability
$P(C_i)$	Probability of existence of minimal cut set i
$P(\varepsilon_i)$	Probability of existence of prime implicant sets i
$P(\theta_i)$	Probability of occurrence of minimal cut set I
Pr_{x_i}	Probability of the path section from the root vertex to the node x_i in the BDD
$Po^1_{x_i}$	Probability of the path section from the one branch of a node encoding x_i to a terminal one node in the BDD
$Po^0_{x_i}$	Probability of the path section from the zero branch of a node encoding x_i to a terminal zero node in the BDD
$po^1_{x_i-x_j}$	Probability from the one branch of node x_i to the node x_j, excluding the probability of node x_i
$po^0_{x_i-x_j}$	Probability from the zero branch of node x_i to the node x_j excluding the probability of x_i
$Po^{1,c}_{x_i}$	Probability of the path section from the 1 branch of node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i)
$Po^{0,c}_{x_i}$	Probability of the path section from the 0 branch of the node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i)
Po^{c}_{xi}	Probability of the path section from the consensus branch of the node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i)
$po^{c}_{x_i-x_j}$	Probability from the consensus branch of node x_i to the node x_j excluding the probability of x_i

$p_i(t)$	Component availability
p_{MIN}	Probability culling value
p_{x_i}	Probability of basic events encoded in the path from the root vertex to the current node x_i
$Q_{M_{i,j}}(t)$	Modified unavailability function for component i and j
$Q_{M_{C_i^n}}$	Modified unavailability function fro minimal cut set C_i^n
$Q_{sys}(t)$	System unavailability function (failure probability)
$q_i(t)$	Component unavailability (failure probability)
q_{Cl}	Minimal cut set unavailability
R	Risk
R(t)	Reliability function
$T_{i=1}$	Failure relevance or irrelevance of component i
$T_{i=0}$	Repair relevance or irrelevance of component i
$T_{i=*}$	Irrelevance of component i
$w_{sys}(t)$	System unconditional failure intensity
$w_i(t)$	Component unconditional failure intensity
w_{Cl}	Minimal cut set unconditional failure intensity
$W_{SYS}(0, t)$	Expected number of failures during the interval (0, 1)
$v_i(t)$	Component unconditional repair intensity
x_i	Binary indictor variable for component states
$Z(\underline{g})$	Probability of the paths from the root node to a terminal 1 node that do not pass through a node encoding x_i
λ	Conditional failure rate
μ	Conditional repair rate
ε_i^n	Prime implicant set i of order n
$\rho_i(\underline{x})$	Binary indicator function for each minimal cut set
$\varphi(\underline{x})$	Structure function
φ_i^f	Failure criticality function
φ_i^r	Repair criticality function

Chapter 1: Introduction

1.1 An Introduction to Risk and Reliability Assessment

Risk is an inherent element of all industrial processes. Accidents such as Piper Alpha, Bhopal and Chernobyl, which resulted in massive loss of human life, demonstrate the need to eliminate hazards as far as possible. It is for this reason that risk and reliability assessment has become an integral part of the safety regulations for many industries.

Quantitative reliability assessment became a major interest during the Second World War, when work was carried out to improve the reliability of the German missile systems. Since this time numerous techniques for assessing the risk and reliability of hazardous systems have been developed to meet the needs of industries such as the nuclear industry and the offshore industry.

Reliability assessment techniques are concerned with calculating the probability or frequency of system failure. There are a number of measures that can be used to quantify system failure, including the system reliability, availability, the unconditional failure intensity and the conditional failure intensity.

The reliability of a system, R(t), is defined as the probability that the system operates without failure for a stated period of time under specified conditions. The unreliability of a system, F(t), is defined as the probability that the system has failed at least once in the interval [0,t) given that it was working at t=0. Since reliability is probabilistic:

$$F(t)+R(t)=1 \tag{1.1}$$

The system availability, A(t), is defined as the fraction of total time that a system is able to perform its required function. The unavailability of a system, $Q_{SYS}(t)$, is the complement of the system availability and is defined as the probability that the system is in a failed state at time t. Again since availability is probabilistic:

$$A(t)+Q_{SYS}(t)=1 \tag{1.2}$$

The unavailability of a system is a relevant measure when system failure can be tolerated. For hazardous industries, system failure may be catastrophic and in such cases system unreliability should be assessed as opposed to system unavailability. The unconditional failure intensity, w(t), is defined as the probability that a system fails per unit time at t given that it was working at time t=0.

The conditional failure rate, $\lambda(t)$, is defined as the probability that a system fails per unit time at t given that it was working at time t and time 0.

Reliability assessments are often performed as part of a quantified risk analysis. The risk associated with a given incident, R, can be defined quantitatively as the product of the consequence associated with the incident, C, and the probability or frequency of the incident, P.

$$R = C \cdot P \tag{1.3}$$

Although the consequence of an incident is generally measured in terms of the number of fatalities, the techniques for modelling the outcome of this incident are industry dependent. However, reliability assessment techniques, which evaluate the probability or frequency of an incident, are standard across all industries. Many such techniques have been developed; the most widely used of these is Fault Tree Analysis (FTA), which will be considered in more detail in section 1.2.

Once the risk of an incident has been quantified it must be decided if it is "acceptable" or whether the risk is too high and so improvements should be made to the system reliability or the consequences mitigated. It is important to note that although the risk of a hazardous event can be reduced, usually it can never be eliminated completely, regardless of the money invested in its improvement.

1.2 An introduction to Fault Tree Analysis

FTA is a well-known and widely used deductive technique developed by Watson in the early 1960's. This technique can be used to assess the reliability of a wide variety of systems. The first stage of FTA is to identify the mode of system failure to be analysed. There may be a number of different modes of system failure; if this is the case a separate fault tree must be constructed and analysed for each mode identified.

A fault tree diagram expresses the causes of a particular mode of system failure also known as the top event. The top event of the fault tree is developed by branches leading to sub-events; these sub-events are then continually refined until the branches are terminated with component failure (or repair) modes.

The events in the fault tree are connected by logical operators (called gates) according to the underlying logic of the system. The three fundamental gate types used in the fault tree are the AND gate, the OR gate and the NOT gate. Although other gate types exist such as the XOR gate and the VOTE gate, these must be expressed in terms of the AND, the OR and the NOT gate before analysis can be performed. The commonly used gate and event types are given in tables1.1 and 1.2 respectively.

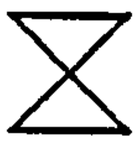
Gate Symbol	Gate Type	Causal Relation
	AND gate	Output occurs if all input events occur simultaneously
	OR gate	Output event occurs if at least one of the input events occur
	NOT gate	Output occurs if the input does not.

Table 1.1: The Three Fundamental Gate Symbols

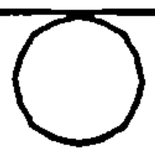
Event Symbol	Meaning of Symbol
	Intermediate event further developed by a gate
	Basic event

Table 1.2: The Common Event Symbols

An example fault tree is given in Figure 1.1. The top gate is an AND gate which means that both of the inputs to this gate must occur for the top event to occur. Gate G1 and G2 are both OR gates thus only 1 of the inputs to this gate is required to occur to cause the output event. The AND gate, the OR gate and the NOT gate combine events in exactly the same way as the Boolean operations of 'union', 'intersection' and 'complementation'

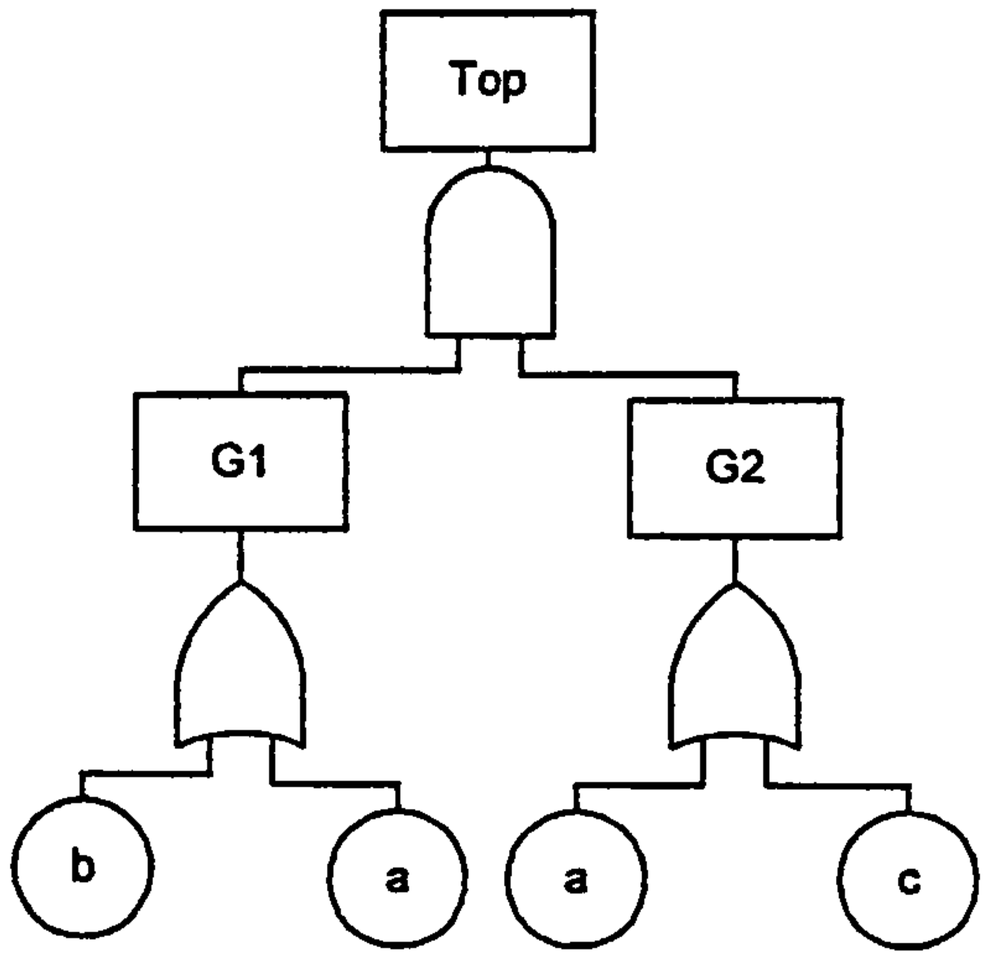


Figure 1.1: Simple Fault Tree Diagram

A fault tree can be classified as coherent or non-coherent according to its underlying logic. If during fault tree construction the failure logic is restricted to the use of the AND gate and the OR gate the fault tree is said to be coherent. If however, the NOT gate is used or directly implied by the XOR gate the resulting fault tree can be non-coherent. The definition of coherency will be considered in more detail in section 1.3.1.

1.2.1 The Analysis Procedure

FTA can be split into two stages, the first is qualitative analysis and the second is quantitative analysis. Qualitative analysis involves the identification of all the possible causes of system failure. For coherent fault trees each possible cause of system failure is known as a minimal cut set. When dealing with a non-coherent fault tree the causes of system failure are defined by the concept of prime implicant sets.

Quantitative analysis involves the quantification of various parameters relating to the system availability and reliability. The main parameters calculated during quantification are the system unavailability, the unconditional failure intensity and the expected number of system failures in a given interval. Another key part of the quantification process is analysing component and minimal cut set (or prime implicant set) importance.

1.2.1.1 Importance Analysis

When assessing a system, its performance is dependent on that of its components. Some of the system components will play a more significant role in causing or contributing to system failures than others. Importance measures can be used to numerically rank the contribution of each component or basic event according to the susceptibility of the system to the occurrence of this event.

Birnbaum introduced the concept of importance and developed a measure of component reliability 1969 [1]. This measure is denoted by $G_i(\underline{q})$ and is defined as the probability that the system is in a critical state at time t for component i, i.e., the system will fail if component i fails. An expression for this measure is given below:

$$G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q}) \tag{1.4}$$

Where $Q_{SYS}(1_i, \underline{q})$ is the probability the system has failed and component i is failed and $Q_{SYS}(0_i, \underline{q})$ is the probability the system has failed and component i is working.

This field has since received a great deal of attention and a variety of measures have now been developed for assessing both component and minimal cut set importance. Birnbaum's measure provides the foundation for a number of these measures.

1.3 Structure Functions

A binary system can exist in only one of two states, either a working or a failed state. The system is composed of components that must also exist in either a failed or working state. A binary indicator variable x_i is associated with component i where:

$$x_i = 1 \quad \text{Component fails}$$

$$x_i = 0 \quad \text{Component works}$$

Since the system state can be defined in terms of the its component states it is possible to define a function, $\varphi(\underline{x})$, to express the system state as follows:

$$\varphi(\underline{x}) = 1, \text{ if the system is failed}$$

$$\varphi(\underline{x}) = 0, \text{ if the system is working}$$

Where $\underline{x} = (x_1, x_2, \ldots, x_n)$ is a vector of all n component states and $\varphi(\underline{x})$ is known as the structure function.

1.3.1 Definition of Coherency

Fault tree structures can be categorised as either coherent or non-coherent. If during fault tree construction the failure logic is restricted to the use of the AND gate and the OR gate the fault tree is said to be coherent. If however, the NOT gate is used or directly implied, the fault tree can be non-coherent. A more precise definition of coherency can be obtained by considering the structure function of the fault tree [2].

A fault tree is coherent if its structure function, $\varphi(\underline{x})$, complies with the definition of coherency given by the properties of relevance and monotonicity. The first condition requires that each component is relevant, this means that each component contributes to the system state.

$$\varphi(1_i, \underline{x}) \neq \varphi(0_i, \underline{x}) \qquad \text{For some } \underline{x}$$

The second condition requires the structure function of the fault tree to be monotonically increasing, i.e. non-decreasing.

$$\varphi(1_i, \underline{x}) \geq \varphi(0_i, \underline{x}) \qquad \forall i$$

Where:

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

The structure function of a fault tree is monotonically increasing (non-decreasing) if as the state of a component deteriorates the system state either remains the same or also deteriorates. The three possibilities are shown in figure 1.2.

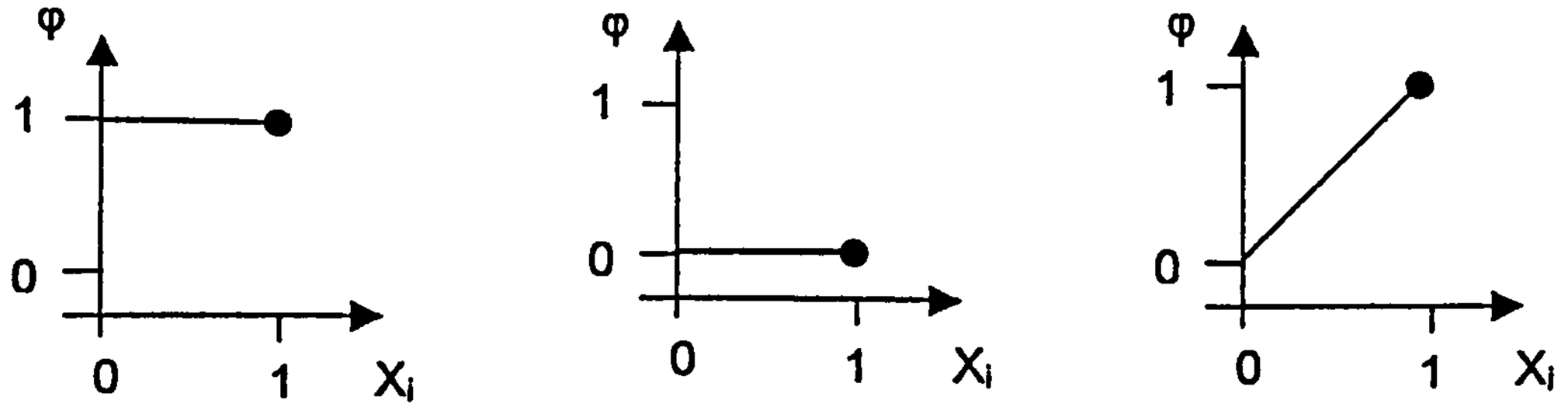


Figure 1.2: Non-decreasing Structure Functions

A structure function of a non-coherent system is shown in figure 1.3. This system is non-coherent for component i, hence the system is in a failed state when component i works and it is restored to a working state when component i fails.

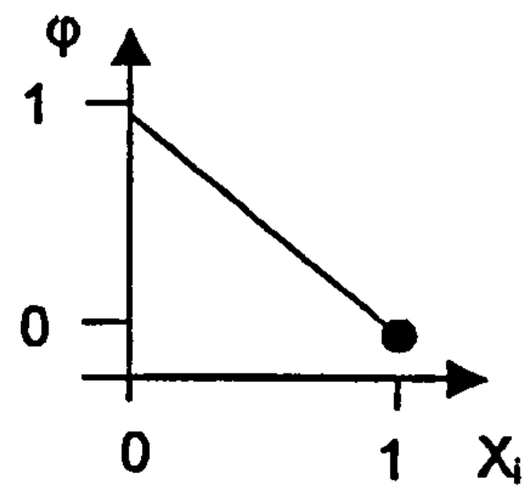


Figure 1.3: Non-coherent Structure function

1.4 An Introduction to The Binary Decision Diagram Method

An alternative technique for analysing fault trees is the Binary Decision Diagram (BDD) method, which was introduced by Rauzy in 1993 [3,4,5]. Conventional FTA techniques can be computationally intensive and sometimes inaccurate. The BDD method has been shown to enable extremely efficient qualitative analysis and accurate quantitative analysis of coherent fault trees. A further advantage of this technique is that the minimal cut sets are not required for quantification.

The first stage of this method is to convert the fault tree diagram to a BDD, known as the SFBDD because it encodes the structure function of the system. This BDD can be used to quantify the system exactly and efficiently. However, the SFBDD must be minimised before the minimal cut sets of the fault tree can be identified.

1.5 Current Limitations

Whilst numerous measures of importance have been developed for analysing component and minimal cut set importance, the majority of these measures are strictly for the analysis of coherent fault trees. If these measures are employed to analyse non-coherent fault trees the results can be misleading and inaccurate.

The Binary decision diagram technique has been shown to enable extremely efficient and accurate analysis of coherent fault trees. More recently Rauzy and Dutuit extended this technique to enable full qualitative analysis of non-coherent fault tree

structures [6,7]. Although this technique is significantly more efficient that conventional FTA techniques, it is necessary to compute two separate BDD's to identify the prime implicant sets exactly. Furthermore for complex fault trees it is not always possible to obtain a full list of prime implicant sets. To overcome this problem Rauzy developed a technique, which can be used to cull the prime implicant sets above a given order [15]. However, at present there are no techniques for culling the prime implicant sets according to their frequency or probability.

Although the system unavailability of a non-coherent fault tree can be calculated using the BDD technique, further quantification is not possible at present. Hence conventional FTA techniques must be employed making approximations unavoidable even for moderate sized trees

1.6 Objectives

The aim of this thesis is to further develop the current techniques available for the analysis of non-coherent fault trees. There are three main areas for consideration; importance analysis, efficient qualitative and quantitative analysis and culling techniques for partial analysis. Specific objectives for each of these areas are given below:

Importance Analysis:

- Modify Lambert's measure of enabler importance and Barlow and Proschan's measure of cut set frequency importance such that they are consistent with the definitions provided.
- Develop suitable measures for assessing the component and prime implicant set importance of non-coherent fault trees.

Qualitative and Quantitative Analysis

- Develop an alternative technique for qualitative analysis using the BDD method.
- Develop efficient techniques for quantification using the BDD technique.

Culling Techniques

- Develop efficient techniques for culling the prime implicant sets of a non-coherent fault tree according to their probability or frequency.

Chapter 2: Conventional Techniques for Coherent Fault Tree Analysis

2.1 Introduction

Risk and reliability assessment has become an increasingly important part of analysing and improving system safety. Numerous techniques for assessing system reliability have been developed since the Second World War; one such technique is Fault Tree Analysis (FTA).

FTA is a well known and widely used deductive technique conceived by Watson in the early 1960's to enable reliability assessment of a variety of systems. A fault tree provides a complete description of all the possible causes of a particular system failure mode in terms of the contributions of component failures. The system failure mode is also known as the top event, and appears at the top of the fault tree. This event is then continually developed into sub-events to show the possible causes of system failure until basic component failure events are encountered.

FTA can be split into two stages. The first stage is qualitative analysis, which aims to identify the causal relationships between the system components. The second stage is quantitative analysis, which aims to quantify various system parameters relating to its reliability and availability.

This chapter considers in detail the two stages of FTA of coherent fault trees. The technique will then be evaluated, highlighting its advantages and disadvantages.

2.2 Qualitative Analysis

2.2.1 Introduction

Qualitative analysis involves the identification of all the possible causes of system failure. System failure can usually occur in several unique ways, each possible cause of system failure is referred to as a system failure mode and will involve the failure of individual components or combinations of components. Failure modes can be defined by the concept of a cut set, where a cut set is a collection of basic events such that if they all occur the top event also occurs.

Fault trees for industrial systems are generally very large with thousands of cut sets. However, it is only the minimal cut sets that are of interest. A cut set is said to be minimal if the combinations of basic events are necessary and sufficient to produce system failure, i.e. if any one basic event is removed from a minimal cut set the top event will not occur.

A minimal cut set is the smallest combination of component failures, which, if they all occur will cause the top event to occur.

The order of a minimal cut set is determined by the number of components within the set, for example a one-component minimal cut set is said to be first order and a two-component minimal cut set is said to be second order. Generally the lower order minimal cut sets contribute most to system failure.

The top event, T, can be expressed in terms of its minimal cut sets as follows:

$$T = C_1 + C_2 + ... + C_n \tag{2.1}$$

Where $C_i, i = 1,....,n$, are the minimal cut sets and '+' represents the logical OR operator. Each cut set consists of one or more basic events, thus a general k^{th}-order cut set can be expressed as follows:

$$C_i = X_1 \cdot X_2 \cdot .... \cdot X_k \tag{2.2}$$

Where $X_i, i = 1,...,k$, are basic component failures and '.' represents the logical AND operator.

Consider the following logic expression of the top event, T:

$$T = B + A \cdot C + A \cdot D + C \cdot D \cdot E$$

There are four minimal cut sets, one first order, $\{B\}$ two second order, $\{A,C\}$and $\{A,D\}$ and one third order, $\{C,D,E\}$.

2.2.2 Obtaining the Minimal Cut Sets of a Coherent Fault Tree

2.2.2.1 Boolean Algebraic Laws

Traditionally the minimal cut sets of a coherent fault tree are obtained by converting a logic expression for the top event into disjunctive normal form, also known as minimal sum-of-products form. The logic expression is usually obtained using either a top-down or bottom-up approach, and then Boolean algebra laws are used to remove any redundancies in the expression leaving it in the required minimal form. The Boolean algebra laws used for coherent fault trees are given below.

1. Commutative laws:

$$A+B=B+A$$
$$A\cdot B=B\cdot A$$

2. Associative laws:

$$(A+B)+C=A+(B+C)$$
$$(A\cdot B)\cdot C=A\cdot(B\cdot C)$$

3. Distributive laws:

$$A+(B\cdot C)=(A+B)\cdot(A+C)$$
$$A\cdot(B+C)=A\cdot B+A\cdot C$$

4. Identities:

$$A+0=A \quad A+1=1$$
$$A\cdot 0=0 \quad A\cdot 1=A$$

5. Idempotent Laws:

$$A+A=A$$
$$A\cdot A=A$$

6. Absorption laws:

$$A+A\cdot B=A$$
$$A\cdot(A+B)=A$$

2.2.2.2 The Top-Down Approach

The top-down approach is commonly used to obtain the minimal cut sets of a coherent fault tree, by developing a Boolean expression for the top-event completely in terms of basic component failures. This approach starts with the top gate and expands each gate by substituting in the inputs that lie directly below it. This process is repeated until the expression has only basic component failures. The Boolean reduction laws introduced above are also applied where possible to simplify the expression.

The basic gate types that occur in a coherent fault tree are the AND gate and the OR gate, which are equivalent to the intersection and union operations of Set Theory. The minimal cut sets of the fault tree in Figure 2.1 can be obtained using the top-down approach as follows.

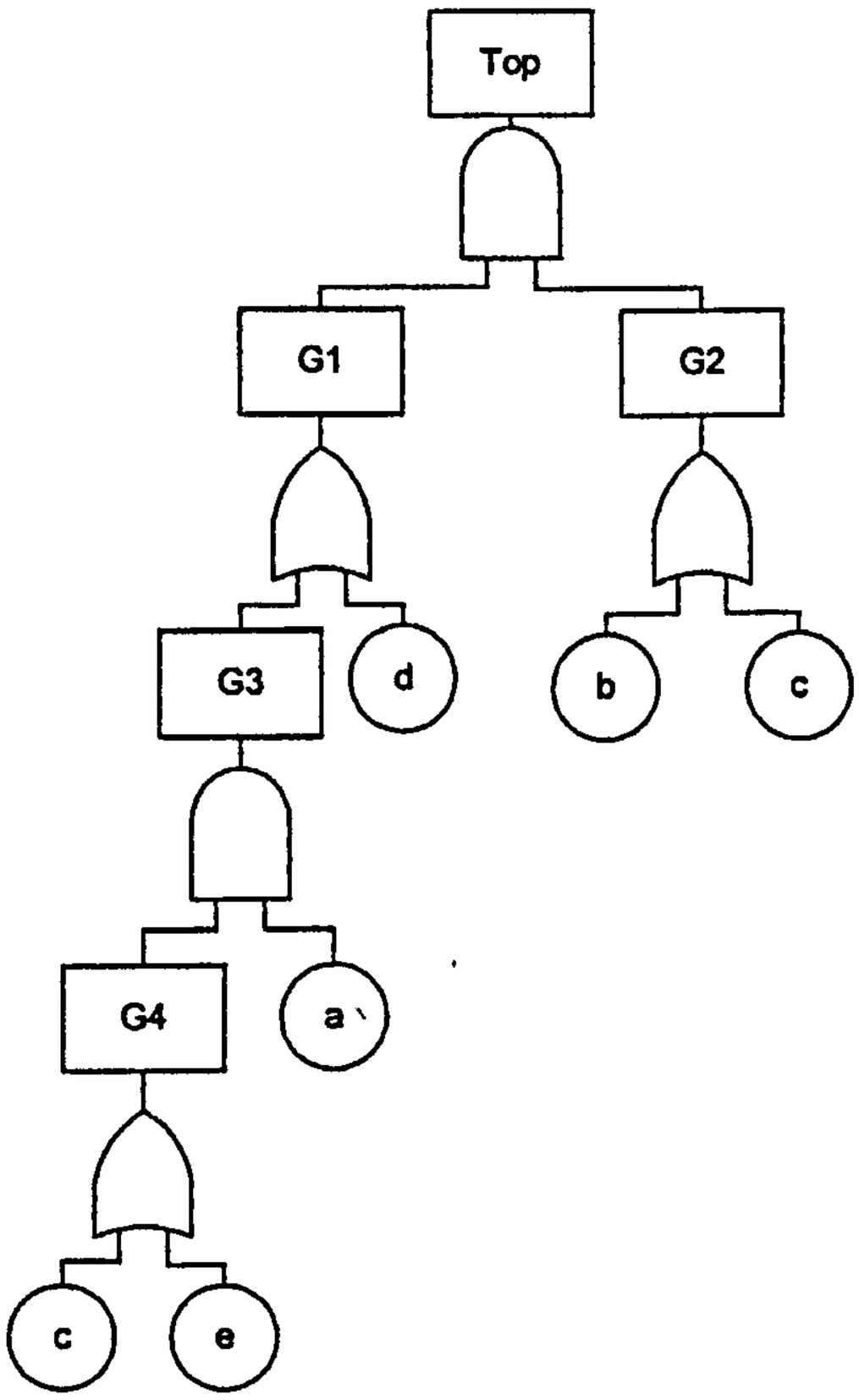


Figure 2.1: Fault Tree Diagram

Starting with the top gate which is an AND gate with two inputs G1 and G2 the following expression is obtained:

$$Top = G1 \cdot G2$$

G1 and G2 can be expressed as follows:

$$G1 = G3 + d$$

$$G2 = b + c$$

Hence Top becomes:

$$Top = (G3 + d)(b + c)$$

Expanding this expression gives:

$$Top = G3 \cdot b + G3 \cdot c + b \cdot d + c \cdot d$$

Expanding $G3 = G4 \cdot a$ Top becomes:

$$Top = G4 \cdot a \cdot b + G4 \cdot a \cdot c + b \cdot d + c \cdot d$$

Expanding $G4 = c + e$ results in the following expression:

$$Top = (c + e)a \cdot b + (c + e)a \cdot c + b \cdot d + c \cdot d$$

Expanding this gives the following expression:

$$Top = a \cdot b \cdot c + a \cdot b \cdot e + a \cdot c \cdot c + a \cdot c \cdot e + b \cdot d + c \cdot d$$

Finally simplifying this expression using the Boolean reduction laws, $A \cdot A = A$ and $A + A \cdot B = A$ the logic expression given in equation (2.3) is obtained for the top event.

$$Top = a \cdot b \cdot e + b \cdot d + a \cdot c + c \cdot d \qquad (2.3)$$

The minimal cut sets can be extracted from the minimal disjunctive normal form given in equation (2.3). There are four minimal cut sets for this example:

$$\{a,b,e\},\{b,d\},\{a,c\},\{c,d\}$$

2.3 Quantitative Analysis

2.3.1 Introduction

Quantitative analysis involves quantification of the system availability and reliability parameters and analysis of component and/or minimal cut set importance. Component and minimal cut set importance will not be considered in this chapter instead they will be the focus of chapters six and seven.

The traditional methodology used for quantitative analysis is known as "Kinetic Tree Theory" and was developed by Vesely [8]. This methodology enables time-dependent analysis of the reliability characteristics of a system and forms the basis for the majority of commercial fault tree packages. Although the development of Kinetic Tree Theory saw a major advancement in the field of reliability analysis, it does have its limitations; even for moderate sized problems approximations are unavoidable.

2.3.2 Component Quantification

Before considering quantification of system parameters it is important to introduce the various parameters relating to component performance. These parameters are expressed in terms of the component's failure probability and need to be evaluated before the system can be quantified. The failure and repair time distributions for each component can be assumed to have the density functions, f(t) and g(t). Component performance for repairable components can be measured in terms of the unconditional repair and failure intensities and the conditional repair and failure intensities.

The unconditional failure intensity, w(t), is defined as the probability per unit time that a component fails at time t given that it was working at t=0.

The unconditional repair intensity, v(t), is defined as the probability that a failed component is repaired per unit time at t given that it worked at t=0.

Andrews and Moss [9] derived integral equations whose solution yields the unconditional failure intensity and the unconditional repair intensity. The equations will not be derived, just stated, but a detailed explanation can be found in Andrews and Moss.

$$w(t) = f(t) + \int_0^t f(t-u)v(u)du$$
$$v(t) = \int_0^t g(t-u)w(u)du \qquad (2.4)$$

The conditional failure rate, $\lambda(t)$, is defined as the probability that a component fails per unit time at t given that it was working at time t and at t=0.

The conditional repair rate, $\mu(t)$, is defined as the probability that a component is repaired per unit time t given that it failed at time t and was working at t=0.

The difference between the unconditional failure (repair) intensity and the conditional failure (repair) rate is that the former is the failure (repair) rate based on the whole population whereas the latter is based only on those components working (failed) at time t.

The integral equations given in equation (2.4) can be solved using either Laplace transforms or numerical methods depending on the distributions defining f(t) and g(t). Once they have been solved and the unconditional failure and repair intensities are known, the component unavailability q(t) can be calculated. This is defined as the probability that the component is in a failed state at time t and is calculated as follows:

$$q(t) = \int_0^t [w(u) - v(u)]du \qquad (2.5)$$

If a component has failure and repair density functions of, $f(t) = \lambda e^{-\lambda t}$ and $g(t) = \mu e^{-\mu t}$, respectively, then Laplace transforms can be used to calculate the component unavailability to give:

$$q(t) = \frac{\lambda}{\lambda + \mu}\{1 - \exp[-(\lambda + \mu)t]\} \qquad (2.6)$$

2.3.3 Calculating the Top Event Probability

Calculating the top event probability, also known as, the system unavailability is a fundamental part of the quantification process. Three calculation procedures will be considered below in sections 2.3.3.1-2.3.3.3. Each procedure requires the identification of all the minimal cut sets during qualitative analysis for an exact calculation of the top event probability.

When calculating the top event probability for large fault trees approximations are often unavoidable for two reasons; firstly, it is not always possible to determine a full list of minimal cut sets due to the computational inefficiency of qualitative analysis. Secondly, the number of terms required to calculate the top event probability is dependent on the number of minimal cut sets. It is not possible to evaluate all of these terms for fault trees with a large number of minimal cut sets. These approximation methods are discussed in section 2.3.3.4.

2.3.3.1 The Structure Function Method

One method for calculating the top event probability uses the structure function, which was introduced in chapter one. Since the structure function, $\varphi(\underline{x})$, defines the state of the system it is also possible to express the structure function in terms of an indicator function ρ_i for the minimal cut sets, $C_i, i = 1,...,n$:

$$\varphi(\underline{x}) = 1 - \prod_{i=1}^{n} (1 - \rho_i) \tag{2.7}$$

If all the minimal cut sets are independent the probability of the top event can be obtained by taking the expectation of the structure function:

$$Q_{SYS}(t) = E[\varphi(\underline{x})] \tag{2.8}$$

In this case, $E[\varphi(\underline{x})] = \varphi[E(\underline{x})]$, however, when the minimal cut sets are not independent, $E[\varphi(\underline{x})] \neq \varphi[E(\underline{x})]$. To calculate the top event probability exactly the structure function must be fully expanded and reduced using the Idempotent law, $x_i x_i = x_i$, before the expectation is taken.

To demonstrate how this approach is used to calculate the top event probability, consider the following Boolean expression for the top event.

$$\text{Top} = ab + bc + ac \tag{2.9}$$

From equation (2.7) the structure function can be expressed as follows:

$$\varphi(\underline{x}) = 1 - (1 - x_a x_b)(1 - x_b x_c)(1 - x_a x_c)$$

Expanding the brackets and applying the Idempotent law gives the following result:

$$\varphi(\underline{x}) = x_a x_b + x_b x_c + x_a x_c - 2x_a x_b x_c$$

Supposing the probability of occurrence for each component is 0.1, the top event probability is obtained by taking the expectation of the structure function as follows:

$$Q_{SYS}(t) = E[\varphi(\underline{x})] = E[x_a x_b] + E[x_b x_c] + E[x_a x_c] - 2 \cdot E[x_a x_b x_c]$$
$$Q_{SYS}(t) = 0.01 + 0.01 + 0.01 - 0.002 = 0.028$$

This method is not an efficient means of calculating the top event probability when there are dependencies between the minimal cut sets. An alternative approach, which employs Shannon's theorem, will be considered in section 2.3.3.2.

2.3.3.2 Shannon's Theorem for Calculating the Top Event Probability

Shannon's theorem states:

A Boolean Function, $f(\underline{x})$, where, $\underline{x} = (x_1, x_2, \ldots, x_n)$ can be expressed as:

$$f(\underline{x}) = x_i f(1_i, \underline{x}) + \bar{x}_i f(0_i, \underline{x}) \tag{2.10}$$

Where:

$\bar{x}_i = 1 - x_i$

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

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

$f(1_i, \underline{x})$ and $f(0_i, \underline{x})$ are the residues of $f(\underline{x})$

Shannon's theorem [10] can be employed to calculate the top event probability of fault trees with repeated events more efficiently than the method considered in the previous section.

From equation (2.8):

$$Q_{SYS}(t) = E[\varphi(\underline{x})]$$

Now letting $q_i = E(x_i)$ and using Shannon's expansion in equation (2.10):

$$Q_{SYS}(t) = q_i E\{\varphi(1_i, \underline{q})\} + (1 - q_i) E\{\varphi(0_i, \underline{q})\} \qquad (2.11)$$

The expectation cannot be taken until there are no powers of indicator variables in the residues. Thus Shannon's expansion is used to expand the structure function with respect to the most repeated event until all powers of indicator variables in the residues are eliminated.

To demonstrate how this method is applied, consider the structure function for the top event, Top, given by equation (2.9):

$$\varphi(\underline{x}) = 1 - (1 - x_a x_b)(1 - x_b x_c)(1 - x_a x_c)$$

This expression contains repeated events, so Shannon's theorem is used to pivot the structure function. Firstly pivoting for event x_a:

$$\varphi(\underline{x}) = x_a[1 - (1 - x_b)(1 - x_b x_c)(1 - x_c)] + (1 - x_a)[1 - (1 - x_b x_c)]$$

There is still a repeated event, x_b, in the first term:

$$\varphi(\underline{x}) = x_a[x_b(1) + (1 - x_b)(x_c)] + (1 - x_a)[x_b x_c]$$

Now substituting in probabilities for each event, $P(x_a) = P(x_b) = P(x_c) = 0.1$, gives:

$$Q_{SYS}(t) = 0.1[0.1 + 0.9 \times 0.1] + 0.9 \times (0.1 \times 0.1) = 0.0028$$

2.3.3.3 The Inclusion-Exclusion Expansion Method

The inclusion-exclusion expansion is commonly used to calculate the top event probability. One advantage of this approach is that it produces the correct result for

trees with repeated events, provided the assumption that basic events are independent is appropriate.

Consider a fault tree with n minimal cut sets, $C_i, i = 1,...,n$. The top event exists if at least one minimal cut set exists:

$$Top = C_1 + C_2 + ... + C_n = \bigcup_{i=1}^{n} C_i \qquad (2.12)$$

The top event probability is calculated as follows:

$$Q_{SYS}(t) = P\left(\bigcup_{i=1}^{n} C_i\right) = P(C_1 + C_2 + ... + C_n) \qquad (2.13)$$

This result is expanded below:

$$Q_{SYS}(t) = \sum_{i=1}^{n} P(C_i) - \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(C_i \cap C_j) + ... + (-1)^{n-1} P(C_1 \cap C_2 \cap ... \cap C_n) \qquad (2.14)$$

This is known as the inclusion-exclusion expansion and if it is evaluated fully for any coherent fault tree it generates the exact top event probability.

To illustrate how the top event probability is calculated using the inclusion-exclusion expansion, consider again the expression for the top event given in equation (2.9). The top event probability, Q_{SYS} is given below:

$$Q_{SYS}(t) = P(Top) = P(a \cdot b + b \cdot c + a \cdot c)$$

Expanding this using the inclusion-exclusion expansion gives:

$$Q_{SYS}(t) = q_a q_b + q_b q_c + q_a q_c - q_a q_b q_c - q_a q_b q_c - q_a q_b q_c + q_a q_b q_c$$

Now suppose each basic event has probability of occurrence of 0.1, then the top event probability is:

$$Q_{SYS}(t) = (0.1)^2 + (0.1)^2 + (0.1)^2 - (0.1)^3 - (0.1)^3 - (0.1)^3 + (0.1)^3 = 0.0028$$

Although this approach generates an exact result, as shown by this simple example the evaluation of each term in the inclusion-exclusion expansion can be extremely computationally intensive even for moderate sized trees. For fault trees with hundreds or even thousands of minimal cut sets, full evaluation of this expansion is beyond the capabilities of even the most powerful computers. Consequently approximations that produce acceptably accurate results are essential.

2.3.3.4 Approximate Methods

In practise approximate methods for calculating the top event probability are unavoidable. Three of the main approximate methods for calculating the top event probability are considered below, these are: the Rare Event Approximation, the lower bound approximation and the Minimal Cut Set Upper Bound.

The Rare Event Approximation

The Rare Event Approximation is the simplest approximation that can be obtained for the top event probability. This approximation consists of just the first term of the inclusion-exclusion expansion and is an upper bound for the top event probability:

$$Q_{SYS}(t) \leq \sum_{i=1}^{n} P(C_i) \qquad (2.15)$$

The name Rare Event Approximation is given to this upper bound because it is only accurate if component failure events are rare.

Note: If $Q_{SYS}(t)$ is truncated with an odd number of terms, an upper bound for the top event probability will always be obtained.

Lower Bounds for the Top Event Probability

If the inclusion-exclusion expansion is truncated with an even number of terms a lower bound for the top event probability is obtained. The more terms that are evaluated the more accurate the approximation will be. One possible lower bound for

the top event probability is calculated by truncating the inclusion-exclusion formula after two terms:

$$\sum_{i=1}^{n} P(C_i) - \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(C_i \cap C_j) \leq Q_{SYS}(t) \qquad (2.16)$$

The Minimal Cut Set Upper Bound

The Minimal Cut Set Upper Bound is a more accurate upper bound than the Rare Event Approximation. In fact if all the minimal cut sets are independent the Minimal Cut Set Upper Bound calculates the exact top event probability. This upper bound is derived as follows:

System failure exists if at least one minimal cut sets exists:

$$P(\text{system failure}) = P(\text{at least one minimal cut set exists})$$

$$P(\text{System failure}) \leq 1 - \prod_{i=1}^{n} P(\text{no minimal cut set exists})$$

Thus:

$$Q_{SYS}(t) \leq 1 - \prod_{i=1}^{n} [1 - P(C_i)] \qquad (2.17)$$

The main disadvantage with all of these approximations is that they are only accurate when basic event failures are rare.

2.3.4 The Unconditional Failure Intensity

Another important system parameter to calculate during quantification is the unconditional failure intensity, $w_{SYS}(t)$. Since, having determined $w_{SYS}(t)$ the expected number of system failures in a given interval can be calculated by integrating $w_{SYS}(t)$. The unconditional failure intensity is defined as the probability that a system fails per unit time at t given that it was working at t=0. The top event will occur between t and $t+dt$ provided none of the minimal cut sets exists at t and one or more of the minimal cut sets occurs in the interval, $[t, t+dt)$. It is possible for more

than one minimal cut set to occur in the small time element dt, because a basic event can be common to more than one minimal cut set.

$$w_{SYS}(t)dt = P\left[A\bigcup_{i=1}^{n}\theta_i\right] \tag{2.18}$$

Where, A is the event that none of the minimal cut sets exist at time t, $A = \bigcap_{i=1}^{n} u_i$

And, u_i denotes the i^{th} minimal cut set does not exist at time t. And, $\bigcup_{i=1}^{n}\theta_i$ is the event that one or more minimal cut sets occur in the interval, $[t, t+dt)$.

Given $P(\overline{A}) = 1 - P(A)$ equation (2.18) can be expressed as follows:

$$w_{SYS}(t)dt = P\left[\bigcup_{i=1}^{n}\theta_i\right] - P\left[\overline{A}\bigcup_{i=1}^{n}\theta_i\right] \tag{2.19}$$

Considering the two terms of equation (2.19) separately. The first term represents the contribution from the occurrence of one or more minimal cut sets and will be denoted by, $w_{SYS}^{(1)}(t)$. The second term is a correction term representing the contribution of minimal cut sets occurring whilst other minimal cut sets already exist, it will be denoted by, $w_{SYS}^{(2)}(t)$. Equation (2.19) becomes:

$$w_{SYS}(t)dt = w_{SYS}^{(1)}(t)dt - w_{SYS}^{(2)}(t)dt \tag{2.20}$$

The first term, the occurrence of at least one minimal cut set can be expanded using the inclusion-exclusion expansion:

$$w_{SYS}^{(1)}(t)dt = \sum_{i=1}^{n} P(\theta_i) - \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j) + \ldots + (-1)^{n-1} P(\theta_1 \cap \ldots \cap \theta_n) \tag{2.21}$$

Where, $\sum_{i=1}^{n} P(\theta_i)$ is the sum of probabilities that minimal cut set i fails in the interval $[t, t+dt)$.

Since dt is a small time element only one basic event can occur in the interval $[t,t+dt)$. Hence, the occurrence of more than one minimal cut set in the interval $[t,t+dt)$ must be the result of the failure of a component common to all the failed minimal cut sets. In general if k is the number of common components in the m minimal cut sets:

If k=0

$$P[\theta_1 \cap \theta_2 \cap ... \cap \theta_m] = 0$$

If $k > 0$

$$P[\theta_1 \cap \theta_2 \cap ... \cap \theta_m] = w_A(t,B_1,....,B_k)dt \prod Q_A(\underline{B})$$

Where $\prod Q_A(\underline{B})$ is the product of the probabilities of the component failures that exist in at least one of the m minimal cut sets but that are not common to them all. And $w_A(t,B_1,...,B_k)$ is the failure intensity for a set, which consists of k common components.

The second term, which represents the contribution of minimal cut sets occurring whilst other minimal cut sets already exist, can also be expanded by the inclusion-exclusion expansion.

$$w_{SYS}^{(2)}(t)dt = \sum_{i=1}^{n} P(\theta_i \cap \bar{A}) - \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j \cap \bar{A}) + ... + (-1)^{n-1} P(\theta_1 \cap ... \theta_n \cap \bar{A})$$

(2.22)

Since $\bar{A} = \bigcup_{i=1}^{n} \bar{u}_i$ each term in equation (2.22) can be expanded again, a general term from this equation is:

$$P(\theta_1 \cap ... \cap \theta_m \cap \bar{A}) = \sum_{i=1}^{n} P(\theta_1 \cap ... \cap \theta_m \cap \bar{u}_i)$$
$$- \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(\theta_1 \cap ... \cap \theta_m \cap \bar{u}_i \cap \bar{u}_j) + ...$$
$$+ (-1)^{n-1} P(\theta_1 \cap ... \cap \theta_m \cap \bar{u}_1 \cap \bar{u}_2 \cap ... \cap \bar{u}_n)$$

(2.23)

Where θ_i denotes the occurrence of minimal cut set i in the interval $[t,t+dt)$ and $\bar{u}_i$ denotes the existence of minimal cut set i at t.

The general term $P(\theta_1,...,\theta_m,\bar{u}_1,...,\bar{u}_k)$ denotes the probability that minimal cut sets 1 to k exist at time t and minimal cut sets 1 to m occur in the interval $[t,t+dt)$. Thus the general term given in equation (2.23) can be expressed as follows:

$$P(\theta_1 \cap ... \cap \theta_m \cap \bar{u}_1 \cap ... \cap \bar{u}_k) = w_B(t,B_1,B_2,...,B_r)dt\prod Q_B(\underline{B}) \quad (2.24)$$

Where $w_B(t,B_1,B_2,...,B_r)$ is the failure intensity for a set of all component failures, which are common to all of the minimal cut sets $\theta_1,\theta_2,...,\theta_m$ but not in sets $\bar{u}_1,\bar{u}_2,...,\bar{u}_k$. And $\prod Q_B(\underline{B})$ is the product of the probabilities of component failures, which exist in any minimal cut set $\bar{u}_j$ together with component failures that exist in one or more of the minimal cut sets $\theta_1,\theta_2,...,\theta_m$ but are not common to them all.

The expected number of system failures in time t, which is denoted by, $W_{SYS}(0,t)$ can be calculated by evaluating the integral of the system unconditional failure intensity over the specified time interval.

$$W_{SYS}(0,t) = \int_0^t w_{SYS}(u)du \quad (2.25)$$

To illustrate how the unconditional failure intensity and the expected number of system failures are calculated, consider a fault tree with three minimal cut sets, $\{a,b\},\{b,c\}\{a,c\}$. Table 2.1 summarises the failure probabilities and the unconditional failure intensities of the basic events.

Basic Event	Failure Probability	Unconditional Failure Intensity
A	0.01	6×10^{-5}
B	0.01	5×10^{-4}
C	0.01	3×10^{-3}

Table 2.1: Component Failure Probabilities and Unconditional Failure Intensities

Firstly the minimal cut set parameters are calculated. The unconditional failure intensity of a minimal cut set C_1 is calculated as follows:

$$w_{C_1} = \sum_{j=1}^{n}\left(w_j \prod_{\substack{i=1 \\ i\neq j}}^{n} q_i \right) \tag{2.26}$$

Also, the probability that a minimal cut set exists at time t is calculated using equation (2.27):

$$q_{C_1} = \prod_{i=1}^{n} q_i \tag{2.27}$$

The following results are obtained for the three minimal cut sets, $\{a,b\},\{b,c\}\{a,c\}$:

$$w_{C_1} = w_a q_b + w_b q_a \qquad q_{C_1} = q_a q_b$$

$$w_{C_2} = w_b q_c + w_c q_b \qquad q_{C_2} = q_b q_c$$

$$w_{C_3} = w_a q_c + w_c q_a \qquad q_{C_3} = q_a q_c$$

Calculating $w_{SYS}^{(1)}(t)$:

$$w_{SYS}^{(1)}(t)dt = \sum_{i=1}^{3} P(\theta_i) - \sum_{i=2}^{3}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j) + P(\theta_1 \cap \theta_2 \cap \theta_3)$$

Given:

$$\sum_{i=1}^{3} P(\theta_i) = \sum_{i=1}^{3} w_{C_i} dt = (w_a q_b + w_b q_a + w_b q_c + w_c q_b + w_a q_c + w_c q_a)dt$$

$$\sum_{i=2}^{3}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j) = (w_b q_a q_c + w_a q_b q_c + w_c q_a q_b)dt$$

$$P(\theta_1 \cap \theta_2 \cap \theta_3) = 0$$

Hence:

$$w_{SYS}^{(1)}(t) = w_a(q_b + q_c - q_b q_c) + w_b(q_a + q_c - q_a q_c) + w_c(q_a + q_b - q_a q_b)$$

Now calculating $w_{SYS}^{(2)}(t)$:

$$w_{SYS}^{(2)}(t)dt = \sum_{i=1}^{3} P(\theta_i \cap \overline{A}) - \sum_{i=2}^{3}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j \overline{A}) + P(\theta_1 \cap \theta_2 \cap \theta_3 \cap \overline{A})$$

Given:

$$\sum_{i=1}^{3} P\left(\theta_i \cap \bar{A}\right) = \left(2w_a q_b q_c + 2w_b q_a q_c + 2w_c q_a q_b\right)dt$$

$$\sum_{i=2}^{3}\sum_{j=1}^{i-1} P\left(\theta_i \cap \theta_j \cap \bar{A}\right) = \left(w_a q_b q_c + w_b q_a q_c + w_c q_a q_b\right)dt$$

$$P\left(\theta_1 \cap \theta_2 \cap \theta_3 \cap \bar{A}\right) = 0$$

Hence:

$$w_{SYS}^{(2)}(t) = w_a q_b q_c + w_b q_a q_c + w_c q_a q_b$$

Thus from equation (2.20):

$$w_{SYS}(t) = w_a(q_b + q_c - 2q_b q_c) + w_b(q_a + q_c - 2q_a q_c) + w_c(q_a + q_b - 2q_a q_b)$$

Substituting in the failure probabilities and unconditional failure intensities of the basic events given in table 2.1, the following result is obtained for the unconditional failure intensity.

$$w_{SYS}(t) = 7.07 \times 10^{-5}$$

Having calculated the unconditional failure intensity the expected number of system failures in a given time period, say one year (8760 hours) can be calculated as follows.

$$W_{SYS}(0,8760) = \int_0^{8760} w_{SYS}(t)dt = \left(7.07 \times 10^{-5}\right) \times 8760 = 0.619$$

2.4 Evaluating the Fault Tree Methodology

FTA has both advantages and disadvantages. The main advantage of the technique is that it assesses each mode of system failure systematically producing a fault tree, which yields a complete diagrammatic description of the particular system failure mode. Having obtained the fault tree, both qualitative and quantitative analysis can be performed. This enables the analyst to identify the exact causes of system failure (minimal cut sets), and then calculate the probability of occurrence and the frequency of system failure.

However, a major disadvantage of this technique is that it can be inefficient, even the most powerful computers may not be able to obtain all the minimal cut sets, especially for large trees with repeated events. Culling techniques can be employed to overcome this problem but they in turn have the disadvantage of producing only a partial list of minimal cut sets. This leads to problems in quantification, since the minimal cut sets are required to quantify the system. If approximate methods are employed during quantification, results can be misleading and inaccurate unless component failures are rare.

An alternative technique known, as the Binary Decision Diagram technique has been developed to overcome the deficiencies of FTA. This technique has been found to be more efficient in terms of qualitative analysis, and also enables exact quantification of the system parameters. This technique will be the focus of chapters three and five.

2.5 Summary

Whilst FTA is a systematic technique for the analysis of a particular mode of system failure, the procedures for both qualitative and quantitative analysis are inefficient, making approximations unavoidable. These approximations can be inaccurate. The Binary Decision Diagram technique offers an alternative means of qualitative and quantitative analysis, which has been found to be more efficient and exact.

Chapter 3:The Binary Decision Diagram Method for the Analysis of Coherent Fault Trees

3.1 Introduction

FTA is an extremely useful technique for assessing the risk and reliability of a wide variety of systems. However, as highlighted in chapter two the conventional techniques for FTA have major limitations in terms of both efficiency and accuracy, making them cumbersome even for moderate sized fault trees.

Refinements to these techniques have resulted in improvements to the efficiency and accuracy, but it seems unlikely that any further developments will result in significant improvements. More recently attention has focused on an alternative technique called the Binary Decision Diagram (BDD) method. Rauzy first introduced this technique for the purposes of reliability analysis in 1993 [3]. It has been shown to be extremely efficient in terms of qualitative analysis and accurate in terms of quantitative analysis, eliminating the need for approximations.

The BDD technique first converts the fault tree diagram into a BDD format, known as the SFBDD because it encodes the structure function of the fault tree. Both qualitative and quantitative analysis can be performed using the SFBDD. The fault tree to BDD conversion process itself is both straightforward and efficient, however, the technique does have two main disadvantages.

The first is that it is necessary to choose a variable ordering for the conversion process. This variable ordering determines the size of SFBDD obtained, and the size of the SFBDD is critical to the efficiency of both stages of analysis. The second major disadvantage is that the BDD method cannot always be used to analyse a fault tree exactly, because computing a SFBDD can be unmanageable for extremely large fault trees. Rauzy has developed techniques for computing a culled SFBDD, which can be used to perform partial analysis. These techniques will be considered in chapter eight.

3.2 An Introduction to Binary Decision Diagrams

A BDD is a directed acyclic graph. Thus all paths through the BDD are directed in one straight route from the top node known as the **Root Vertex** through **Non-Terminal Vertices** until a **Terminal Vertex** is reached. All paths terminate in one of two states, 1, corresponding to system failure or, 0, corresponding to the system functioning. Non-terminal vertices represent system components and are connected to other vertices by branches. Each non-terminal vertex has a one branch, corresponding to component failure and a zero branch corresponding to the component functioning.

All the **paths** through the SFBDD terminating in a 1 state represent a cut set of the fault tree. Figure 3.1 highlights the features of a SFBDD introduced here.

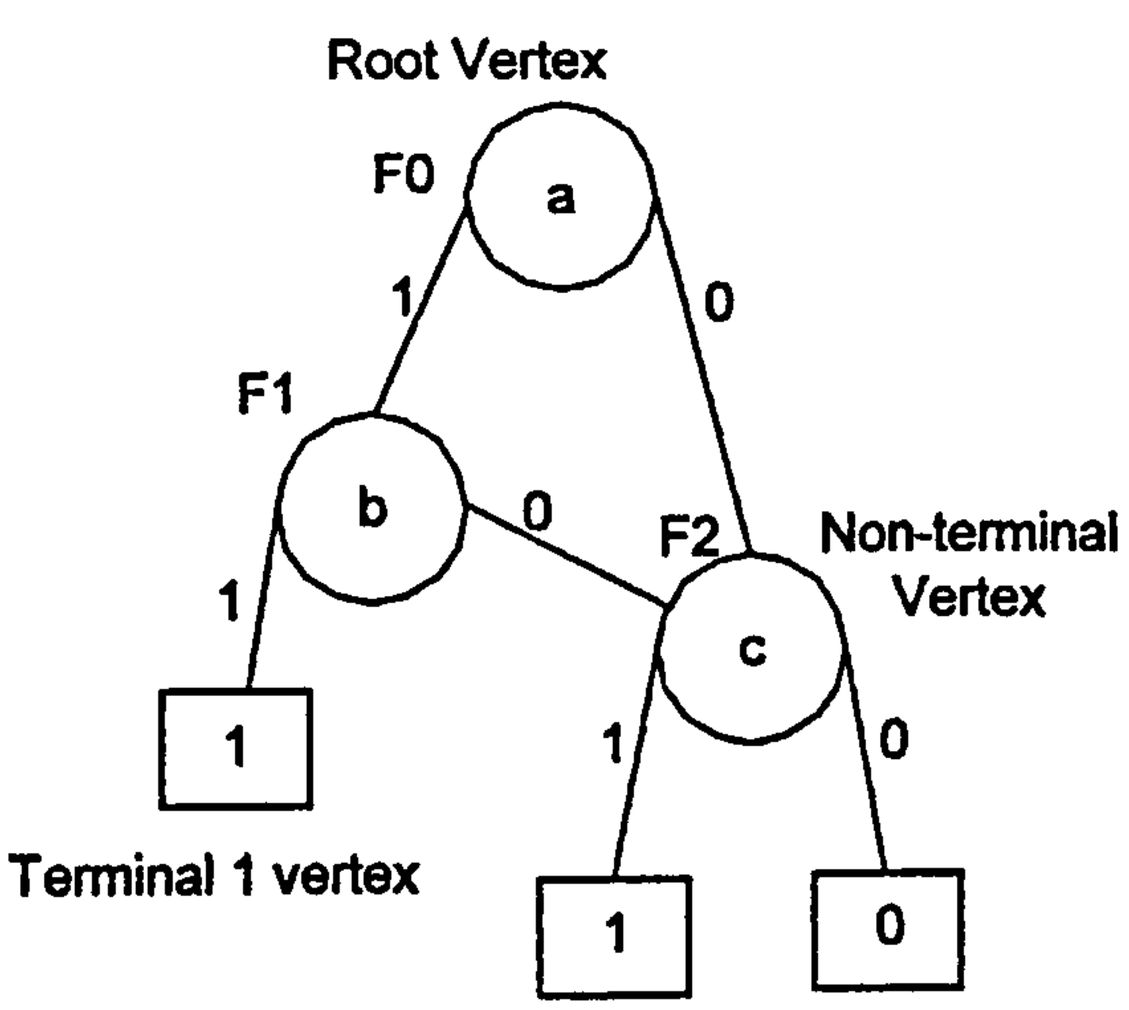


Figure 3.1: A General SFBDD

Notice that node F2 is shared by the zero branch of node F0 and the zero branch of node F1, this is known as sub-node sharing and is an important feature of BDD's since it can help to reduce the size of the BDD.

3.3 The Conversion Process

The BDD technique requires the fault tree to be converted to a SFBDD; there are two main methods for this conversion process. The first method requires knowledge of the logic function and is considered in section 3.3.2. The second method, which is the preferred of the two for computer implementation, employs an If-Then-Else technique and is considered in section 3.3.3. Each of these methods requires a variable ordering scheme to be chosen for the conversion process to proceed.

3.3.1 Choosing a Variable Ordering Scheme

The construction process begins with choosing a variable ordering scheme. The chosen scheme determines the size of the resulting SFBDD, which in turn has an impact on both qualitative and quantitative analysis; the smaller (more minimal) the SFBDD, the more efficient the analysis. A "good" ordering scheme can produce a minimal or near minimal SFBDD, however a "bad" ordering scheme can result in an extremely large non-minimal SFBDD or at worst it may not be possible to produce a SFBDD at all in which case either a culling technique must be employed or an alternative means of analysis must be found. It is for this reason that choosing a variable ordering scheme is such a critical part of the conversion process.

Many ordering schemes have been developed including the "Top-Down" approach, and the "Depth first" approach. However, there is no universal ordering scheme that produces a minimal SFBDD for all fault trees. Attention is now focused on choosing the "best" ordering scheme from a number of possible schemes; Bartlett & Andrews [11] demonstrated that neural networks can be used with success to predict a "good" ordering scheme.

3.3.2 The Logic Function Method

This method for computing the SFBDD requires knowledge of the logic function of the fault tree. Then the conversion process involves substituting the value of one (component fails) and then zero (component working) for each vertex in the SFBDD according to the chosen variable ordering scheme, and simplifying the result where possible using the Boolean algebra laws introduced in chapter two. To illustrate this method consider a fault tree which has the logic function given below.

$$Top = abd + cd + ce$$

Assuming the variable ordering, $a < b < c < d < e$, which means that variable a is considered first in the conversion process followed by variable b, then c and so on until all the variables have been considered.

Thus beginning with variable a and assigning the value one to this variable within the logic function the Boolean equation representing the one branch of this vertex is obtained.

$$Top(1_a, \underline{x}) = bd + cd + ce$$

The zero branch for this vertex is obtained by assigning the value zero to variable a within the logic function.

$$Top(0_a, \underline{x}) = cd + ce$$

This process is repeated for each variable in turn according to the ordering scheme adopted. The resulting SFBDD is shown in figure 3.2.

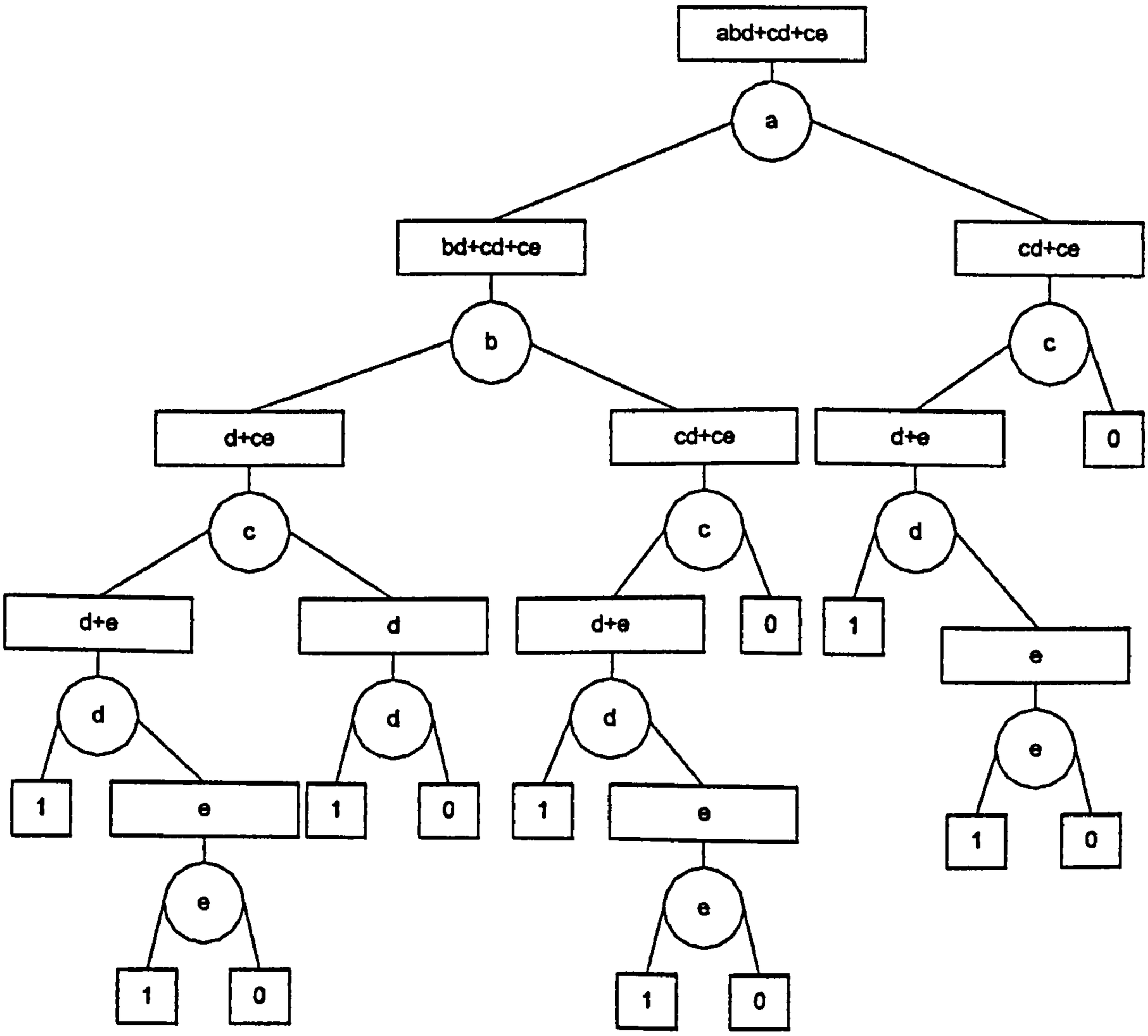


Figure 3.2: SFBDD Obtained Using the Logic Function Method

The SFBDD in figure 3.2 is quite large and it can be seen that there are equivalent nodes within the structure. Friedman and Supowit [12], proposed two collapsing operations that can be used to reduce the size of the BDD.

1. If the two sons of a node A are equivalent, then delete node A and direct all of its incoming edges to its left son.
2. If nodes A and B are equivalent, then delete node B and direct all of its incoming edges to A.

Applying Friedman and Supowit's second collapsing operation and removing the intermediate logic functions results in the simplified SFBDD shown in figure 3.3 from which seven cut sets are identified:

{a,b,c,d}, {a,b,c,e}, {a,b,d}, {a,c,d}, {a,c,e}, {c,d}, {c,e}

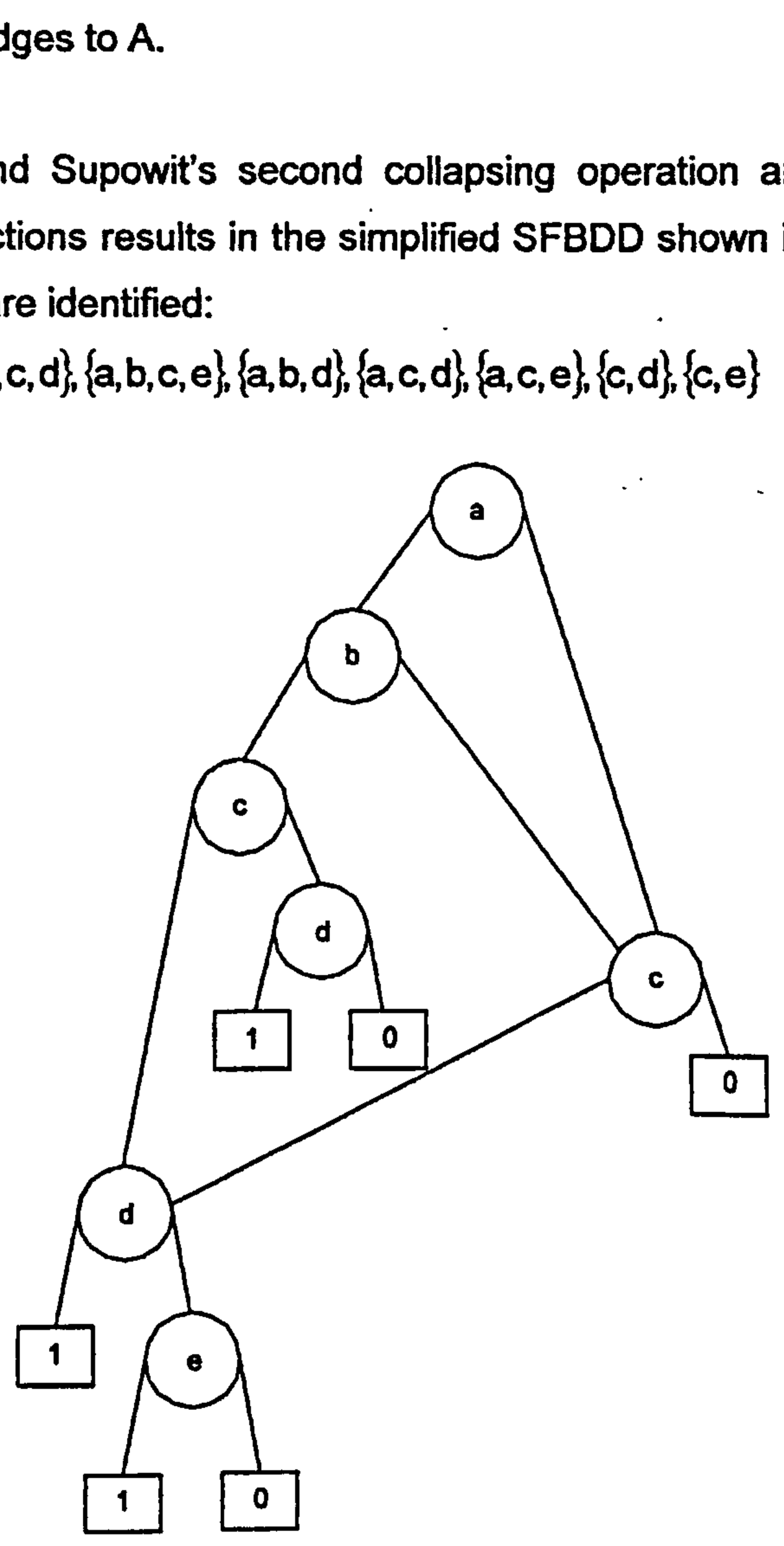


Figure 3.3: Simplified SFBDD

Obtaining the minimal cut sets from the SFBDD will be considered in detail in section 3.4. Although the logic function method for computing the SFBDD is quite straightforward to apply manually, it is not a particularly efficient means of obtaining the SFBDD and it is not easy to implement on a computer.

3.3.3 The If-Then-Else Method

Rauzy developed the If-Then-Else (**ite**) method for computing a SFBDD in 1993. This method has the advantage of producing a SFBDD, which encodes Shannon's formula, i.e. If $f(\underline{x})$ is the Boolean function for the top event of the fault tree, then by pivoting about any variable x_i, Shannon's theorem states:

$$f(\underline{x}) = x_i f_1 + \bar{x}_i f_2$$

Where, f_1 and f_2 are Boolean functions.

This is represented by the **ite** structure given in equation (3.1):

$$ite(x_i, f_1, f_2) \tag{3.1}$$

Where, x_i represents a variable and f_1 and f_2 represent logic functions. This **ite** structure is interpreted as follows:

If x_i fails **then** consider the logic function f_1

else consider the logic function f_2.

Thus in the BDD, f_1 forms the logic function for the one branch of x_i and f_2 forms the logic function for the zero branch of x_i. Figure 3.4 shows the diagram that represents this **ite** structure.

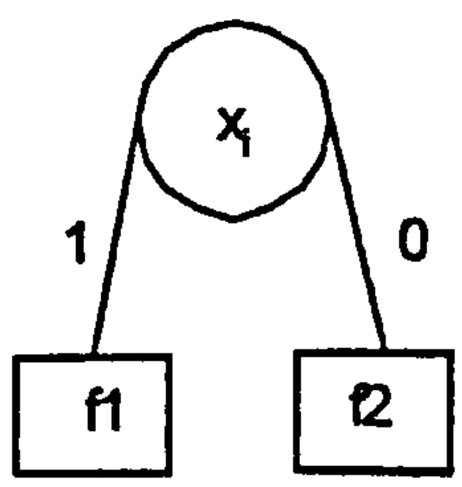


Figure 3.4: **ite** structure for $ite(x_i, f1, f2)$

3.3.3.1 The Conversion Procedure

Once a variable ordering scheme has been selected the **ite** procedure outlined below is applied to compute a SFBDD.

1. Assign each basic event x_i in the fault tree an **ite** structure, $\text{ite}(x_i,1,0)$.
2. Modify the fault tree structure so that each gate has only two inputs.
3. Consider each gate in a bottom-up fashion.
4. If the two gate inputs are J and H such that:

$$J = \text{ite}(x,F1,F2) \qquad H = \text{ite}(y,G1,G2)$$

Then the following rules are applied:

- If $x < y$, $J < op > H = \text{ite}(x, F1 < op > H, F2 < op > H)$
- If $x = y$, $J < op > H = \text{ite}(x, F1 < op > G1, F2 < op > G2)$

These rules are used in conjunction with the following identities:

$1 < op > H = H$ if $< op >$ is an AND gate
$0 < op > H = 0$ if $< op >$ is an AND gate
$1 < op > H = 1$ if $< op >$ is an OR gate
$0 < op > H = H$ if $< op >$ is an OR gate

Where $< op >$ describes the Boolean operation of the logic gates of the fault tree. For an AND gate $< op >$ is the dot product (.) and for an OR gate $< op >$ is the sum symbol (+).

To illustrate the **ite** procedure, consider the fault tree shown in figure 3.5.

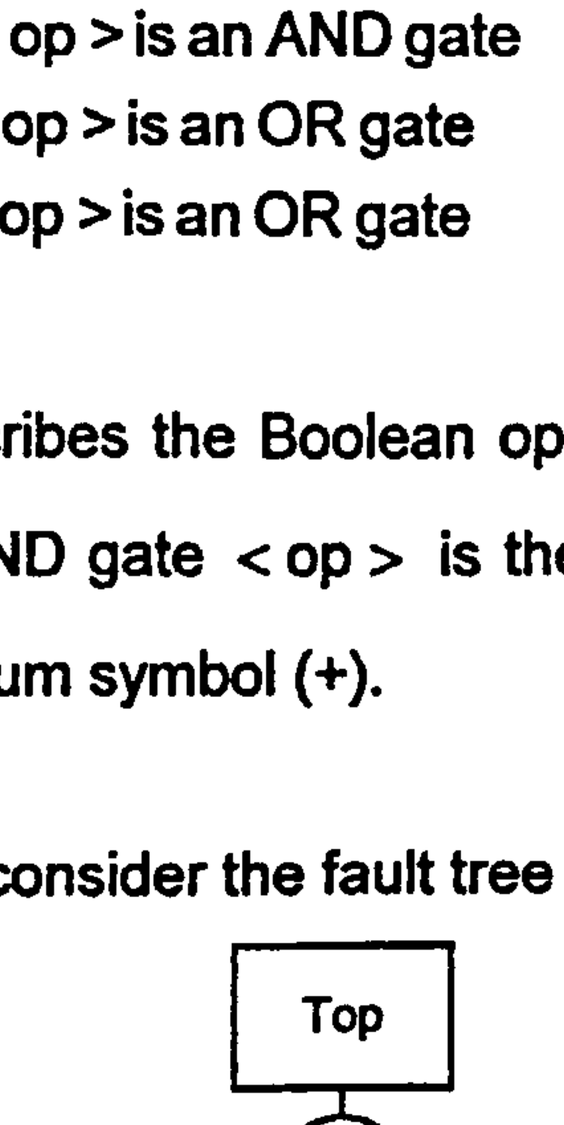


Figure 3.5: Fault Tree Diagram

Assuming a Depth first variable ordering $a < b < c$:

Assigning each variable an **ite** structure:

$$a = ite(a,\ 1,\ 0)$$
$$b = ite(b,\ 1,\ 0)$$
$$c = ite(c,\ 1,\ 0)$$

Now, computing the **ite** structure of gate G1:

$$G1 = a + b$$
$$G1 = ite(a,\ 1,\ 0) + ite(b,\ 1,\ 0)$$
$$G1 = ite(a, [1 + ite(b,\ 1,\ 0)], [0 + ite(b,\ 1,\ 0)])$$
$$G1 = ite(a,\ 1,\ ite(b,\ 1,\ 0)) \qquad \text{Since } 1 + H = 1$$

Finally dealing with the top gate, Top:

$$Top = G1 \cdot c$$
$$Top = ite(a,\ 1,\ ite(b,\ 1,\ 0)) \cdot ite(c,\ 1,\ 0)$$
$$Top = ite(a, [1 \cdot ite(c,\ 1,\ 0)], [ite(b,\ 1,\ 0) \cdot ite(c,\ 1,\ 0)])$$

The one branch for a is simple:

$$1 \cdot ite(c,\ 1,\ 0) = ite(c,\ 1,\ 0) \qquad \text{Since } 1 \cdot H = H$$

But the zero branch of a needs to be further developed:

$$ite(b,\ 1,\ 0) \cdot ite(c,\ 1\ 0) = ite(b, [1 \cdot ite(c,1,0)], [0 \cdot ite(c,1,0)])$$
$$= ite(b,\ ite(c,\ 1,\ 0),\ 0) \qquad \text{Since } 1 \cdot H = H \text{ and } 0 \cdot H = 0$$

Thus the **ite** structure for the fault tree shown in figure 3.5 is given in equation (3.2):

$$ite(a,\ ite(c,\ 1,\ 0),\ ite(b,\ ite(c,\ 1,\ 0),\ 0)) \qquad (3.2)$$

The resulting SFBDD is shown in figure 3.6, two cut sets are identified:

$$\{a,c\}, \{b,c\}$$

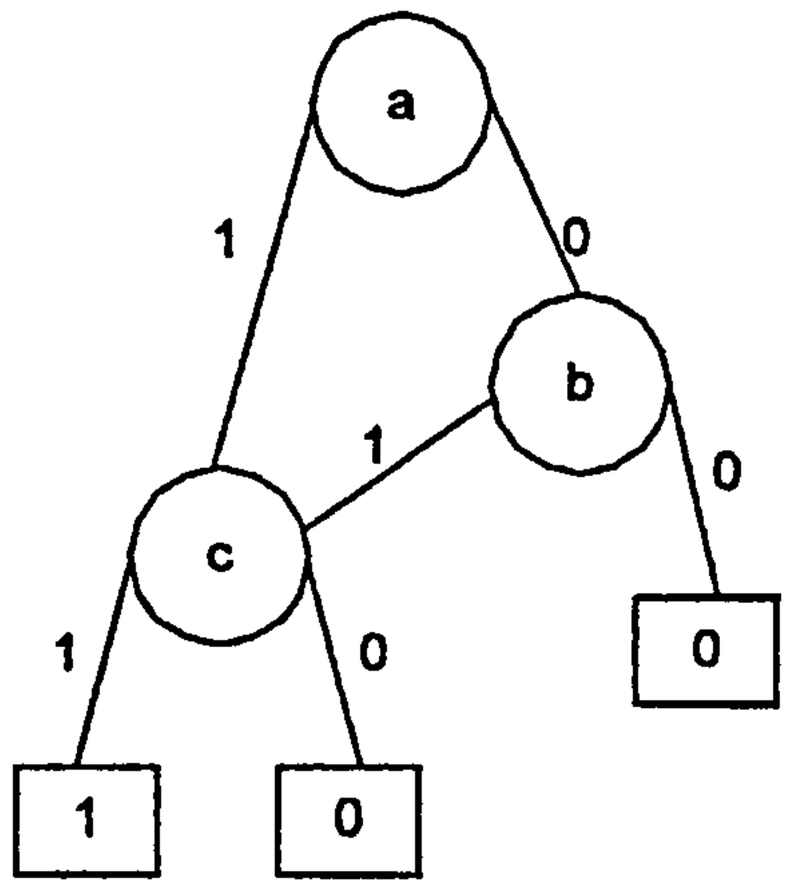


Figure 3.6: SFBDD Obtained for the Fault Tree in Figure 3.5

Although this conversion process is straightforward and easy to implement, for extremely large fault trees it is not always possible to compute a SFBDD. If a SFBDD cannot be computed, the fault tree cannot be analysed exactly and a culling technique must be employed to compute a culled SFBDD.

Until recently there was no way of using the BDD technique to perform approximate analysis, this discouraged the majority of industries from using the BDD technique. Whilst conventional techniques for FTA are not as efficient or accurate as the BDD method, they always produce at least a partial list of minimal cut sets and thus enable approximate quantification.

However, Rauzy developed techniques for computing a culled SFBDD, from which a partial list of minimal cut sets can be identified and the system can be quantified approximately. These techniques will be considered in detail in chapter eight.

3.4 Qualitative Analysis

During qualitative analysis all of the minimal cut sets of the fault tree are identified, where a minimal cut set is defined as a necessary and sufficient combination of component failures that will cause the system to fail.

Although qualitative analysis can be performed directly on the SFBDD, it is only possible to identify the minimal cut sets exactly if the SFBDD is minimal. If the SFBDD is non-minimal, redundancies will exist in the cut sets identified. The Boolean algebra laws introduced in chapter two can be applied to eliminate any redundancies,

but this can be quite computationally intensive. Rauzy [3] has developed a more efficient means of identifying the minimal cut sets, which involves modifying the SFBDD structure so that it encodes only the minimal cut sets of the fault tree. This process is called minimising the SFBDD.

3.4.1 Minimising the SFBDD

Rauzy developed an algorithm, which can be used to minimise the SFBDD. This algorithm modifies the structure of the SFBDD so that all redundancies are eliminated. The resulting BDD encodes the minimal cut sets of the fault tree exactly.

This algorithm is applied to every node in the SFBDD, for a general node the algorithm states:

If the output of a node is represented by the function F, where, $F = \text{ite}(x,G,H)$, let δ be a minimal solution of G, which is **not** a minimal solution of H, then the intersection of δ and x will be a minimal solution of F given by $F_{min} = \{\delta\} \cap x$. The set of all minimal solutions of F, $\text{sol}_{min}(F)$ will also include the minimal solutions of $H(\text{sol}_{min}(H))$ so:

$$\text{sol}_{min}F = [\{\delta\} \cap x] \cup [\text{sol}_{min}(H)]$$

There are essentially three stages to minimising $F = \text{ite}(x,G,H)$:

1. Identify the minimal solution of G, G_{min}.
2. Identify the minimal solution of H, H_{min}.
3. Remove all minimal solutions of G_{min} that are also solutions of H_{min}, $\text{without}(G_{min}, H_{min})$.

The third stage of the algorithm is crucial since, if a minimal solution of H, say y is retained on the one branch as a minimal solution of G it will result in a non-minimal combination $y \cdot x$. To enable the minimisation procedure to be coded Rauzy developed two algorithms called 'minsol' and 'without'. 'Minsol' computes the minimal solution of a node and calls 'without' to compute δ, i.e. the minimal solutions of G that are not minimal solutions of H. These algorithms are outlined in appendix (I).

Once the SFBDD has been minimised the minimal cut sets can be identified through just one pass of the minimised BDD. Every path through the minimised BDD that terminates in a 1 (failed) state represents a minimal cut set. Each path begins at the root vertex and proceeds through the BDD until a terminal vertex is reached. Only those vertices that lie on the one branch on the way to a terminal 1 vertex are included in the minimal cut set.

To illustrate how the minimisation procedure is applied consider the SFBDD shown in figure 3.7.

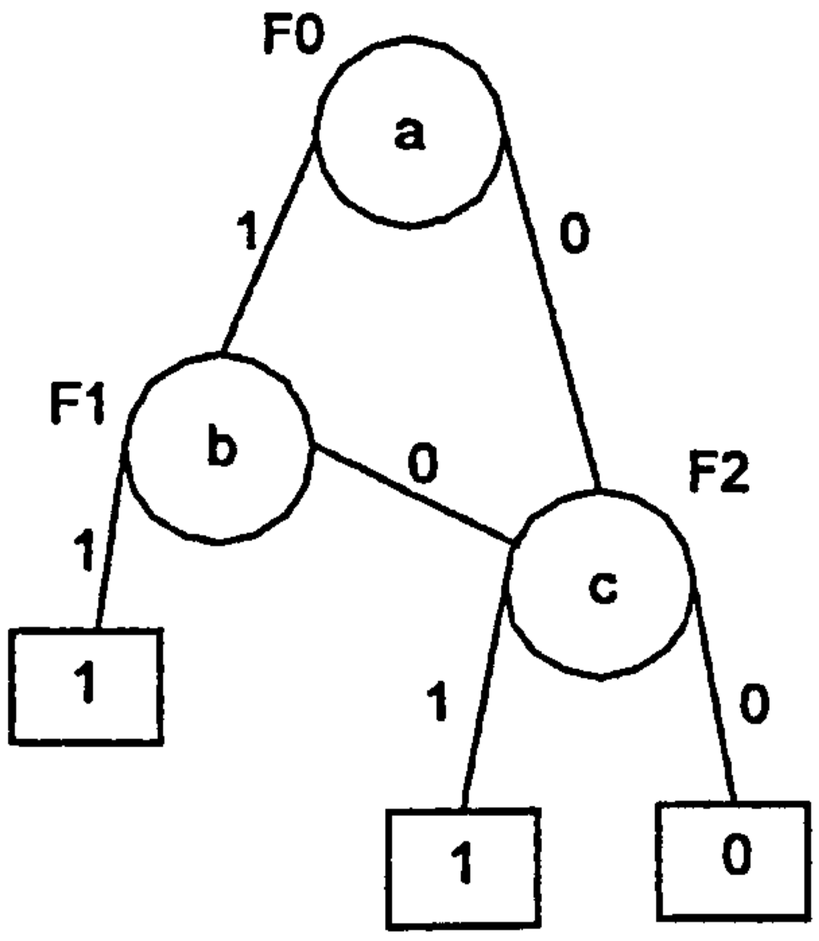


Figure 3.7: Non-minimal BDD

Considering the nodes in a top-down order. For the root vertex node F0, the one branch leads to node F1 that corresponds to G. Since the solutions of node F1 are minimal, the minimal solutions of G are b and c. The zero branch of F0 corresponding to H leads to node F2. Again the solutions of F2 are already minimal, thus the minimal solution of H is c.

The solution c is a minimal solution of both G and H. If it is retained on the one branch of F0 it will result in a non-minimal combination. Therefore all paths that include the solution c are removed from F1. This is done by removing F2 from the zero branch of F1 and replacing it with a terminal 0 vertex. This procedure is repeated for all the nodes in the SFBDD. In this case the SFBDD has been minimised and the minimised BDD is given in figure 3.8, two minimal cut sets are identified from this BDD:

$$\{a,b\},\{c\}$$

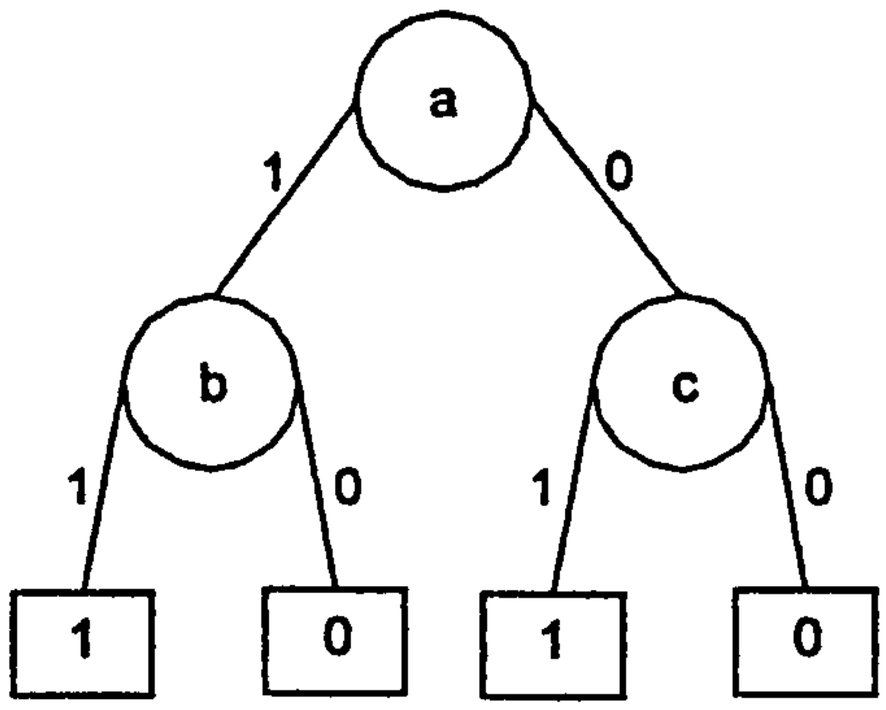


Figure 3.8: Minimised BDD

3.5 Quantitative Analysis

The BDD method does not require knowledge of the minimal cut sets to perform quantitative analysis and it also eliminates the need to evaluate lengthy series expansions. As such it provides an extremely efficient and accurate means of quantification. Andrews and Sinnamon developed procedures for quantifying the system exactly using the SFBDD [13]. The minimised BDD cannot be used for quantification since the minimisation process changes the logic function of the SFBDD to encode only the minimal cut sets. The procedures for calculating both the system unavailability and the unconditional failure intensity will be considered in sections 3.5.1 and 3.5.2 respectively.

3.5.1 Calculating the System Unavailability

The BDD method converts the fault tree diagram into a structure that encodes the structure function in the form of Shannon's decomposition; hence it is possible to calculate the system unavailability directly from this BDD.

The system unavailability is defined as the probability that the system is in a failed state at time t and it can be calculated by taking the expectation of the structure function for the top event, $F(\underline{x})$. The structure function can be expressed according to Shannon's decomposition as shown in equation (3.3):

$$F(\underline{x}) = x_i F_1(x_1, x_2, .., x_{i-1}, 1, x_{i+1}, ..., x_n) + \bar{x}_i F_2(x_1, x_2, ..., x_{i-1}, 0, x_{i+1}, ...x_n) \tag{3.3}$$

Thus by taking the expectation of equation (3.3), the top event probability is obtained:

$$Q_{SYS}(t) = E[x_i] \cdot E[F_1(\underline{x})] + E[\overline{x_i}] \cdot E[F_2(\underline{x})] \tag{3.4}$$

Consider a general SFBDD, each of the terminal one paths through the SFBDD represent combinations of component failure states and component working states that result in system failure. Since these paths are mutually exclusive, (disjoint), the top event probability can be obtained by summing the probabilities of all the disjoint paths through the SFBDD. Each disjoint path begins at the root vertex and terminates in a failed (1) state. The path includes all working and failed component states encountered.

To illustrate how the top event probability is calculated consider the SFBDD in figure 3.9, the disjoint paths though the SFBDD are:

$$\{a,b,c,e\}, \{a,b,\bar{c},d\}, \{a,\bar{b},c,d\}, \{a,\bar{b},c,\bar{d},f\}, \{\bar{a},c,d\}, \{\bar{a},c,\bar{d},f\}$$

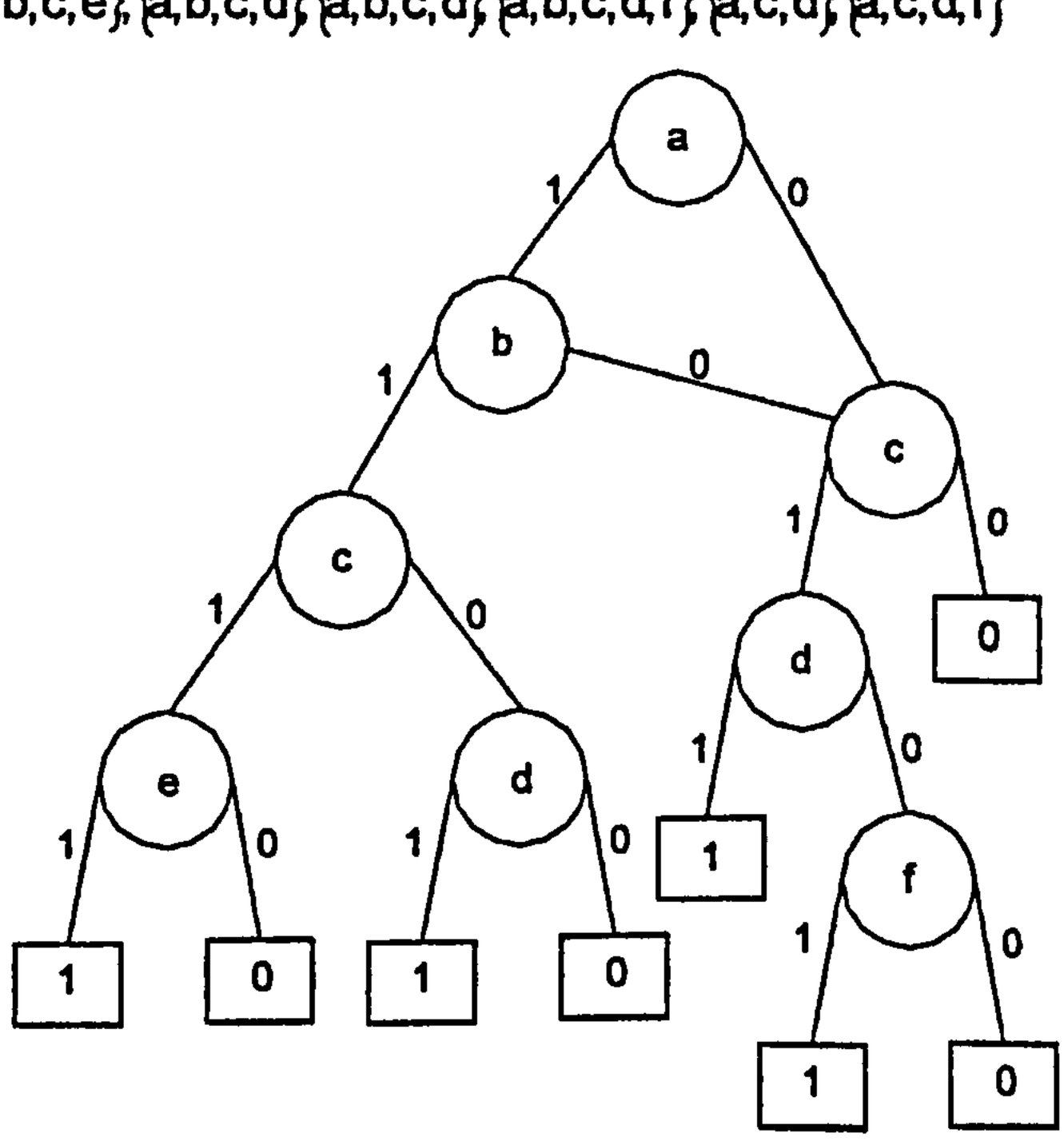


Figure 3.9: SFBDD

The top event probability is obtained by summing the probabilities of the disjoint paths:

$$\begin{aligned} Q_{SYS}(t) &= P(abce + ab\bar{c}d + a\bar{b}cd + a\bar{b}c\bar{d}f + \bar{a}cd + \bar{a}c\bar{d}f) \\ &= q_a q_b q_c q_e + q_a q_b (1-q_c) q_d + q_a (1-q_b) q_c q_d + q_a (1-q_b) q_c (1-q_d) q_f \\ &\quad + (1-q_a) q_c q_d + (1-q_a) q_c (1-q_d) q_f \end{aligned}$$

Where q_i denotes the unavailability of component i.

3.5.2 Calculating the Unconditional Failure Intensity

The unconditional failure intensity, $w_{SYS}(t)$ is defined as the probability per unit time that the system fails at time t and is a key quantitative measure. Having calculated the unconditional failure intensity it is possible to calculate the expected number of system failures in a given interval $[0,t)$ by integrating $w_{SYS}(t)$ over, 0 to t.

The unconditional failure intensity can be expressed in terms of the criticality function:

$$w_{SYS}(t) = \sum_{i=1}^{n_C} G_i(\underline{q}) w_i(t) \tag{3.5}$$

Where, $w_i(t)$ is the failure intensity for component i and $G_i(\underline{q})$ is the criticality function for component i.

The criticality function is defined as the probability that the system is in a critical state for component i such that the system will fail if component i fails.

$$G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q}) \tag{3.6}$$

The two terms on the right hand side of equation (3.6) can be calculated from the SFBDD, and the calculation procedure is more efficient using the SFBDD than the fault tree since only one pass of this BDD is required as opposed to two passes of the fault tree.

Sinnamon and Andrews [13] developed a procedure for calculating the criticality function using the SFBDD. This procedure is outlined below:

$$Q_{SYS}(1_i, \underline{q}) = \sum_{i=1}^{n} Pr_{x_i}(\underline{q}) \cdot Po^1_{x_i}(\underline{q}) + Z(\underline{q}) \tag{3.7}$$

$$Q_{SYS}(0_i, \underline{q}) = \sum_{i=1}^{n} Pr_{x_i}(\underline{q}) \cdot Po^0_{x_i}(\underline{q}) + Z(\underline{q}) \tag{3.8}$$

Where:

$Pr_{x_i}(\underline{q})$ is the probability of the path section from the root vertex to node x_i.

$Po^1_{x_i}(\underline{q})$ is the probability of the path section from the one branch of node x_i to a

terminal 1 node (excluding the probability of x_i).

$Po^0_{x_i}(\underline{q})$ is the probability of the path section from the zero branch of the node x_i to a terminal 1 node (excluding the probability of x_i).

$Z(\underline{q})$ is the probability of the paths from the root node to a terminal 1 node not passing the node for variable x_i.

Hence the criticality function is expressed as follows:

$$G_i(\underline{q}) = \sum_{i=1}^{n} Pr_{x_i}(\underline{q})\left[Po^1_{x_i}(\underline{q}) - Po^0_{x_i}(\underline{q})\right] \qquad (3.9)$$

To illustrate how this procedure is applied in practise to calculate both the criticality function and the unconditional failure intensity consider the SFBDD shown in figure 3.10.

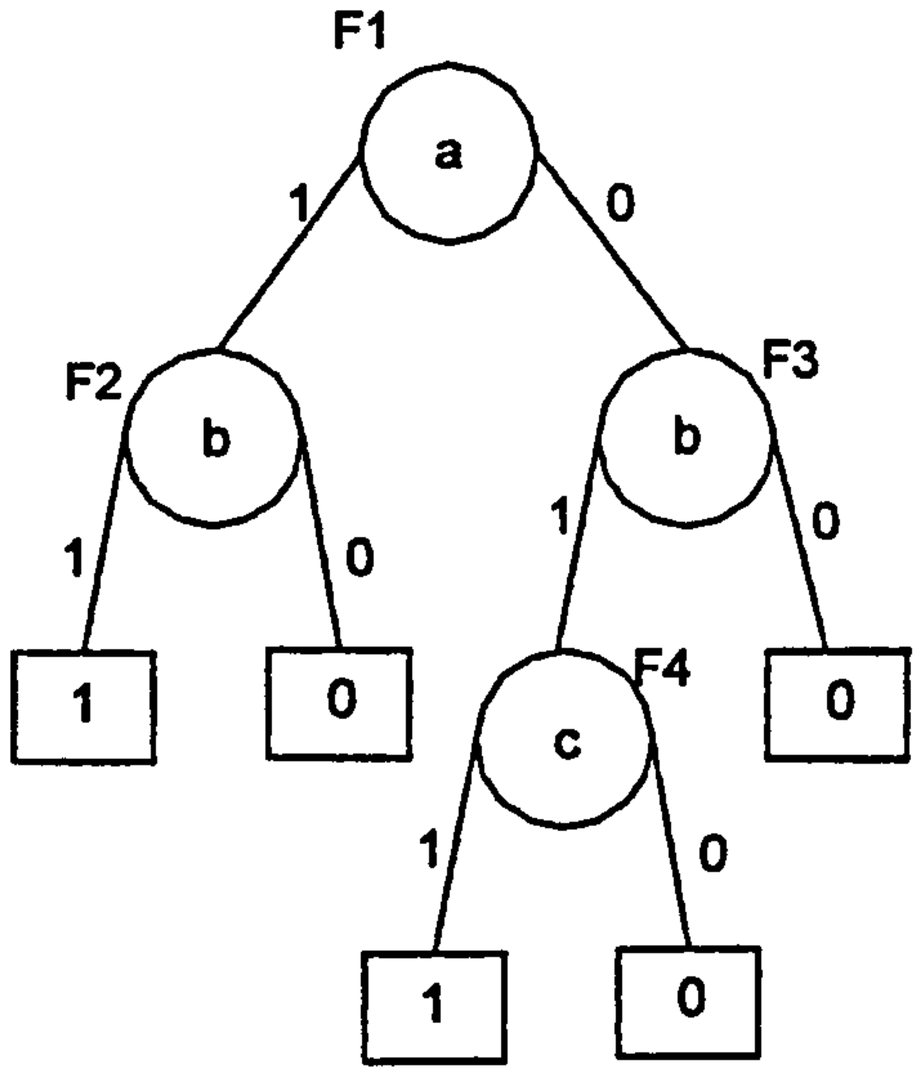


Figure 3.10: SFBDD

Firstly the connections between the nodes are recorded in an **ite** table as shown in table 3.1.

Node	Variable	1 Branch	0 Branch
F1	a	F2	F3
F2	b	1	0
F3	b	F4	0
F4	c	1	0

Table 3.1: **Ite** Table

The second stage is to calculate each of the terms in equation (3.9). The first term, $Pr_{x_i}(\underline{q})$ is the probability of the path from the root vertex to node x_i, which is calculated by evaluating the probability of the path from the root vertex up to but not including the node x_i. Table 3.2 records $Pr_{x_i}(\underline{q})$ for each node in the BDD.

Node	$Pr_{x_i}(\underline{q})$	Comments
F1	1	Root vertex itself
F2	q_a	F2 reached through the 1 branch of F1
F3	$1-q_a$	F3 reached through the 0 branch of F1
F4	$(1-q_a)q_b$	F4 reached through the 0 branch of F1 followed by the one branch of F3

Table 3.2: $Pr_{x_i}(\underline{q})$ for each node in the BDD

$Po^1_{x_i}(\underline{q})$ is calculated by summing the probability of all the paths from the selected node, x_i along the one branch to a terminal 1 vertex, excluding the probability of the selected node. Table 3.3 records $Po^1_{x_i}(\underline{q})$ for each node.

Node	$Po^1_{x_i}(\underline{q})$	Comments
F1	q_b	1 branch of F1 passes to F2 and the 1 branch of F2 passes to a terminal 1 vertex
F2	1	1 branch of F2 passes to a terminal 1 vertex
F3	q_c	1 branch of F3 passes to F4 and the 1 branch of F4 passes to a terminal 1 vertex
F4	1	1 branch of F4 passes to a terminal 1 vertex

Table 3.3: $Po^1_{x_i}(\underline{q})$ for each node

Similarly $Po^0_{x_i}(\underline{q})$ is calculated by summing the probability of all paths from the selected node, x_i along the zero branch to a terminal 1 vertex, excluding the probability of the selected node. Table 3.4 records $Po^0_{x_i}(\underline{q})$ for each node.

Node	$Po^0_{x_i}(\underline{q})$	Comments
F1	$q_b q_c$	0 branch of F1passes through the 1 branch of F3 and then the 1 branch of F4
F2	0	No terminal 1 paths from the 0 branch of F2
F3	0	No terminal 1 paths from the 0 branch of F3
F4	0	No terminal 1 paths from the 0 branch of F4

Table 3.4: $Po^0_{x_i}(\underline{q})$ for each node

The criticality function for each variable can be calculated by summing the contribution of nodes of the same variable. Thus using equation (3.9) and the results shown in tables 3.2, 3.3 and 3.4:

$$G_a(\underline{q}) = 1 \cdot [(q_b) - (q_b q_c)]$$
$$G_b(\underline{q}) = q_a \cdot [1-0] + (1-q_a) \cdot [q_c - 0]$$
$$G_c(\underline{q}) = (1-q_a) q_b \cdot [1-0]$$

Simplifying:

$$G_a(\underline{q}) = q_b(1-q_c)$$
$$G_b(\underline{q}) = q_a + q_c - q_a q_c$$
$$G_c(\underline{q}) = (1-q_a) q_b$$

The final stage is to substitute these measures into equation (3.5) to obtain $w_{SYS}(t)$:

$$w_{SYS}(t) = q_b(1-q_c) w_a + (q_a + q_c - q_a q_c) w_b + (1-q_a) q_b w_c$$

3.6 Summary

The BDD method developed for the purposes of fault tree analysis by Rauzy in 1993 has provided a significant improvement in terms of both accuracy and efficiency. Qualitative analysis is performed without the need for the numerous expansions, comparisons and reductions of conventional FTA techniques. Quantitative analysis can be performed without the knowledge of the minimal cut sets and eliminates the need to evaluate lengthy series expansions.

However, this method does have two main disadvantages. The first is that a variable ordering scheme must be chosen for the basic events of the fault tree in order to compute a SFBDD, and the second is that for extremely large fault trees it is not always possible to compute the SFBDD exactly.

The variable ordering scheme is critical, because it determines the size of the SFBDD; if a "bad" ordering scheme is chosen it may result in an extremely large non-minimal SFBDD which will make both qualitative and quantitative analysis inefficient. In more serious cases it may not be possible to compute a SFBDD at all due to huge processing and computer memory requirements.

There is no universal ordering scheme that produces a minimal SFBDD for every fault tree. Attention is now focused on choosing the "best" ordering scheme for a particular fault tree from a number of different schemes.

The second major disadvantage of the BDD method is that for extremely large fault trees it is not always possible to compute a SFBDD. Until recently there were no techniques for using the BDD method to perform partial analysis. Hence if the SFBDD could not computed exactly, the analysis produced nothing and an alternative technique had to be employed to start the analysis from scratch. This discourages the use of the BDD method as a generic way of solving fault trees. However, Rauzy developed techniques that can be used to compute a culled SFBDD. The culled SFBDD can then be used to perform approximate qualitative and quantitative analysis. Rauzy's culling techniques are significantly more efficient that the conventional FTA culling technique and will be considered in chapter eight.

Chapter 4: Fault Tree Analysis of Non-coherent Fault Trees

4.1 Introduction

The three main gate types encountered during FTA are the AND gate, the OR gate, and the NOT gate. Generally the use of the NOT gate is discouraged since a fault tree can be non-coherent if the NOT gate is used or directly implied (e.g. XOR). In a non-coherent system, component failure states and component working states can contribute to system failure [2]. This may be considered philosophically to be a poor system design when functioning components can contribute to system failure. Furthermore, from a practical viewpoint NOT logic increases the complexity of analysis and in many cases provides little additional information about the system and its effectiveness.

Although the use of NOT logic is often discouraged, Andrews [14] demonstrated that in the case of multi-tasking systems NOT logic is essential if successful and meaningful analysis is to be performed. This is also true for event tree analysis in which the consideration of success states is an integral feature of the technique [15]. Hence it will be essential to consider NOT logic and be able to analyse the resulting non-coherent fault trees accurately and efficiently.

Chapters two and three have focused on techniques for the analysis of coherent fault tree structures. When dealing with non-coherent fault tree structures additional work is required to perform exact qualitative and quantitative analysis. This chapter will consider the conventional techniques for analysing non-coherent fault tree structures.

4.2 The Use of NOT Logic

This section will illustrate that under certain circumstances the use of NOT logic during fault tree construction is essential for meaningful and accurate analysis. Consider the simplified gas detection system shown in figure 4.1.

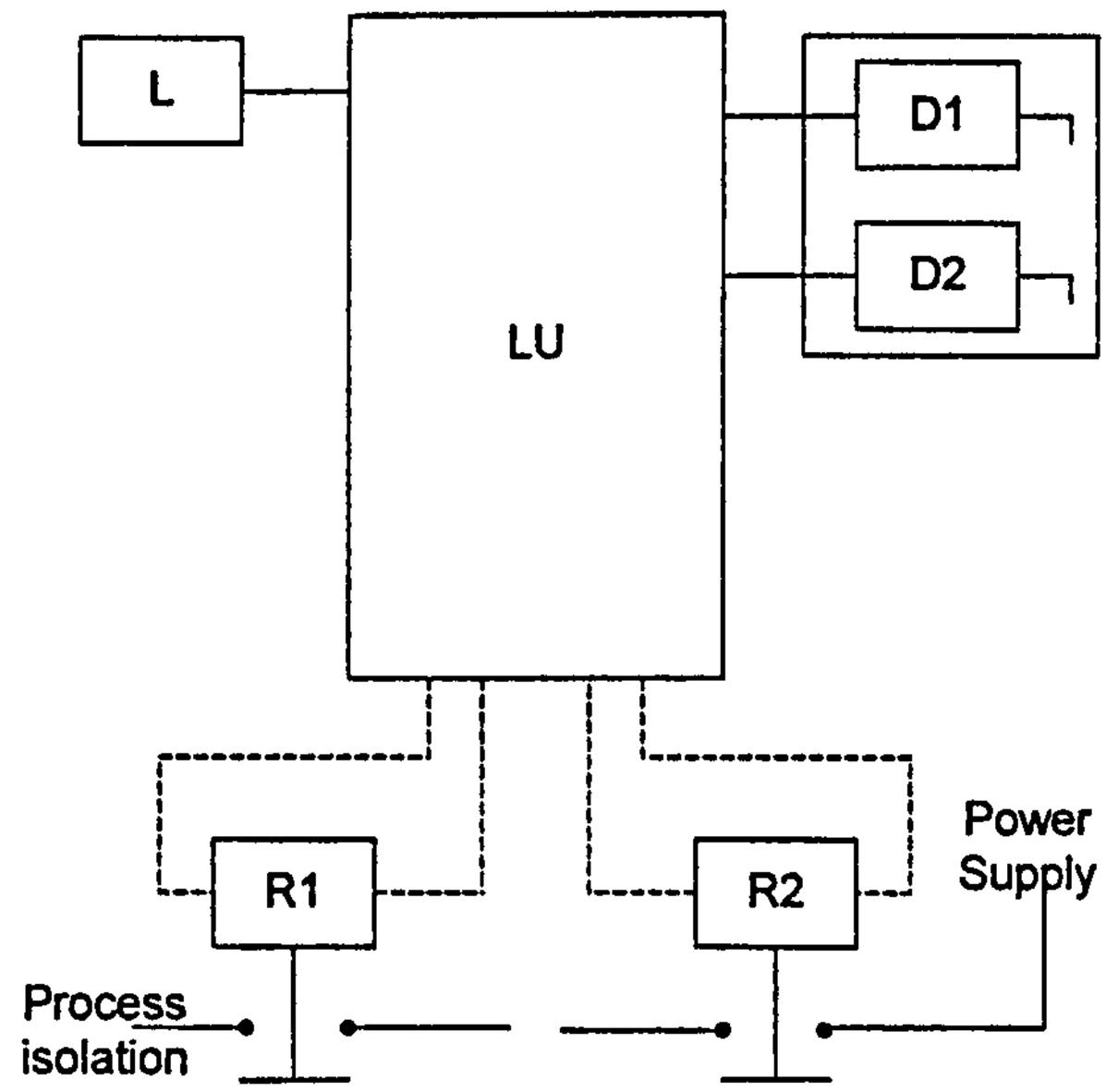


Figure 4.1: Simplified Gas Detection System

This is a multitasking system with two gas sensors, D1 and D2 that are used to detect leakage in a confined space. The detectors send signals along individual cables to the computer logic control unit, LU. If the LU receives a signal that there is a gas leak from any sensor, three functions must be performed.

- Process shut down: de-energise relay R1
- Inform the operator of the leak by a lamp and siren labelled L
- Remove the power supply to affected areas: de-energise relay R2

The system is considered failed if it does not perform one or more of the three functions given a leak occurs. There are seven possible failure states for this system, which are outlined in table 4.1. Although each state represents system failure, some states are more severe than others. For example, outcome three is particularly undesirable since the operator is informed of the gas release and thus believes that everything is fine but the process has not shut down and the power has not been isolated.

Failure State	Operator Informed	Process Shut-down	Power Isolation
1	W	W	F
2	W	F	W
3	W	F	F
4	F	W	W
5	F	W	F
6	F	F	W
7	F	F	F

Table 4.1: The Seven Possible Failure States of the Gas Detection System in Figure 4.1

If a Fault Tree Analysis is performed avoiding the use of NOT logic the fault tree shown in figure 4.2 is obtained.

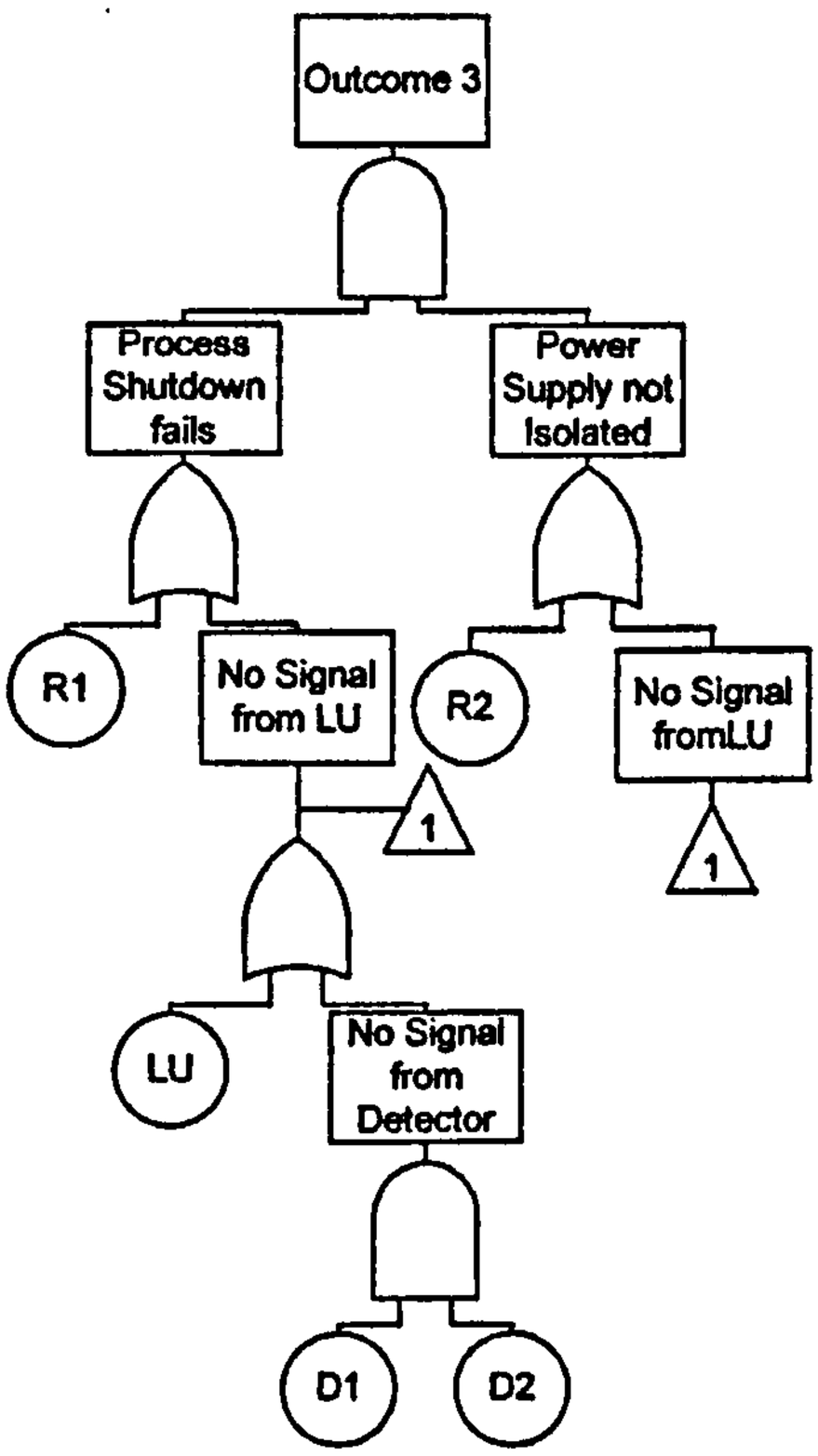


Figure 4.2: Fault Tree Obtained from a Coherent Assessment of the Gas Detection System

Three minimal cut sets can be identified from this fault tree:

{R1,R2}, {D1,D2}, {LU}

Although this fault tree has been constructed in a logical manner it is inaccurate since, accounting for the conditions that the operator is informed, then either D1 and LU or D2 and LU must be working, thus minimal cut sets two and three would not cause outcome three. Consequently quantification of this fault tree will result in a substantial overestimate of the probability of outcome three. For a correct assessment of this outcome it is essential to use NOT logic during fault tree construction so that the working part of the system is taken into account. The non-coherent fault tree shown in figure 4.3 is obtained from a non-coherent assessment of outcome three.

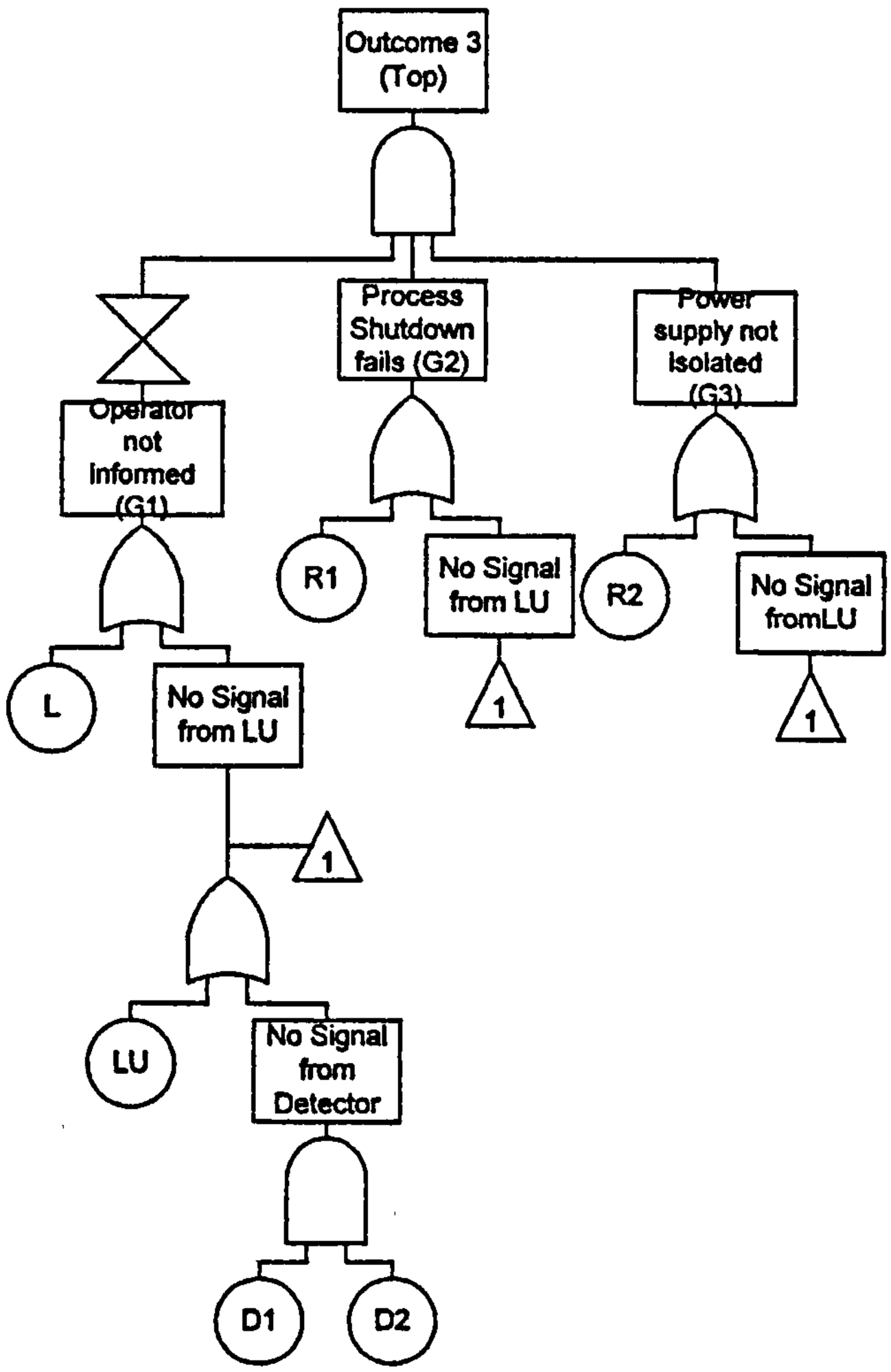


Figure 4.3: Non-coherent Fault Tree Obtained from a Non-coherent Assessment of the Gas Detection System

Working in a top-down fashion the following logic expression is obtained:

$$Top = G1 \cdot G2 \cdot G3$$

Developing G1, G2 and G3:

$$G1 = \overline{\overline{L} + LU + D1 \cdot D2}$$
$$= \overline{L} \cdot \overline{LU} \cdot \left(\overline{D1} + \overline{D2}\right)$$

$$G2 = R1 + LU + D1 \cdot D2$$

$$G3 = R2 + LU + D1 \cdot D2$$

Substituting in for G1, G2 and G3, Top becomes:

$$Top = \left(\overline{L} \cdot \overline{LU} \cdot \left(\overline{D1} + \overline{D2}\right)\right) \cdot (R1 + LU + D1 \cdot D2) \cdot (R2 + LU + D1 \cdot D2)$$
$$= \overline{L} \cdot \overline{LU} \cdot R1 \cdot R2 \cdot \left(\overline{D1} + \overline{D2}\right)$$

The coherent approximation is $R1 \cdot R2$ thus the inclusion of NOT logic has successfully removed the inappropriate failure combinations and will enable accurate quantitative analysis. Thus whilst NOT logic can increase the complexity of analysis, in the case of multitasking systems the use of NOT logic is essential. The method used to obtain the prime implicant sets is explained in detail in section 4.3.1.

4.3 Qualitative Analysis

4.3.1 Introduction

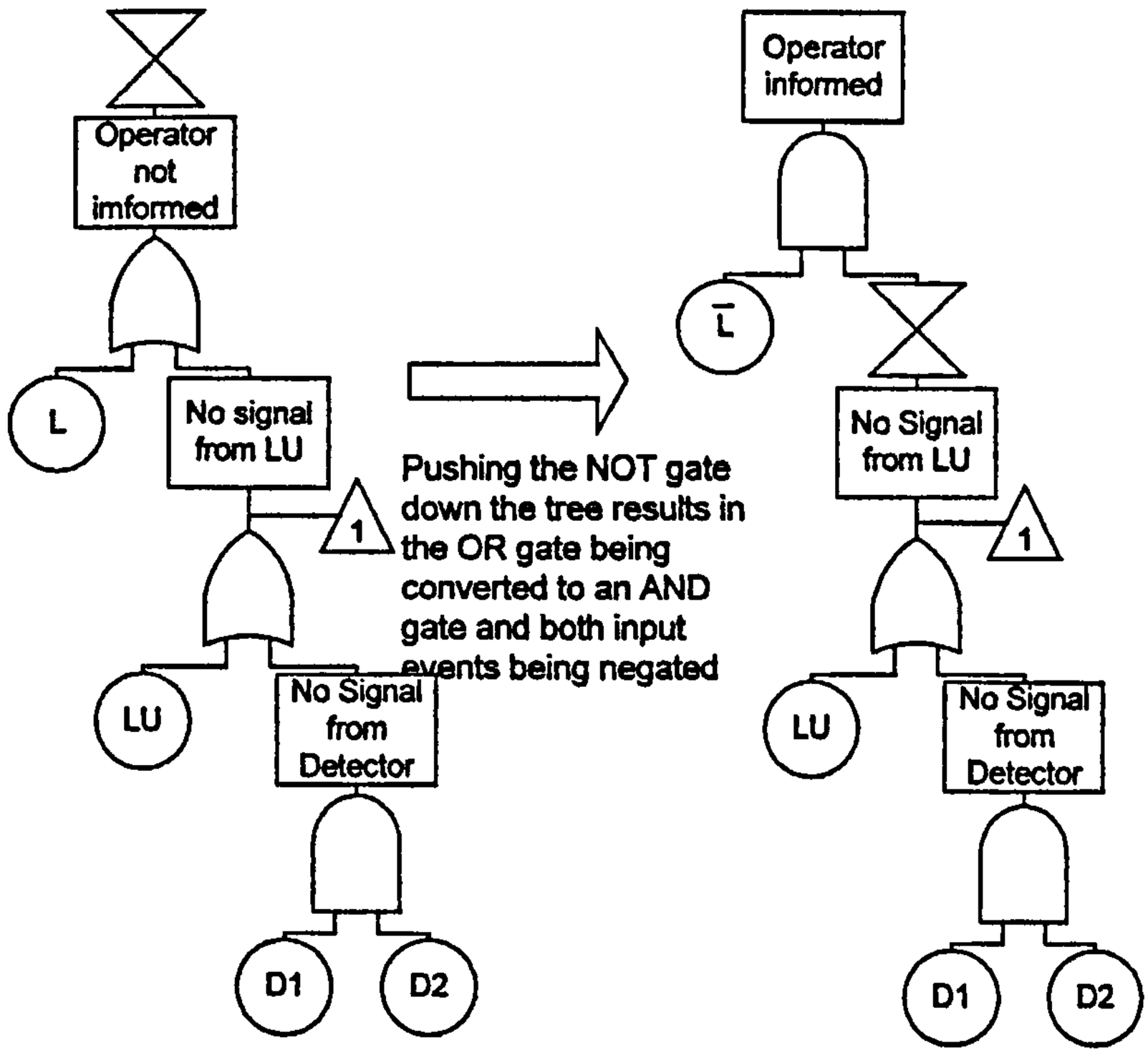
During qualitative analysis all the possible causes of system failure are identified. For coherent fault trees each possible cause of system failure is called a minimal cut set; a combination of component failures that are both necessary and sufficient to cause system failure. In the case of non-coherent fault trees both component failed states and component working states can contribute to system failure. Hence each possible cause of system failure is called a **prime implicant set** and is a combination of component failure states and component working states that are both necessary and sufficient to cause the system to be in the failed state.

4.3.2. Identifying the Prime Implicants Sets of a Non-coherent Fault Tree

To identify the prime implicant sets of a non-coherent fault tree it is first necessary to ensure that the fault tree structure contains only AND gates and OR gates, i.e. any NOT gates need to be removed. De-Morgan's laws outlined in equation (4.1) can be used to push the NOT logic down the fault tree to complement the basic events.

$$\overline{(A+B)} = \overline{A}\cdot\overline{B}, \quad \overline{(A\cdot B)} = \overline{A}+\overline{B} \tag{4.1}$$

To illustrate this process consider again the non-coherent fault tree in figure 4.3:



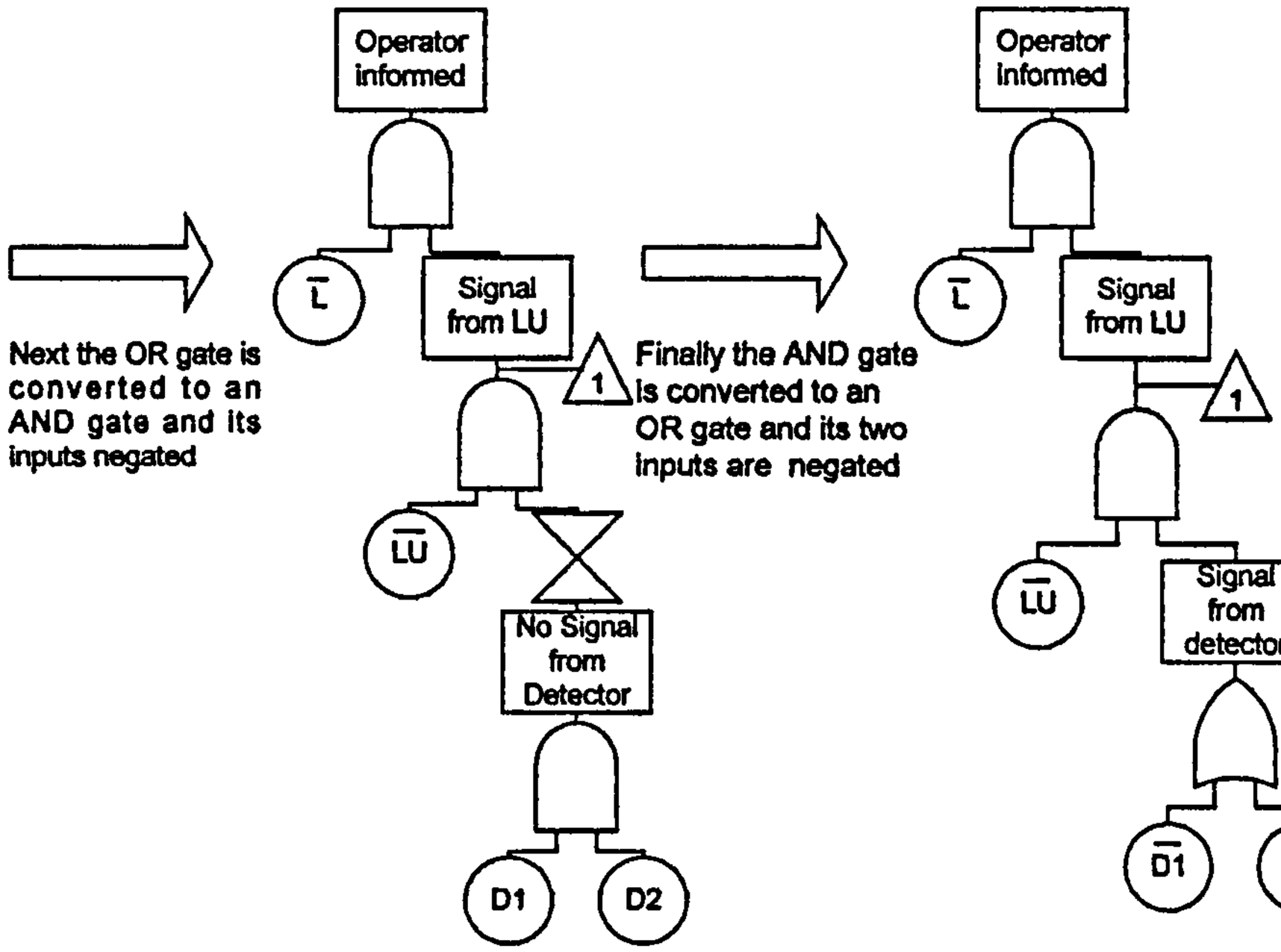


Figure 4.4: Diagram Showing how the NOT Logic is Pushed Down the Fault Tree

The application of De-Morgan's laws to the fault tree in figure 4.3 results in an equivalent fault tree that contains only AND and OR gates, see figure 4.5.

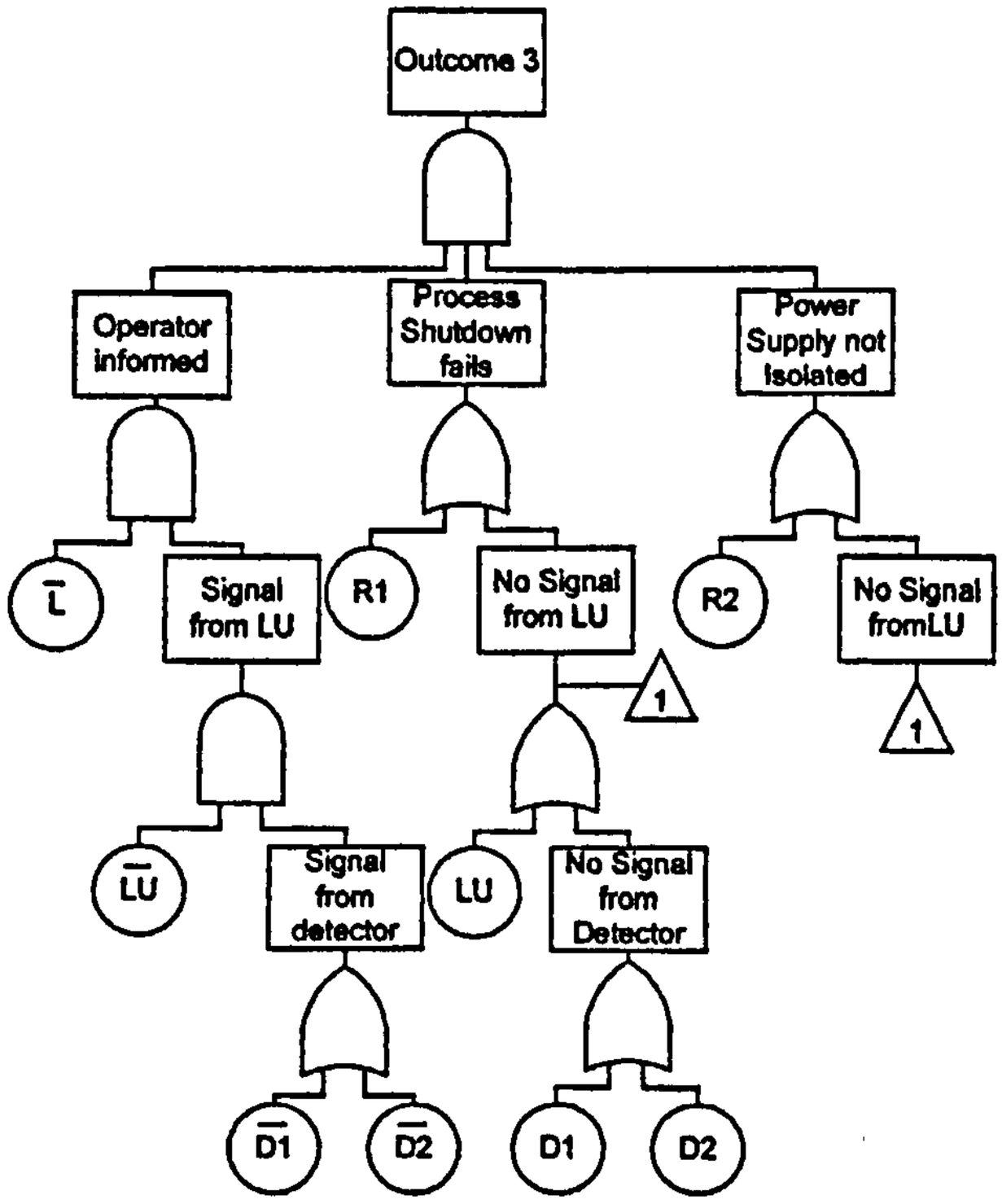


Figure 4.5: The Equivalent Non-coherent Fault Tree Obtained for the Fault Tree in Figure 4.3

Once all of the NOT gates have been eliminated, the same top-down approach applied to coherent fault trees is applied to produce the failure modes of the modified non-coherent fault tree. To illustrate this consider the non-coherent fault tree in figure 4.6.

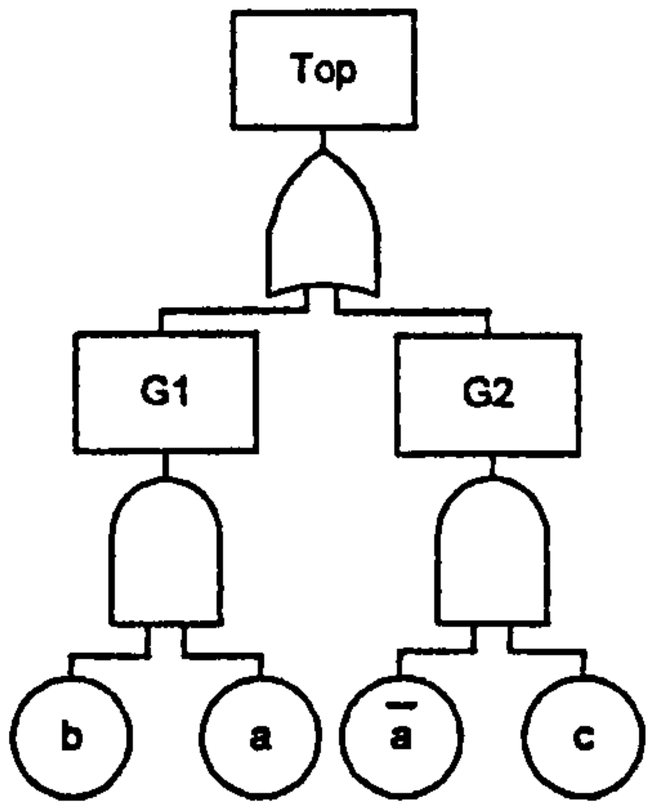


Figure 4.6: Non-coherent Fault Tree Structure

Working in a top-down fashion the following logic expression is obtained for the Top event:

$$Top = G1 + G2$$

Substituting in for, $G1 = a \cdot b$ and $G2 = \bar{a} \cdot c$, the logic expression for Top becomes:

$$Top = a \cdot b + \bar{a} \cdot c$$

Two prime implicant sets, $\{a,b\}, \{\bar{a},c\}$ have been identified. However, this is not a complete list of prime implicant sets. In fact if both component b and component c are in a failed state then the system will be in a failed state regardless of the state of component a. Hence a third prime implicant set, $\{b,c\}$ can be identified by applying the consensus law given in equation (4.2):

$$AX + \bar{A}Y = AX + \bar{A}Y + XY \qquad (4.2)$$

The following expression is obtained for the top gate of the fault tree in figure 4.6:

$$Top = a \cdot b + \bar{a} \cdot c + b \cdot c$$

To identify a full list of prime implicant sets directly from the fault tree, an expression for the top event is obtained using a Top-Down approach. Then the consensus theorem is applied exhaustively to pairs of prime implicant sets involving a normal and negated literal. Whilst this technique can be applied to simple fault trees with ease, for larger trees it may not be possible to identify the prime implicant sets exactly.

4.3.3. Obtaining a Coherent Approximation

Conventional techniques for analysing non-coherent fault trees are more computationally intensive than those for analysing coherent fault trees. Hence it is not always possible to perform exact analysis of non-coherent fault trees. In such cases a coherent approximation can be obtained during qualitative analysis. This involves identifying only the positive parts of the prime implicant sets (i.e. failing), known as the minimal p-cuts of the fault tree. The minimal p-cuts can then be used to quantify the system approximately using the conventional FTA techniques outlined in chapter two.

To obtain a full list of minimal p-cuts a traditional top-down or bottom-up approach is applied to the fault tree to obtain an expression for the top gate in terms of the system components as usual, but all negated literals are ignored. To illustrate this procedure consider the fault tree in figure 4.7.

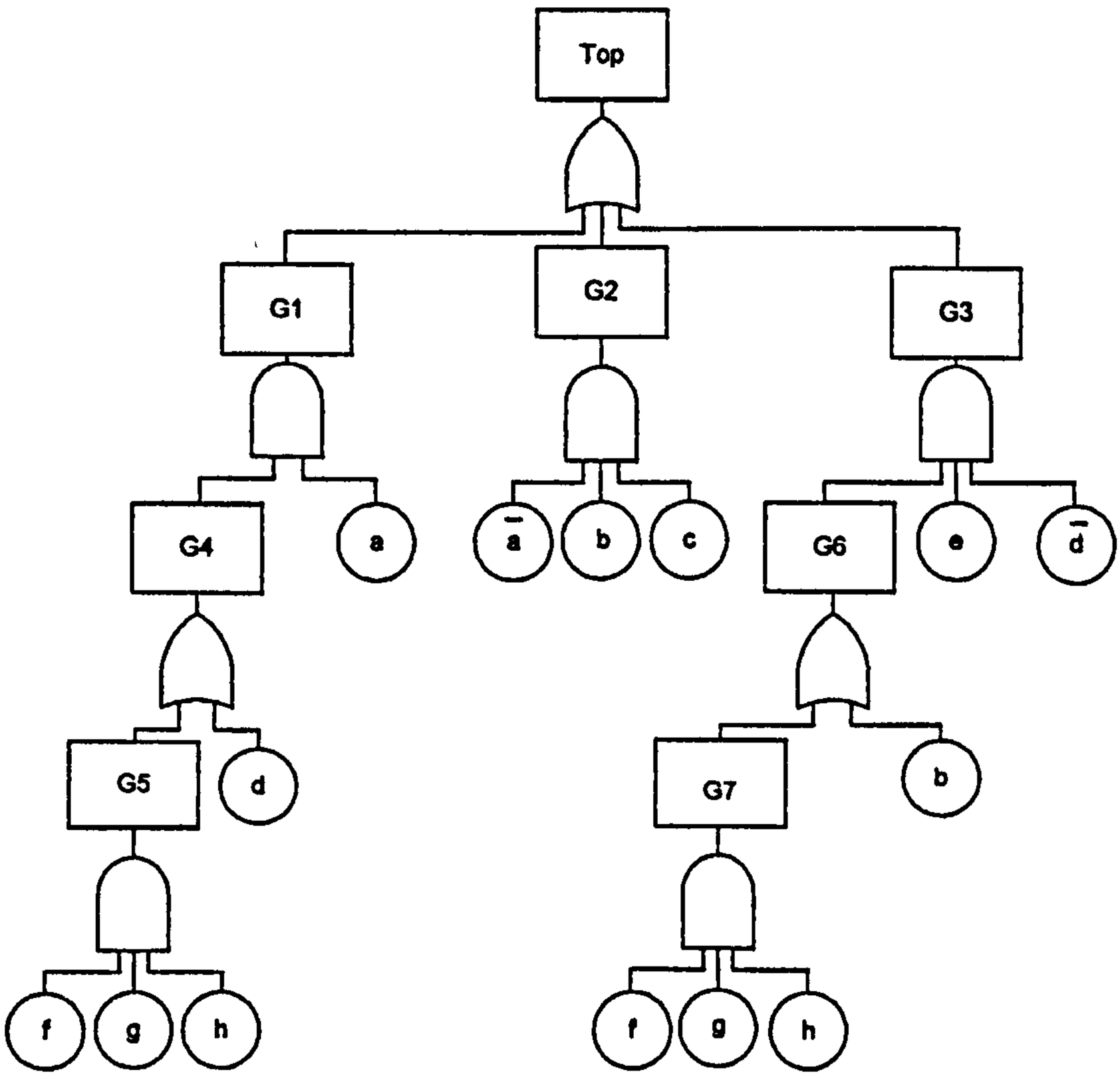


Figure 4.7: Non-coherent Fault Tree Structure

Developing an expression for G1, by evaluating, G4 and G5:

$G5 = f \cdot g \cdot h$

$G4 = G5 + d = f \cdot g \cdot h + d$

$G1 = G4 \cdot a = (f \cdot g \cdot h + d) \cdot a = a \cdot f \cdot g \cdot h + a \cdot d$

Developing an expression for G2:

$G2 = \bar{a} \cdot b \cdot c = b \cdot c$ $\bar{a}$ is removed from this expression

Developing an expression for G3, by evaluating G6, G7 and G8:

$G7 = b \cdot e$

$G8 = e \cdot f \cdot g \cdot h$

$G6 = G7 + G8 = b \cdot e + e \cdot f \cdot g \cdot h$

$G3 = G6 \cdot \bar{d} = G6 = b \cdot e + e \cdot f \cdot g \cdot h$ $\bar{d}$ is removed from this expression

Finally developing an expression for the top gate, Top:

$$Top = a \cdot f \cdot g \cdot h + a \cdot d + b \cdot c + b \cdot e + e \cdot f \cdot g \cdot h$$

Five minimal p-cuts are identified for the fault tree in figure 4.7:

{afgh}, {ad}, {bc}, {be}, {efgh}

For large fault trees this technique can significantly reduce the work required to analyse quantify the system. Furthermore, provided the reliability of component is high the approximation obtained will be reasonably accurate.

4.4 Quantitative Analysis

Quantification of a non-coherent fault tree cannot be achieved using the quantification methods for coherent fault trees introduced in chapter two, since they do not take into account the working states that contribute to system failure. This section will consider how the system unavailability and the unconditional failure intensity of a non-coherent fault tree are calculated.

4.4.1 Calculating the System Unavailability

In chapter two, the inclusion-exclusion method for calculating the system unavailability was introduced. Inagaki and Henley [16] modified this method to enable the calculation of the system unavailability of non-coherent fault trees. Inagaki and Henley's calculation procedure is given in equation (4.3):

$$Q_{SYS}(t) = \sum_{i=1}^{n} P(\varepsilon_i) - \sum_{i=2}^{n} \sum_{j=1}^{i-1} P(\varepsilon_i \cap \varepsilon_j) + \ldots + (-1)^{n-1} P(\varepsilon_1 \cap \varepsilon_2 \cap \ldots \cap \varepsilon_n) \tag{4.3}$$

Where, $P(\varepsilon_i)$ denotes the probability that prime implicant set i exists at time t, this probability is calculated as follows:

$$P(\varepsilon_i) = \prod_{j=1}^{n_p} q^{\alpha j}(t) \tag{4.4}$$

Where, $q^{\alpha j} = \begin{cases} q_j(t) \text{ if } \alpha = 1 \text{ i.e. j appears} \\ p_j(t) \text{ if } \alpha = 0 \text{ i.e. } \bar{j} \text{ appears} \end{cases}$ and $p_j = 1 - q_j$

Thus, $q^{\alpha j}$ represents the probabilities of the literals, (normal or negated) contained in any one of the prime implicant sets. These products of prime implicant sets are such that conflicting literals, $x_i \cdot \bar{x}_i = 0$ and $P(\varepsilon_i) = 0$, and all redundancies are eliminated from these products using the Idempotent law, $x_i \cdot x_i = x_i$. And, n_p denotes the number of basic events in a given prime implicant set.

This procedure enables the exact system unavailability to be calculated. However, the calculation can be unmanageable even for moderate sized fault trees so it may be necessary to terminate the modified inclusion-exclusion expansion to obtain an approximate value for the system unavailability. One possible upper bound is given in equation (4.5).

$$Q_{SYS}(t) \leq \sum_{i=1}^{n} P(\varepsilon_i) \tag{4.5}$$

This upper bound is equivalent to the Rare Event Approximation given in equation (2.10). Although the Rare Event Approximation can be a good approximation for coherent systems provided that component failure events are rare. It is not possible to give any guide to how accurate this bound is for non-coherent systems, but it is not unusual for it to be greater than unity.

An alternative and more commonly used means of approximating the system unavailability is to obtain a coherent approximation for qualitative analysis as illustrated in section 4.3.3 and use this to calculate the rare event approximation or the Minimal Cut Set Upper Bound considered in chapter two.

4.4.2 Calculating the Unconditional Failure Intensity

The unconditional failure intensity was first introduced in chapter one and is defined as the probability that a system fails per unit time at t, given that it was working at $t=0$. This section will consider two methods that have been developed for calculating the unconditional failure intensity of a non-coherent fault tree. The first method, developed by Inagaki and Henley in 1980 [16] relies on knowledge of the prime implicant sets and is considered in section 4.4.2.1. The second method developed by Becker and Camarinopoulos [17] is an extension of the identity given in equation 3.5 and will be considered in section 4.4.2.2.

4.4.2.1 Inagaki and Henley's Method

Henley and Inagaki extended the procedure for calculating the unconditional failure intensity that was introduced in chapter two for use with non-coherent fault trees. This calculation procedure is outlined below.

The top event occurs in interval, $[t,t+dt)$, if and only if no prime implicant sets exist at time t and at least one prime implicant set occurs in the interval $[t,t+dt)$. Thus:

$$w_{SYS}(t)dt = P\left\{\bigcup_{i=1}^{n}\theta_i\right\} - P\left\{B\bigcup_{i=1}^{n}\theta_i\right\} \qquad (4.6)$$
$$= w^1_{sys}(t)dt - w^2_{sys}(t)dt$$

Where, $B \equiv \bigcup_{i=1}^{n}\varepsilon_i$ and ε_i is the event that prime implicant set i exists at time t. And θ_i is defined as the event that prime implicant set i occurs at time t per unit time.

$$P(\theta) = \sum_{j=1}^{n_P} z^{\alpha j}(t)\prod_{\substack{i=1 \\ i\neq j}}^{n_P} q^{\alpha i}(t)$$

Where, $q^{\alpha k} = \begin{cases} q_k(t) \text{ if } \alpha_k = 1 \\ p_k(t) \text{ if } \alpha_k = 0 \end{cases}$, $z^{\alpha j} = \begin{cases} w_j(t) \text{ if } \alpha_j = 1 \\ v_j(t) \text{ if } \alpha_j = 0 \end{cases}$ α is set to 1 if the literal exists in its normal form and 0 if the literal is negated. $w_j(t)$ is the unconditional failure intensity of component j and $v_j(t)$ is the unconditional repair intensity of component j.

The first term in equation (4.6) is the probability that one or more prime implicant sets occurs in the interval $[t, t+dt)$. Hence this term can be expanded using the inclusion-exclusion expansion as follows:

$$P\left\{\bigcup_{i=1}^{n}\theta_i\right\} = \sum_{i=1}^{n}P\{\theta_i\} - \sum_{i=2}^{n}\sum_{j=1}^{i-1}P\{\theta_i \cap \theta_j\} + \ldots + (-1)^{n-1}P(\theta_1 \cap \theta_2 .. \cap \theta_n) \qquad (4.7)$$

The second term of equation (4.6) is the probability that one or more prime implicant sets occurs in the interval $[t, t+dt)$ given that one or more prime implicant sets already exists. Thus it can also be expanded using the inclusion-exclusion expansion as follows:

$$P\left\{B\bigcup_{i=1}^{n}\theta_i\right\} = \sum_{i=1}^{n}P\{\theta_i \cap B\} - \sum_{i=2}^{n}\sum_{j=1}^{i-1}P\{\theta_i \cap \theta_j \cap B\} + \ldots + (-1)^{n-1}P(\theta_1 \cap \theta_2 .. \cap \theta_n \cap B) \qquad (4.8)$$

To illustrate how this method is applied in practise consider again the fault tree in figure 4.6 which has three prime implicant sets, $\{a,b\},\{\bar{a},c\}\{b,c\}$.

Calculating $w_{SYS}^{(1)}(t)$:

$$w_{SYS}^{(1)}(t)dt = \sum_{i=1}^{3} P(\theta_i) - \sum_{i=2}^{3}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j) + P(\theta_1 \cap \theta_2 \cap \theta_3) \qquad (4.9)$$

Given:

$$\sum_{i=1}^{3} P(\theta_i) = \sum_{i=1}^{3} w_{C_i}dt = (w_a q_b + w_b q_a + v_a q_c + w_c \bar{q}_a + w_b q_c + w_c q_b)dt$$

$$\sum_{i=2}^{3}\sum_{j=1}^{i-1} P(\theta_i \cap \theta_j) = (w_b q_a q_c + w_c \bar{q}_a q_b)dt$$

$$P(\theta_1 \cap \theta_2 \cap \theta_3) = 0$$

Hence:

$$w_{SYS}^{(1)}(t) = w_a(q_b) + v_a(q_c) + w_b(q_a + q_c - q_a q_c) + w_c(\bar{q}_a + q_b - \bar{q}_a q_b)$$

Now calculating $w_{SYS}^{(2)}(t)$:

$$w_{SYS}^{(2)}(t) = P\left(B\bigcup_{i=1}^{3}\theta_i\right) = P(\theta_1 \cap B) + P(\theta_2 \cap B) + P(\theta_3 \cap B) - P(\theta_1 \cap \theta_2 \cap B)$$
$$- P(\theta_1 \cap \theta_3 \cap B) - P(\theta_2 \cap \theta_3 \cap B) + P(\theta_1 \cap \theta_2 \cap \theta_3 \cap B) \qquad (4.10)$$

And:

$$P(\theta_1 \cap B) = P(\theta_1 \cap \varepsilon_1) + P(\theta_1 \cap \varepsilon_2) + P(\theta_1 \cap \varepsilon_3) - P(\theta_1 \cap \varepsilon_1 \cap \varepsilon_2)$$
$$- P(\theta_1 \cap \varepsilon_1 \cap \varepsilon_3) - P(\theta_1 \cap \varepsilon_2 \cap \varepsilon_3) + P(\theta_1 \cap \varepsilon_1 \cap \varepsilon_2 \cap \varepsilon_3)$$
$$= 0 + 0 + w_a q_b q_c - 0 - 0 - 0 + 0$$

Expanding and evaluating each of the remaining terms in equation (4.10) gives:

$$P(\theta_2 \cap B) = v_a q_b q_c$$

$$P(\theta_3 \cap B) = w_c q_a q_b + w_b \bar{q}_a q_c$$

$$P(\theta_1 \cap \theta_2 \cap B) = 0$$

$$P(\theta_1 \cap \theta_3 \cap B) = 0$$

$$P(\theta_2 \cap \theta_3 \cap B) = 0$$

$$P(\theta_1 \cap \theta_2 \cap \theta_3 \cap B) = 0$$

From equation (4.6):

$$w_{SYS}(t)dt = (w_a(q_b - q_b q_c) + v_a(q_c - q_b q_c) + w_b(q_a + q_c - q_a q_c - \bar{q}_a q_c) + w_c(\bar{q}_a + q_b - \bar{q}_a q_b - q_a q_b))dt$$

Simplifying this expression gives:

$$w_{SYS}(t) = w_a q_b \bar{q}_c + v_a q_c \bar{q}_b + w_b q_a + w_c \bar{q}_a$$

From this simple example it is clear that calculating the unconditional failure intensity is a time consuming and exhaustive process. Approximations are unavoidable for large fault trees with many repeated events. Vesely identified a possible approximation for the unconditional failure intensity, which is given in equation (4.11). This approximation produces an accurate upper bound for $w_{sys}(t)$ provided component failures are rare.

$$w_{SYS}(t) \leq w^{(1)}_{SYS}(t) \qquad (4.11)$$

4.4.2.2 Becker and Camarinopoulos' Method

For a coherent system the unconditional failure intensity can be expressed as the probability that the system is in a critical state for one or more components at time t, and one of those critical components fails in the interval $[t, t+dt)$. Hence it is possible to express the unconditional failure intensity for a coherent fault tree as follows:

$$w_{SYS}(t)dt = \sum_{i=1}^{n_C} G_i(\underline{q}) w_i dt \qquad (4.12)$$

Where, $G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q})$

For non-coherent systems, components can be both failure and repair critical, hence equation (4.12) cannot be used to calculate the unconditional failure intensity for non-coherent systems. To extend equation (4.12) Becker and Camarinopoulos defined two types of system criticality functions:

1. Failure Criticality Function: System is working when component i is working and failed when component i is failed.

$$\varphi_i^f = Q_{SYS}(1_i, \underline{q})(1 - Q_{SYS}(0_i, \underline{q})) \tag{4.13}$$

2. Repair Criticality Function: System is working when component i is failed and failed when component i is working.

$$\varphi_i^r = Q_{SYS}(0_i, \underline{q})(1 - Q_{SYS}(1_i, \underline{q})) \tag{4.14}$$

From this Becker and Camarinopoulos extended equation (4.12) as follows:

$$w_{SYS}(t)dt = \sum_{i=1}^{n_C} \varphi_i^f w_i dt + \sum_{i=1}^{n_C} \varphi_i^r v_i dt \tag{4.15}$$

However, this method is only applicable when $Q_{SYS}(1_i, \underline{q})$ and $Q_{SYS}(0_i, \underline{q})$ are independent. Consequently it is not possible to use this method to calculate $w_{sys}(t)dt$ for the majority of non-coherent systems.

4.5 Summary

The use of NOT logic during fault tree construction can add to the complexity of analysis and in many cases provide little additional information. However, Andrews demonstrated that NOT logic can be essential for meaningful and accurate analysis of certain systems. Consequently it is essential to be able to analyse non-coherent fault trees accurately.

Conventional techniques for analysing coherent fault trees have been extended for the purposes of non-coherent fault tree analysis. However NOT logic increases the complexity of the analysis, and it is not always possible to analyse even moderate sized trees exactly. Although a coherent approximation can be used to reduce the work

required to analyse a system, the techniques employed are still computationally intensive and they only produce a list of minimal p-cuts rather than prime implicant sets.

Although the fault tree diagram provides a useful description of the system being analysed, an alternative technique for both qualitative and quantitative analysis is required to improve efficiency and accuracy of non-coherent FTA. Rauzy and Dutuit have extended the BDD method introduced in chapter three to enable full qualitative analysis of non-coherent fault trees and Andrews has developed a procedure for calculating the top event probability from the SFBDD. These techniques will be the focus of chapter five.

Chapter 5: The Binary Decision Diagram Method for the Analysis of Non-coherent Fault Trees

5.1 Introduction

It is useful to be able to analyse non-coherent fault trees when they are encountered. Conventional techniques for FTA can be used to perform both qualitative and quantitative analysis of non-coherent fault trees. However, they can be problematical and approximations are unavoidable even for moderate sized trees.

The BDD method for analysing coherent fault trees introduced in chapter three is more efficient and accurate than conventional FTA methods. In order to utilise the BDD methodology the fault tree is converted to the SFBDD, from which exact qualitative and quantitative analysis is performed. Although the SFBDD of a non-coherent fault tree can be used to calculate the system unavailability exactly [14], other system parameters such as the unconditional failure intensity and the expected number of system failures in a given interval, required for a system assessment had not previously been obtained from the SFBDD. This will be considered in detail in chapter seven.

It is not possible to identify the prime implicant sets directly from the SFBDD; this requires further work. Two methods for identifying the prime implicant sets will be considered. The first of these methods was developed by Rauzy and Dutuit and involves computing what is known as the meta-products BDD from which the prime implicant sets can be identified [4,6,7]. The second method was developed as part of this research project, which encodes the Consensus BDD directly from the fault tree.

Although the BDD method is more efficient and accurate than conventional FTA techniques performed by Kinetic Tree Theory it has a major disadvantage. Namely that a variable ordering scheme must be chosen for the conversion process .

5.2 Computing the SFBDD

The SFBDD for a non-coherent fault tree is computed using the same **ite** procedure introduced in chapter three, section 3.3.3.1. The only additional rule required is the **ite** structure for negated events, which is introduced below. Notice that the one and zero branches have been switched compared to the **ite** expression for the positive literal.

$$\overline{x} = \text{ite}(x, 0, 1)$$

To illustrate how the SFBDD is computed consider the non-coherent fault tree in figure 5.1:

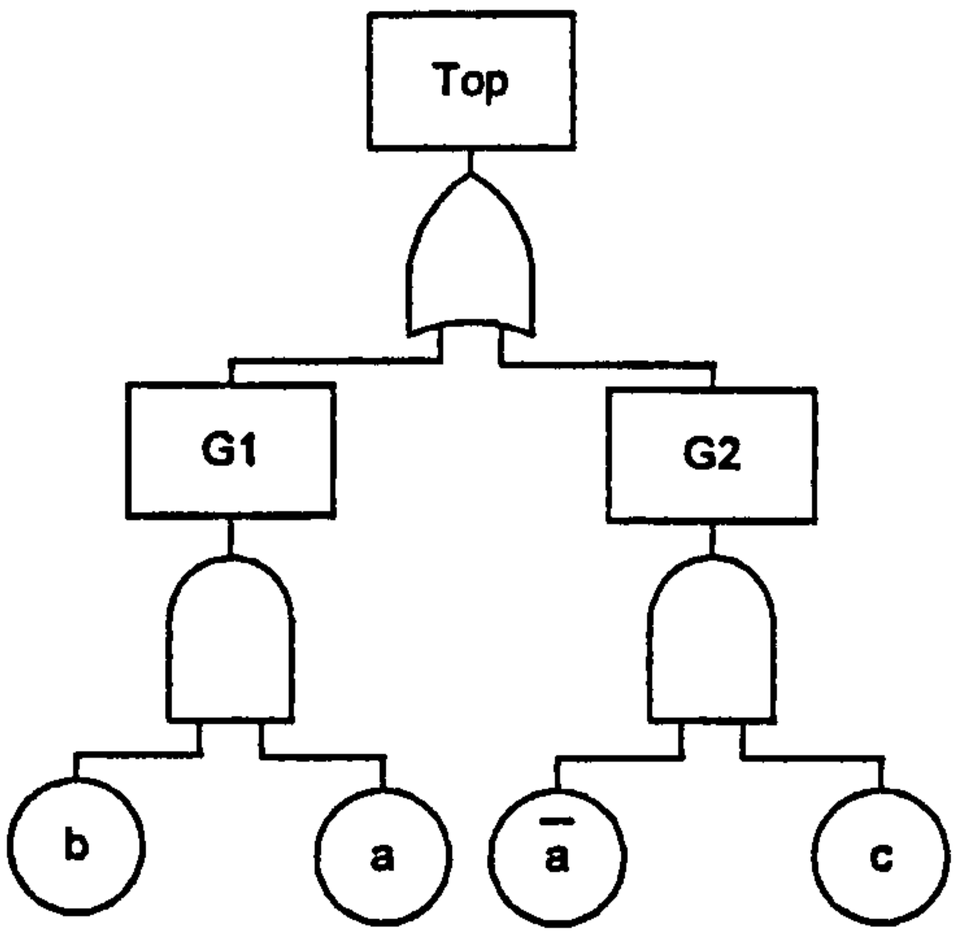


Figure 5.1: Non-coherent Fault Tree Structure

Beginning by assuming a variable ordering $b < a < c$:

Assigning each basic event in the fault tree an **ite** structure:

$$a = \text{ite}(a, 1, 0)$$
$$\overline{a} = \text{ite}(a, 0, 1)$$
$$b = \text{ite}(b, 1, 0)$$
$$c = \text{ite}(c, 1, 0)$$

Considering the gates in a bottom-up fashion according to rules and identities introduced in section 3.3.3.1, beginning with gate G1:

$$G1 = a \cdot b$$
$$G1 = \text{ite}(a, 1, 0) \cdot \text{ite}(b, 1, 0)$$

$$G1 = ite(b, [1 \cdot ite(a, 1, 0)], [0 \cdot ite(a, 1, 0)])$$

$$G1 = ite(b, ite(a, 1, 0), 0)$$ Since $1 \cdot H = H$ and $0 \cdot H = 0$

Now dealing with gate G2:

$$G2 = \bar{a} \cdot c$$
$$G2 = ite(a, 0, 1) \cdot ite(c, 1, 0)$$
$$G2 = ite(a, [0 \cdot ite(c, 1, 0)], [1 \cdot ite(c, 1, 0)])$$
$$G2 = ite(a, 0, ite(c, 1, 0))$$ Since $1 \cdot H = H$ and $0 \cdot H = 0$

Finally dealing with the Top gate, Top:

$$Top = G1 + G2$$
$$Top = ite(b, ite(a, 1, 0), 0) + ite(a, 0, ite(c, 1, 0))$$
$$Top = ite(b, [ite(a, 1, 0) + ite(a, 0, ite(c, 1, 0))], [0 + ite(a, 0, ite(c, 1, 0))])$$
$$Top = ite(b, ite(a, [1 + 0], [0 + ite(c, 1, 0)]), ite(a, 0, ite(c, 1, 0)))$$ Since $0 + H = H$
$$= ite(b, ite(a, 1, ite(c, 1, 0)), ite(a, 0, ite(c, 1, 0)))$$

The ite structure computed for the fault tree shown in figure 5.1 is given in equation (5.1) and the SFBDD is shown in figure 5.2.

$$ite(b, ite(a, 1, ite(c, 1, 0)), ite(a, 0, ite(c, 1, 0))) \quad (5.1)$$

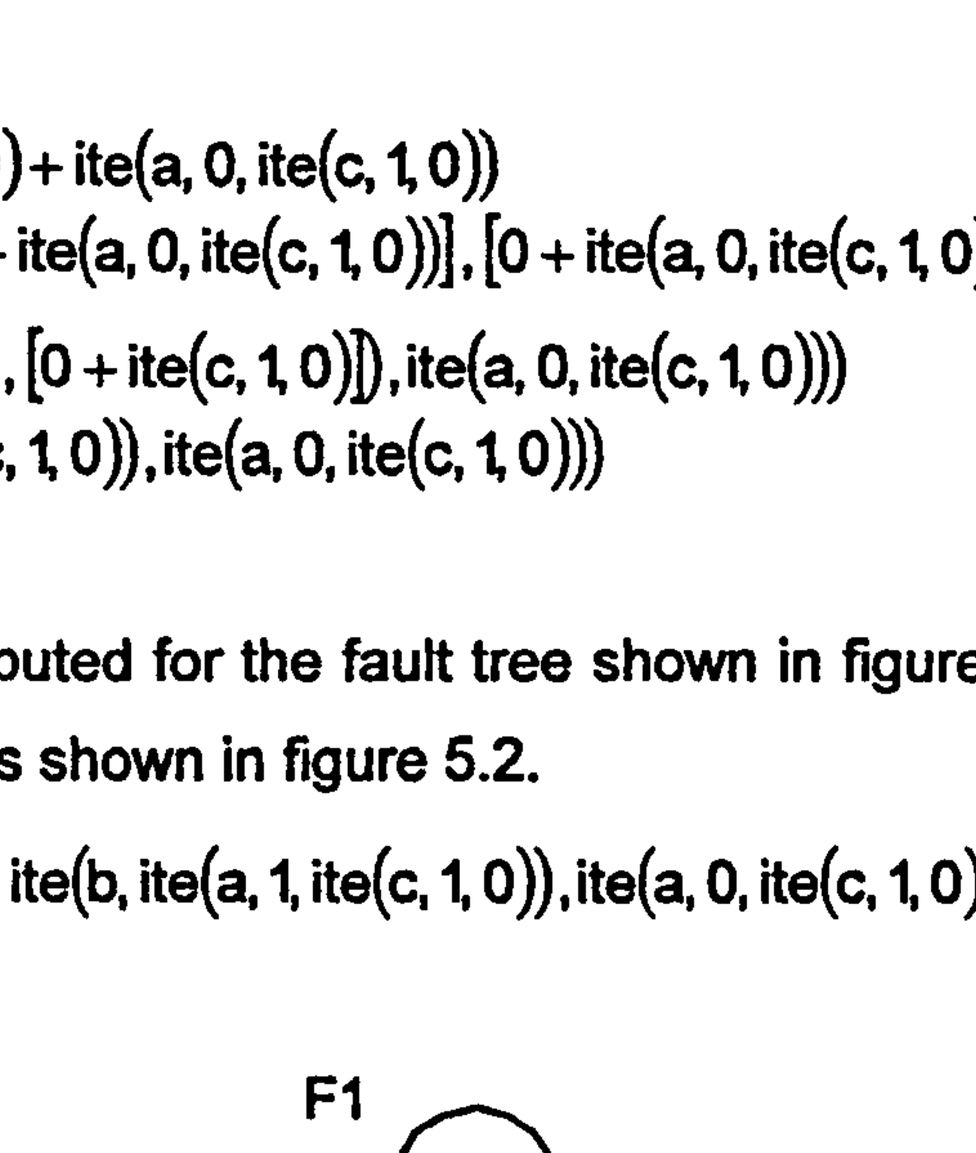


Figure 5.2: SFBDD for Fault Tree in Figure 5.1

5.3 Qualitative Analysis

Whilst the SFBDD does encode the structure function of the fault tree its minimal form can only be used to obtain a coherent approximation for qualitative analysis, this will be considered in section 5.3.1. Additional work is required to identify the prime implicant sets. Rauzy and Dutuit developed a technique to identify the prime implicant sets, which involves computing another BDD known as the meta-products BDD from the SFBDD [4,5,6,7]. This technique will be the focus of section 5.3.2.1. A new alternative technique, which computes what is known as the Consensus BDD has also been developed which may possess some advantages to the meta-products BDD, this technique will be considered in section 5.3.2.2.

5.3.1 Obtaining a Coherent Approximation

The SFBDD of a non-coherent fault tree can be used to obtain a coherent approximation for qualitative analysis. This involves identifying only the positive parts of the prime implicant sets known as the minimal p-cuts. In order to identify the minimal p-cuts, the SFBDD obtained for the non-coherent fault tree is treated in the same way as a BDD from a coherent fault tree. Hence it is first necessary to minimise the SFBDD removing any solutions on the one branch of a node that are also solutions on the zero branch of the node, thus eliminating any redundancies from the SFBDD to produce a minimal BDD. A full list of minimal p-cuts is then obtained by tracing the terminal one paths through the minimised BDD.

Consider the non-coherent fault tree given in figure 5.1, which has three prime implicant sets, $\{ab\},\{\bar{a}c\},\{bc\}$. The SFBDD given in figure 5.2 is non-minimal and thus must be minimised before the minimal p-cuts can be identified exactly. If F4 is retained on the zero branch of node F2 it will result in the non-minimal combination bc. Hence the zero branch of F2 is terminated with a zero resulting in the minimised BDD shown in figure 5.3.

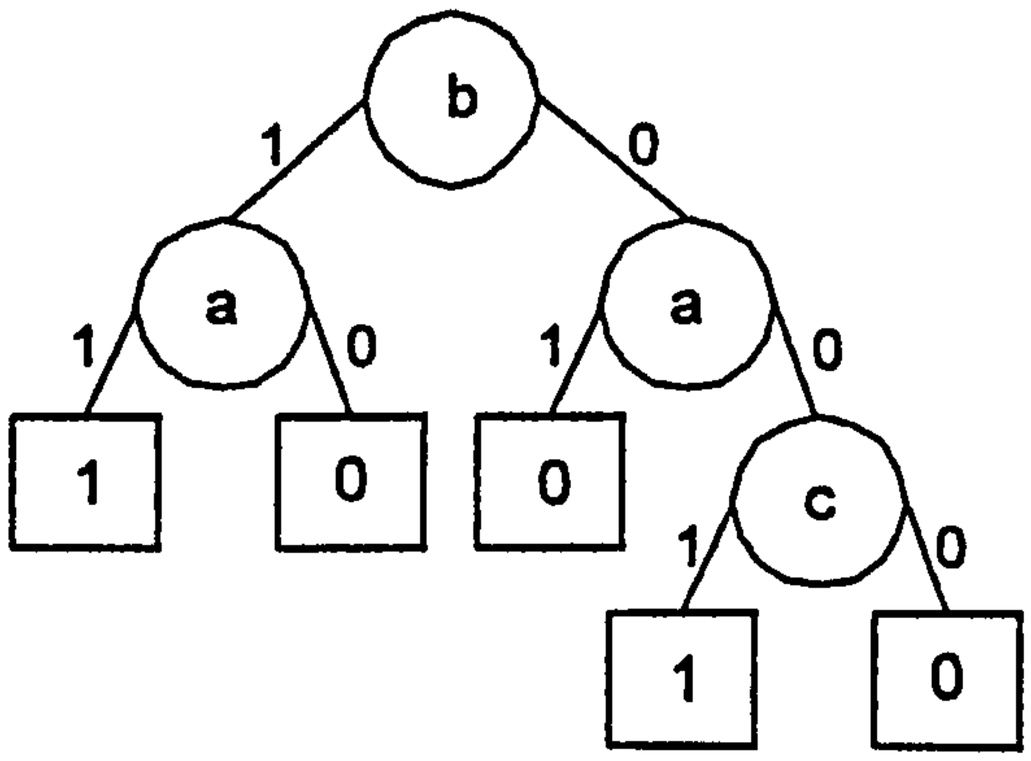


Figure 5.3: Minimised BDD

Now tracing all the terminal one paths through the minimised BDD in figure 5.3 and disregarding all the negated variables in these paths the following coherent approximation is obtained:

$$\{ab\}, \{c\}.$$

5.3.2 Exact Qualitative Analysis

Although a coherent approximation for qualitative analysis may be sufficient in some cases, knowledge of the prime implicant sets can be extremely valuable for two main reasons. Firstly, it can help to develop a repair schedule for failed components if a system cannot be taken off line for repair. Secondly, it can highlight where safety systems should be incorporated into the system in order to prevent hazardous events.

To illustrate this consider a system, which has three components A, B and C. Suppose that one failure state for the system is represented by the prime implicant set $\{AB\overline{C}\}$ and that if this failure state occurs it results in a serious hazard. Knowledge of this prime implicant set would be valuable. Firstly if components A, B and C were in a failed state, it would be evident from the prime implicant set that component C should not be repaired until either component A or B had been repaired. Hence unnecessary system failures could be avoided. Secondly the analyst would realise that by incorporating a safety system into the system, which causes component C to fail if components A, and B were both in a failed state, the occurrence of hazardous events would be avoided.

5.3.2.1 Rauzy and Dutuit's Meta-products Method

The SFBDD cannot be used directly to produce the complete list of prime implicant sets of a non-coherent fault tree. To understand why this is consider a general component x in a non-coherent fault tree. Component x can either contribute to a particular failure mode in its failed state, working state or be excluded from the failure mode. In the first two of these situations x is said to be relevant, in the third case it is irrelevant to the system state. If x is relevant then it can be either failure relevant (x appears in the prime implicant set) or repair relevant (x appears in the prime implicant set).

Now consider a general node in the SFBDD for a non-coherent fault tree representing component x. The one branch of this node corresponds to the failure of x; hence x is either failure relevant or irrelevant. Similarly, the zero branch corresponds to the functioning of x and so x is either repair relevant or irrelevant. It is impossible to distinguish between the two cases for each branch and consequently the prime implicant sets cannot be identified from the SFBDD.

In order to overcome this problem Rauzy and Dutuit developed an alternative notation that associates two variables with every component x. The first variable is P_x, which denotes relevancy and the second is S_x, which denotes the type of relevancy, i.e. failure relevant or repair relevant. This notation enables the meta-products of a system to be encoded. Where a meta-product, $MP(\pi)$, is the intersection of all the system components according to their relevancy to the system state and π represents the prime implicant set encoded in meta-product $MP(\pi)$.

$$MP(\pi)=\begin{cases}(P_x \wedge S_x) \text{ if } x \in \pi \\ (P_x \wedge \bar{S}_x) \text{ if } \bar{x} \in \pi \\ \bar{P}_x \text{ if neither } x \text{ nor } \bar{x} \text{ belongs to } \pi\end{cases}$$

Rauzy proposed a method for calculating the meta-products BDD of a fault tree from the SFBDD. The meta-products BDD is always minimal, thus it encodes the prime implicant sets exactly. Rauzy and Dutuit developed a procedure called MPPI that coverts the SFBDD into the meta-products BDD. It is then possible to identify the prime implicant sets from this BDD by eliminating all the irrelevant components in the meta-products. The algorithm for computing the meta-products BDD from the SFBDD is applied in a top-down fashion, beginning with the root vertex of the SFBDD. It is outlined below.

MPPI

Given a node with ite structure:

$$ite(x_i, f1, f0)$$

The meta-products structure for this node is denoted by:

$$PI[ite(x_i, f1, f0),\ L]$$

Note: L denotes the ordered list of all basic events except for those that appear on the current path from the root node to the node under consideration.

$PI[ite(x_i, f1, f0),\ L]$ is evaluated according to the following rules:

- If x_i is the first basic event in L:

$$PI[ite(x_i, f1, f0),\ L] = ite(P_{x_i}, ite(S_{x_i}, P1, P0), P2)$$

Where:

$$P2 = PI[f1 \cdot f0,\ L']$$

$$P1 = PI[f1, L'] \cdot \overline{P2}$$

$$P0 = PI[f0, L'] \cdot \overline{P2}$$

And:

$L = x_i, x_{i+1}, \ldots, x_n$

$L' = x_{i+1}, x_{i+2}, \ldots, x_n$

- If x_i is not the first basic event in L, i.e. $L = x_j, x_{j+1}, \ldots, x_n$ such that $i > j$:

$$PI[ite(x_i, f1, f0), L] = ite(P_{x_j}, 0, PI[ite(y, f1, f0), L'])$$

These rules are applied in conjunction with the following identities:

$$PI[0, L] = 0$$

$$PI[1, x \cdot L] = ite(P_x, 0, PI[1, L])$$

In order to obtain the meta-products structure of the root vertex, $ite(x, f1, f0)$, denoted by $PI[ite(x, f1, f0),\ L]$ the following three calculations must be performed:

- The calculation of P2, which encodes the prime implicant sets for which x is irrelevant.
- The calculation of P1, which encodes the prime implicant sets for which x is failure relevant
- The calculation of P0, which encodes the prime implicant sets for which x is repair relevant.

To calculate P2 it is first necessary to obtain the basic ite structure of $f2 = f1 \cdot f0$ where f1 and f0 represent the one and zero branches of the root vertex respectively. Next the meta-products structure of f2, denoted by $PI[f2, L]$ must be computed, where L represents the ordered list of basic events, $x_i,...,x_n$ that have not yet been considered in the conversion process. If f2 is a terminal node the conversion process can be performed immediately as follows:

- If f2=0, $PI[f2, L] = 0$
- If f2=1, $PI[f2, L] = ite(P_{x_i}, 0, ite(P_{x_{i+1}}, 0, ..., ite(P_{x_n}, 0, 1)))$

If however f2 is not terminal, MPPI calls itself to compute the meta-products structure of f2, before continuing the calculation of $PI[ite(x_i, f1, f0), L]$ at the previous level. The recursive nature of MPPI means that nodes for which f2 is terminal will necessarily be evaluated first.

The same procedure is implemented for calculating the meta-products structure of f1 and f0, denoted by $PI[f1, L]$ and $PI[f0, L]$ respectively. In order to ensure that the meta-products BDD is minimal, the meta-products structure obtained for f1 and f0 is ANDED with $\overline{P2}$. This eliminates any minimal solutions from $PI[f1, L]$ and $PI[f0, L]$ that are also minimal solutions of P2 and thus produces a minimal meta-products BDD.

The calculations of P1 and P0 for each node are performed together with the calculation for P2 so that only one depth first traversal of the SFBDD is required to compute the meta-products BDD.

To illustrate how this algorithm is applied in practice consider the SFBDD in figure 5.2, which has the following ite structure.

$$ite(b, ite(a, 1, ite(c, 1, 0)), ite(a, 0, ite(c, 1, 0)))$$

To encode the meta-products BDD the meta-products structure must be computed for this ite structure:

$$PI[ite(b, ite(a, 1, ite(c, 1, 0)), ite(a, 0, ite(c, 1, 0))), bac] = ite(Pb, ite(Sb, P1, P0), P2)$$

Where:

$$P2 = PI[ite(a, 1, ite(c, 1, 0)) \cdot ite(a, 0, ite(c, 1, 0)) \ , ac]$$

$$P1 = PI[ite(a, 1, ite(c, 1, 0)) \ , ac] \cdot \overline{P2}$$

$$P0 = PI[ite(a, 0, ite(c, 1, 0)) \ , ac] \cdot \overline{P2}$$

Evaluating P2

$$\begin{aligned} P2 &= PI[ite(a, 1, ite(c, 1, 0)) \cdot ite(a, 0, ite(c, 1, 0)) \ , ac] \\ &= PI[ite(a, 0, ite(c, 1, 0)) \ , ac] \\ &= ite(P_a, ite(S_a, P1.1, P0.1), P2.1) \\ &= ite(P_a, ite(S_a, 0, ite(P_c, ite(S_c, 1, 0), 0)), 0) \end{aligned}$$

Where:

$$\begin{aligned} P2.1 &= PI[0 \cdot ite(c,1,0) \ , c] \\ &= PI[0 \ , c] \\ &= 0 \end{aligned}$$

$$\begin{aligned} P1.1 &= PI[0 \ , c] \cdot \overline{P2.1} \\ &= 0 \cdot 1 \\ &= 0 \end{aligned}$$

$$\begin{aligned} P0.1 &= PI[ite(c,1,0) \ , c] \cdot \overline{P2.1} \\ &= ite(P_c, ite(S_c, 1, 0), 0) \cdot 1 \\ &= ite(P_c, ite(S_c, 1, 0), 0) \end{aligned}$$

Evaluating P1

$$\begin{aligned} P1 &= PI[ite(a, 1, ite(c, 1, 0)) \ , ac] \cdot \overline{P2} \\ &= ite(P_a, ite(S_a, P1.2, P0.2), P2.2) \cdot \overline{P2} \\ &= ite(P_a, ite(S_a, ite(P_c, 0, 1), 0), ite(P_c, ite(S_c, 1, 0), 0)) \cdot ite(P_a, ite(S_a, 1, ite(P_c, ite(S_c, 0, 1), 1)), 1 \\ &= ite(P_a, ite(S_a, ite(P_c, 0, 1), 0), ite(P_c, ite(S_c, 1, 0), 0)) \end{aligned}$$

Where:

$$\begin{aligned} P2.2 &= PI[1 \cdot ite(c, 1, 0) \ , c] \\ &= PI[ite(c, 1, 0) \ , c] \\ &= ite(P_c, ite(S_c, 1, 0), 0) \end{aligned}$$

$$\begin{aligned} P1.2 &= PI[1 \ , c] \cdot \overline{P2.2} \\ &= ite(P_c, 0, 1) \cdot ite(P_c, ite(S_c, 0, 1), 1) \\ &= ite(P_c, 0, 1) \end{aligned}$$

$$P0.2 = PI[ite(c,1,0)\ ,c]\cdot\overline{P2.2}$$
$$= ite(P_c, ite(S_c,1,0),0)\cdot ite(P_c, ite(S_c,0,1),1)$$
$$= ite(P_c, ite(S_c,0,0),0)$$
$$= 0$$

Evaluating P0

$$P0 = PI[ite(a,0,ite(c,1,0))\ ,ac]\cdot\overline{P2}$$
$$= ite(P_a, ite(S_a,0, ite(P_c, ite(S_c,1,0),0)),0)\cdot ite(P_a, ite(S_a,1, ite(P_c, ite(S_c,0,1),1)),1)$$
$$= ite(P_a, ite(S_a,0, ite(P_c, ite(S_c,0,0),0)),0)$$
$$= ite(P_a, ite(S_a,0,0),0)$$
$$= 0$$

Hence the **ite** structure of the meta-products BDD for the SFBDD in figure 5.2 is given in equation (5.2):

$$ite(P_b, ite(S_b, ite(P_a, ite(S_a, ite(P_c,0,1),0), ite(P_c, ite(S_c,1,0),0)),0), ite(P_a, ite(S_a,0, ite(P_c, ite(S_c,1,0),0)),0)) \quad (5.2)$$

The corresponding meta-products BDD is given in figure 5.4:

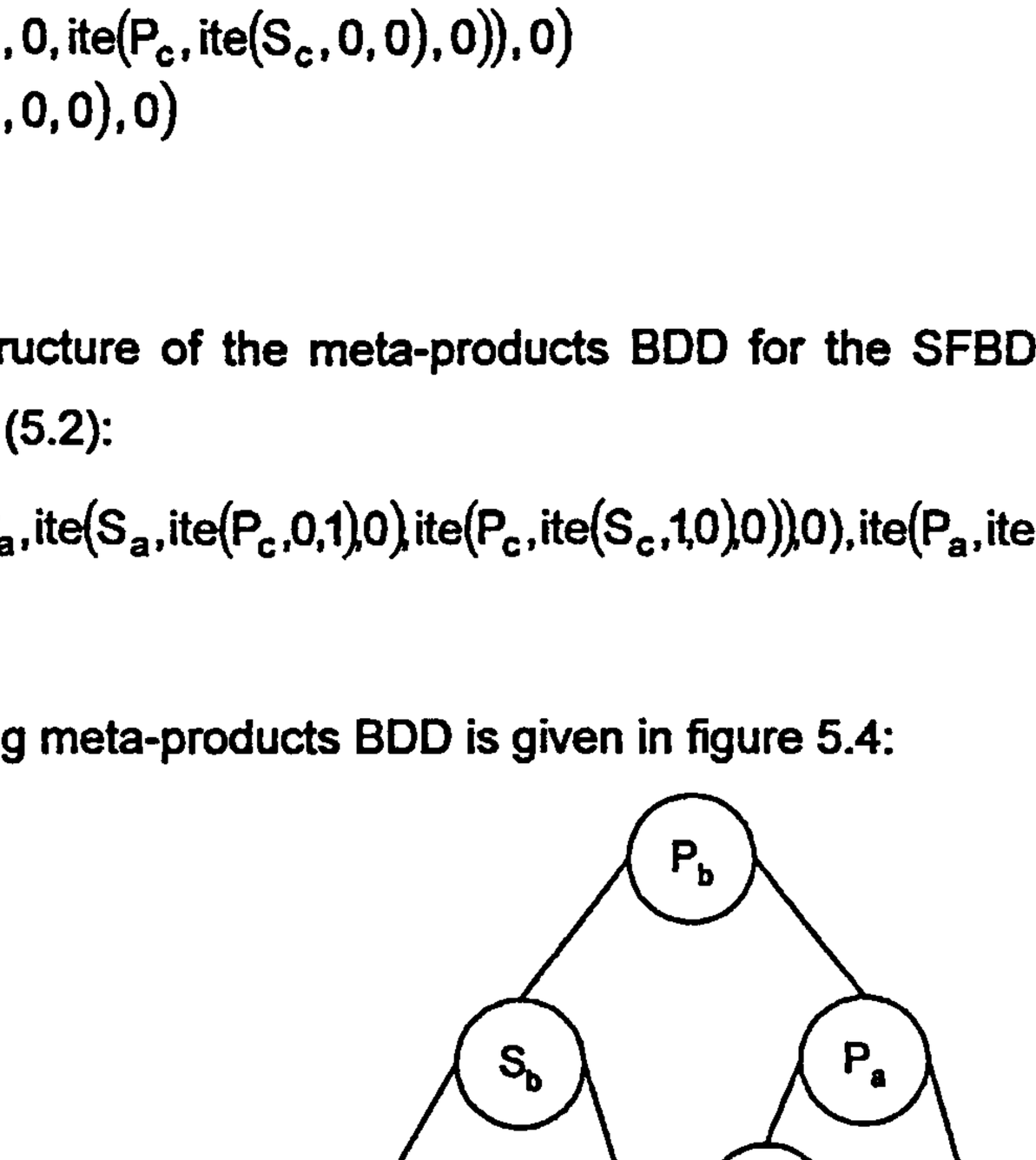


Figure 5.4: Meta-products BDD Computed from the SFBDD Given in Figure 5.2

From the meta-products BDD shown in figure 5.4, it is possible to obtain the meta-products, by tracing the terminal one paths through the BDD, the prime implicant sets are subsequently identified from the mate-products as shown below.

$$P_b \wedge S_b \wedge P_a \wedge S_a \wedge \overline{P}_c = ab$$

P_b signifies that component b is relevant and S_b signifies that component b is failure relevant. Component a is also failure relevant. Finally $\overline{P}_c$ signifies that component c is irrelevant. Hence the prime implicant set ab is obtained from this meta-product. Interpreting the remaining two meta-products in the same way gives:

$$P_b \wedge S_b \wedge \overline{P}_a \wedge P_c \wedge S_c = bc$$
$$\overline{P}_b \wedge P_a \wedge \overline{S}_a \wedge P_c \wedge S_c = \overline{a}c$$

5.3.2.2 An Alternative Method for Identifying the Prime Implicant Sets

As part of this research project an alternative method has been developed for qualitative analysis of non-coherent fault trees using the BDD technique. This method produces a Consensus BDD directly from the fault tree. It is known as the Consensus BDD because the method used to compute this BDD employs the consensus theorem, (introduced in chapter four) in order to encode the "hidden" prime implicant sets.

A three-way **ite** structure is used to distinguish not only between relevant and irrelevant components but also to distinguish between the type of relevancy, i.e. failure relevant and repair relevant. The **ite** structure for a general component x is an ordered quadruple given below:

T=ite(x, f1, f0, f2)

Where the first component is the variable of the node and each node in the Consensus BDD has three branches. Thus for a general node given in figure 5.5, the one branch encodes prime implicant sets for which component x is failure relevant, the zero branch encodes prime implicant sets for which x is repair relevant, and the consensus branch encodes prime implicant sets for which component x is irrelevant.

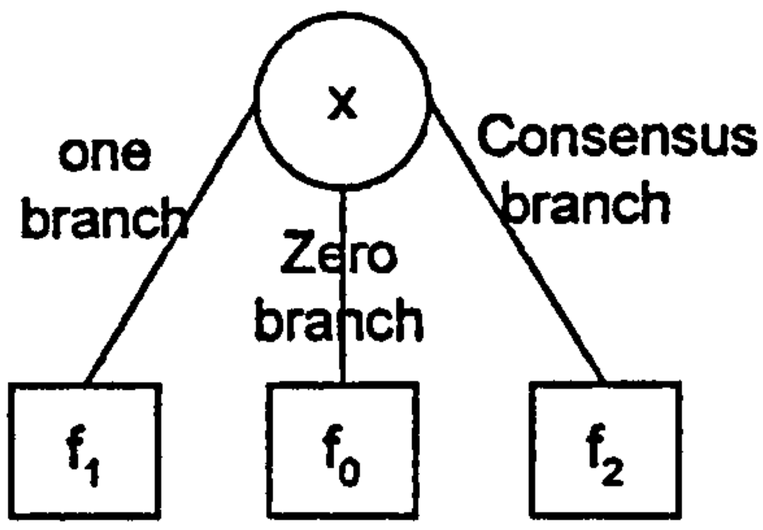


Figure 5.5: Three-way **ite** Structure

The **ite** structure shown in figure 5.5 is interpreted as follows:

If x is failure relevant **then** consider function f_1

Else if x is repair relevant **then** consider function f_0

Else consider function f_2.

5.4.2.2.1 Computing the Consensus BDD

The conversion process for computing the Consensus BDD is similar to that for computing the SFBDD. Before the conversion process can proceed the basic events of the fault tree must be ordered. The conversion process is outlined below:

1. Assign each basic event x in the fault tree an **ite** structure, $x = \text{ite}(x,1,0,0)$.
2. Modify the fault tree structure so that each gate has only two inputs.
3. By the application of De Morgan's laws push any NOT gates down through the fault tree structure until it reaches a basic event level.
4. Consider each gate in a bottom-up fashion.
5. If the two gate inputs are J and I such that:

$$J = \text{ite}(x, F1, F0, F2) \qquad I = \text{ite}(y, H1, H0, H2)$$

Then the following rules are applied:

- If $x < y$, $J < op > I = \text{ite}(x, K1, K0, [K1 \cdot K0])$

  Where, $K1 = F1 < op > I$ and $K0 = F0 < op > I$, and $K1 \cdot K0$ represents the consensus of K1 and K0

- If $x = y$, $J < op > I = \text{ite}(x, L1, L0, [L1 \cdot L0])$

  Where, $L1 = F1 < op > H1$ and $L0 = F0 < op > H0$, and $L1 \cdot L0$ represents the consensus of L1 and L0.

These rules are used in conjunction with the following identities:

- $1 < op > H = H \quad 0 < op > H = 0 \quad$ if $< op >$ is and AND gate
- $1 < op > H = 1 \quad 0 < op > H = H \quad$ if $< op >$ is and OR gate

Within each **ite** calculation an additional consensus calculation is performed to ensure all the hidden prime implicant sets are encoded in the BDD obtained. This operation calculates the product of the one and the zero branch of each node and thus identifies the consensus of each node.

To demonstrate how this procedure is employed consider the non-coherent fault tree in figure 5.1:

Assuming a variable ordering $b < a < c$

Assigning each variable an **ite** structure:

$$a = \text{ite}(a,1,0,0) \quad c = \text{ite}(c,1,0,0)$$
$$\bar{a} = \text{ite}(a,0,1,0) \quad b = \text{ite}(b,1,0,0)$$

Computing the **ite** structure of each gate in the fault tree, beginning with gate G2:

$$G2 = a \cdot b$$
$$= \text{ite}(a,1,0,0) \cdot \text{ite}(b,1,0,0)$$
$$= \text{ite}(b, f1, f0, f1 \cdot f0)$$

Where:

$$f1 = 1 \cdot \text{ite}(a,1,0,0) = \text{ite}(a,1,0,0)$$

$$f0 = 0 \cdot \text{ite}(a,1,0,0) = 0$$

$$f1 \cdot f0 = \text{ite}(a,1,0,0) \cdot 0 = 0$$

Hence the final **ite** structure for gate G2 is given below:

$$\text{ite}(b,\text{ite}(a,1,0,0),0,0)$$

Dealing with gate G1:

$$G1 = \bar{a} \cdot c$$
$$= ite(a,0,1,0) \cdot ite(c,1,0,0)$$
$$= ite(a, f1, f0, f1 \cdot f0)$$

Where:

$$f1 = 0 \cdot ite(c,1,0,0) = 0$$

$$f0 = 1 \cdot ite(c,1,0,0) = ite(c,1,0,0)$$

$$f1 \cdot f0 = 0 \cdot ite(c,1,0,0) = 0$$

Hence the final ite structure for gate G2 is given below:

ite(a,0,ite(c,1,0,0),0)

Finally Dealing with the top gate, Top:

$$Top = G1 + G2$$
$$= ite(b, ite(a,1,0,0)0,0) + ite(a,0, ite(c,1,0,0),0)$$
$$= ite(b, f1, f0, f1 \cdot f0)$$

Where:

f1=ite(a,1,0,0)+ite(a,0,ite(c,1,0,0),0)

f0=0+ite(a,0,ite(c,1,0,0),0)=ite(a,0,ite(c,1,0,0),0)

Evaluating f1:

f1=ite(a,f1.1,f0.1, f1.1 · f0.1)

Where:

f1.1=1+0=1

f0.1=0+ite(c,1,0,0)=ite(c,1,0,0)

$$f1.1 \cdot f0.1 = 1 \cdot ite(c,1,0,0)$$

Evaluating f1 · f0:

$$f1 \cdot f0 = ite(a,1,ite(c,1,0,0),ite(c,1,0,0)) \cdot ite(a,0,ite(c,1,0,0),0)$$
$$= ite(a, f1.2, f0.2, f1.2 \cdot f0.2)$$

Where:

$f1.2 = 1 \cdot 0 = 0$

$f0.2 = ite(c,1,0,0) \cdot ite(c,1,0,0) = ite(c,1,0,0)$

$f1.2 \cdot f0.2 = 0 \cdot ite(c,1,0,0) = 0$

The final **ite** structure obtained for the fault tree in figure 5.1 is given below:

ite(b,(ite(a,1,ite(c,1,0,0),ite(c,1,0,0)),ite(a,0,ite(c,1,0,0),0),ite(a,ite(c,1,0,0),0))

Figure 5.6 given the Consensus BDD obtained for the fault tree given in figure 5.1:

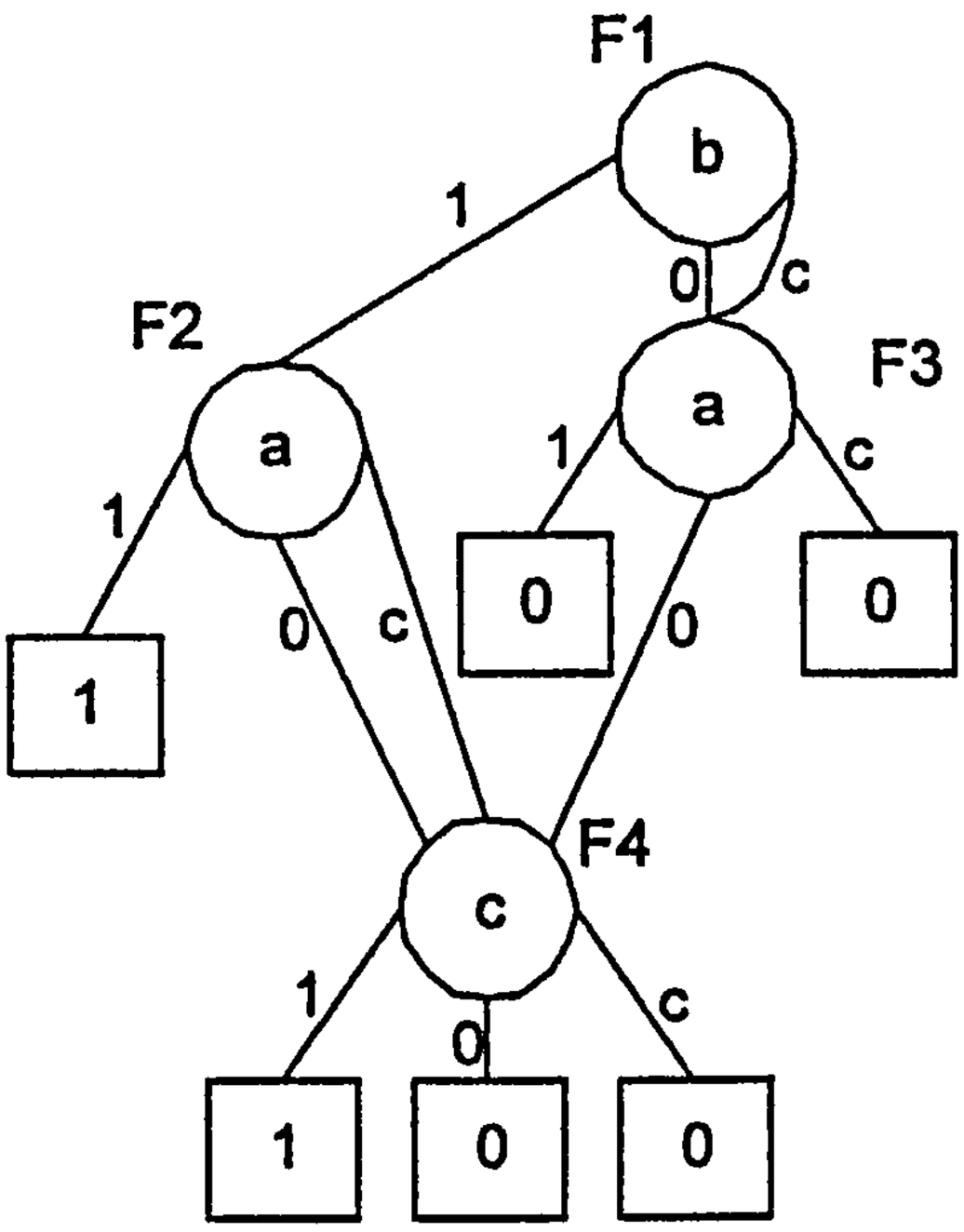


Figure 5.6: Consensus BDD for Fault Tree Shown in Figure 5.1

5.4.2.2.2 Minimising the Consensus Binary Decision Diagram

Once the consensus **ite** structure of the top event has been computed, there is no guarantee that the resulting Consensus BDD will be minimal, i.e. produce the prime implicant sets exactly. Although the form of the Consensus BDD generated initially needs to be retained for quantitative analysis, in order to perform qualitative analysis a minimisation procedure needs to be implemented.

The Consensus BDD can be used to obtain a list of implicant sets then; the Boolean Reduction Laws introduced in chapter two can be applied to these implicant sets to

produce a list of prime implicant sets. However, this is an inefficient means of obtaining the prime implicant sets. Instead a procedure that minimises the BDD itself, similar to the procedure developed by Rauzy for minimising the SFBDD [3] can be used to create a new minimal BDD that encodes the prime implicant sets exactly.

The procedure is applied to each node in the Consensus BDD. For a general node the procedure states:

Given an output of a node represented by the function F, where $F = \text{ite}(x,G,H,I)$. The set of all minimal solutions of F will include the minimal solutions of G and H that are not minimal solutions of I and also all minimal solutions of I. Let δ be a minimal solution of G which is **not** a minimal solution of I, then the intersection of δ and x will be a minimal solution of F given by, $F_{min} = \{\delta\} \cap x$. Similarly let γ be a minimal solution of H which is **not** a minimal solution of I, then the intersection of γ and $\bar{x}$ will be a minimal solution of F, given by, $F_{min} = \{\gamma\} \cap \bar{x}$. Finally if the set of all minimal solutions of H, is given by, $(\text{sol}_{min}(H))$:

$$\text{sol}_{min}F = [\{\delta\} \cap x] \cup [\{\gamma\} \cap \bar{x}] \cup [\text{sol}_{min}(H)]$$

There are essentially five stages to minimising $F = \text{ite}(x,G,H,I)$:

1. Identify the minimal solution of G, G_{min}.
2. Identify the minimal solution of H, H_{min}.
3. Identify the minimal solution of I, I_{min}.
4. Remove all minimal solutions of G_{min} that are also solutions of I_{min}, $\text{without}(G_{min}, I_{min})$.
5. Remove all minimal solutions of H_{min} that are also solutions of I_{min}, $\text{without}(H_{min}, I_{min})$.

Stages 4 and 5 of the procedure are critical since, if a minimal solution of I, say y is retained on the either the one branch as a minimal solution of G or on the zero branch as a minimal solution of H, it will result in a non-minimal combination $y \cdot x$, or $y \cdot \bar{x}$ respectively. This minimisation procedure is an extension of Rauzy's method for minimising the SFBDD hence the algorithms 'minsol' and 'without' were modified in

order to produce a working code to minimise a given consensus BDD. 'Minsol' computes the minimal solution of a node and calls 'without' to compute δ and γ , i.e. the minimal solutions of G and H that are not minimal solutions of I. These algorithms are outlined in appendix (I).

To illustrate this procedure consider the Consensus BDD given in figure 5.6. The nodes are considered in a top down fashion, starting with the top node. The minimal solutions of the one branch of a node are computed first, then the minimal solutions of the zero branch and then the minimal solutions of the consensus branch are obtained. Finally all solutions that exist on either the one or zero branch of a node that also exist on the consensus branch of the node are removed from the one or zero branch and replaced with zero.

If a node is terminal then it is automatically minimal. If however, it is non-terminal the new node must be minimised before the minimisation of the current node can be completed.

Table 5.1 records the one, zero and consensus branches of each node in the Consensus BDD and the modified one and zero branches that are obtained as part of the minimisation process. Two modifications are made to the Consensus BDD.

- Firstly both the zero branch and the consensus branch of node F1 point to node F4. Thus if F4 is retained on the one branch of node F1 it will results in non-minimal combinations. Hence it is replaced with a 0.

- Secondly the zero branch and the consensus branch of node F2 both point to node F3. Hence to remove redundant implicant sets the zero branch of node F2 is replaced with 0.

Node	One Branch	Zero Branch	Consensus Branch	Modified one branch	Modified zero branch
F1	F2	F3	F3	F2	0
F2	1	**F4**	**F4**	1	**0**
F3	0	F4	0	0	F4
F4	1	0	0	1	0

Table 5.1: The Node Connections for the Consensus BDD in Figure 5.6

The Consensus BDD in figure 5.6 has now been minimised and minimal BDD is shown in figure 5.7.

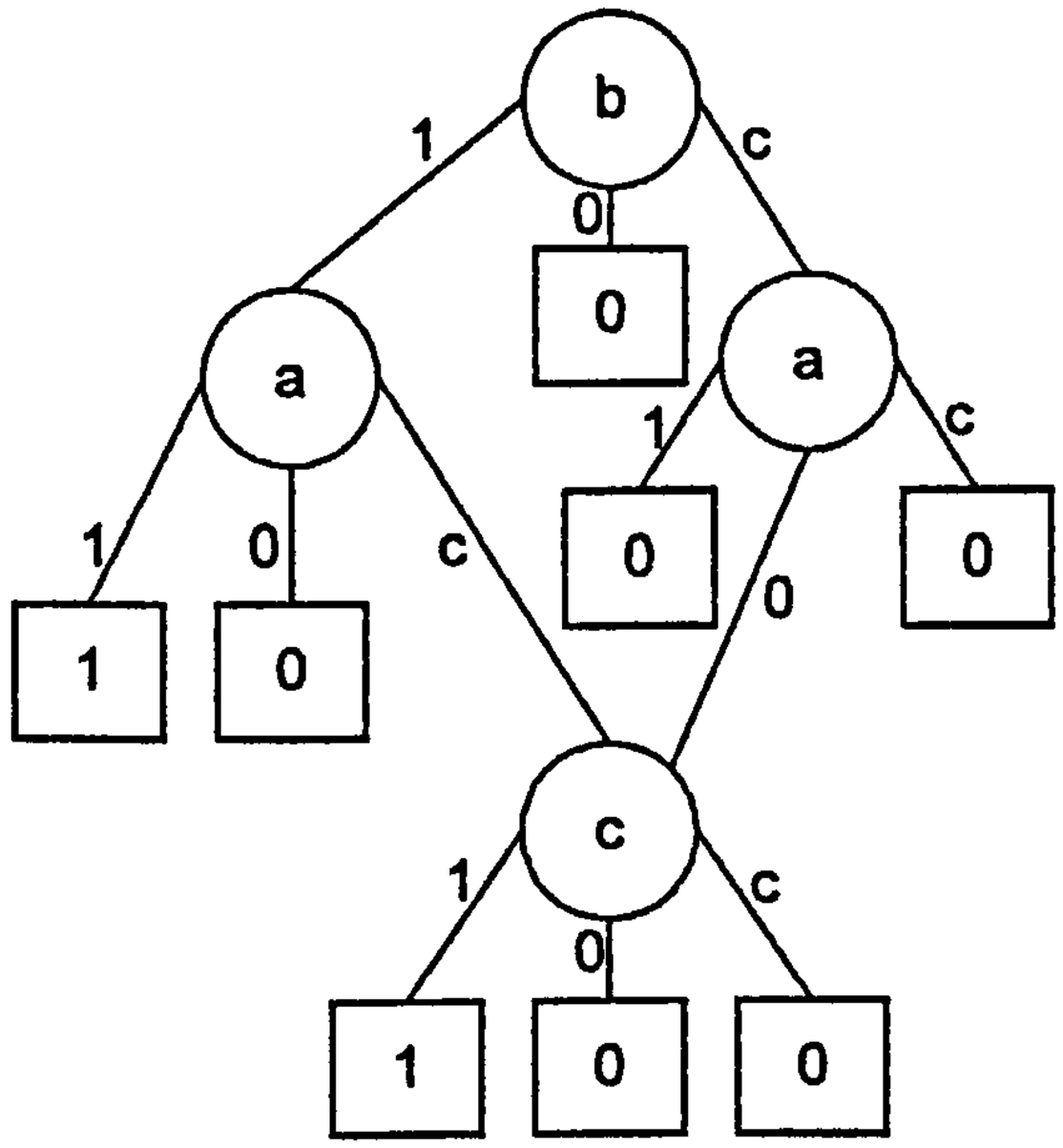


Figure 5.7: Minimised BDD Obtained from the Consensus BDD in Figure 5.6

It is then possible to obtain a full list of prime implicant sets by tracing all the terminal one paths through the minimised BDD:

$$\{ab\},\{\bar{a}c\},\{bc\}$$

5.4 Quantitative Analysis

One of the major advantages of the BDD method is that unlike conventional Fault Tree Analysis techniques it does not require knowledge of the prime implicant sets (minimal cut sets in the case of coherent fault trees) for quantification. It is possible to use the SFBDD or the Consensus BDD to perform full and exact quantitative analysis. The procedures for calculating both the system unavailability from the SFBDD and the Consensus BDD will be considered in sections 5.4.1.1 and 5.4.1.2 respectively. The calculation procedures for the most commonly used measures of importance and the unconditional failure intensity will be the focus of chapter seven.

5.4.1 Calculating the System Unavailability

The system unavailability is defined as the probability that the system is in a failed state at time t. Conventional Fault Tree Analysis techniques for calculating the system unavailability are lengthy and inefficient. The BDD technique enables more efficient and accurate quantification of the system unavailability. Techniques for calculating the system unavailability from the SFBDD and the Consensus BDD will be considered in sections 5.4.1.1 and 5.4.1.2 respectively.

5.4.1.1 Calculating the System Unavailability from the SFBDD

The SFBDD for a non-coherent fault tree encodes Shannon's decomposition. Hence, as Andrews [14] demonstrated the calculation procedure introduced in chapter three section 3.5.1 for calculating the top event probability from the SFBDD can be employed for non-coherent fault trees. Thus the top event probability for a non-coherent fault tree is obtained by summing the probabilities of the disjoint paths through the SFBDD.

To illustrate this consider the SFBDD shown in figure 5.2 This SFBDD has been computed for the non-coherent fault tree in figure 5.1. Three prime implicant sets can be identified from this fault tree:

$$\{a,b\},\{\bar{a},c\},\{b,c\}$$

Thus using the inclusion-exclusion expansion introduced in chapter three the expression in equation (5.3) is obtained for the top event probability.

$$Q_{SYS}(t) = q_a q_b + (1-q_a)q_c + q_b q_c - q_a q_b q_c - (1-q_a)q_b q_c = q_a q_b + (1-q_a)q_c \quad (5.3)$$

There are two disjoint paths through the SFBDD shown in figure 5.2, ab and $\bar{a}c$. Thus given that the top event probability is obtained by summing the probability of each disjoint path, the following expression is obtained:

$$Q_{SYS}(t) = q_a q_b + (1-q_a)q_c$$

This agrees with the expression derived by the Fault Tree Analysis method, and is a far more efficient means of calculating the top event probability since it eliminates both the need to identify the prime implicants sets and evaluate the many terms of the inclusion-exclusion expansion.

5.4.1.2 Calculating the System Unavailability using the Consensus Binary Decision Diagram

The SFBDD is encoded within the Consensus BDD. If all the consensus branches are removed from the Consensus BDD the SFBDD for the fault tree is obtained. Hence, the system unavailability can be calculated directly from the Consensus BDD, by summing the probability of all the terminal one paths through the BDD that only pass through the one or zero branches of non-terminal nodes.

To illustrate this technique consider the Consensus BDD given in figure 5.6. There are three terminal one paths through the BDD (since paths passing through consensus branches of non-terminal nodes are ignored). The three paths are: ba, $\bar{b}ac$ and $b\bar{a}c$, by taking the sum of the probability of these paths the following expression for the system unavailability is obtained:

$$Q_{SYS}(t) = q_b q_a + q_b p_a q_c + p_a p_b q_c = q_a q_b + (1 - q_a) q_c$$

5.5 A Comparison of the Two Methods

To test the efficiency of Rauzy and Dutuits method and the Consensus method, 20 example non-coherent fault trees of varying complexity were analysed using MPPI (a computational method for constructing and analysing the SFBDD and meta-products BDD) and Consensus (a computational method for constructing and analysing the Consensus BDD) and the results were compared. A top-down, left-right ordering of the basic events was used for both methods. Both of the codes run on a Sun workstation and the execution time is given in hours, minutes and seconds. The execution times include the computation of all BDD's required for both qualitative and quantitative analysis and the time taken for the analysis itself. Table 5.2 records the results.

Fault Tree	No. of Gates	No. of Basic Event	No. of Prime Implicant Sets	Time Taken for Meta-products	Time Taken for Consensus
Lisa102	63	105	736	6 hrs 38 mins	N/A
Lisa106	153	207	953	8 hrs 12 mins	N/A
Dre1058	13	41	1805	12 hrs 7 mins	N/A
Bpfsw02	19	40	249	4 hrs 21 mins	10hrs 49 mins
Bpfeg03	24	63	387	7 hrs 9 mins	N/A
Nakashi	22	15	68	2 hrs 14 mins	3 hrs 39 mins
Lisab13	25	30	153	5 hrs 26 mins	7 hrs 19 mins
Lisa118	42	77	948	12 hrs 8 mins	N/A
Lisa117	90	147	337	6 hrs 27 mins	N/A
Lisab15	53	98	496	8 hrs 33 mins	9 hrs 13 mins
Jdtree5	12	20	7	13 seconds	4 seconds
Jdtree3	12	21	18	14 seconds	3 seconds
Lisab56	13	17	10	23 seconds	4 seconds
Sjdtree	18	15	36	7 seconds	7 seconds
Lisa103	6	9	15	4 seconds	3 seconds
Fatram2	6	9	11	4 seconds	4 seconds
Rando15	7	11	10	4 seconds	3 seconds
Rando83	17	21	9	5 seconds	4 seconds
Rand146	11	21	6	5 seconds	7 seconds
Rand117	12	17	1	3 seconds	4 seconds

Table 5.2: Results for Analysing Non-coherent Fault Trees using Rauzy's Meta-products Method and the Consensus Method

The results in table 5.2 demonstrate that whilst for the smaller less complex fault trees (from jdtree-rand117) there was little difference in the performance of the two techniques. For the large more complex fault trees (from lisa102-lisab15) the meta-products technique was significantly more efficient than the consensus technique. In fact the consensus technique was unable to analyse six out of the ten more complex fault trees in a reasonable time. The time taken to analyse the remaining four fault trees is significantly greater than that taken by the meta-products technique.

The Consensus method for the analysis of non-coherent fault trees only has one node for each basic event with three branches denoting failure relevance, repair relevance and irrelevance. Rauzy and Duituits method assigns two variables to each basic event, the first denotes relevance and the second denotes the type of relevance. Thus the most significant difference between the two methods is the structure of the diagram used. However, the Consensus method has the disadvantage of requiring a minimisation process to be applied to the Consensus BDD computed, in order to encode the prime implicant sets exactly. This minimisation process can be lengthy for larger fault trees since it requires both the one and zero branch of each node to be compared to the consensus branch and reduced where redundancies exist.

5.6 Summary

The BDD method for the analysis of non-coherent fault trees can be more efficient and accurate than conventional techniques for non-coherent FTA especially when dealing with large fault trees with many repeated events. Full qualitative analysis can be performed using the BDD technique and the system unavailability can be calculated without knowledge of the prime implicant sets and there are no lengthy series expansions to evaluate. The calculation of importance measures using the BDD technique will be considered in detail in chapter seven.

Knowledge of the prime implicant sets can be of value to the analyst for planning repair schedules and deciding where to incorporate safety systems. It is not possible to identify the prime implicant sets from the SFBDD directly, since no distinction is made between component, failure relevance, repair relevance and irrelevance. Two main methods for identifying the prime implicant sets using the BDD technique have been considered in this chapter. The first method developed by Rauzy and Dutuit requires the SFBDD to be converted to the meta-products BDD that encodes the prime implicant sets exactly. The second method converts the fault tree directly to a Consensus BDD, which is then minimised to encode the prime implicant sets exactly.

A comparison of the two methods revealed that although for moderate sized fault trees there is little difference in efficiency. For larger fault trees the meta-products technique is significantly more efficient than the consensus technique. The main disadvantage with the consensus technique is that a minimisation procedure has to be applied to the consensus BDD to identify a full list of prime implicant sets.

Although the BDD method for the analysis of non-coherent fault trees has advantages in terms of efficiency and accuracy over FTA, it suffers the same shortfalls as the BDD method for the analysis of coherent fault trees. Namely a variable ordering scheme must be chosen for the basic events of the fault tree in order to compute a SFBDD.

Chapter 6: Importance Analysis of Coherent Fault Trees

6.1 Introduction

When assessing a system, its performance is dependent on that of its components. Certain components will play a more significant role in causing or contributing to system failure than others. Birnbaum first introduced this concept of importance in 1969 [1], highlighting the value of numerically ranking the contribution of each component or basic event to reflect the susceptibility of the system to the occurrence of this event. Measures of importance assign a value between 0 and 1 to each component (or minimal cut set) with 1 signifying the highest level of contribution.

Importance analysis is now a key part of the quantification process, which enables the analyst to rank the contribution each component makes to system failure. The weakest areas of the system can be identified thus highlighting the modifications that will best improve the system reliability.

Measures of importance can be categorised as either deterministic or probabilistic. Numerous measures of importance have been developed for assessing the different roles a component failure can play in the deterioration of the system state. All measures of importance provide particular information about the system and its components. Hence it is critical to firstly choose suitable measures of importance according to the objectives of the analysis and secondly to understand how the results should be interpreted. This chapter will firstly introduce some of the most commonly used measures of importance and illustrate two calculation procedures for these measures. Finally an example will be used to illustrate how the measures of importance introduced are calculated and the results are interpreted.

6.2 Deterministic Measures of Importance

Deterministic measures of importance also known as structural measures of importance assess the importance of components and or minimal cut sets without taking into account the reliability of components. These measures are useful early on in the design phase because information concerning the failure probability of components tends to be limited at this time.

Birnbaum introduced a measure of structural importance, which enables the researcher to identify the proportion of critical vector states that exist for each component.

$$I_i(\varphi) = \frac{\text{No. of critical states for component i}}{\text{Total no. of states for the (n - 1) remaining components}} \qquad (6.1)$$

To illustrate how this measure is calculated consider a system with three minimal cut sets, AB, BC, AC, the structural importance of component A will be calculated. Firstly all the possible states for components B and C are identified. Then those states for which component A is critical are identified. Table 6.1 records the results.

States for Components b and c	Critical State for Component a
(., B, C)	No
(., B, $\overline{C}$)	Yes
(., $\overline{B}$, C)	Yes
(., $\overline{B}$, $\overline{C}$)	No

Table 6.1: System States

Then using equation (6.1) the following result is obtained for the structure importance of component a:

$$I_a(\varphi) = \frac{2}{4} = \frac{1}{2}$$

Although deterministic measures of importance can be of use, probabilistic measures of importance are generally preferred because they provide more valuable information.

6.3 Probabilistic Measures of Importance

Probabilistic measures of importance are far more useful than deterministic measures in practical reliability problems. This is because they take component failure probabilities and intensities into account and thus provide more detailed information about the system and its components. There are two types of probabilistic measures, those for assessing component importance and those for assessing minimal cut set importance.

Probabilistic measures can be further categorised as either measures concerned with system unreliability (contributing to the failure frequency) or measures concerned with system unavailability (contributing to the failure probability). There is a clear and important distinction between reliability and availability, which was, explained in chapter 1 section 1.1.

6.3.1 Measures for Assessing Component Importance

In a system, some components will contribute to system failure more than others. In order to improve the reliability of the system, those components most likely to contribute to system failure must be identified. Numerous measures of importance have been developed for assessing component importance, five of the most commonly used measures will be considered in detail below.

6.3.1.1 Birnbaum's Measure of Component Reliability Importance

Birnbaum introduced the concept of importance and a probabilistic measure of component reliability importance in 1969 [1]. This measure is denoted by $G_i(\underline{q})$ and is defined as the probability that component i is critical to system failure. Thus, the system is in a working state such that the failure of component i causes it to fail. An expression for this measure is given in equation (6.2):

$$G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q}) \tag{6.2}$$

Where, Q_{SYS} is the system unavailability function and $Q_{SYS}(1_i, \underline{q})$ is the probability that the system fails with component i failed and $Q_{SYS}(0_i, \underline{q})$ is the probability that the system fails with component i working.

Given that Q_{SYS} is linear in each q_i, equation (6.2) can be expressed as follows:

$$G_i(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial q_i(t)} \tag{6.3}$$

In addition to providing a measure of importance in its own right, this measure also forms the basis for a number of other measures of importance, including the Measure of Component Criticality, Barlow and Proschan's measure of component initiator importance [18] and Lambert's measure of component enabler importance [19]. These measures will now be considered.

6.3.1.2 The Component Criticality Measure

This measure is defined as the probability that component i is critical to the system and i has failed, weighted by the system unavailability at time t. An expression for this measure is given in equation (6.4).

$$I_{C_i} = \frac{G_i(q)q_i(t)}{Q_{SYS}(t)} \tag{6.4}$$

This measure takes into account the failure probability of component i, unlike Birnbaum's measure, and as such can provide additional information regarding how likely component i is to actually be in a failed state when the system is failed.

6.3.1.3 Fussell-Vesely's Measure of Component Importance

Fussell-Vesely's measure is concerned with the contribution component failures make to system failure [20]. The measure is defined as the probability that the failure of component i contributes to system failure and is numerically similar to the Component Criticality Measure:

$$I_{FV_i} = \frac{P\left[\bigcup_{\substack{k=1 \\ i \in C_K}}^{n_c} C_k\right]}{Q_{SYS}(t)} \tag{6.5}$$

6.3.1.4 Measure of Initiator and Enabler Importance

The measures introduced previously have not been concerned with the order in which the basic events fail. Although these measures can be useful for assessing importance, in certain circumstances the order in which basic events fail it is of vital importance to the occurrence of system failure. One circumstance when order of failure is of great importance is during the assessment of safety protection systems. If a hazardous event occurs whilst the safety system is failed the outcome would be a serious/dangerous system failure. If however, the hazardous event were to occur prior to the failure of the safety system then the appropriate action would have been taken to shutdown the system and thus prevent a dangerous system failure. This situation can be modelled by considering the component failures as either **initiating** or **enabling** events, and is illustrated more clearly in figure 6.1.

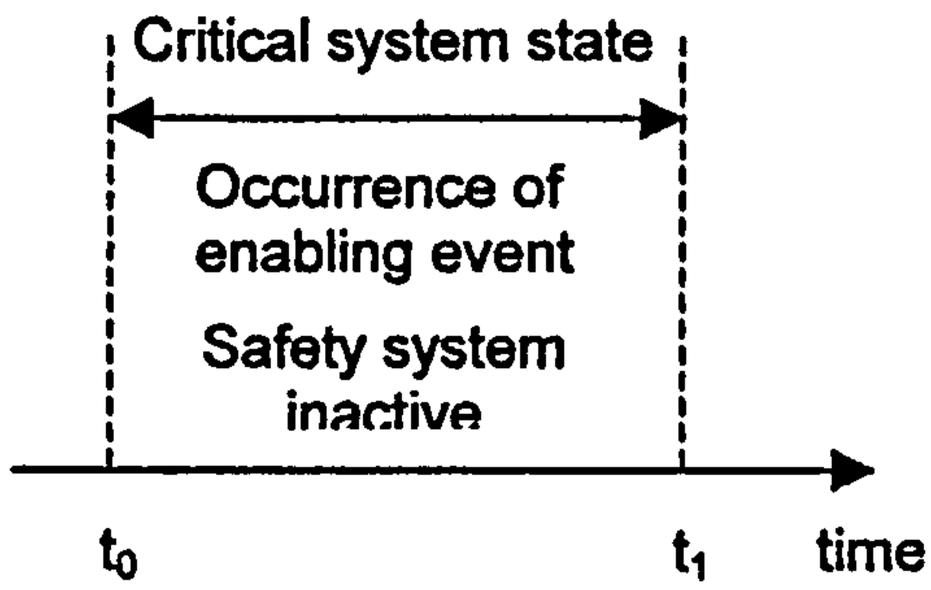


Figure 6.1: Initiating Event Window

In this diagram the safety system is inactive between t_0 and t_1. During this time the safety systems cannot respond to a hazardous event and therefore the system is in a critical state due to the occurrence of an enabling event. If the initiating event occurs within the interval (t_0, t_1) it will result in a dangerous system failure. If however, it occurs outside of this interval the safety system will respond as designed. Hence the order of component failures is clearly of importance. The formal definitions for initiating and enabling events are given below:

Initiating events perturb system variables placing a demand on control/protection systems to respond.

Enabling events are inactive control/protective systems, which permit initiating events to cause the top event.

Barlow and Proschan developed a measure to assess component initiator importance in 1974, and Lambert developed a measure of component enabler importance in 1975. Both these measures are considered separately below.

6.3.1.4.1 Barlow and Proschan's Measure of Component Initiator Importance

This measure is concerned with the failure of components acting as initiating events. If components are to be initiators then their occurrence must coincide with system failure. Barlow and Proschan's measure [18] calculates the probability that component i causes system failure in the interval $[0,t)$, an expression for this measure is given in equation (6.6).

$$I_{IN_i} = \frac{\int_0^t G_i(\underline{q}) w_i(u) du}{W_{SYS}(0,t)} \tag{6.6}$$

Where, $w_i(t)$ and $W_{SYS}(0,t)$ are the unconditional failure intensity of component i and the expected number of system failures in a given interval respectively.

It is important to realise that not all components can act as initiators. For example, safety systems are passive, and can only contribute to system failure, not directly cause it. Hence Barlow and Proschan's measure will not be suitable for assessing the importance of passive components.

6.3.1.4.2 Lambert's Measure of Component Enabler Importance

Lambert introduced the measure of enabler importance in 1975 [19], which is defined as the probability that the failure of component i allows the system failure in the interval $[0,t)$ caused by the failure of another component j occurring. Lambert's expression for this measure is given in equation (6.7):

$$I_{E_i} = \frac{\sum_{\substack{j \\ i \neq j \\ i,j \in C_k}} \int_0^t \left[Q_{SYS}(1_i,1_j,\underline{q}) - Q_{SYS}(1_i,0_j,\underline{q}) \right] q_i(u) w_j(u) du}{W_{SYS}(0,t)} \tag{6.7}$$

Where, i is the enabler and j is the initiator.

This expression does not exactly specify the probability that component i contributes to system failure in a given interval when another component, j causes system failure. It is an approximation because it does not take into account the separate roles of components i and j in causing or contributing to system failure. For

component i to enable system failure by the failure of component j, i and j must occur in at least one minimal cut set together and it must be the existence of one such minimal cut set that causes the system failure.

To illustrate this, consider the following example with four minimal cut sets:

$$\{abe, def, bg, dh\}$$

The Boolean expression for the top event and the system unavailability function are given below:

$$T = abe + def + bg + dh$$

$$\begin{aligned} Q_{SYS}(t) = {} & q_a q_b q_e + q_d q_e q_f + q_b q_g + q_d q_h - q_a q_b q_d q_e q_f - q_a q_b q_e q_g - q_a q_b q_d q_e q_h \\ & - q_b q_d q_e q_f q_g - q_d q_e q_f q_h - q_b q_d q_g q_h + q_a q_b q_d q_e q_f q_g + q_a q_b q_d q_e q_g q_h \\ & + q_a q_b q_d q_e q_f q_h + q_b q_d q_e q_f q_g q_h - q_a q_b q_d q_e q_f q_g q_h \end{aligned}$$

Consider the situation where d is an enabler and e is an initiator, then, for d to act as an enabler and contribute to system failure when it is e that actually causes system failure, they must occur in the same minimal cut set. This is minimal cut set 2, $\{def\}$. Therefore, f must fail and cut sets 1, 3 and 4 must not exist, if minimal cut set 2 is to cause system failure. This means that either b works or a works from minimal cut set 1, and g works or b works to prevent minimal cut set 3 and h works to prevent minimal cut set 4. Thus the required circumstances are:

$$f \text{ and } \overline{ab} \text{ and } \overline{bg} \text{ and } \bar{h}$$

$$\text{i.e. } f + (\bar{a} + \bar{b}) \cdot (\bar{b} + \bar{g}) \cdot \bar{h} = f \cdot \bar{h} \cdot (\bar{b} + \bar{a} \cdot \bar{g})$$

Taking the probability of this expression gives:

$$P[f \cdot \bar{h} \cdot (\bar{b} + \bar{a} \cdot \bar{g})] = q_f + (1 - q_h)[1 - q_a q_b - q_b q_g + q_a q_b q_g]$$

Using this probability, the correct expression can be obtained for the enabler measure of importance for component d when component e initiates system failure.

$$I_{AE_{d,e}} = \frac{\int_0^t [q_f (1 - q_h)[1 - q_a q_b - q_b q_g + q_a q_b q_g]] q_d w_e du}{W_{SYS}(0, t)}$$

Where, $I_{AE_{i,j}}$ is the actual enabler probability, the probability calculated using Lambert's formula is:

$$I_{E_{d,e}} = \frac{\int_0^t Q_{SYS}(1_d, 1_e, \underline{q}(u)) - Q_{SYS}(1_d, 0_e, \underline{q}(u)) q_d(u) w_e(u) du}{W_{SYS}(0,t)}$$

$$I_{E_{d,e}} = \frac{\int_0^t [(1-q_h)[q_a q_b (1-q_g) + q_f (1-q_a q_b - q_b q_g + q_a q_b q_g)]] q_d(u) w_e(u) du}{W_{SYS}(0,t)}$$

The two results differ. Notice that Lambert's measure has the extra terms:

$$(1-q_h)[q_a q_b - q_a q_b q_g] = q_a q_b (1-q_g)(1-q_h)$$

These occur because Lambert has failed to account for the fact that the components i and j can occur separately in failure combinations and consequently have an effect on system state individually as well as a jointly, Lambert has accounted for the independent role of the initiator but not the enabler.

The required probability is the probability that components i and j are critical to system state at time t and component i has failed and component j fails at time t all weighted by the expected number of system failures.

In order to calculate the required probability firstly the criticality of component i and j must be calculated and then a correction term which eliminates the separate contributions of i and j is required. The criticality of i and j is the probability that component i and j are critical to system state, i.e. the system is in a state at time t such that the failure of components i and j would cause system failure.

The probability that component i is critical to the system state at time t is calculated as follows:

$$G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}(t)) - Q_{SYS}(0_i, \underline{q}(t))$$

Where, $Q_{SYS}(1_i, \underline{q}(t))$ is the probability that the system is in a failed state at time t and i is failed and $Q_{SYS}(0_i, \underline{q}(t))$ is the probability that the system is in a failed state at time t and i is working.

Since $Q_{SYS}(t)$ is linear in $q_i(t)$ this can be re-written as follows.

$$G_i(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial q_i(t)}$$

Hence the probability that components i and j are critical to the system state, $G_{i,j}(\underline{q})$, can be obtained by extending this definition of criticality. Firstly the notation must be extended.

Let $Q_{SYS}(1_i, 1_j, \underline{q}(t))$ be the probability that the system is in a failed state with components i and j failed. Then $Q_{SYS}(1_i, 0_j, \underline{q}(t))$ is the probability that the system is in a failed state with component i failed and component j working. Then the probability that components i and j are critical to the system at time t can be expressed as follows:

$$G_{i,j}(\underline{q}) = Q_{SYS}(1_i, 1_j, \underline{q}(t)) - Q_{SYS}(1_i, 0_j, \underline{q}(t)) - Q_{SYS}(0_i, 1_j, \underline{q}(t)) + Q_{SYS}(0_i, 0_j, \underline{q}(t)) \quad (6.8)$$

This is because $Q_{SYS}(1_i, 1_j, \underline{q}(t))$ represents the probability that the system is in a failed state and components i and j are failed. In order to eliminate the individual contributions of components i and j, (i.e. if they are contained separately in minimal cut sets they make an individual contribution to system failure.) the probability that the system is failed when i is failed and j is working and the probability that the system is failed when i is working and j is failed are subtracted. Finally because this subtraction results in an underestimation the probability that the system is in a failed state and components i and j are working must be added.

Since $Q_{SYS}(t)$ is linear in $q_i(t)$ and $q_j(t)$ equation (6.8) can be re-written as follows.

$$G_{i,j}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_i \partial q_j} \quad (6.9)$$

Although $G_{i,j}(\underline{q})$ represents the probability that components i and j are critical to the systems state, it does not always produce the required probability, since the separate effects of i and j result in redundant combinations in the system unavailability function. This is illustrated by three examples below.

Example 1

$$T=abc+bd+cf$$

$$Q_{sys}(t)=q_aq_bq_c+q_bq_d+q_cq_f-q_aq_bq_cq_d-q_aq_bq_cq_f-q_bq_cq_dq_f+q_aq_bq_cq_dq_f$$

Let C_1=abc, C_2=bd, C_3=cf. Then considering the role of component a as initiator and component b as enabler:

Result obtained for $G_{i,j}(\underline{q})$:

$$\frac{\partial^2 Q_{SYS}(t)}{\partial q_a \partial q_b} = q_c - q_c q_d - q_c q_f + q_c q_d q_f$$

Correct result for $G_{i,j}(\underline{q})$:

c fails and d works and f works

$$P\left(c \cdot \bar{d} \cdot \bar{f}\right) = q_c\left(1 - q_d - q_f + q_d q_f\right) = q_c - q_c q_d - q_c q_f + q_c q_d q_f$$

Notice the two results agree, this is because component a occurs only in minimal cut set C_1 along with component b. Hence none of the terms in the system unavailability function involving both a and b represent the separate contributions of the failure of components a and b. However, this is not always the case as will be illustrated by the following examples.

Example 2

$$T=abc+ad+bf$$

$$Q_{sys}(t)=q_aq_bq_c+q_aq_d+q_bq_f-q_aq_bq_cq_d-q_aq_bq_cq_f-q_aq_bq_dq_f+q_aq_bq_cq_dq_f$$

Let C_1=abc, C_2=ad, C_3=bf. Then considering the role of component a as initiator and component b as enabler:

Result obtained for $G_{i,j}(\underline{q})$:

$$\frac{\partial^2 Q_{SYS}(t)}{\partial q_a \partial q_b} = q_c - q_c q_d - q_c q_f - q_d q_f + q_c q_d q_f$$

Correct result for $G_{i,j}(\underline{q})$:

c fails and d works and f works

$$P(c \cdot \bar{d} \cdot \bar{f}) = q_c(1 - q_d - q_f + q_d q_f) = q_c - q_c q_d - q_c q_f + q_c q_d q_f$$

Notice the additional term $-q_d q_f$ appears in the obtained result for $G_{i,j}(\underline{q})$. This additional term is the result of the probability of the combined failure of C_2 and C_3 which represents the separate contribution of the failure of components a and b to system failure. Since, if C_2 and C_3 are both failed, then both a and b have failed, however, because they are not contained together in either C_2 or C_3, they are not acting in an initiator / enabler sense.

Example 3

$$T = ab + ac + bd + ce$$

$$Q_{sys}(t) = q_a q_b + q_a q_c + q_b q_d + q_c q_e - q_a q_b q_c - q_a q_b q_d - q_a q_c q_e - q_b q_c q_d q_e$$

Let C_1=ab, C_2=ac, C_3=bd, C_4=ce. Then considering the role of component a as initiator and component b as enabler:

Result obtained for $G_{i,j}(\underline{q})$:

$$\frac{\partial^2 Q_{SYS}(t)}{\partial q_a \partial q_b} = 1 - q_c - q_d$$

Correct result for $G_{i,j}(\underline{q})$:

c works and d works

$$P(\bar{c} \cdot \bar{d}) = (1 - q_c)(1 - q_d) = 1 - q_c - q_d + q_c q_d$$

The obtained result is missing the term $+q_c q_d$. This term represents the contribution of the failure of minimal cut sets C_1, C_2 and C_3. However this contribution is eliminated by the contribution of the failure of minimal cut sets C_2 and C_3. If C_2 and C_3 have both failed then although both a and b are failed, they are not acting in an initiator / enabler sense. Whereas if C_1, C_2 and C_3 have all failed a and b are failed and since they are both contained in C_1 they are acting in an initiator / enabler sense.

The last two examples illustrate the need to apply a correction term to $G_{i,j}(\underline{q})$ in order to eliminate the separate contributions of components i and j.

The correction term to be applied to $G_{i,j}(\underline{q})$ can be derived to take into account the separate effects of components i and j on the system state. It can be defined as the probability that the system is in a critical state for component i and j, such that the failure of either i or j alone would be sufficient to cause the system to fail. In order to calculate the correction term a modified unavailability function, $Q_{M_{i,j}}(t)$ is required, this considers all minimal cut sets of the system apart from those containing both i and j.

$$Q_{M_{i,j}}(t) = \bigcup_{\substack{k=1 \\ i \cap j \notin C_K}}^{n_p} P(C_K) \tag{6.10}$$

The probability that components i and j are critical to the system state such that the failure of either i or j alone would be sufficient to cause the system to fail is calculated as follows.

$$G_{M_{i,j}}(\underline{q}) = \frac{\partial^2 Q_{M_{i,j}}(t)}{\partial q_i \partial q_j} \tag{6.11}$$

Thus the enabler importance of component i when component j causes system failure is given in equation (6.12):

$$I_{E_{i,j}} = \frac{\int_0^t \left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q})\right] q_i(u) w_j(u) du}{W_{SYS}(0,t)} \tag{6.12}$$

The total enabler importance of component i is the sum of the enabler importance of i when component j initiates system failure for, $j = 1,.....,n \quad j \neq i$ and is given in equation (6.13).

$$I_{E_{i,j}} = \frac{\sum_{\substack{j=1 \\ j \neq i}}^{n_{Fj}} \int_0^t \left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q})\right] q_i(u) w_j(u) du}{W_{SYS}(0,t)} \tag{6.13}$$

6.5.2 Measures for Assessing Cut Set Importance

The measures introduced above are suitable for analysing component importance. It can also be useful to rank cut set importance in order to identify the cut set most likely to contribute to system failure. Two measures for analysing cut set importance will be considered in the following sections.

6.3.2.1 Fussell-Vesely's Measure of Cut Set Importance

In 1974 Fussell and Vesely developed a measure of importance to assess cut set importance [20]. The measure is defined as the probability that a cut set C_i contributes to system failure:

$$I_{FV}(C_i) = \frac{P(C_i)}{Q_{SYS}(t)} \tag{6.14}$$

6.3.2.2 Barlow and Proschan's Measure of Cut Set Importance

In 1975 Barlow and Proschan developed a measure of cut set importance, which they defined as the probability that the failure of cut set C_i^n coincides with the failure of the system failure, i.e. that cut set C_i^n actually causes system failure. The following expression was developed for this measure:

$$I_{B-P}\left(C_i^n\right) = \sum_{i \in C} \int_0^t \psi\left(1_i, 0^{C_i^n - \{i\}}, \bar{q}\right) \prod_{j \in C_i^n - \{i\}} q_j w_i dt \tag{6.15}$$

Where:

$\psi\left(1_i, 0^{C_i^n - \{i\}}, \bar{q}\right)$ represents the probability that component i is critical at time t.

$\prod_{j \in C_i^n - \{i\}} q_j w_i dt$ represents the probability that component i fails at time t and the remaining components in cut set C_i^n have failed by time t.

C_i^n represents cut set i of order n.

Barlow and Proschan suggested that the product of these two probabilities gives the probability that component i causes system failure. Thus by summing over

$i \in C_i^n$ (which corresponds to all the mutually exclusive ways that C_i^n can fail) the probability that cut set C_i^n causes system failure is obtained. However, this measure does not always give the required result, to illustrate this consider the following example.

$$T=ab+ac$$

$$\varphi(\underline{x}) = x_a(x_b + x_c - x_b x_c)$$

Consider the first cut set, ab, Barlow and Proschan's measure considers each component in the cut set separately:

$$\begin{aligned} I_{B-P}(ab) &= \int_0^t \psi(1_a, 0_b, \bar{q}) q_b w_a dt + \int_0^t \psi(1_b, 0_a, \bar{q}) q_a w_b dt \\ &= \int_0^t \bar{q}_c q_b w_a dt + \int_0^t 0 \cdot q_a w_b dt \end{aligned} \tag{6.16}$$

Now the correct result will be calculated. Each component will be dealt with separately.

Component a: The cut set ab is critical for component a if, b is failed and c works thus the probability that the failure of component a causes cut set ab and the system to fail is given below:

$$P(b \cdot \bar{c}) \cdot w_a = q_b \bar{q}_c w_a$$

Component b: The cut set ab is critical for component b if a is failed and c works, similarly the probability that the failure of component b causes the cut set ab and the system to fail is given below:

$$P(a \cdot \bar{c}) \cdot w_b = q_a \bar{q}_c w_b$$

Notice that the expression obtained for the probability that component b causes system failure by Barlow and Proschan's measure does not give the correct probability.

This is because the term does not give the probability that component i is critical unless all the cut sets in the system are independent. An explanation of how the required probability can be obtained is given below.

The probability that cut set C_i^n causes system failure can be obtained by summing the probability that the failure of each component in the cut set coincides with the failure of cut set C_i^n and the failure of the system.

Each component in the cut set C_i^n could cause the failure of C_i^n and the system failure by failing in the interval [t,t+dt), provided that all the other components in C_i^n have failed prior to time t, and no other cut sets exist at time t. Thus by summing the contribution for each component in C_i^n the probability that cut set C_i^n causes system failure is obtained.

The first stage is to calculate the probability that cut set C_i^n is critical to the system state. The probability that the components contained in cut set C_i^n are critical to the system failure, such that the failure of all n components in C_i^n would cause the failure of C_i^n and the system is expressed as follows:

$$G_{C_i^n}(\underline{q}) = \left. \frac{\partial^n Q_{SYS}(t)}{\partial q_i{}^n} \right|_{i \in C_i^n} \tag{6.17}$$

However, this does not always give the required probability exactly. This is because some of the terms in $Q_{SYS}(t)$, can represent the separate contirubtions of the components contained within the cut set C_i^n. This will be illustrated by means of a worked example.

$$T=abc+acd+be$$

$$Q_{SYS}(t)=q_aq_bq_c+q_aq_cq_d+q_bq_e-q_aq_bq_cq_d-q_aq_bq_cq_e-q_aq_bq_cq_dq_e+ q_aq_bq_cq_dq_e$$

Suppose the importance of cut set abc is being derived. Then, cut set abc is only critical to the system state if the other two cut sets acd and be do not already exist. Thus abc is critical provided component d works and component e works. The expected probability that cut set abc is critical to the system is given below:

$$P(\text{cut set abc critical})=p_dp_e=1-q_d-q_e+q_dq_e$$

The probability obtained from equation (6.17) is given below:

$$G_{\{abc\}}(\underline{q}) = \frac{\partial^3 Q_{SYS}(t)}{\partial q_a \partial q_b \partial q_c} = 1 - q_d - q_e - q_dq_e + q_dq_e = 1 - q_d - q_e$$

Notice that the expression obtained from equation (6.17) is missing the term $+q_d q_e$, this is because it is eliminated by the redundant combinations of the failure of cut sets 2 and 3. If both of these cut sets have failed, then components a, b and c are in a failed state and thus recorded as part of the solution for $G_{\{abc\}}(\underline{q})$. However, it is only those combinations involving cut set abc that are of interest.

To eliminate these redundant combinations a correction term must be applied to $G_{\{C_i^n\}}(\underline{q})$. Firstly a modified system unavailability function is formed for cut set C_i^n, this function is denoted by, $Q_{M_{C_i^n}}$:

$$Q_{M_{C_i^n}} = \bigcup_{\substack{j=1 \\ j\neq i}}^{n_c} P(C_j) \tag{6.18}$$

Then this function is differentiated with respect to all components in cut set C_i^n to eliminate any redundant combinations included in $G_{\{C_i^n\}}(\underline{q})$.

Thus for the above example the following modified system unavailability function is obtained for the cut set abc:

$$Q_{M_{\{abc\}}} = q_a q_c q_d + q_b q_e - q_a q_b q_c q_d q_e$$

$$G_{M\{abc\}}(\underline{q}) = \frac{\partial^3 Q_{M_{\{abc\}}}(t)}{\partial q_a \partial q_b \partial q_c} = -q_d q_e$$

Now subtracting the result obtained for $G_{M\{abc\}}(\underline{q})$ from the result obtained for $G_{\{abc\}}(\underline{q})$ the correct probability is obtained:

$$G_{\{abc\}}(\underline{q}) - G_{M_{\{abc\}}}(\underline{q}) = 1 - q_d - q_e + q_d q_e$$

Once the probability that cut set C_i^n is critical has been obtained it is multiplied by the probability that cut set component j=1,..,n , $j \neq i$ contained in C_i^n are failed at time t and component i fails in the interval [t,t+dt). This is summed over $i \in C_i^n$ giving the probability that cut set C_i^n causes system failure.

$$I_{B-P}(C_i^n) = \sum_{i \in C_i^n} \int_0^\infty \left[G_{C_i^n} - G_{M_{C_i^n}} \right] \prod_{j \in C_i^n - \{i\}} q_j(t) w_i(t) dt \tag{6.19}$$

6.4 Methods for Calculating Measures of Importance

Two methods for calculating the various measures of importance outlined above will be considered in sections 6.4.1 and 6.4.2. The first method employs conventional FTA techniques and the second is concerned with calculating the measures directly from the SFBDD.

6.4.1 Fault Tree Analysis Technique

Measures of importance can be calculated during quantitative Fault Tree Analysis, the calculation procedures rely on knowledge of the minimal cut sets of the fault tree and tend to involve evaluating lengthy series expansions. To illustrate the Fault Tree Analysis techniques for calculating the seven measures of importance introduced above consider the fault tree diagram in figure 6.2.

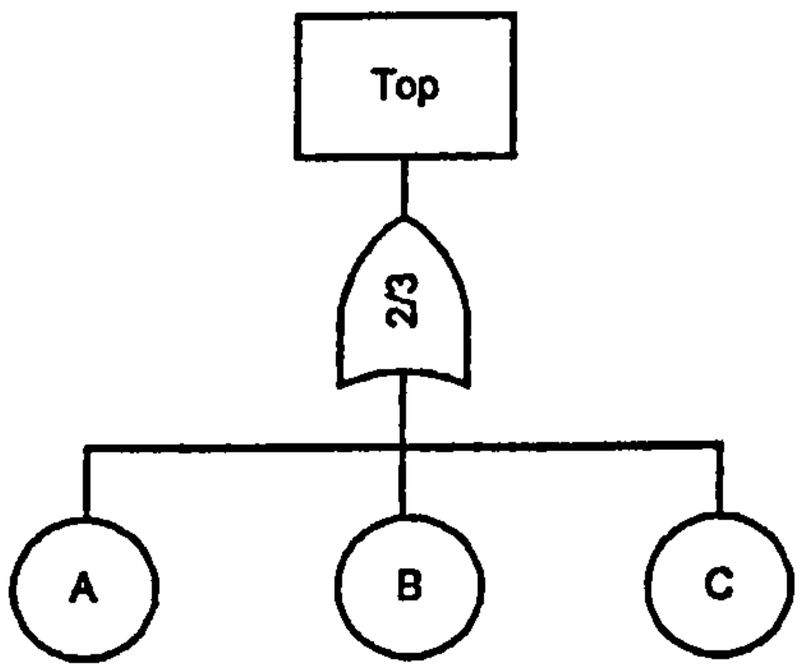


Figure 6.2: Fault Tree Diagram

This fault tree consists of a 2-out-of-three-3 vote gate and has three minimal cut sets:

$$C_1=\{AB\},\ C_2=\{AC\},\ C_3=\{BC\}$$

The first stage is to obtain an expression for the system unavailability, using the inclusion-exclusion expansion given in equation 2.14:

$$Q_{SYS}(t)=q_Aq_B+q_Aq_C+q_Bq_C-2q_Aq_Bq_C$$

From equation (6.3), an expression for Birnbaum's measure of component reliability importance can be obtained for each component as follows:

$$G_A(\underline{q})=\frac{\partial Q_{SYS}}{\partial q_A}=q_B+q_C-2q_Bq_C$$

$$G_B(\underline{q}) = \frac{\partial Q_{SYS}}{\partial q_B} = q_A + q_C - 2q_A q_C$$

$$G_C(\underline{q}) = \frac{\partial Q_{SYS}}{\partial q_C} = q_A + q_B - 2q_A q_B$$

Now expressions for the system unconditional failure intensity and thus the expected number of system failures in a given interval can be obtained:

$$w_{SYS}(t) = \sum_{i=1}^{n} G_i(\underline{q}) w_i$$
$$= (q_B + q_C - 2q_B q_C) w_A + (q_A + q_C - 2q_A q_C) w_B$$
$$+ (q_A + q_B - 2q_A q_B) w_C$$

$$W(0,t) = \int_0^t w_{SYS}(u)du$$

From equation (6.4) and Birnbaum's measure of component reliability importance expressions for the Criticality measure can be obtained for each component:

$$I_{C_A} = \frac{G_A(\underline{q}) q_A}{Q_{SYS}} = \frac{(q_B + q_C - 2q_B q_C) q_A}{Q_{SYS}}$$

$$I_{C_B} = \frac{G_B(\underline{q}) q_B}{Q_{SYS}} = \frac{(q_A + q_C - 2q_A q_C) q_B}{Q_{SYS}}$$

$$I_{C_C} = \frac{G_C(\underline{q}) q_C}{Q_{SYS}} = \frac{(q_A + q_B - 2q_A q_B) q_C}{Q_{SYS}}$$

Fussell-Vesely's measure of importance is calculated for component i by considering only those minimal cut sets containing component i. From equation (6.5) and knowledge of the three minimal cut sets, expressions for Fussell-Vesely's measure of component importance are obtained as follows:

$$I_{FV_A} = \frac{P\left[\bigcup_{\substack{k=1 \\ A \in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{q_A q_B + q_A q_C - q_A q_B q_C}{Q_{SYS}}$$

$$I_{FV_B} = \frac{P\left[\bigcup_{\substack{k=1 \\ B \in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{q_A q_B + q_B q_C - q_A q_B q_C}{Q_{SYS}}$$

$$I_{FV_C} = \frac{P\left[\bigcup_{\substack{k=1 \\ C \in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{q_A q_C + q_B q_C - q_A q_B q_C}{Q_{SYS}}$$

Barlow and Proschan's measure of initiator importance is calculated from equation (6.6), and the results obtained for Birnbaum's measure:

$$I_{IN_A} = \frac{\int_0^t G_A(\underline{q}) w_A dt}{W(0,t)} = \frac{\int_0^t (q_B + q_C - 2q_B q_C) w_A dt}{W(0,t)}$$

$$I_{IN_B} = \frac{\int_0^t G_B(\underline{q}) w_B dt}{W(0,t)} = \frac{\int_0^t (q_A + q_C - 2q_A q_C) w_B dt}{W(0,t)}$$

$$I_{IN_C} = \frac{\int_0^t G_C(\underline{q}) w_C dt}{W(0,t)} = \frac{\int_0^t (q_A + q_B - 2q_A q_B) w_C dt}{W(0,t)}$$

The procedure for calculating the component enabler importance is quite involved and will only be illustrated for component A.

$$I_{E_{i,j}} = \frac{\int_0^t \left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q})\right] q_i(t) w_j(u) du}{W_{SYS}(0,t)}$$

Firstly the contribution of A as an enabler when B is the initiator will be calculated. From equation (6.9):

$$G_{A,B}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_A \partial q_B} = \frac{\partial^2}{\partial q_A \partial q_B}\left[q_A q_B + q_A q_C + q_B q_C - 2q_A q_B q_C\right] = 1 - 2q_C$$

The modified system unavailability function for components A and B is obtained as follows:

$$Q_{M_{A,B}}(t) = \bigcup_{\substack{k=1 \\ A\cap B \in C_K}}^{3} P(C_K) = q_A q_C + q_B q_C - q_A q_B q_C$$

Hence:

$$G_{M_{A,B}}(\underline{q}) = \frac{\partial^2 Q_{M_{A,B}}(t)}{\partial q_A \partial q_B} = \frac{\partial^2}{\partial q_A \partial q_B}[q_A q_C + q_B q_C - q_A q_B q_C] = -q_C$$

Now evaluating the contribution of A as the enabler when component C is the initiator.

$$G_{A,C}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_A \partial q_C} = \frac{\partial^2}{\partial q_A \partial q_C}[q_A q_B + q_A q_C + q_B q_C - 2q_A q_B q_C] = 1 - 2q_B$$

The modified system unavailability function for components A and C is given below:

$$Q_{M_{A,C}}(t) = \bigcup_{\substack{k=1 \\ A\cap C \in C_K}}^{3} P(C_K) = q_A q_B + q_B q_C - q_A q_B q_C$$

Thus:

$$G_{M_{A,C}}(\underline{q}) = \frac{\partial^2 Q_{M_{A,C}}(t)}{\partial q_A \partial q_C} = \frac{\partial^2}{\partial q_A \partial q_C}[q_A q_B + q_B q_C - q_A q_B q_C] = -q_B$$

From equation (6.13) the following expression is obtained for the enabler importance of component A.

$$I_{E_A} = \frac{\int_0^t (1-q_C) q_A w_B + (1-q_B) q_A w_C}{W(0,t)}$$

Expressions for the enabler importance of components B and C are obtained in the same way:

$$I_{E_B} = \frac{\int_0^t (1-q_C) q_B w_A + (1-q_A) q_B w_C}{W(0,t)}$$

$$I_{E_C} = \frac{\int_0^t (1-q_B) q_C w_A + (1-q_A) q_C w_B}{W(0,t)}$$

Next expressions for the Fussell-Vesely measure of cut set importance for each minimal cut set $\{AB\},\{AC\},\{BC\}$ are obtained from equation (6.14):

$$I_{FV}(C_1)=\frac{q_A q_B}{Q_{SYS}(t)}$$

$$I_{FV}(C_2)=\frac{q_A q_C}{Q_{SYS}(t)}$$

$$I_{FV}(C_3)=\frac{q_B q_C}{Q_{SYS}(t)}$$

Finally expressions for the modified Barlow and Proshcan measure of cut set importance is obtained for each minimal cut set from equation (6.19). The process will be outlined for cut set C_1=AB:

Firstly an expression for $G_{\{AB\}}(\underline{q})$:

$$G_{\{AB\}}(\underline{q})=\frac{\partial^2 Q_{SYS}(t)}{\partial q_A \partial q_B}=1-2q_C$$

Then an expression for the modified unavailability function for C_1 is obtained, from which an expression for $G_{M_{\{AB\}}}(\underline{q})$ is obtained:

$$Q_{M_{\{AB\}}}(t)=q_A q_C+q_B q_C-q_A q_B q_C$$

$$G_{M_{\{AB\}}}(\underline{q})=\frac{\partial^2 Q_{M_{\{ab\}}}(t)}{\partial q_A \partial q_B}=-q_C$$

Finally from equation (6.19) the following expression is obtained for the importance of cut set C_1^2:

$$I_{B-P}(C_1^2)=\int_0^t (1-q_C)q_B w_A du+\int_0^t (1-q_C)q_A w_B du$$

Repeating this process the following results are obtained for cut sets C_2^2 and C_3^2:

$$I_{B-P}(C_2^2)=\int_0^t (1-q_B)q_C w_A du+\int_0^t (1-q_B)q_A w_C du$$

$$I_{B-P}(C_3^2)=\int_0^t (1-q_A)q_C w_B du+\int_0^t (1-q_A)q_B w_C du$$

6.4.2 The Binary Decision Diagram Method

For coherent fault trees it is possible to use the SFBDD to exactly calculate all of the measures of importance introduced above except the modified measure of cut set frequency importance which must be approximated. The BDD method enables exact and efficient calculation of the measures of importance eliminating firstly the intermediate stage of identifying the minimal cut sets and secondly the evaluation of lengthy series expansions. The procedure for calculating Birnbaum's measure developed by Sinnamon and Andrews [13] was introduced briefly in chapter three it will be considered in greater detail below.

6.4.2.1 Calculating Birnbaum's Measure of Importance from the SFBDD

Birnbaum's measure of component reliability importance is expressed as follows:

$$G_i(\underline{q}) = Q_{SYS}(1_i \underline{q}) - Q_{SYS}(0_i, \underline{q}) \tag{6.20}$$

The first term of equation (6.20), $Q_{SYS}(1_i, \underline{q})$, can be calculated from the SFBDD by substituting $q_i = 1$ and calculating the probability of the disjoint paths through the SFBDD, i.e. repeating the system unavailability calculations. $Q_{SYS}(0_i, \underline{q})$ is calculated in the same way except q_i is set to 0. There are three different path types:

(i) Paths including a failed state of component i.

(ii) Paths including a working state of component i.

(iii) Paths not including component i.

The following expressions can be obtained for $Q_{SYS}(1_i, \underline{q})$ and $Q_{SYS}(0_i, \underline{q})$:

$$Q_{SYS}(1_i, \underline{q}) = \sum_{\substack{\text{Paths in} \\ \text{category (i)}}} \left(Pr_{x_i}(\underline{q}) \cdot Po^1_{x_i}(\underline{q}) \right) + Z(\underline{q}) \tag{6.21}$$

$$Q_{SYS}(0_i, \underline{q}) = \sum_{\substack{\text{Paths in} \\ \text{category (ii)}}} \left(Pr_{x_i}(\underline{q}) \cdot Po^0_{x_i}(\underline{q}) \right) + Z(\underline{q}) \tag{6.22}$$

Where:

$Pr_{x_i}(\underline{q})$ is the probability of the path section from the root vertex to node x_i

$Po^1_{x_i}(\underline{q})$ is the probability of the path section from the one branch of node x_i to a terminal 1 node (excluding the probability of x_i)

$Po^0_{x_i}(\underline{q})$ is the probability of the path section from the zero branch of the node x_i to a terminal 1 node (excluding the probability of x_i)

$Z(\underline{q})$ is the probability of the paths from the root node to a terminal 1 node not passing the node for variable x_i

Hence from equations (6.20)-(6.22):

$$G_i(\underline{q}) = \sum_{i=1}^{n} Pr_{x_i}(\underline{q})\left[Po^1_{x_i}(\underline{q}) - Po^0_{x_i}(\underline{q})\right] \qquad (6.23)$$

Re-running the system failure probability calculations is an inefficient means of calculating $G_i(\underline{q})$. The efficiency can be improved by calculating each term in equation (6.23) during the initial pass of the SFBDD to obtain the system unavailability as illustrated in chapter three.

Having calculated the top event probability and Birnbaum's measure of importance for each component it is possible to calculate the unconditional failure intensity and the expected number of system failures in a given interval. Then all the quantities required for calculating the Measure of Component Criticality and Barlow and Proschan's measure of initiator importance are known. The procedure for calculating Birnbaum's measure of importance was illustrated in detail in chapter three.

6.4.2.2 Calculating Fussell-Vesely's Measure of Component Importance from the SFBDD

The calculation procedure for the Fussell-Vesely measure of importance is quite involved since both the minimal BDD and the SFBDD must be used during the procedure. Firstly the minimal BDD is searched to identify the number of times each variable is encountered in a minimal cut set. If the basic event occurs in one minimal cut set Fussell-Vesely's measure of component importance is calculated as follows:

$$I_{FV_i} = \frac{P(C_i)}{Q_{SYS}(t)} \qquad (6.24)$$

Otherwise equation (6.25) and the SFBDD are used:

$$I_{FV_i} = \frac{\sum_{x_i \text{ nodes}} q_{x_i} Pr(x_i) po^1(x_i)}{Q_{SYS}(t)} \qquad (6.25)$$

6.4.2.3 Calculating the Enabler Measure of Importance from the SFBDD

The modified measure of enabler importance can be calculated exactly from the SFBDD, $G_{i,j}(\underline{q})$ can be calculated by extending the calculation procedure for $G_i(\underline{q})$ developed by Andrews and Sinnamon, which was considered in section 6.4.2.1. An expression for $G_{i,j}(\underline{q})$ is given in equation (6.26).

$$G_{i,j}(\underline{q}) = Q_{SYS}(1_i,1_j,\underline{q}(t)) - Q_{SYS}(1_i,0_j,\underline{q}(t)) - Q_{SYS}(0_i,1_j,\underline{q}(t)) + Q_{SYS}(0_i,0_j,\underline{q}(t)) \quad (6.26)$$

$Q_{SYS}(1_i,1_j,\underline{q}(t))$ can be calculated by substituting $q_i = 1$ and $q_j = 1$ and re-running the system failure probability calculations. Similarly by substituting $q_i = 1$ and $q_j = 0$, and re-running the system failure calculations $Q_{SYS}(1_i,0_j,\underline{q}(t))$ can be calculated. Repeating this process it is possible to calculate all of the terms in equation (6.26).

However, this does not provide a very efficient means of calculating $G_{i,j}(\underline{q})$. It is possible to calculate $G_{i,j}(\underline{q})$ through just one pass of the SFBDD. The procedure for calculating this probability is derived below.

There are nine paths types through the SFBDD for components I and j:

(i) Paths passing through the one branch of a node x_i and the one branch of a node x_j.

(ii) Paths passing through the one branch of a node x_i and the zero branch of a node x_j.

(iii) Paths passing through the zero branch of a node x_i and the one branch of a node x_j.

(iv) Paths passing through the zero branch of a node x_i and the zero branch of a node x_j.

(v) Paths only passing through the one branch of a node x_i.

(vi) Paths only passing through the zero branch of a node x_i.

(vii) Paths only passing through the one branch of a node x_j.

(viii) Paths only passing through the zero branch of a node x_j.

(ix) Paths not passing through a node x_i or x_j.

To calculate $Q_{SYS}(1_i,1_j,\underline{q}(t))$ q_i and q_j are set to one then the system unavailability function is evaluated. Thus to calculate this probability from the SFBDD, q_i and q_j are set to 1. Since $q_i=q_j=1$, $1-q_i=1-q_j=0$, thus any paths passing through the zero branch of a node xi or xj have a probability of zero. Hence, only those paths passing through the one branch of a node x_i and x_j, or just x_i, or x_j or neither need to be evaluated to obtain an expression for $Q_{SYS}(1_i,1_j,\underline{q}(t))$:

$$Q_{SYS}(1_i,1_j,\underline{q}(t)) = P(\text{paths type (i)}) - P(\text{paths type (v)}) - P(\text{paths type (vii)}) + P(\text{paths type (ix)})$$

Similar expressions are obtained for the other terms in equation (6.26):

$$Q_{SYS}(1_i,0_j,\underline{q}(t)) = P(\text{paths type (ii)}) - P(\text{paths type (v)}) - P(\text{paths type (viii)}) + P(\text{paths type (ix)})$$

$$Q_{SYS}(0_i,1_j,\underline{q}(t)) = P(\text{paths type (iii)}) - P(\text{paths type (vi)}) - P(\text{paths type (vii)}) + P(\text{paths type (ix)})$$

$$Q_{SYS}(0_i,0_j,\underline{q}(t)) = P(\text{paths type (iv)}) - P(\text{paths type (vi)}) - P(\text{paths type (viii)}) + P(\text{paths type (ix)})$$

From equation (6.26) the following expression is obtained for $G_{i,j}(\underline{q})$:

$$G_{i,j}(\underline{q}) = P(\text{paths type (i)}) - P(\text{paths type (ii)}) - P(\text{paths type (iii)}) + P(\text{paths type (iv)}) \quad (6.27)$$

From this it is clear that when calculating $G_{i,j}(\underline{q})$ from the SFBDD, it is only those paths types that pass through nodes representing both component i and component j that are of interest, i.e. paths types (i)-(iv) from the list given above.

6.4.2.3.1 Calculating the Probability of these Paths from the SFBDD

Supposing that component i is before component j in the order scheme, i<j, then the probability of each of these paths can be calculated from the SFBDD as follows:

Paths of type (i): the probability preceding node x_i, multiplied by the probability from the one branch of node x_i to the node x_j (excluding the probability of x_j), multiplied by

the probability from the one branch of x_j to a terminal 1 node (excluding the probability of x_j).

$$P(\text{paths of type (i)}) = pr_{x_i} \cdot po^1_{x_i-x_j} \cdot po^1_{x_j} \tag{6.28}$$

Where:

pr_{x_i} is the probability preceding node x_i.

$po^1_{x_i-x_j}$ is the probability from the one branch of node x_i to the node x_j, excluding the probability of node x_i.

$po^1_{x_j}$ is the probability from the one branch of node x_j to a terminal 1 node.

Paths of type (ii): the probability preceding node x_i, multiplied by the probability from the one branch of node x_i to x_j (excluding the probability of x_i), multiplied by the probability from the zero branch of x_j to a terminal one node (excluding the probability of x_j).

$$P(\text{paths of type (ii)}) = pr_{x_i} \cdot po^1_{x_i-x_j} \cdot po^0_{x_j} \tag{6.29}$$

Where:

$po^0_{x_j}$ is the probability from the zero branch of node x_j to a terminal one node.

Paths of type (iii): the probability preceding node x_i, multiplied by the probability from the zero branch of node x_i to x_j (excluding the probability of x_i), multiplied by the probability from the one branch of x_j to a terminal 1 node (excluding the probability of x_j).

$$P(\text{paths of type (iii)}) = pr_{x_i} \cdot po^0_{x_i-x_j} \cdot po^1_{x_j} \tag{6.30}$$

Where:

$po^0_{x_i-x_j}$ is the probability from the zero branch of node x_i to the node x_j excluding the probability of x_i.

Paths of type (iv): the probability preceding node x_i, multiplied by the probability from the zero branch of x_i to x_j (excluding the probability of x_i), multiplied by the probability from the zero branch of x_j to a terminal 1 node (excluding the probability of x_j).

$$P(\text{paths of type (iv)}) = pr_{x_i} \cdot po^0_{x_i-x_j} \cdot po^0_{x_j} \tag{6.31}$$

Thus the following expression is obtained for $G_{i,j}(\underline{q})$:

$$G_{i,j}(\underline{q}) = pr_{x_i} \cdot \left[po^1_{x_i - x_j} \cdot po^1_{x_j} - po^1_{x_i - x_j} \cdot po^0_{x_j} - po^0_{x_i - x_j} \cdot po^1_{x_j} + po^0_{x_i - x_j} \cdot po^0_{x_j} \right] \tag{6.32}$$

It is possible to calculate, pr_{x_i}, $po^1_{x_i - x_j}$, $po^0_{x_i - x_j}$, $po^1_{x_j}$, and $po^0_{x_j}$ through just one pass of the SFBDD. Beginning at the root vertex of the SFBDD, a depth first traversal of the tree structure is performed. At each node x_j it is possible to record the probability preceding this node, pr_{x_j}. Furthermore since it is possible track whether on the one or zero branch of all nodes preceding x_j, it is possible to record the probability from either the one or zero branch of all previous nodes x_i to the current node x_j, i.e., $po^1_{x_i - x_j}$ and $po^0_{x_i - x_j}$. Finally the probability from the one and zero branches of the current node x_j to a terminal one node, $po^1_{x_j}$, and $po^0_{x_j}$ can be recorded in the same way that it is recorded for calculating $G_i(\underline{q})$.

To illustrate this calculation procedure consider the SFBDD given in figure 6.3.

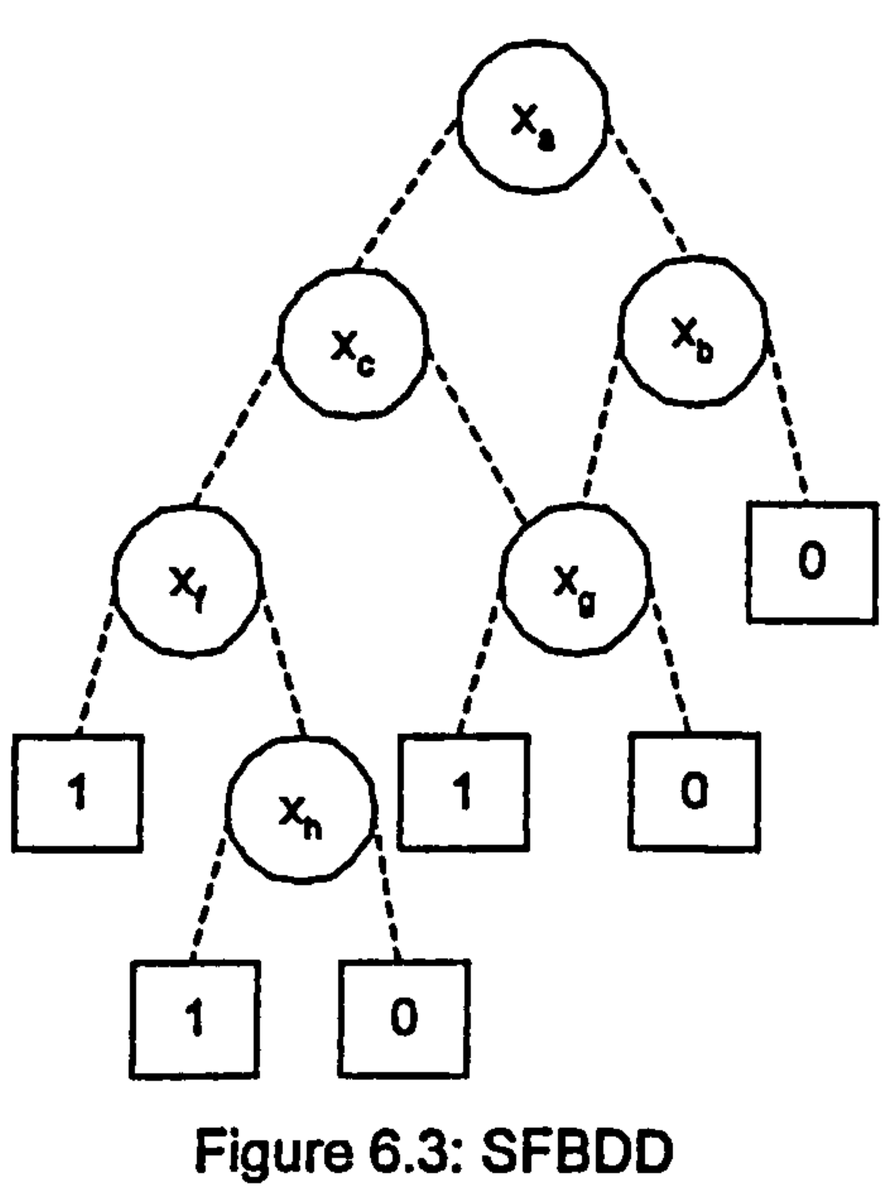


Figure 6.3: SFBDD

Suppose node x_h is being dealt with, at this point three nodes have been passed through:

(i) the one branch of node x_a

(ii) the one branch of node x_c

(iii) the zero branch of node x_f

Given that the probability preceding each node is recorded as the node is encountered, $pr_{x_a}, pr_{x_c}, pr_{x_f}$ and pr_{x_h} are known.

Thus since the current path passes through the one branch of nodes x_a and x_c to x_h, $po^1_{x_a-x_h}$ and $po^1_{x_c-x_h}$ can be calculated as follows.

$$po^1_{x_a-x_h} = \frac{pr_{x_h}}{pr_{x_a} \cdot q_{x_a}}$$

$$po^1_{x_c-x_h} = \frac{pr_{x_h}}{pr_{x_c} \cdot q_{x_c}}$$

Also, since the current path passes through the zero branch of node x_f to node x_h, $po^0_{x_f-x_h}$ can be calculated as follows:

$$po^0_{x_f-x_h} = \frac{pr_{x_h}}{pr_{x_f} \cdot (1-q_{x_f})}$$

To calculate the correction term, $G_{M_{i,j}}(\underline{q})$, a modified system unavailability function $Q_{M_{i,j}}(\underline{q})$ is formed. This modified function is formed by considering only a partial list of the minimal cut sets. All those minimal cut sets involving both i and j are ignored. Since the SFBDD obtained for the system encodes the full list of minimal cut sets it does not encode the modified system unavailability function. Thus to calculate $G_{M_{i,j}}(\underline{q})$ a new SFBDD which only encodes the partial list of minimal cut sets must be computed. This modified BDD encodes the structure function of the modified system and thus both $Q_{M_{i,j}}(\underline{q})$ and $G_{M_{i,j}}(\underline{q})$ can be calculated directly from this tree structure.

Although the structure function method outlined in chapter three can be used to compute the modified SFBDD, this is not an efficient means of computation. Rauzy's ite procedure also outlined in chapter three would enable the modified SFBDD to be calculated efficiently. In order to use this procedure, a fault tree must be developed for the modified system. The fault tree structure will always take the same form. The top gate will be an OR gate with all gate inputs, the number of gate inputs will be equal to the number of minimal cut sets being considered in the modified system. Each gate input to the top gate will be an AND gate, with it inputs being the basic events of each minimal cut set.

Once the fault tree structure has been obtained the **ite** procedure is applied to compute the modified SFBDD. It is then possible to calculate $G_{M_{i,j}}(\underline{q})$ using the method outlined previously for calculating $G_{i,j}(\underline{q})$. This technique for calculating the modified enabler measure of importance will now be illustrated by means of a worked example. Consider the fault tree in figure 6.4:

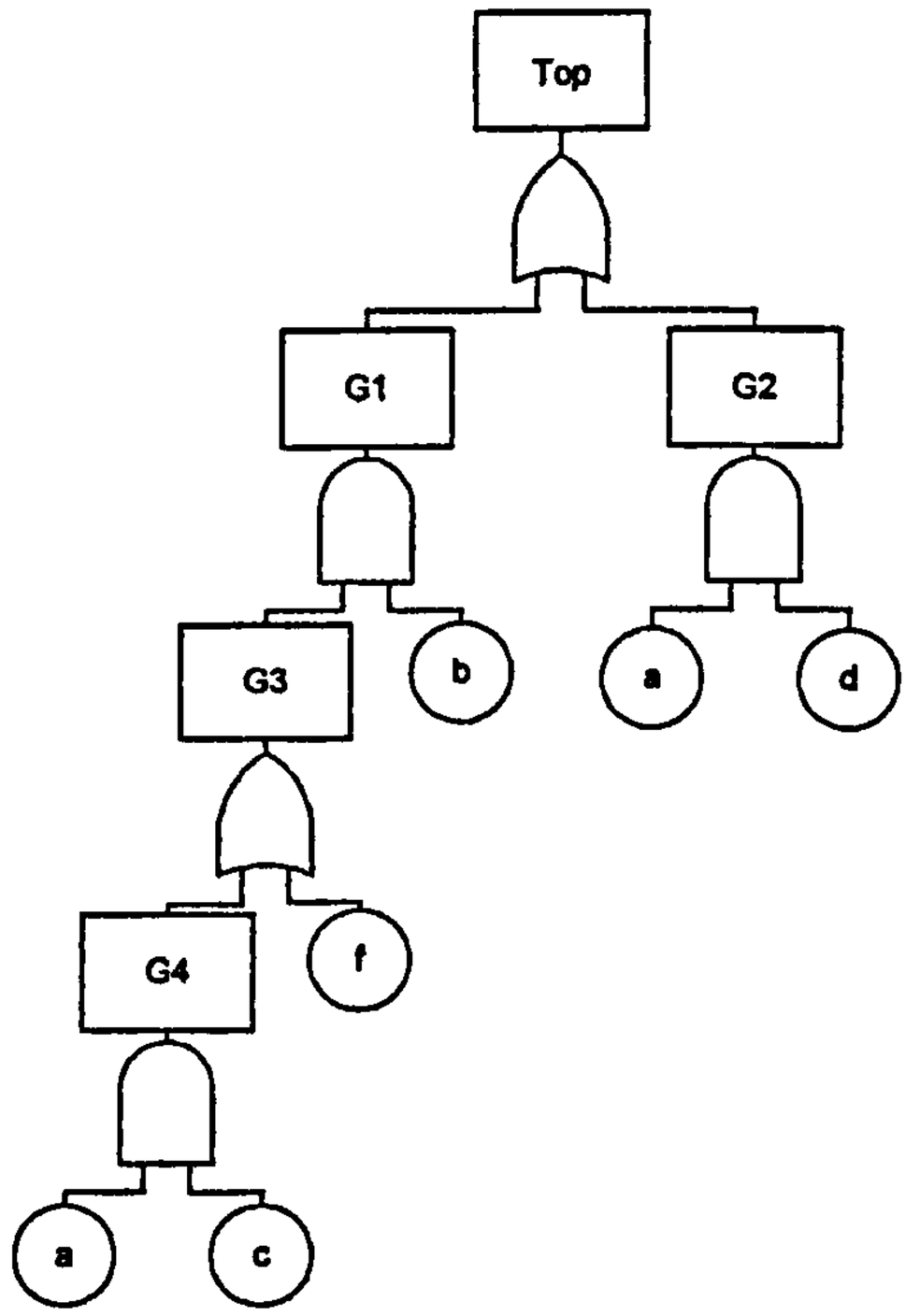


Figure 6.4: Coherent Fault Tree

Assigning the ordering $a < b < c < d < f$ to the basic events in the fault tree the SFBDD shown in figure 6.5 is obtained.

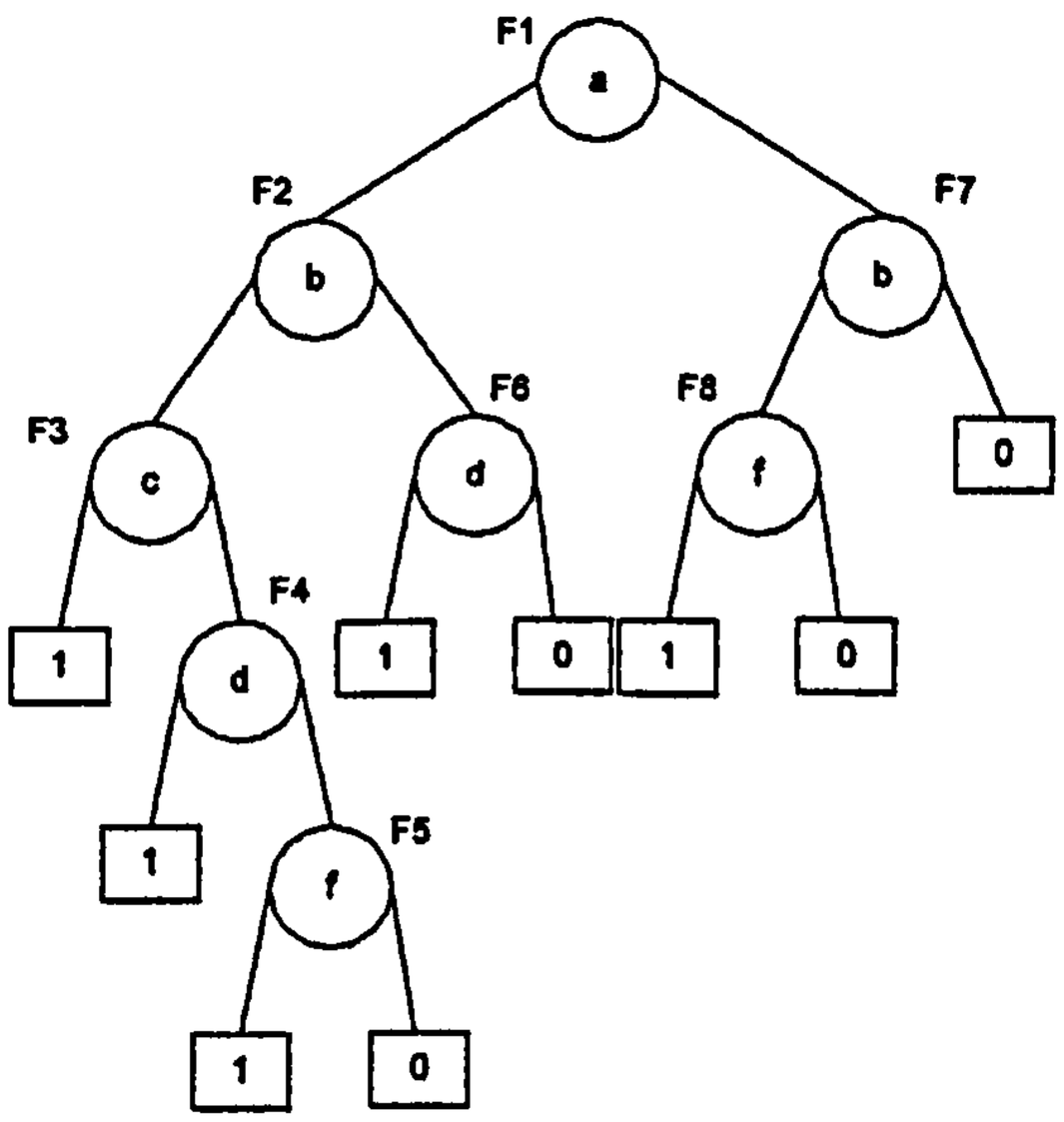


Figure 6.5: SFBDD for the Fault Tree in Figure 6.4

The enabler importance of component a, when component b acts as an initiator will be calculated. Firstly $G_{a,b}(\underline{q})$ will be calculated from the SFBDD in figure 6.5. Table 6.2 Records the results for pr_{x_i}, $po^1_{x_i-x_j}$, $po^0_{x_i-x_j}$, $po^1_{x_i}$, and $po^0_{x_i}$ for each node in the SFBDD shown in figure 6.5.

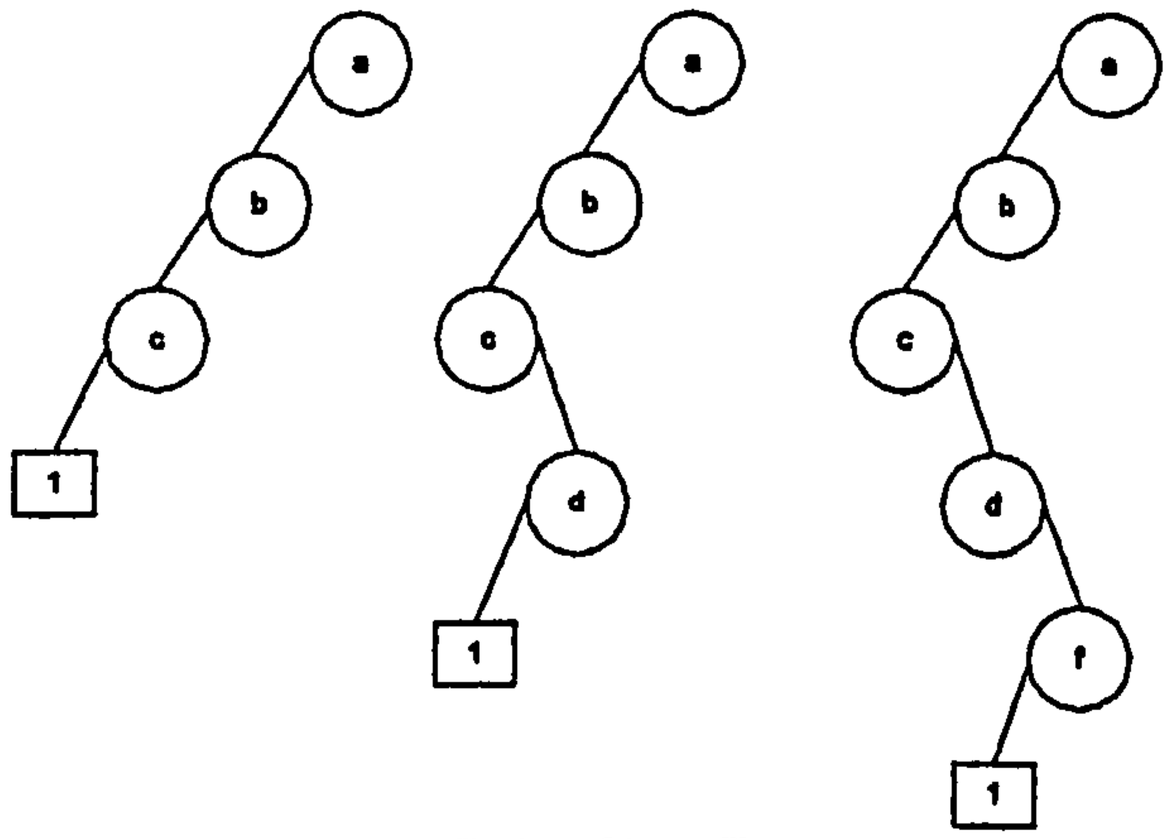
Node	Variable	pr_{x_i}	$po^1_{x_i-x_j}$	$po^0_{x_i-x_j}$	$po^1_{x_i}$	$po^0_{x_i}$
F1	a	1	$po^1_{F1-F2}=1$ $po^1_{F1-F3}=1$ $po^1_{F1-F4}=p_c$ $po^1_{F1-F5}=p_cp_d$ $po^1_{F1-F6}=1$	$po^0_{F1-F7}=1$ $po^0_{F1-F8}=1$	$q_c+p_cq_d$ $+p_cp_dq_f+q_d$	q_f
F2	b	q_a	$po^1_{F2-F3}=1$ $po^1_{F2-F4}=p_c$ $po^1_{F2-F5}=p_cp_d$	$po^0_{F2-F6}=1$	$q_c+p_cq_d+p_cp_dq_f$	q_d
F3	c	q_a		$po^0_{F3-F4}=1$ $po^0_{F3-F5}=p_d$	1	p_dq_f
F4	d	q_ap_c		$po^0_{F4-F5}=1$	1	q_f
F5	f	$q_ap_cp_d$			1	0
F6	d	q_a			1	0
F7	b	p_a	$po^1_{F7-F8}=1$		q_f	0
F8	f	p_a			1	0

Table 6.2: Results for pr_{x_i}, $po^1_{x_i-x_j}$, $po^0_{x_i-x_j}$, $po^1_{x_j}$, and $po^0_{x_j}$ for each Node in the SFBDD in Figure 6.2.

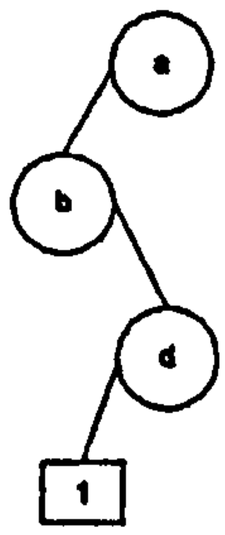
The enabler importance of component a when it is component b that actually causes systems failure is calculated using equation (6.33) and the results given in table 6.2.

$$I_{E_{a,bj}} = \frac{\int_0^t [G_{a,b}(\underline{q}) - G_{M_{a,b}}(\underline{q})] q_a(u) w_b(u) du}{W_{SYS}(0,t)} \qquad (6.33)$$

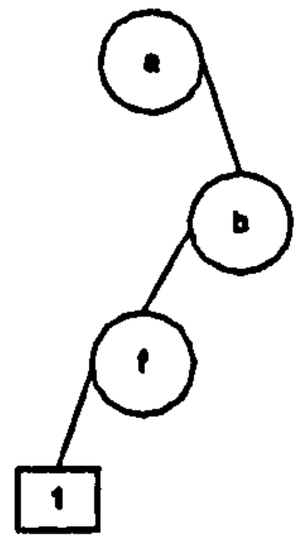
From the SFBDD in figure 6.5 three paths of type (i), one of type (ii) and one of type three (iii) for components a and b can be identified, each paths is shown below:



Paths of type (i)



Path of type (ii)



Path of type (iii)

Figure 6.6: Paths of Type (i), (ii) and (iii) through the SFBDD in Figure 6.5 for Components a and b

From equations (6.29)-(6.31) and figure 6.65, the following probabilities are obtained for the three paths:

$$P(\text{paths of type (i)}) = \sum_{\substack{\text{paths of} \\ \text{type (i)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j}$$

$$= 1 \cdot 1 \cdot q_c + 1 \cdot 1 \cdot p_c q_d + 1 \cdot 1 \cdot p_c p_d q_f$$

$$= q_c + p_c q_d + p_c p_d q_f$$

$$P(\text{paths of type (ii)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{0}_{x_j}$$

$$= 1 \cdot 1 \cdot q_d$$

$$= q_d$$

$$P(\text{paths of type (iii)}) = pr_{x_i} \cdot po^{0}_{x_i - x_j} \cdot po^{1}_{x_j}$$

$$= 1 \cdot 1 \cdot q_f$$

$$= q_f$$

From equation (6.32) the following result is obtained for $G_{a,b}(\underline{q})$:

$$G_{a,b}(\underline{q}) = q_c + p_c q_d + p_c p_d q_f - q_d - q_f$$

To calculate the correction term the minimal cut sets, {ad} and {bf} are considered. The fault tree shown in figure 6.7 is constructed for this system.

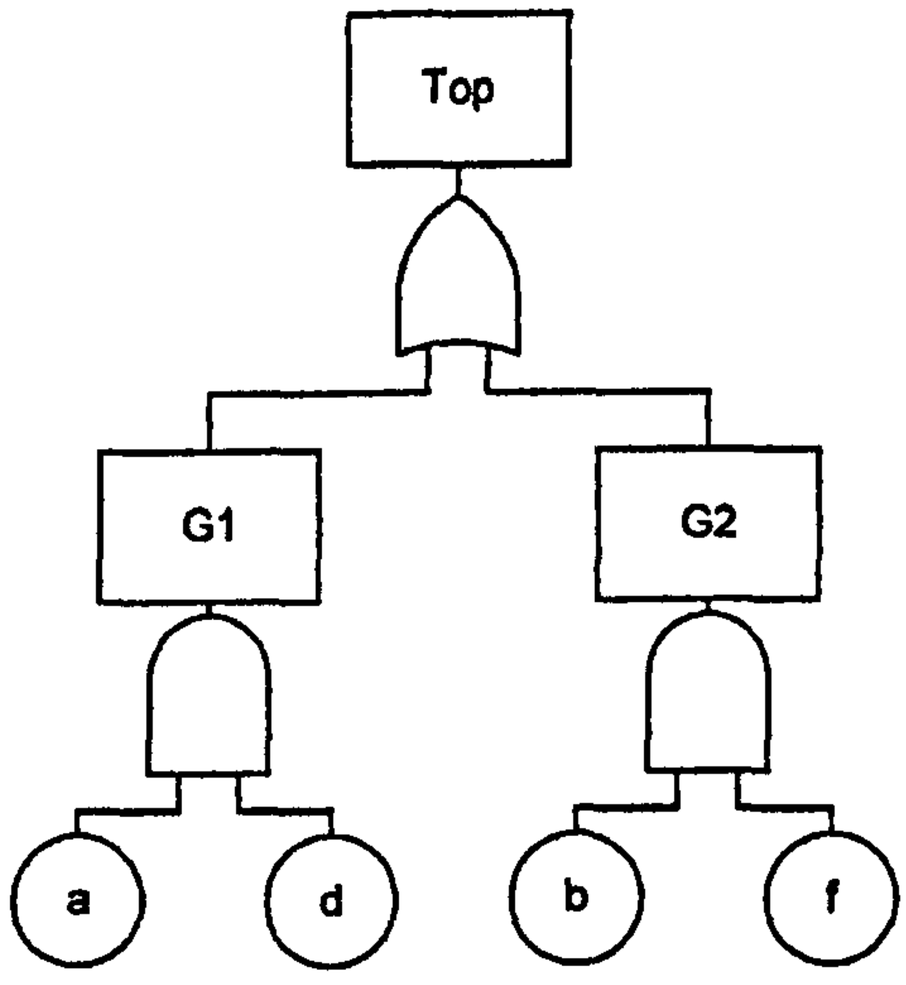


Figure 6.7: Modified Fault Tree Diagram

The basic events in the fault tree are ordered as follows, $a < b < d < f$ and the SFBDD shown in figure 6.8 is obtained:

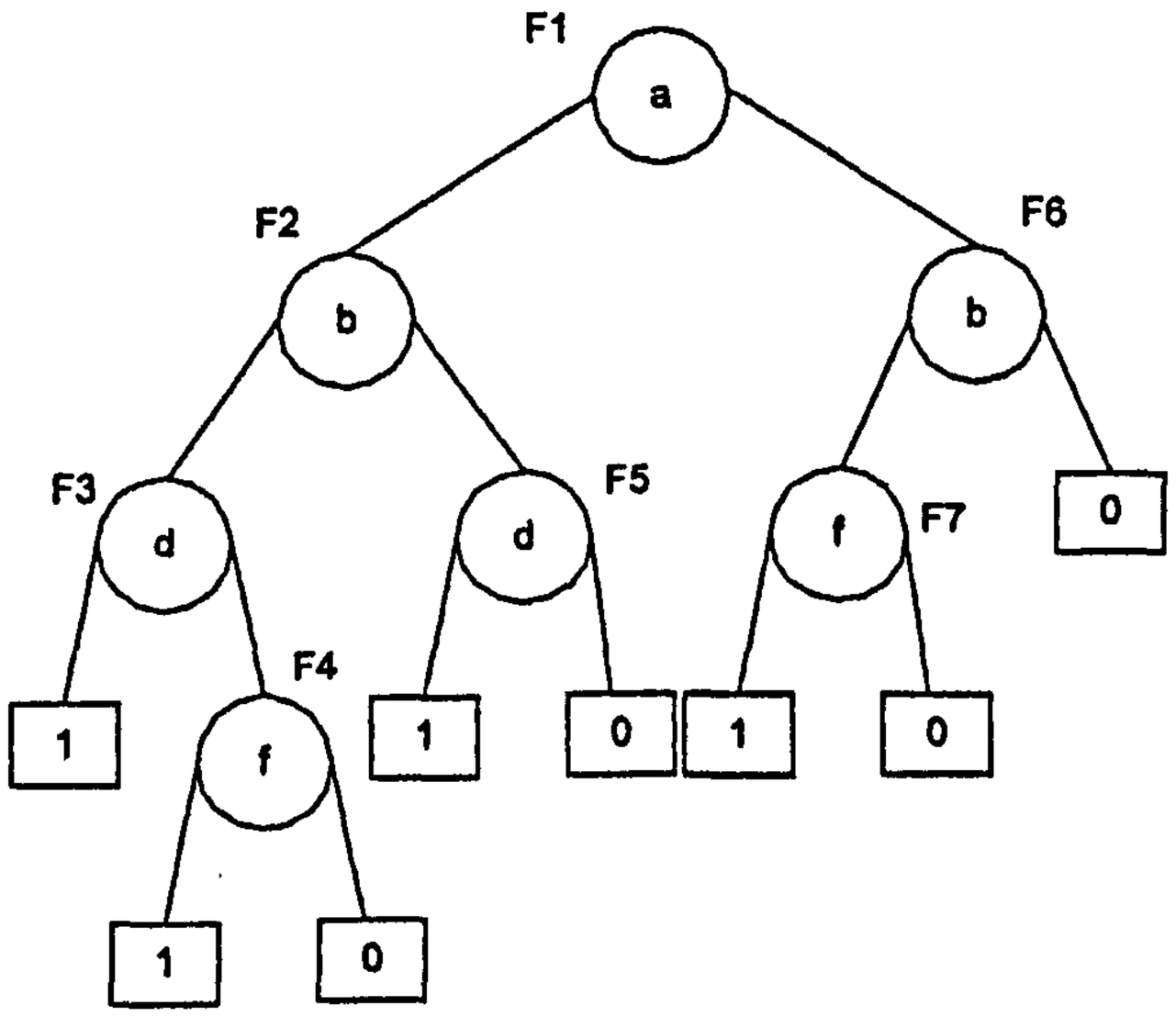


Figure 6.8: The SFBDD for the Fault Tree in Figure 6.7

To calculate $G_{M_{a,b}}(\underline{q})$ pr_{x_i}, $po^1_{x_i-x_j}$, $po^0_{x_i-x_j}$, $po^1_{x_i}$, and $po^0_{x_i}$ for each node in the SFBDD shown in figure 6.8 are recorded in table 6.3.

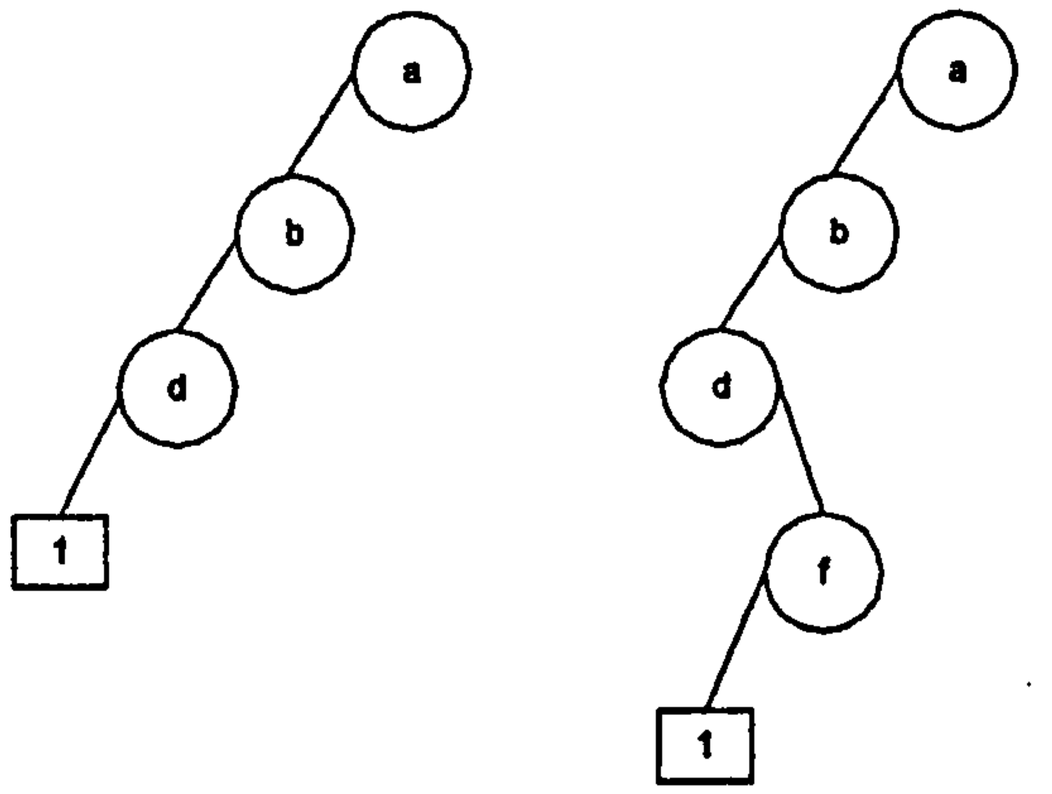
Node	Variable	pr_{x_i}	$po^1_{x_i-x_j}$	$po^0_{x_i-x_j}$	$po^1_{x_i}$	$po^0_{x_i}$
F1	a	1	$po^1_{F1-F2}=1$ $po^1_{F1-F3}=1$ $po^1_{F1-F4}=p_d$ $po^1_{F1-F5}=1$	$po^0_{F1-F6}=1$ $po^0_{F1-F7}=1$	$q_d+p_dq_f$ $+q_d$	q_f
F2	b	q_a	$po^1_{F2-F3}=1$ $po^1_{F2-F4}=p_d$ $po^1_{F2-F5}=1$	$po^0_{F2-F5}=1$	q_d+q_f	q_d
F3	d	1		$po^0_{F3-F4}=1$	1	p_d
F4	f	p_d			1	0
F5	d	1			1	0
F6	b	1	$po^1_{F6-F7}=1$		q_f	0
F7	f	1			1	0

Table 6.3: Results for pr_{x_i}, $po^1_{x_i-x_j}$, $po^0_{x_i-x_j}$, $po^1_{x_j}$, and $po^0_{x_j}$ for each Node in the SFBDD in Figure 6.8.

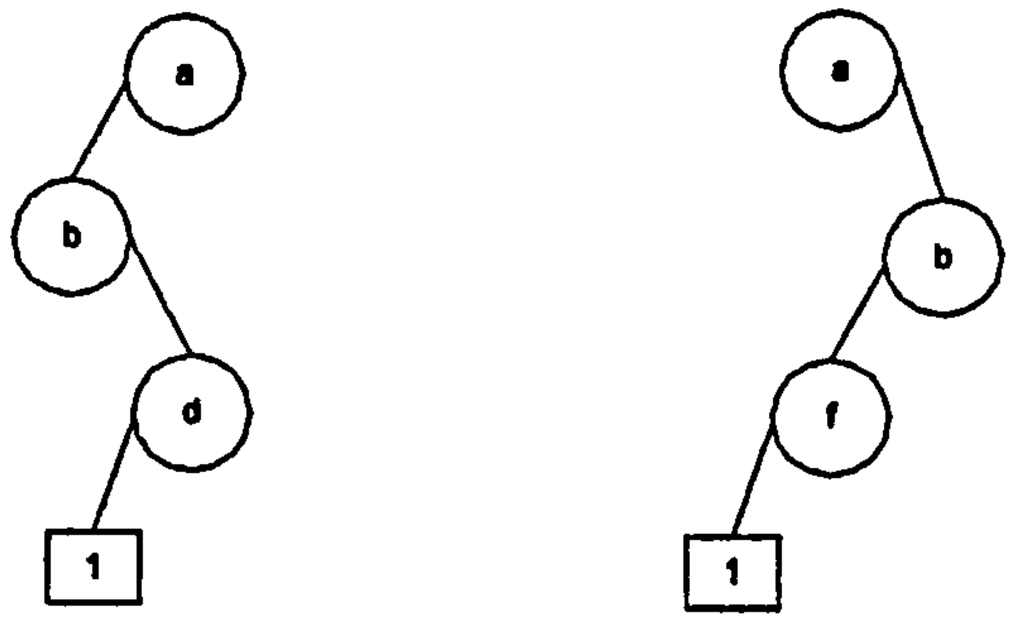
To calculate $G_{M_{i,j}}(\underline{q})$ the paths of type (i), type (ii), type (iii) and type (iv) for components a and b are identified from the modified SFBDD in figure 6.7. Then equation (6.34) is used to calculate $G_{M_{i,j}}(\underline{q})$.

$$G_{M_{i,j}}(\underline{q}) = P(\text{paths of type (i)}) - P(\text{paths of type (ii)}) - P(\text{paths of type (iii)}) + P(\text{paths of type (iv)}) \quad (6.34)$$

From the modified SFBDD in figure 6.8 two paths of type (i), one path of type (ii) and one paths of type (iii) are identified. Each paths is shown below:



Paths of type (i)



Path of type (ii)

Path of type (iii)

Figure 6.9: Path Types through the Modified SFBDD in Figure 6.8 for Components a and b

Thus from equations (6.29)-(6.31) and figure 6.9 the following probabilities are obtained for the three paths:

$$P(\text{paths of type (i)}) = \sum_{\substack{\text{paths of} \\ \text{type (i)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j}$$

$$= 1 \cdot 1 \cdot q_d + 1 \cdot 1 \cdot p_d q_f$$

$$= q_d + p_d q_f$$

$$P(\text{paths of type (ii)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{0}_{x_j}$$

$$= 1 \cdot 1 \cdot q_d$$

$$= q_d$$

$$P(\text{paths of type (iii)}) = pr_{x_i} \cdot po^{0}_{x_i - x_j} \cdot po^{1}_{x_j}$$

$$= 1 \cdot 1 \cdot q_f$$

$$= q_f$$

From equation (6.33) the following result is obtained for $G_{M_{a,b}}(\underline{q})$:

$$G_{M_{a,b}}(\underline{q}) = q_d + p_d q_f - q_d - q_f$$

Thus from equation the following expression is obtained for the enabler importance of component a when component b causes system failure.

$$I_{E_{i,j}} = \frac{\int_0^t \left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q})\right] q_i(u) w_j(u) du}{W_{SYS}(0,t)} = \frac{\int_0^t \left[q_c - q_c q_d - q_c q_f + q_c q_d q_f\right] q_a w_b du}{W_{SYS}(0,t)}$$

This simple example has demonstrated that the calculation procedure for the modified measure of enabler importance is quite involved. Furthermore this procedure must be applied to every pair of components in the fault tree.

6.4.2.4 Calculating Fussell-Vesely's Measure of Cut Set Importance

To calculate this measure knowledge of the minimal cut sets is essential. Thus the minimal BDD is tracked to identify each minimal cut set. The probability of this cut set is calculated and then divided by the value obtained for the system unavailability.

6.4.2.5 Calculating The Measure of Cut Set Frequency Importance

The procedure for calculating this measure exactly from the SFBDD is CPU intensive and not a practical proposition. Thus it will not be considered in detail here. An alternative measure for assessing the importance of a cut set C_i^n in causing system failure is to calculate the expected number of system failures caused by cut set C_i^n and divide this by the total number of expected system failures in a given interval.

$$I(C_i^n) = \frac{\int_0^t \sum_{j=1}^{n} w_j \prod_{\substack{i=1 \\ i \neq j}}^{n} q_i du}{W_{SYS}(0,t)} \qquad (6.35)$$

The expected number of system failures can be calculated directly from this SFBDD and each minimal cut set can be identified from the minimal BDD and subsequently quantified, provided the failure probability and frequency of each system component is known.

6.5 Importance Analysis a Worked Example

Prior to performing analysis it is essential to consider the aims of the analysis so that appropriate measures of importance can be chosen to analyse the importance of the components and the minimal cut sets of the system. It is also critical to interpret the results obtained correctly. Different measures assess different 'types' of importance thus different components could be ranked as 'most important' for different measures. In this section an importance analysis will be performed on a simple system and the results will be interpreted.

Consider again the fault tree shown in figure 6.2. Table 6.4 summarises the conditional failure rate, λ, and the failure probability, q, for each component. From this the unconditional failure intensity of each component has been calculated using the formulae given in equation (6.36). These component parameters are used to quantify the expressions obtained for the five different measures of component importance and the two measures of cut set importance considered in sections 6.4.1 and 6.4.2. The results are given in tables 6.5 and 6.6.

$$w(t) = \lambda(t)[1 - q(t)] \tag{6.36}$$

Component	Failure Rate $\lambda(t)$	Failure Probability $q(t)$	Failure Intensity $w(t)$
A	1.0×10^{-3}	1.0×10^{-2}	9.9×10^{-4}
B	1.0×10^{-4}	2.0×10^{-2}	9.998×10^{-5}
C	5.0×10^{-6}	2.5×10^{-4}	4.99875×10^{-6}

Table 6.4: Summary of Component Parameters

Component / Measure	A	B	C	Component Ranked 1st
Birnbaum	2.03×10^{-2}	1.03×10^{-2}	2.96×10^{-2}	C
Criticality	0.976	0.988	3.57×10^{-2}	B
Fussell-Vesely	0.976	0.988	3.59×10^{-2}	B
Initiator	0.945	4.83×10^{-2}	6.98×10^{-3}	A
Enabler	4.94×10^{-2}	0.938	1.15×10^{-2}	B

Table 6.5: Results for the Various Measures of Component Importance

Minimal Cut Set	Fussell-Vesely's Importance	Barlow & Proshcan's Modified Importance
AB	0.9643	0.18215
AC	1.205×10^{-2}	2.554×10^{-3}
BC	2.41×10^{-2}	1.084×10^{-3}

Table 6.6: Results for Measures of Cut Set Importance

The five measures of component importance were calculated for each component and the highest ranked component is recorded in the far right column of table 6.5. Notice that it is not always the same component that is ranked as "most important" for the different measures, i.e. Birnbaum's measure ranks component C as the most important, whereas component B is ranked highest for the Criticality Measure of importance. To interpret the results it is important to understand the definition of each measure.

From the results for Birnbaum's measure of importance it can be seen that the system is most likely to be in a working yet critical state for component C. Thus it can be concluded that if the system performance is considered inadequate extra resources should be allocated to reduce the existence of the necessary and sufficient conditions that make component C critical to the system state. Thus the availability of component A and B should be improved.

The results from both the Measure of Component Criticality and Fussell-Vesely's measure of component importance confirm this conclusion since although component B ranks highest for these measures. There is little difference between the measures for components A and B. It is also be seen from these measures that there is little need at present to improve the availability of component C since it is extremely unlikely to be in a failed state when the system is failed.

The initiator measure of importance indicates that it is component A that is most likely to actually **cause** the system to go from a working to a failed state. Component C is least likely to **cause** the system to fail and there is a slim chance that component B will actually **cause** the system to fail. However, by considering the results for the enabler importance it is clear that component B is most likely to enable another component to act as an initiator causing system failure. From this it is clear that the most likely cause of system failure is the existence of minimal cut set AB where B

acts as an enabler and A acts as the initiator. This conclusion is confirmed by the results from Fussell-Vesely's measure of cut set importance and the measure of cut set frequency importance recorded in table 6.6.

Therefore, if resources are to be allocated to improve the system, efforts should be concentrated on either reducing the failure rate of A or the unavailability of B.

6.6 Summary

Importance analysis is an essential part of the quantification process enabling the analyst to identify the weakest areas of the system and then make informed decisions about how the system can be improved. Numerous measures of importance have been developed and they can be categorised as either deterministic or probabilistic. Probabilistic measures tend to provide more valuable information because they take account of the component failure probabilities.

Probabilistic measures fall into one of two categories, those suitable for assessing system unreliability and those for assessing system unavailability. It is vitally important to understand the definition of each measure of importance and be able to choose suitable measures for analysis. It is also important to interpret the results correctly if any real value is to be gained from the analysis.

The measures of importance introduced in this chapter can be calculated using one of two methods. They can be calculated directly from the fault tree or they can be calculated from the SFBDD. Fault Tree Analysis requires knowledge of all the minimal cut sets if exact results are to be obtained during quantification. The quantification procedures for calculating the measures of importance are also complex and CPU intensive and thus can be inefficient. The BDD method provides a more efficient and exact means of calculating the measures of importance, significantly reducing the need to employ approximations that are often unavoidable when conventional FTA techniques are employed.

Chapter 7: Importance Analysis of Non-coherent Fault Trees

7.1 Introduction

An importance analysis is a valuable part of the quantification process, which helps the analyst to identify the weakest areas of the system and thus make informed decisions about resource allocation for system improvement. This field has received a great deal of attention over the last 30 years and numerous measures of importance have been developed. However, importance analysis of non-coherent fault trees is extremely limited since the majority of the measures that have been developed are strictly for the analysis of coherent fault trees.

If these measures are used to analyse non-coherent fault trees the results are generally inaccurate and misleading. Chapters four and five highlighted that non-coherent fault trees can occur and accurate importance analysis is required. In 1983 Jackson [21] developed extensions for some of the most commonly used measures of importance, however, these extensions are inconsistent.

This chapter will introduce an extension for Birnbaum's measure of component reliability importance and demonstrate how this can be used to extend a number of other measures based on Birnbaum's measure [22]. Other commonly used measures of component and cut set importance will also be extended for use with non-coherent systems [23]. These include, the Fussel-Vesely measure of component importance, the modified measure of enabler importance, Fussell-Vesely's measure of cut set importance and the measure of cut set frequency importance.

7.2 Coherent Approximations

The majority of importance measures that have been developed over the last 30 years have been developed specifically for the analysis of coherent systems and therefore rank component failures. If these measures are used to analyse a non-coherent system, the results are inaccurate and little value is gained from the analysis. To illustrate this consider the non-coherent system with three prime implicant sets:

$$\{a,c\},\{b,\bar{c}\},\{a,b\}$$

The following expression is obtained for the system unavailability function:

$$Q_{SYS}(t) = q_a q_c + q_b(1-q_c) + q_a q_b - q_a q_b q_c - q_a q_b(1-q_c)$$
$$= q_a q_c + q_b - q_b q_c$$

Birnbaum's measure of component reliability importance is calculated for each component as follows:

$$G_i(\underline{q}) = Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q}) \qquad (7.1)$$

Hence:

$$G_a(\underline{q}) = q_c$$
$$G_b(\underline{q}) = 1 - q_c$$
$$G_c(\underline{q}) = q_a - q_b$$

Importance measures assign components a numerical value between 0 and 1, however the result obtained for component c would be negative if, $q_a < q_b$, and thus no conclusions can be drawn about the importance of this component.

Birnbaum's measure calculates the probability that the system is in a critical state for component i at time t. In a non-coherent system, a component can be failure critical, $G_i^F(\underline{q})$ or repair critical, $G_i^R(\underline{q})$, because system failure can be caused not only by component failure but also by component repair.

One way of obtaining Birnbaum's measure of importance for the above example is to consider component criticality by an exhaustive tabular approach. Consider a system with n components: the system state can then be expressed in terms of the component states. It is possible to determine whether a component is critical to system failure given the states of the remaining (n-1) components. There are 2^{n-1} possible states of the other (n-1) components. By identifying the critical situations for component i and summing their probability of occurrence it is possible to calculate the probability that component i is critical to system failure.

Thus for the above example table 7.1 identifies the critical states for each of the three components, table 7.2 records the sum of the critical situations for each event and the probability that each event is critical to system failure. The probability is calculated using the component unavailabilities assigned by Jackson [21], which are given below:

$$q_a = 9.90099 \times 10^{-3}, q_b = 3.84615 \times 10^{-2}, q_c = 1.52534 \times 10^{-2}$$

State of a	State of b	Is c Critical	State of a	State of c	Is b Critical	State of b	State of c	Is a Critical
W	W	No	W	W	Yes (F)	W	W	No
W	F	Yes (R)	W	F	No	W	F	Yes (F)
F	W	Yes (F)	F	W	Yes (F)	F	W	No
F	F	No	F	F	No	F	F	Yes (F)

Table 7.1: Possible and Critical States for the Events

Event	Sum of critical Situations	Expected Result	Ranking
a	$p_b q_c + q_b q_c$	0.152534	2
b	$p_a p_c + q_a p_c$	0.84747	1
c	$q_a p_b$	0.00952	4
$\bar{c}$	$p_a q_b$	0.03808	3

Table 7.2: Expected Results for Component Criticality

These results demonstrate that although Birnbaum's measure calculates the importance of components a and b correctly, it does not obtain the correct result for component c. Consequently, Birnbaum's measure of importance is not suitable for the analysis of non-coherent systems. Furthermore, any other measures based on Birnbaum's measure of importance including, the Measure of Component Criticality, Barlow and Proschan's measure of initiator importance and the modified measure of enabler importance will not be suitable for the analysis of non-coherent systems either.

7.3 Jackson's Extension of Birnbaum's Measure of Component Reliability Importance

In 1983 Jackson [21] developed an extension of Birnbaum's measure and then used this to extend a number of others measures based on Birnbaum's measure. Jackson's proposed extension of Birnbaum's measure is given below.

$$G_i^*(\underline{q}) = \left| Q_{SYS}(1_i, \underline{q}) - Q_{SYS}(0_i, \underline{q}) \right| \qquad (7.2)$$

It is unclear exactly how this measure should be interpreted, equation (7.2) suggests only one calculation per component, however, in a worked example, Jackson actually ranks component failure importance and component repair importance separately. To demonstrate the problems encountered with Jackson's proposed extension, consider again the non-coherent system with three prime implicant sets {ab}, {bc}, {b $\bar{c}$}.

The component unavailabilities assigned by Jackson are given below and the results Jackson obtained for $G_i^*(\underline{q})$ are given in table 7.3 (where the component reliability $p_i = 1 - q_i$):

$$q_a = 9.90099 \times 10^{-3}, q_b = 3.84615 \times 10^{-2}, q_c = 1.52534 \times 10^{-2}$$

Event	Jackson's Results	Ranking
a	5.665×10^{-3}	4
b	8.105×10^{-3}	3
c	9.575×10^{-3}	2
$\bar{c}$	3.839×10^{-2}	1

Table 7.3: The Results Obtained by Jackson

Comparing Jackson's results in table 7.3 to those given in table 7.2 it is clear that not only does Jackson's extension calculate component criticality incorrectly but that it also ranks the components incorrectly. Hence it can be concluded that Jackson's extension is not conceptually equivalent.

7.4 The concept of Component Relevancy/ Irrelevancy

When analysing non-coherent fault trees, both component failed states and component working states can contribute to system failure. A component can be either relevant to the system state or irrelevant to the system state. If a component is relevant to the system state, it can be either failure relevant or repair relevant. Component i is said to be failure relevant if the system is in a critical state such that the failure of component i would cause the system to fail. Similarly component i is said to be repair relevant if the system is in a critical state such that the repair of component i would cause system failure. Finally component i is said to be irrelevant if its state has no bearing on the state of the system.

Expressions for the failure relevance and irrelevance of a component can be obtained from the Boolean expression for the top event. Consider the Boolean expression for the top event Top given in equation (7.3).

$$Top = ab + \bar{a}d + ce + bd \tag{7.3}$$

An expression for the failure relevance or irrelevance of component a denoted by, $Top_{a=1}$, can be obtained by substituting the value 1 for component a into equation (7.3).

$$\begin{aligned} Top_{a=1} &= b + ce + bd \\ &= b + ce \end{aligned}$$

Similarly an expression for the repair relevance or irrelevance of component a denoted by, $Top_{a=0}$, can be obtained by substituting the value 0 for component a into equation (7.3).

$$\begin{aligned} Top_{a=0} &= d + ce + bd \\ &= d + ce \end{aligned}$$

An expression for the irrelevance of component a, denoted by, $Top_{a=\text{'-'}}$ is obtained by taking the product of $Top_{a=1}$ and $Top_{a=0}$, see figure 7.1.

$$\begin{aligned} Top_{a=\text{'-'}} &= Top_{a=1} \cdot Top_{a=0} \\ &= (b + ce)(d + ce) \\ &= bd + ce \end{aligned}$$

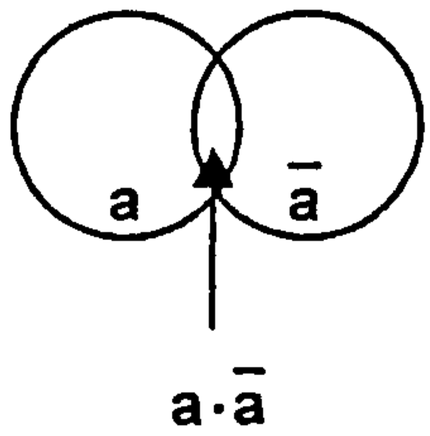


Figure 7.1: Component Irrelevance

In general an expression for the failure relevance or irrelevance of component i, $T_{i=1}$, is obtained by setting component i=1 and evaluating the structure function $\varphi(\underline{x})$:

$$T_{i=1} = \varphi(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \tag{7.4}$$

Similarly an expression for the repair relevance or irrelevance of component i, $T_{i=0}$, is obtained by setting component i=0 and evaluating the structure function $\varphi(\underline{x})$:

$$T_{i=0} = \varphi(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) \tag{7.5}$$

Finally an expression for the irrelevance of component i, $T_{i=-}$, is obtained by removing all minimal cut sets containing either, i or $\bar{i}$ from the structure function $\varphi(\underline{x})$:

$$T_{i=-} = \varphi(x_1,\ldots,x_{i-1},-,x_{i+1},\ldots,x_n) \tag{7.6}$$

7.5 An Alternative Extension of Birnbaum's Measure of Component Reliability Importance

Birnbaum's measure of component reliability importance (defined as the probability that component i is critical to system failure) is a fundamental probabilistic measure of importance. Many other measures of importance are based on this measure. Birnbaum developed this measure for the analysis of coherent systems only. It is calculated from the system unavailability function, $Q_{sys}(t)$, which is obtained using the exclusion-inclusion principle and Boolean reduction laws. $G_i(\underline{q})$, can be evaluated from equation (7.1), since $Q_{SYS}(t)$ is linear in each q_i can be expressed as:

$$G_i(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial q_i} \tag{7.7}$$

Birnbaum's measure for coherent systems is central to so many other measures of importance hence, its extension to enable analysis of non-coherent systems needs to provide a consistent foundation to then derive these measures for non-coherent analysis.

Birnbaum's measure calculates the probability that component i is critical to the system state. When dealing with a coherent system, system failure can only be caused by component failure. Hence a component in a coherent system can only be **failure critical**. However, when dealing with a non-coherent system, system failure can be caused not only by component failure referred to as event I, but also by component repair referred to as event $\overline{i}$, thus a component in a non-coherent system can be **failure critical** or **repair critical**. These two criticalities must be considered separately since component i can exist in only one state at any time [22].

The probability required is the probability that component i is critical to system failure, which can be expressed as the probability that component i is repair critical, $G_i^R(\underline{q})$or the probability that component i is failure critical , $G_i^F(\underline{q})$.

$$G_i(q) = G_i^R(\underline{q}) + G_i^F(\underline{q}) \tag{7.8}$$

An expression for the system unavailability function can be obtained from Henley and Inagaki's calculation procedure outlined in chapter four. Component i is failure critical if the system is working but will fail if component i fails. Thus the probability that component i is failure critical is the probability that the system is in a working state such that the failure of component i causes at least one prime implicant set containing event i to occur. This probability is calculated by obtaining the probability that at least one prime implicant set containing event i exists at time t and then dividing this probability by the unavailability of component i.

To calculate this probability it is first helpful to re-express the system unavailability as three distinct terms as follows:

$$Q_{SYS}(t) = q_i \Pr[A] + p_i \Pr[B] + \Pr[C] \tag{7.9}$$

The three terms of this equation represent those products involving the failure of component i those products involving the repair of i and those products for which component i is irrelevant respectively.

Now the probability that component i is failure critical is calculated as follows:

$$G_i^F(\underline{q}) = Pr[A] \tag{7.10}$$

Similarly the probability that component i is repair critical is the probability that the system is in a working state such that the repair of component i causes at least one prime implicant set containing event $\overline{i}$ to occur. This is calculated as follows:

$$G_i^R(\underline{q}) = Pr[B] \tag{7.11}$$

The top event can only exist at time t if at least one prime implicant set exists at time t. Hence, the failure and repair criticality can be calculated separately by differentiating the system unavailability function, $Q_{SYS}(t)$ with respect to q_i and p_i respectively.

$$G_i^F(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial q_i} \tag{7.12}$$

$$G_i^R(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial p_i} \tag{7.13}$$

Consider the non-coherent system introduced above, the Boolean expression for the top event is given below:

$$T = ab + ac + b\overline{c}$$

As before component availability is denoted by p_i and component unavailability is denoted by q_i for $i = a,b,c$. An expression for the system unavailability has already been calculated and is shown below.

$$Q_{SYS}(t) = q_a q_b + q_a q_c + q_b p_c - q_a q_b q_c - q_a q_b p_c$$

Now the proposed extension can be used to calculate the repair and the failure importance of any component. The failure importance for component c can be calculated from equation (7.12).

$$G_c^F(\underline{q}) = q_a - q_a q_b = q_a p_b \tag{7.14}$$

Similarly the repair importance for component c can be calculated from equation (7.13).

$$G_c^R(\underline{q}) = q_b - q_a q_b = q_b p_a \tag{7.15}$$

Hence from equation (7.8):

$$G_c(\underline{q}) = p_a q_b + q_a p_b$$

The result obtained using the proposed equations is the same as the results obtained in section 7.2, table 7.2. Hence the proposed extension calculates the probability that component i is critical to system failure.

7.5.1 The Expected Number of System Failures

The expression for calculating the expected number of system failures, $W_{sys}(0,t)$ when analysis is coherent, can be given in terms of Birnbaum's measure of component reliability importance.

$$W_{SYS}(0,t) = \int_0^t \sum_{i=1}^{n_c} G_i(\underline{q}) w_i(u) du \tag{7.16}$$

Where, $w_i(t)$ denotes the component unconditional failure intensity and n_c denotes the total number of system components.

The identity in equation (7.16) can be extended to non-coherent systems as follows:

$$W_{SYS}(0,t) = \int_0^t \left[\sum_{i=1}^{n_c} G_i^F(\underline{q}) w_i(u) + \sum_{i=1}^{n_c} G_i^R(\underline{q}) v_i(u) \right] du \tag{7.17}$$

Where, $v_i(t)$ denotes the component unconditional repair intensity.

The first term on the right hand side of equation (7.17) calculates the number of occurrences of system failure due to the failure of component i in a given interval. The second term calculates the number of occurrences of system failure due to the repair of component i in a given interval.

7.6 Deriving Other Measures of Component Importance

Birnbaum's measure of importance forms the basis for a number of other measures of component importance, including the measure of component criticality, Barlow and Proschan's measure of initiator importance and the modified measure of enabler importance [23]. It is possible to derive these measures of importance for non-coherent analysis using the extension of Birnbaum's measure developed in section 7.4.

Since components in a non-coherent system can be failure and repair critical it will be necessary to derive an expression for both the failure and repair importance of a component i for each measure.

7.6.1 The Component Criticality Measure

For coherent systems this measure is defined as the probability that component i is critical to the system and i has failed weighted by the system unavailability. Hence the criticality measure of failure importance can be defined as the probability that component i is failure critical to the system and i has failed weighted by the system unavailability:

$$I_{C_i}^{F} = \frac{G_i^F q_i}{Q_{SYS}(t)} \tag{7.18}$$

Similarly the criticality measure of component repair importance is defined as the probability that component i is success (repair) critical to the system and is in a working state, weighted by the system unavailability.

$$I_{C_i}^{R} = \frac{G_i^R p_i}{Q_{SYS}(t)} \tag{7.19}$$

The total criticality measure of importance is obtained by summing the failure and repair contributions:

$$I_{C_i} = I_{C_i}^F + I_{C_i}^R \tag{7.20}$$

7.6.2 Fussell-Vesely's Measure of Component Importance

For coherent systems the Fussell-Vesely measure of component importance [20] is concerned with component failures contributing to system failure. The measure is defined as the probability that the failure of component i contributes to system failure. An expression for the Fussell-Vesely measure is given in equation (7.21).

$$I_{FV_i} = \frac{P\left[\bigcup_{\substack{k=1 \\ i \in C_K}}^{n_p} C_k\right]}{Q_{SYS}(t)} \tag{7.21}$$

This measure can also be extended for non-coherent analysis. The Fussell-Vesely failure importance is defined as the probability that the failed state of component i contributes to system failure, an expression for this is given in equation (7.22).

$$I_{FV_i}^F = \frac{P\left[\bigcup_{\substack{k=1 \\ i \in C_K}}^{n_p} C_k\right]}{Q_{SYS}(t)} \tag{7.22}$$

Similarly the Fussell-Vesely repair importance is defined as the probability that the working state of component i contributes to system failure weighted by the system unavailability, an expression for this is given in equation (7.23).

$$I_{FV_i}^{R} = \frac{P\left[\bigcup_{k=1}^{n_p} C_k\right]}{Q_{SYS}(t)} \qquad (7.23)$$

7.6.3 Barlow and Proschan's Measure of Initiator Importance

For non-coherent systems the failure or repair of a component could act as an initiator causing system failure. The initiator failure importance is defined as the probability that the failure of component i causes system failure in the interval $[0,t)$. An expression for the initiator failure importance is given in equation (7.24):

$$I_{IN_i}^{F} = \frac{\int_0^t G_i^F(q) w_i(u) du}{W_{SYS}(0,t)} \qquad (7.24)$$

The initiator repair importance is defined as the probability that the repair of component i causes the system to fail in the interval $[0,t)$.

$$I_{IN_i}^{R} = \frac{\int_0^t G_i^R(q) v_i(u) du}{W_{SYS}(0,t)} \qquad (7.25)$$

Where, $v_i(t)$ is the unconditional repair intensity of component i.

The full initiator importance of component i is obtained by summing the repair and failure initiator importance of i:

$$I_{IN_i} = I_{IN_i}^{F} + I_{IN_i}^{R} \qquad (7.26)$$

7.6.4 The Modified Measure of Enabler Importance

To obtain this measure it is necessary to consider the role of both component failure and working states in the system. The failure of component i could result in component j being either failure or repair critical. Thus the probability required for the failure enabler importance of component i when j causes system failure is:

> The probability that component i fails leaving the system failure critical for component j and j fails, or that component i fails leaving the system repair critical for component j and j is repaired. Weighted by the expected number of system failures in the interval $[0,t)$.

The probability that components i and j are failure critical at time t, and the probability that i is failure critical and j is repair critical at time t are calculated as follows:

$$G_{i,j}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_i(t)\partial q_j(t)} \tag{7.27}$$

$$G_{i,\bar{j}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_i(t)\partial p_j(t)} \tag{7.28}$$

Both criticalities require a correction term to ensure that the separate role of component failures and repairs are eliminated. The first correction term must calculate the probability that i and j are failure critical to the system such that the failure of i or j alone would be sufficient to cause system failure. Similarly the second correction term should calculate the probability that component i is failure critical and component j is repair critical such that the failure of i or the repair of j alone would be sufficient to cause system failure. These corrections terms are given below:

$$G_{M_{i,j}}(\underline{q}) = \frac{\partial^2 Q_{M_{i,j}}(t)}{\partial q_i(t)\partial q_j(t)} \tag{7.29}$$

$$G_{M_{i,\bar{j}}}(\underline{q}) = \frac{\partial^2 Q_{M_{i,\bar{j}}}(t)}{\partial q_i(t)\partial p_j(t)} \tag{7.30}$$

Where, $Q_{M_{i,j}}(t)$ is the modified system unavailability function for I and j such that:

$$Q_{M_{i,j}}(t) = \bigcup_{\substack{k=1 \\ i \cap j \in C_K \\ \bar{i} \notin C_K \\ \bar{j} \notin C_K}}^{n_p} P(C_K) \tag{7.31}$$

And, $Q_{M_{i,\bar{j}}}(t)$ is the modified system unavailability function for i and $\bar{j}$ such that:

$$Q_{M_{i,\bar{j}}}(t) = \bigcup_{\substack{k=1 \\ i \cap \bar{j} \in C_K \\ \bar{i} \notin C_K \\ j \notin C_K}}^{n_p} P(C_K) \tag{7.32}$$

Hence the failure enabler importance of component i when component j initiates system failure is given in equation (7.36).

$$I^F_{E_{i,j}} = \frac{\int_0^t \left[\left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q}) \right] q_i w_j + \left[G_{i,\bar{j}}(\underline{q}) - G_{M_{i,\bar{j}}}(\underline{q}) \right] q_i v_j \right] du}{W_{SYS}(0,t)} \tag{7.33}$$

The total failure enabler importance of component i is given in equation (7.34):

$$I^F_{E_i} = \frac{\sum_{\substack{j=1 \\ j \neq i}}^{n_{Fj}} \int_0^t \left[G_{i,j}(\underline{q}) - G_{M_{i,j}}(\underline{q}) \right] q_i w_j du + \sum_{\substack{j=1 \\ j \neq i}}^{n_{Rj}} \int_0^t \left[G_{i,\bar{j}}(\underline{q}) - G_{M_{i,\bar{j}}}(\underline{q}) \right] q_i v_j du}{W_{SYS}(0,t)} \tag{7.34}$$

The probability required for the repair enabler importance of component i when component j causes system failure is:

The probability that component i has been repaired leaving the system failure critical for component j and component j fails, or that component i has been repaired leaving the system repair critical for component j and component j is repaired.

By similar arguments to those used above to derive the failure enabler importance, the repair enabler importance of component i when component j causes system failure is calculated as follows.

$$I^{R}_{E_{i,j}} = \frac{\int_0^t \left[\left[G_{\bar{i},j}(\underline{q}) - G_{M_{\bar{i},j}}(\underline{q}) \right] p_i w_j + \left[G_{\bar{i},\bar{j}}(\underline{q}) - G_{M_{\bar{i},\bar{j}}}(\underline{q}) \right] p_i v_j \right] du}{W_{SYS}(0,t)} \qquad (7.35)$$

And the total repair enabler importance for component is given in equation (7.36):

$$I^{R}_{E_i} = \frac{\sum_{\substack{j=1 \\ j \neq i}}^{n_{Fj}} \int_0^t \left[G_{\bar{i},j}(\underline{q}) - G_{M_{\bar{i},j}}(\underline{q}) \right] p_i w_j du + \sum_{\substack{j=1 \\ j \neq i}}^{n_{Rj}} \int_0^t \left[G_{\bar{i},\bar{j}}(\underline{q}) - G_{M_{\bar{i},\bar{j}}}(\underline{q}) \right] p_i v_j du}{W_{SYS}(0,t)} \qquad (7.36)$$

Where:

$G_{\bar{i},j}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial p_i(t) \partial q_j(t)}$ Is the probability that component i is repair critical and component j is failure critical at time t.

$G_{\bar{i},\bar{j}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial p_i(t) \partial p_j(t)}$ Is the probability that components i and j are repair critical at time t.

$G_{M_{\bar{i},j}}(\underline{q}) = \frac{\partial^2 Q_{M_{\bar{i},j}}(t)}{\partial p_i(t) \partial q_j(t)}$ Is the probability that component i is repair critical and j is failure critical such the repair of i or the failure of j alone is sufficient to cause system failure.

$G_{M_{\bar{i},\bar{j}}}(\underline{q}) = \frac{\partial^2 Q_{M_{\bar{i},\bar{j}}}(t)}{\partial p_i(t) \partial p_j(t)}$ Is the probability that component i is repair critical and j is repair critical such the repair of i or j alone is sufficient to cause system failure.

7.7 Extending Measures of Minimal Cut Set Importance

Two measures of cut set importance were introduced in chapter six, the first developed by Fussell and Vesely enables the importance of each cut set to be calculated in terms of its contribution to system failure. The second measure initially developed by Barlow

and Proschan, but modified by Beeson and Andrews enables the frequency importance of each cut set to be calculated. Both of these measures will be extended to enable importance analysis of the prime implicant sets of a non-coherent system.

7.7.1 Fussell-Vesely's Measure of Cut Set Importance

In 1974 Fussell and Vesely developed a measure of importance to assess cut set importance. The measure is defined as the probability that a cut set C_i contributes to system failure. This measure can be extended to enable the analysis of prime implicant set importance, by considering the probability that prime implicant set ε_i contributes to the system failure:

$$I_{FV}(\varepsilon_i) = \frac{P(\varepsilon_i)}{Q_{SYS}} \tag{7.37}$$

7.7.2 The Measure of Cut Set Frequency Importance

The measure of cut set frequency importance developed in chapter six calculates the probability that a minimal cut set C_i^n causes system failure. This measure is extended for use with non-coherent systems by considering the probability that a prime implicant set ε_i^n causes system failure, where, ε_i^n is a prime implicant set of order n. The prime implicant set ε_i^n can cause system failure in n ways, since each component in ε_i^n could act as an initiator causing the prime implicant set and the system to fail. Furthermore for the failure of ε_i^n to cause system failure no other prime implicant set can exist at time t.

Hence the frequency importance of ε_i^n is calculated by taking the sum for all $k \in \varepsilon_i^n$ of the probability that the system is in a critical state for ε_i^n and all component states in ε_i^n except k exist at time t multiplied by the probability that component k is failed if it occurs as a normal literal in ε_i^n, or repaired if it occurs as a negated literal in ε_i^n in the interval [t,t+dt).

The first stage is to calculate the probability that prime implicant set ε_i^n is critical to the system state. The probability that the components contained in ε_i^n are critical to the system state, such that the failure (if normal literals) or repair (if negated literals) of all n components in ε_i^n would cause the failure of ε_i^n and the system is expressed as follows:

$$G_{\varepsilon_i^n}(\underline{q}) = \frac{\partial^n Q_{SYS}(t)}{\partial q^{\alpha k^n}} \Bigg|_{k \epsilon \varepsilon_i^n} \tag{7.38}$$

Where: $q^{\alpha k} = \begin{cases} q_k(t) \text{ if } \alpha_k = 1 \\ p_k(t) \text{ if } \alpha_k = 0 \end{cases}$, α is set to 1 if the literal exists in its normal form and 0 if the literal is negated.

As with the coherent measure this does not always give the required probability and a correction term must be applied to $G_{\varepsilon_i^n}(\underline{q})$ to eliminate the separate contributions of the elements contained in ε_i^n. First a modified system unavailability function must be formed for ε_i^n, $Q_{M_{\varepsilon_i^n}}$:

$$Q_{M_{\varepsilon_i^n}} = \bigcup_{\substack{j=1 \\ j \neq i}}^{n_c} P(\varepsilon_j) \tag{7.39}$$

Then this function is differentiated with respect to all elements contained in ε_i^n to give the required correction term, $G_{M_{\varepsilon_i^n}}(\underline{q})$.

$$G_{M_{\varepsilon_i^n}}(\underline{q}) = \frac{\partial^n Q_{M_{\varepsilon_i^n}}(t)}{\partial q^{\alpha k^n}} \Bigg|_{k \epsilon \varepsilon_i^n} \tag{7.40}$$

Once the probability that ε_i^n is critical has been obtained it is multiplied by the probability that components, j=1,..,n , $j \neq k$, are failed or repaired at time t and component k fails (if it occurs as a normal literal) or is repaired (if it occurs as a negated literal) in the interval [t,t+dt). This is summed over $j \in \varepsilon_i^n$ giving the probability that prime implicant set ε_i^n causes system failure.

$$I_F(\varepsilon_i^n) = \sum_{j \in \varepsilon_i^n} \int_0^t \left[G_{\varepsilon_i^n}(q) - G_{M_{\varepsilon_i^n}}(q) \right] \cdot \prod_{k \in \varepsilon_i^n - \{j\}} q^{\alpha k} \cdot z^{\alpha j} du \qquad (7.41)$$

Where:

$z^{\alpha j} = \begin{cases} w_j(t) \text{ if } \alpha_j = 1 \\ v_j(t) \text{ if } \alpha_j = 0 \end{cases}$ α is set to 1 if the literal exists in its normal form and 0 if the literal is negated. $w_j(t)$ is the unconditional failure intensity of component j and $v_j(t)$ is the unconditional repair intensity of component j.

7.8 Methods for Calculating Measures of Importance

Conventional fault tree techniques for calculating the measures of importance outlined above will be illustrated in section 7.8.1. Then techniques for calculating these measures using either the SFBDD or the consensus BDD will be developed in section 7.8.2.

7.8.1 The Fault Tree Analysis Techniques

Measures of importance can be calculated during quantitative Fault Tree Analysis, the calculation procedures rely on knowledge of the prime implicant sets of the fault tree and tend to involve evaluating lengthy series expansions. To illustrate the Fault Tree Analysis techniques for calculating the seven measures of importance introduced above consider the fault tree diagram in figure 7.2.

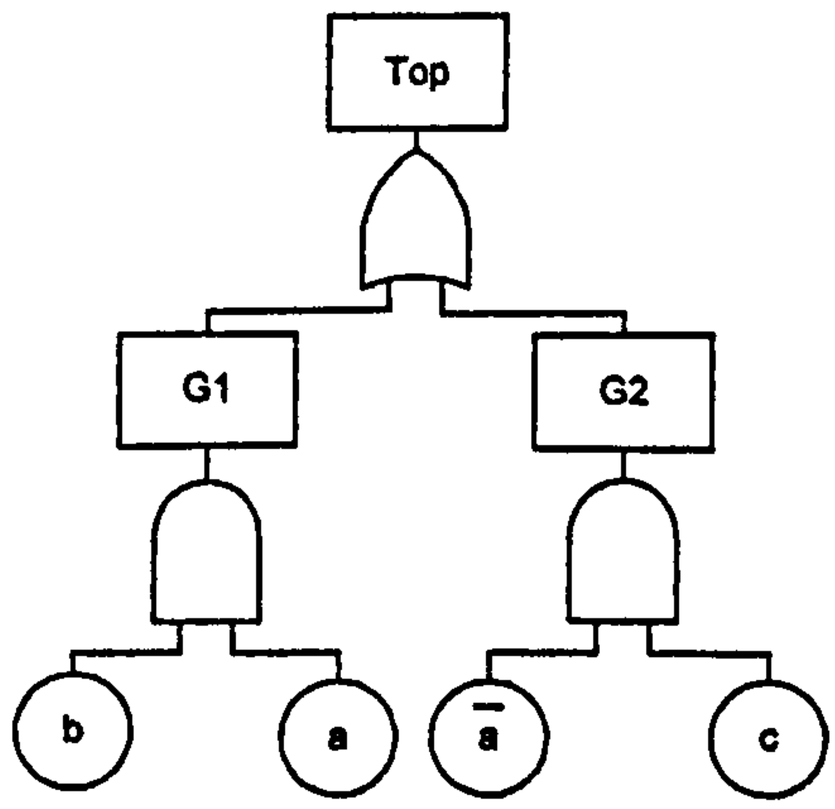


Figure 7.2: Non-Coherent Fault Tree Diagram

This non-coherent fault tree has three prime implicant sets:

$$\{a,b\},\{\bar{a},c\},\{b,c\}$$

The first stage is to obtain an expression for the system unavailability. Inagaki and Henley's method considered in chapter four is used to obtain an expression for the system unavailability.

$$Q_{SYS}(t) = q_a q_b + p_a q_c + q_b q_c - q_a q_b q_c - p_a q_b q_c$$

From equations, (7.12) and (7.13), expressions for Birnbaum's measure of component failure and repair importance can be obtained for each component as follows:

$$G_a^F(\underline{q}) = \frac{\partial Q_{SYS}}{\partial q_a} = q_b - q_b q_c = q_b p_c \qquad G_a^R(\underline{q}) = \frac{\partial Q_{SYS}}{\partial p_a} = q_c - q_b q_c = q_c p_b$$

$$G_b^F(\underline{q}) = \frac{\partial Q_{SYS}}{\partial q_b} = q_a + q_c - q_a q_c - p_a q_c = q_a \qquad G_b^R(\underline{q}) = \frac{\partial Q_{SYS}}{\partial p_b} = 0$$

$$G_c^F(\underline{q}) = \frac{\partial Q_{SYS}}{\partial q_c} = p_a + q_c - q_a q_c - p_a q_c = p_a \qquad G_c^R(\underline{q}) = \frac{\partial Q_{SYS}}{\partial p_c} = 0$$

Now expressions for the system unconditional failure intensity and thus the expected number of system failures in a given interval can be obtained:

$$w_{SYS}(t) = \sum_{i=1}^{n} G_i^F(\underline{q}) w_i + \sum_{i=1}^{n} G_i^R(\underline{q}) v_i$$

$$= q_b p_c w_A + p_b q_c v_a + q_a w_B + p_a w_C$$

$$W(0,t) = \int_0^t w_{SYS}(u)du$$

From equations, (7.18) and (7.19), and the results for Birnbaum's measure of component failure and repair importance, expressions for the failure and repair criticality measure can be obtained for each component:

$$I^F_{C_a} = \frac{G^F_a(\underline{q})q_a}{Q_{SYS}} = \frac{q_b p_c q_a}{Q_{SYS}} \qquad I^R_{C_a}(\underline{q}) = \frac{G^R_a(\underline{q})p_a}{Q_{SYS}} = \frac{p_b q_c p_a}{Q_{SYS}}$$

$$I^F_{C_b} = \frac{G^F_b(\underline{q})q_b}{Q_{SYS}} = \frac{q_a q_b}{Q_{SYS}} \qquad I^R_{C_b}(\underline{q}) = \frac{G^R_b(\underline{q})p_b}{Q_{SYS}} = 0$$

$$I^F_{C_c} = \frac{G^F_c(\underline{q})q_c}{Q_{SYS}} = \frac{p_a q_c}{Q_{SYS}} \qquad I^R_{C_c}(\underline{q}) = \frac{G^R_c(\underline{q})p_c}{Q_{SYS}} = 0$$

Fussell-Vesely's measure of failure importance is calculated for component i by considering only those prime implicant sets that containing the normal literal i. From equation (7.22) and knowledge of the three prime implicant sets, expressions for Fussell-Vesely's measure of failure component importance are obtained as follows:

$$I^F_{FV_a} = \frac{P\left[\bigcup_{\substack{k=1 \\ a\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{q_a q_b}{Q_{SYS}}$$

$$I^F_{FV_b} = \frac{P\left[\bigcup_{\substack{k=1 \\ b\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{q_a q_b + q_b q_c - q_a q_b q_c}{Q_{SYS}}$$

$$I^F_{FV_c} = \frac{P\left[\bigcup_{\substack{k=1 \\ c\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{p_a q_c + q_b q_c - p_a q_b q_c}{Q_{SYS}}$$

The following expressions are obtained for Fussel-Vesely's measure of repair importance from equation (7.23) and knowledge of the prime implicant sets:

$$I^R_{FV_a} = \frac{P\left[\bigcup_{\substack{k=1 \\ \bar{a}\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = \frac{p_a q_c}{Q_{SYS}}$$

$$I^R_{FV_b} = \frac{P\left[\bigcup_{\substack{k=1 \\ \bar{b}\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = 0$$

$$I^R_{FV_c} = \frac{P\left[\bigcup_{\substack{k=1 \\ \bar{c}\in C_K}}^{3} C_k\right]}{Q_{SYS}(t)} = 0$$

Barlow and Proschan's measures of initiator failure and repair importance are calculated from equations (7.24) and (7.25), and the results obtained for Birnbaum's measure of failure and repair importance:

$$I^F_{IN_a} = \frac{\int_0^t G^F_a(q) w_a \, du}{W_{SYS}(0,t)} = \frac{\int_0^t q_b p_c w_a \, du}{W_{SYS}(0,t)} \qquad I^R_{IN_a}(q) = \frac{\int_0^t G^R_a(q) v_a}{W_{SYS}(0,t)} = \frac{\int_0^t p_b q_c v_a \, du}{W_{SYS}(0,t)}$$

$$I^F_{IN_b} = \frac{\int_0^t G^F_b(q) w_b \, du}{W_{SYS}(0,t)} = \frac{\int_0^t q_a w_b \, du}{W_{SYS}(0,t)} \qquad I^R_{IN_b}(q) = \frac{\int_0^t G^R_b(q) v_b \, du}{W_{SYS}(0,t)} = 0$$

$$I^F_{IN_c} = \frac{\int_0^t G^F_c(q) w_c \, du}{W_{SYS}(0,t)} = \frac{\int_0^t p_a w_c \, du}{W_{SYS}(0,t)} \qquad I^R_{IN_c}(q) = \frac{\int_0^t G^R_c(q) v_c \, du}{W_{SYS}(0,t)} = 0$$

The procedures for calculating the component failure and repair enabler importance are quite involved and will only be illustrated for component a.

The failure enabler importance of a component i is obtained by considering the role of the failure of component i in enabling the failure or repair of another component j to act as an initiator causing system failure.

The failure of component a can only allow the failure of component b to act as an initiator. To calculate the failure enabler importance of component a both, $G_{a,b}(q)$ and $G_{M_{a,b}}(q)$ must be calculated.

$$G_{a,b}(\underline{q}) = \frac{\partial^2 Q_{SYS}}{\partial q_a \partial q_b} = 1 - q_c$$

To calculate $G_{M_{a,b}}(\underline{q})$ the modified system unavailability function for a and b, $Q_{M_{a,b}}$ must be obtained:

$$Q_{M_{a,b}}(t) = \bigcup_{\substack{k=1 \\ a \cap b \in C_K \\ \bar{a} \in C_K \\ \bar{b} \in C_K}}^{n_p} P(C_K) = q_b q_c$$

Then the following result can be obtained for $G_{M_{a,b}}(\underline{q})$:

$$G_{M_{a,b}}(\underline{q}) = \frac{\partial^2 Q_{M_{a,b}}}{\partial q_a \partial q_b} = 0$$

Then from equation (7.34) the following expression is obtained for the failure enabler importance of component a.

$$I^F_{E_{a,b}} = \frac{\int_0^t [1 - q_c(u)] q_a(u) w_b(u) du}{W_{SYS}(0,t)}$$

The repair enabler importance of component a is obtained in a similar way. The repair of component a can only allow the failure of component c to act as an initiator. To calculate the repair enabler importance of component a both $G_{\bar{a},c}(\underline{q})$ and $G_{M_{\bar{a},c}}(\underline{q})$ must be calculated.

$$G_{\bar{a},c}(\underline{q}) = \frac{\partial^2 Q_{SYS}}{\partial p_a \partial q_c} = 1 - q_b$$

To calculate $G_{M_{\bar{a},c}}(\underline{q})$ the modified system unavailability function for $\bar{a}$ and b, $Q_{M_{\bar{a},c}}$ must be obtained:

$$Q_{M_{\bar{a},c}}(t) = \bigcup_{\substack{k=1 \\ \bar{a}\cap c \in C_K \\ a \in C_K \\ \bar{c} \in C_K}}^{n_p} P(C_K) = q_b q_c$$

$$G_{M_{\bar{a},c}}(\underline{q}) = \frac{\partial^2 Q_{M_{\bar{a},c}}}{\partial p_a \partial q_c} = 0$$

Then from equation (7.36) the following expression is obtained for the repair enabler importance of component a.

$$I^R_{E_{\bar{a},c}} = \frac{\int_0^t [1-q_b] b_a(u) w_c(u) du}{W_{SYS}(0,t)}$$

Expressions for the failure and repair enabler importance of components b and c are obtained in the same way from equations (7.34) and (7.36):

$$I^F_{IN_b} = \frac{\int_0^t (1-q_c(u)) q_b(u) w_a(u) du}{W_{SYS}(0,t)} \qquad I^R_{IN_b}(\underline{q}) = 0$$

$$I^F_{IN_c} = \frac{\int_0^t (1-q_b(u)) q_c(u) v_a(u) du}{W_{SYS}(0,t)} \qquad I^R_{IN_c}(\underline{q}) = 0$$

Now expressions for the extended measure of Fussell-Vesely's prime implicant set importance are obtained from equation (7.37):

$$I_{FV}(\varepsilon_1) = \frac{q_a q_b}{Q_{SYS}(t)}$$

$$I_{FV}(\varepsilon_2) = \frac{p_a q_c}{Q_{SYS}(t)}$$

$$I_{FV}(\varepsilon_3) = \frac{q_b q_c}{Q_{SYS}(t)}$$

Finally expressions for the frequency importance of each prime implicant set will be obtained. Beginning with prime implicant set, {ab}. From equation (7.38) an expression for $G_{\{ab\}}(\underline{q})$ is obtained:

$$G_{\{ab\}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_a \partial q_b} = 1 - q_c$$

To calculate the correction term, an expression for the modified system unavailability function, $Q_{M\{ab\}}$ is formed using equation (7.39):

$$Q_{M\{ab\}}(t) = p_a q_c + q_b q_c - p_a q_b q_c$$

Then from the expression for $Q_{M\{ab\}}$ the correction term, $G_{M\{ab\}}(\underline{q})$, is calculated:

$$G_{M\{ab\}}(\underline{q}) = \frac{\partial^2 Q_{M\{ab\}}(t)}{\partial q_a \partial q_b} = 0$$

Finally from equation (7.41) and the results obtained for $G_{\{ab\}}(\underline{q})$ and $G_{M\{ab\}}(\underline{q})$ the following result is obtained for the frequency importance of prime implicant set {ab}.

$$I_F(\{ab\}) = \int_0^t (1 - q_c) q_b w_a \, du + \int_0^t (1 - q_c) q_a w_b \, du$$

This process is then repeated for prime implicant set $\{\bar{a}\,c\}$. From equation (7.38) an expression for $G_{\{\bar{a}c\}}(\underline{q})$ is obtained:

$$G_{\{\bar{a}c\}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial p_a \partial q_c} = 1 - q_b$$

To calculate the correction term, an expression for the modified system unavailability function, $Q_{M\{\bar{a}c\}}$ is formed using equation (7.39):

$$Q_{M\{\bar{a}c\}}(t) = q_a q_b + q_b q_c - q_a q_b q_c$$

Then from the expression for $Q_{M\{\bar{a}c\}}$ the correction term, $G_{M\{\bar{a}c\}}(\underline{q})$, is calculated:

Finally expressions for the frequency importance of each prime implicant set will be obtained. Beginning with prime implicant set, {ab}. From equation (7.38) an expression for $G_{\{ab\}}(\underline{q})$ is obtained:

$$G_{\{ab\}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial q_a \partial q_b} = 1 - q_c$$

To calculate the correction term, an expression for the modified system unavailability function, $Q_{M\{ab\}}$ is formed using equation (7.39):

$$Q_{M\{ab\}}(t) = p_a q_c + q_b q_c - p_a q_b q_c$$

Then from the expression for $Q_{M\{ab\}}$ the correction term, $G_{M\{ab\}}(\underline{q})$, is calculated:

$$G_{M\{ab\}}(\underline{q}) = \frac{\partial^2 Q_{M\{ab\}}(t)}{\partial q_a \partial q_b} = 0$$

Finally from equation (7.41) and the results obtained for $G_{\{ab\}}(\underline{q})$ and $G_{M\{ab\}}(\underline{q})$ the following result is obtained for the frequency importance of prime implicant set {ab}.

$$I_F(\{ab\}) = \int_0^t (1 - q_c) q_b w_a du + \int_0^t (1 - q_c) q_a w_b du$$

This process is then repeated for prime implicant set $\{\bar{a}\,c\}$. From equation (7.38) an expression for $G_{\{\bar{a}c\}}(\underline{q})$ is obtained:

$$G_{\{\bar{a}c\}}(\underline{q}) = \frac{\partial^2 Q_{SYS}(t)}{\partial p_a \partial q_c} = 1 - q_b$$

To calculate the correction term, an expression for the modified system unavailability function, $Q_{M\{\bar{a}c\}}$ is formed using equation (7.39):

$$Q_{M\{\bar{a}c\}}(t) = q_a q_b + q_b q_c - q_a q_b q_c$$

Then from the expression for $Q_{M\{\bar{a}c\}}$ the correction term, $G_{M\{\bar{a}c\}}(\underline{q})$, is calculated:

$$G_{M\{\bar{a}c\}}(\underline{q}) = \frac{\partial^2 Q_{M\{\bar{a}c\}}(t)}{\partial p_a \partial q_c} = 0$$

Finally from equation (7.41) and the results obtained for $G_{\{\bar{a}c\}}(\underline{q})$ and $G_{M\{\bar{a}c\}}(\underline{q})$ the following result is obtained for the frequency importance of prime implicant set $\{\bar{a}\,c\}$.

$$I_F(\{\bar{a}c\}) = \int_0^t (1-q_b) q_c v_a du + \int_0^t (1-q_b) p_a w_c du$$

Finally if this process is repeated for prime implicant set {bc} the following expression is obtained for the frequency importance:

$$I_F(\{bc\}) = 0$$

Notice that the result for prime implicant set {bc} is 0, this is because if both b and c are failed the system is automatically failed hence the prime implicant set {bc} cannot actually cause system failure.

7.8.2 The Binary Decision Diagram Method

For non-coherent fault trees it is possible to use either the SFBDD or the consensus BDD to calculate all of the measures of importance introduced above exactly except for the modified measure of enabler importance and the measure of prime implicant set frequency importance both of which must be approximated. The BDD method enables exact and efficient calculation of the measures of importance eliminating both the intermediate stage of identifying the prime implicant sets and the need to evaluate lengthy series expansions.

The procedures for calculating Birnbaum's measure of component failure and repair importance and the enabler measure of failure and repair importance, which were developed as part of this research project will be introduced and illustrated by means of a worked example.

7.8.2.1 Calculating Birnbaum's Measure of Failure and Repair Importance

Expressions for calculating Birnbaum's measure of component failure and repair importance are given below:

$$G_i^F(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial q_i}$$

$$G_i^R(\underline{q}) = \frac{\partial Q_{SYS}(t)}{\partial p_i}$$

The failure importance of component i is defined as the probability that the system is in a working, yet critical state, such that the failure of component i would cause the system to fail. Thus from the definition of component relevance/irrelevance given in section 7.4, it is possible to define Birnbaum's measure of component failure importance as the probability that component i is failure relevant to the system state:

$$G_i^F(\underline{q}) = E[T_{i=1}] - E[T_{i=-1}] \tag{7.42}$$

Where, $E[T_{i=1}]$ is the probability that component i is either failure relevant, or irrelevant to the state of the system, and $E[T_{i=-1}]$ is the probability that component i is irrelevant to the state of the system.

Similarly, Birnbaum's measure of component repair importance can be defined as the probability that component i is repair relevant to the system state:

$$G_i^R(\underline{q}) = E[T_{i=0}] - E[T_{i=-1}] \tag{7.43}$$

Where, $E[T_{i=0}]$ is the probability that component i is either repair relevant or irrelevant to the system state.

These alternative expressions enable the efficient calculation of Birnbaum's measure of component failure and repair importance from either the SFBDD or the Consensus BDD. Once Birnbaum's measure has been calculated for each component it is possible to calculate, the failure and repair criticality measure, the failure and repair initiator importance and the unconditional failure intensity and thus the expected number of system failures in a given interval by means of simple substitution.

7.8.2.1.1 The SFBDD Technique

It is possible to calculate $E[T_{i=1}]$ and $E[T_{i=0}]$ directly from the SFBDD, the procedure for calculating these probabilities is outlined below:

$$E[T_{i=1}] = \sum_{x_i} Pr_{x_i}(\underline{q}) \cdot Po^1_{x_i}(\underline{q}) \tag{7.44}$$

$$E[T_{i=0}] = \sum_{x_i} Pr_{x_i}(\underline{q}) \cdot Po^0_{x_i}(\underline{q}) \tag{7.45}$$

Where:

$Pr_{x_i}(\underline{q})$ is the probability of the path section from the root vertex to node x_i.

$Po^1_{x_i}(\underline{q})$ is the probability of the path section from the one branch of node x_i to a terminal 1 node (excluding the probability of x_i).

$Po^0_{x_i}(\underline{q})$ is the probability of the path section from the zero branch of the node x_i to a terminal 1 node (excluding the probability of x_i).

Although, $E[T_{i=1} \cdot T_{i=0}]$ can be calculated by taking the expectation of the product of $T_{i=1}$ and $T_{i=0}$, $E[T_{i=1} \cdot T_{i=0}] = E[T_{i=1} \cdot T_{i=0}]$. This is an inefficient means of calculating $E[T_{i=1} \cdot T_{i=0}]$. An alternative technique can be employed which involves computing an intermediate BDD for each node from which $E[T_{i=1} \cdot T_{i=0}]$ can be efficiently calculated. The intermediate BDD for each node is obtained by ANDing the one and zero branches of the node. An expression for $E[T_{i=1} \cdot T_{i=0}]$ is then calculated by multiplying the probability preceding the node in the SFBDD by the probability of the sum of all the terminal one paths through the intermediate BDD. To illustrate this technique consider again the non-coherent fault tree in figure 7.2, if the ordering, $b < a < c$ is adopted the SFBDD shown in figure 7.3 is obtained:

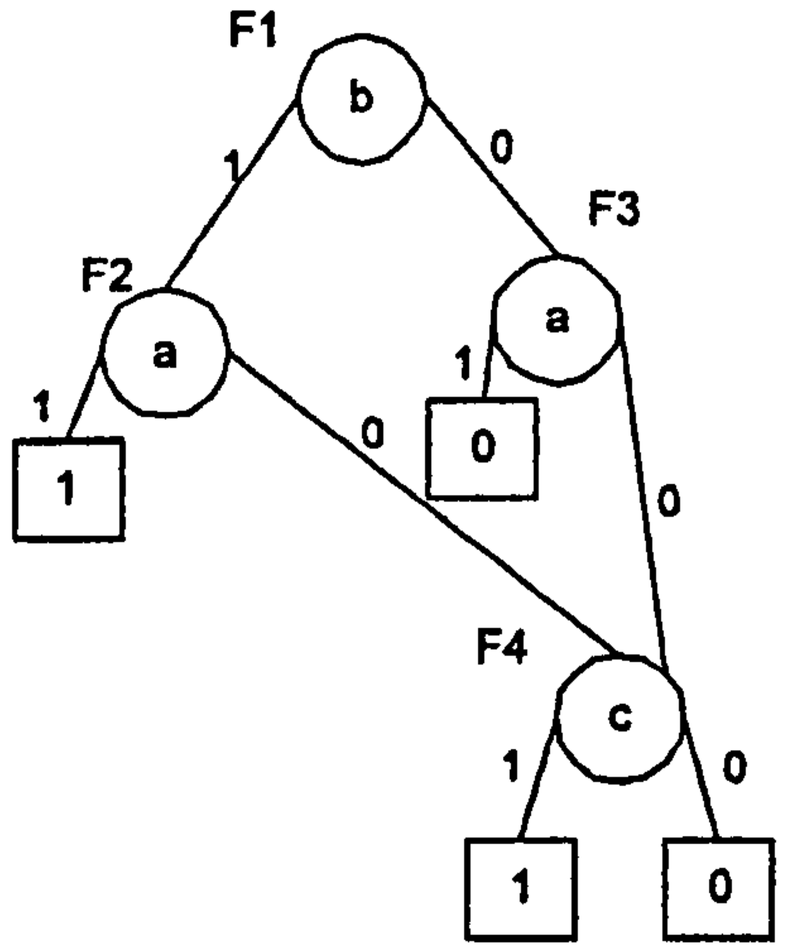


Figure 7.3: SFBDD

Firstly it is necessary to record the connections between the nodes in an ite table as shown in table 7.4.

Node	Variable	1 Branch	0 Branch
F1	b	F2	F3
F2	a	1	F4
F3	a	0	F4
F4	c	1	0

Table 7.4: ite Table

The second stage is to calculate each of the terms in equations (7.44) and (7.45). The first term, $Pr_{x_i}(\underline{q})$ is the probability of the path from the root vertex to node x_i, which is calculated by evaluating the probability of the path from the root vertex up to but not including the node x_i. Table 7.5 records $Pr_{x_i}(\underline{q})$ for each node in the BDD.

Node	$Pr_{x_i}(\underline{q})$	Comments
F1	1	Root vertex itself
F2	q_b	F2 reached through the 1 branch of F1
F3	p_b	F3 reached through the 0 branch of F1
F4	$p_a q_b + p_a p_b$	F4 reached through the 1 branch of F1 followed by the 0 branch of F2 & through the 0 branch of F1 followed by the zero branch of F3

Table 7.5: $Pr_{x_i}(\underline{q})$ for each Node in the SFBDD in Figure 7.3

The, $Po^1_{x_i}(\underline{q})$ term is calculated by summing the probability of all the paths from the selected node, x_i along the **one** branch to a terminal 1 vertex, excluding the probability of the selected node. Table 7.6 records $Po^1_{x_i}(\underline{q})$ for each node.

Node	$Po^1_{x_i}(\underline{q})$	Comments
F1	$q_a + p_a q_c$	1 branch of F1 passes to F2 the 1 branch of F2 passes to a terminal 1 vertex and the 0 branch passes to F4
F2	1	1 branch of F2 passes to a terminal 1 vertex
F3	0	No terminal 1 paths from 1 branch of F3
F4	1	1 branch of F4 passes to a terminal 1 vertex

Table 7.6: $Po^1_{x_i}(\underline{q})$ for each Node in the SFBDD in Figure 7.3

Similarly $Po^0_{x_i}(\underline{q})$ is calculated by summing the probability of all paths from the selected node, x_i along the **zero** branch to a terminal 1 vertex, excluding the probability of the selected node. Table 7.7 records $Po^0_{x_i}(\underline{q})$ for each node.

Node	$Po^0_{x_i}(\underline{q})$	Comments
F1	$p_a q_c$	0 branch of F1passes through the 0 branch of F3 & the 1 branch of F4
F2	q_c	0 branch of F2 passes through the 1 branch of F4
F3	q_c	0 branch of F3 passes through the 1 branch of F4
F4	0	No terminal 1 paths from the 0 branch of F4

Table 7.7: $Po^0_{x_i}(\underline{q})$ for each Node in the SFBDD in Figure 7.3

Finally $E[T_{x_i \cdot}]$ must be calculated for each of the nodes in the SFBDD. The calculation procedure requires some addition work. An intermediate BDD must be calculated for each node x_i. The probability of the sum of the disjoint paths through this intermediate BDD is calculated and multiplied by the probability preceding the node x_i to give $E[T_{x_i \cdot}]$. The intermediate BDD is computed by ANDing the one and zero branches of a node. Each of the nodes in the SFBDD in figure 7.3 will be considered below.

Dealing with node F1

$$\begin{aligned} F2 \cdot F3 &= ite(a,1,ite(c,1,0)) \cdot ite(a,0,ite(c,1,0)) \\ &= ite(a,[1 \cdot 0],[ite(c,1,0) \cdot ite(c,1,0)]) \\ &= ite(a,0,ite(c,1,0)) \end{aligned}$$

The resulting BDD is shown in figure 7.4:

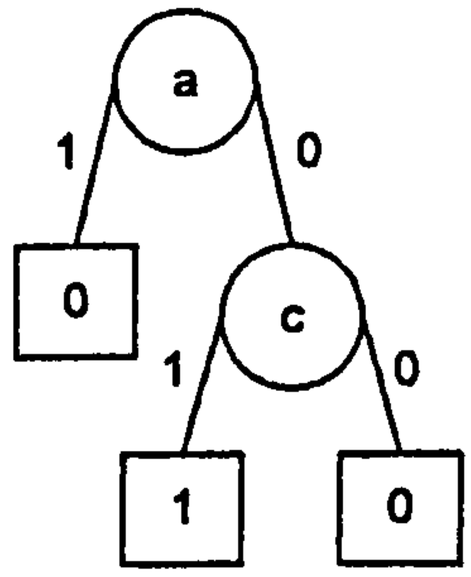


Figure 7.4: Intermediate BDD Obtained for Node F1 from the SFBDD in Figure 7.3

There is one terminal one path through this BDD, $\bar{a}c$, the probability of this path is: $p_a q_c$

Dealing with node F2

$$\begin{aligned} 1 \cdot F4 &= 1 \cdot ite(c,1,0) \\ &= ite(c,1,0) \end{aligned}$$

The resulting BDD is shown in figure 7.5:

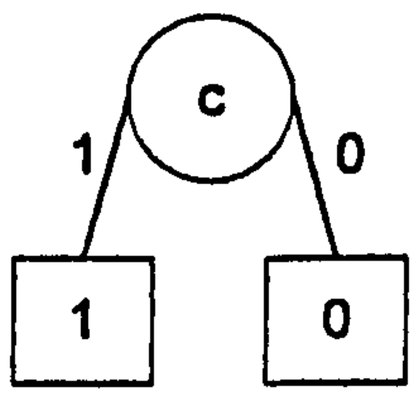


Figure 7.5: Intermediate BDD Obtained for Node F2 from the SFBDD in Figure 7.3

There is one terminal one path through this BDD, c, the probability of this path is q_c.

Dealing with node F3

$$\begin{aligned} F4 \cdot 0 &= 0 \cdot ite(c,1,0) \\ &= 0 \end{aligned}$$

The probability of this is 0.

Dealing with node F4

$$1 \cdot 0 = 0$$

The probability of this is 0.

Table 7.8 Summaries the results obtained for $E[T_{i=1}]$, $E[T_{i=0}]$ and $E[T_{i=-}]$ using the results from tables 7.5, 7.6 and 7.7 and equations (7.44) and (7.45).

Node	Variable	$E[T_{i=1}]$	$E[T_{i=0}]$	$E[T_{i=-}]$
F1	b	$q_a + p_a q_c$	$p_a q_c$	$p_a q_c$
F2	a	q_b	$q_b q_c$	$q_b q_c$
F3	a	0	$p_b q_c$	0
F4	c	$p_a q_b + p_a p_b$	0	0

Table 7.8: Summary of the Results Obtained for $E[T_{i=1}]$, $E[T_{i=0}]$ and $E[T_{i=-}]$

From the results in table 7.8 and equations, (7.42) and (7.43) the failure and repair importance of each component is obtained as follows:

$$G_a^F(q) = q_b - q_b q_c + 0 - 0 \qquad G_a^R(q) = q_b q_c - q_b q_c + p_b q_c - 0$$
$$G_b^F(q) = q_a - p_a q_c + p_a q_c \qquad G_b^R(q) = p_a q_c - p_a q_c$$
$$G_c^F(q) = p_a q_b + p_a p_b \qquad G_c^R(q) = 0 - 0$$

Simplifying:

$$G_a^F(q) = q_b p_c \qquad G_a^R(q) = p_b q_c$$
$$G_b^F(q) = q_a \qquad G_b^R(q) = 0$$
$$G_c^F(q) = p_a \qquad G_c^R(q) = 0$$

7.8.2.1.2 The Consensus Binary Decision Diagram Technique

It is possible to calculate, $E[T_{i=1}]$, $E[T_{i=0}]$ and $E[T_{i=-}]$ directly from the Consensus BDD. The procedure for calculating the repair and failure criticality of a component i from the Consensus BDD is outlined below:

$$E[T_{i=1}] = \sum_{x_i} Pr_{x_i}(\underline{q}) \cdot Po_{x_i}^{1,c}(\underline{q}) \quad (7.46)$$

$$E[T_{i=0}] = \sum_{x_i} Pr_{x_i}(\underline{q}) \cdot Po_{x_i}^{0,c}(\underline{q}) \quad (7.47)$$

$$E[T_{i=-,-}] = \sum_{x_i} Pr_{x_i}(\underline{q}) \cdot Po_{x_i}^{c}(\underline{q}) \quad (7.48)$$

Where:

$Pr_{x_i}(\underline{q})$ is the probability of the path section from the root vertex to node x_i.

$Po_{x_i}^{1,c}(\underline{q})$ is the probability of the path section from the one branch of node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i).

$Po_{x_i}^{0,c}(\underline{q})$ is the probability of the path section from the zero branch of the node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i).

$Po_{xi}^{c}(\underline{q})$ is the probability of the path section from the consensus branch of the node x_i to a terminal 1 node via only one or zero branches of non-terminal nodes (excluding the probability of x_i).

Hence the failure and repair criticality of component I are expressed as follows:

$$G_i^F(\underline{q}) = \sum_{x_i} Pr_{x_i}(\underline{q}) \left[Po_{x_i}^{1,c}(\underline{q}) - Po_{x_i}^{c}(\underline{q}) \right] \quad (7.49)$$

$$G_i^R(\underline{q}) = \sum_{x_i} Pr_{x_i}(\underline{q}) \left[Po_{x_i}^{0,c}(\underline{q}) - Po_{x_i}^{c}(\underline{q}) \right] \quad (7.50)$$

To illustrate how this procedure is applied in practise to calculate both the failure and repair importance of components and also the unconditional failure intensity, consider the Consensus BDD obtained earlier and shown below.

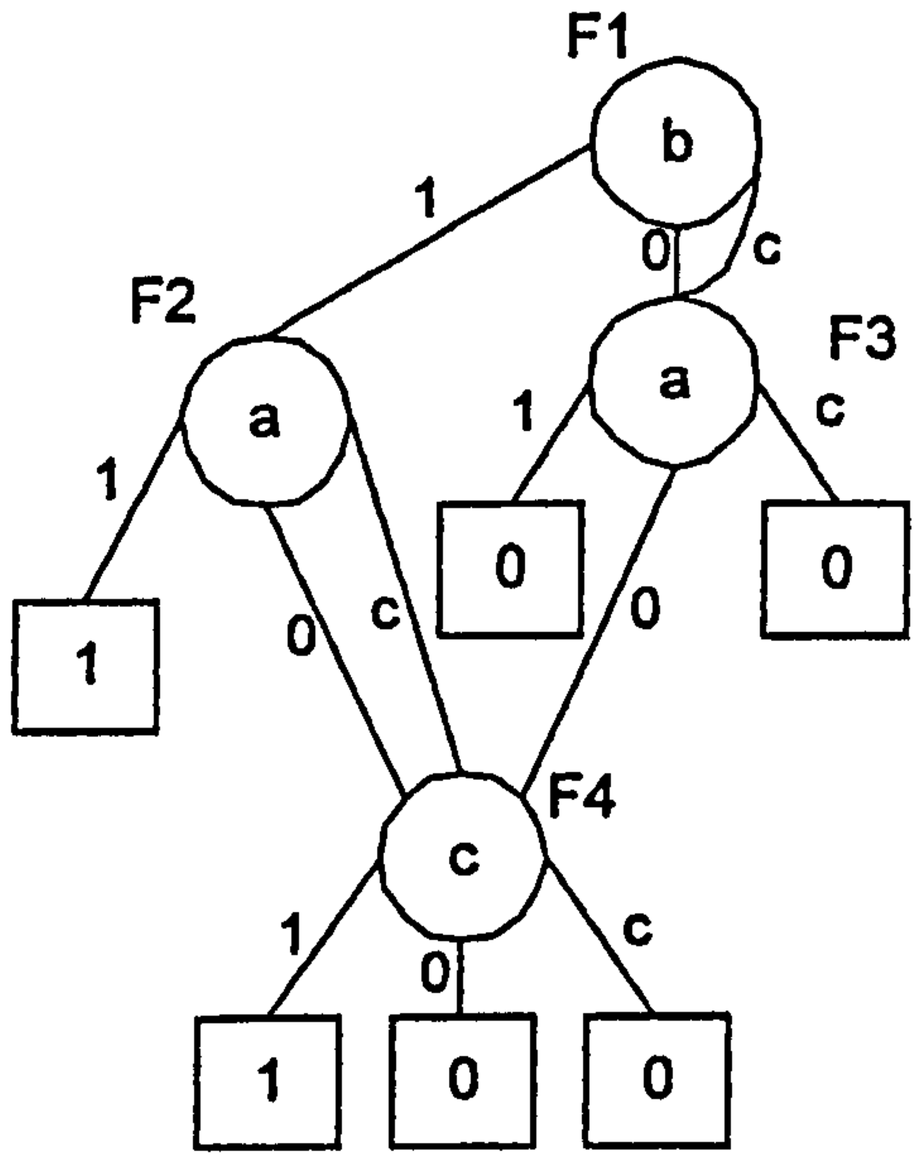


Figure 7.6: Consensus BDD

Firstly the connections between the nodes are recorded in table 7.9.

Node	Variable	1 Branch	0 Branch	Consensus Branch
F1	b	F2	F3	F3
F2	a	1	F4	F4
F3	a	0	F4	0
F4	c	1	0	0

Table 7.9: ite table

The second stage is to calculate each of the terms in equations, (7.49) and (7.50). The first term, $Pr_{x_i}(q)$ is the probability of the path from the root vertex to node x_i, which is calculated by evaluating the probability of the path from the root vertex up to but not including the node x_i. Note any paths passing through a consensus branch of a non-terminal node are excluded. Table 7.10 records $Pr_{x_i}(q)$ for each node in the BDD.

Node	$Pr_{x_i}(\underline{q})$	Comments
F1	1	Root vertex itself
F2	q_b	F2 reached through the 1 branch of F1
F3	p_b	F3 reached through the 0 branch of F1
F4	$p_a q_b + p_a p_b$	F4 reached through the 1 branch of F1 followed by the 0 branch of F2 and through the 0 branch of F1 followed by the zero branch of F3

Table 7.10: $Pr_{x_i}(\underline{q})$ for each Node in the Consensus BDD in Figure 7.6

$Po^{1,c}_{x_i}(\underline{q})$ is calculated by summing the probability of all the paths from the selected node, x_i along the **one** branch to a terminal 1 vertex, excluding the probability of the selected node. Again any paths passing through a consensus branch of a non-terminal node are excluded. Table 7.11 records $Po^{1,c}_{x_i}(\underline{q})$ for each node.

Node	$Po^{1,c}_{x_i}(\underline{q})$	Comments
F1	$q_a + p_a q_c$	1 branch of F1 passes to F2 the 1 branch of F2 passes to a terminal 1 vertex and the 0 branch passes to F4
F2	1	1 branch of F2 passes to a terminal 1 vertex
F3	0	No terminal 1 paths from 1 branch of F3
F4	1	1 branch of F4 passes to a terminal 1 vertex

Table 7.11: $Po^{1,c}_{x_i}(\underline{q})$ for each Node in the Consensus BDD in Figure 7.6

Similarly $Po^{0,c}_{x_i}(\underline{q})$ is calculated by summing the probability of all paths from the selected node, x_i along the **zero** branch to a terminal 1 vertex, excluding the probability of the selected node. Only paths passing through one or zero branches of non-terminal nodes are included in the calculation. Table 7.12 records $Po^{0,c}_{x_i}(\underline{q})$ for each node.

Node	$Po_{x_i}^{0,c}(\underline{q})$	Comments
F1	$p_a q_c$	0 branch of F1passes through the 0 branch of F3 and then the 1 branch of F4
F2	q_c	0 branch of F2 passes through the 1 branch of F4
F3	q_c	0 branch of F3 passes through the 1 branch of F4
F4	0	No terminal 1 paths from the 0 branch of F4

Table 7.12: $Po_{x_i}^{0,c}(\underline{q})$ for each Node in the Consensus BDD in Figure 7.6

Finally $Po_{x_i}^{c}(\underline{q})$ is calculated by summing the probability of all paths from the selected node , x_i, along the **consensus** branch to a terminal 1 vertex, excluding the probability of the selected node. Only paths passing through one or zero branches of non-terminal nodes are included in the calculation. Table 7.13 records $Po_{x_i}^{c}(\underline{q})$ for each node.

Node	$Po_{x_i}^{c}(\underline{q})$	Comments
F1	$p_a q_c$	Consensus branch of F1passes through the 0 branch of F3 and then the 1 branch of F4
F2	q_c	Consensus branch of F2 passes through the 1 branch of node F4
F3	0	No terminal 1 paths from the 0 branch of F3
F4	0	No terminal 1 paths from the 0 branch of F4

Table 7.13: $Po_{x_i}^{c}(\underline{q})$ for each Node in the Consensus BDD in Figure 7.6

The failure and repair importance for each component can be calculated by summing the contributions of nodes of the same component. Thus using equations, (7.49) and (7.50) and the results shown in tables, 7.10, 7.11, 7.12 and 7.13:

$$G_a^F(\underline{q}) = q_b \cdot [1 - q_c] + p_b[0 - 0] \qquad G_a^R(\underline{q}) = q_b[q_c - q_c] + p_b[q_c - 0]$$
$$G_b^F(\underline{q}) = 1 \cdot [q_a + p_a q_c - p_a q_c] \qquad G_b^R(\underline{q}) = 1 \cdot [p_a q_c - p_a q_c]$$
$$G_c^F(\underline{q}) = (p_a q_b + p_a p_b) \cdot [1 - 0] \qquad G_c^R(\underline{q}) = (p_a q_c + p_a p_b)[0 - 0]$$

Simplifying:

$$G_a^F(q) = q_b p_c \qquad G_a^R(q) = p_b q_c$$
$$G_b^F(q) = q_a \qquad G_b^R(q) = 0$$
$$G_c^F(q) = p_a \qquad G_c^R(q) = 0$$

Having calculated the top event probability and Birnbaum's measure of failure and repair importance for each component it is possible to calculate the unconditional failure intensity and the expected number of system failures in a given interval. Then all the quantities required for calculating the Measure of Component failure and repair Criticality and Barlow and Proschan's measure of initiator failure and repair importance are known.

7.8.2.2 Calculating the Measure of Enabler Failure and Repair Importance

It is not possible to calculate the measure of failure and repair enabler importance directly from the SFBDD. This is because the inclusion of NOT logic makes the calculation procedure complex and time consuming, and thus it is not a practical proposition. However, an approximation can be obtained by calculating the coherent measure outlined in chapter six. This will give an indication of the significance of each component in contributing to system failure. The consensus BDD on the other hand lends itself to quantification and a procedure has been developed to calculate the enabler failure and repair importance.

7.8.2.2.1 The Consensus Binary Decision Diagram Technique

To calculate the enabler failure importance of component i when component j actually causes system failure, the following quantities must be calculated, $G_{i,j}(q)$, $G_{i,\overline{j}}(q)$, $G_{M_{i,j}}(q)$ and $G_{M_{i,\overline{j}}}(q)$. The techniques for calculating each of these quantities from the consensus BDD will be outlined in detail. The technique developed for calculating Birnbaum's measure of failure and repair importance is extended.

$G_i^F(\underline{q})$ represents the probability that component i is critical to the system state at time t, such that the failure of component i would cause system failure. $G_{i,j}(\underline{q})$, represents the probability that components i and j are critical to the system state, such that the failure of both i and j would cause system failure. $G_i^F(\underline{q})$, can be represented as follows:

$$G_i^F(\underline{q}) = E[T_{i=1}] - E[T_{i='-'}] \tag{7.51}$$

This notation can be extended to $G_{i,j}(\underline{q})$ as follows:

$$G_{i,j}(\underline{q}) = E[T_{i=1,j=1}] - E[T_{i=1,j='-'}] - E[T_{i='-',j=1}] + E[T_{i='-',j='-'}] \tag{7.52}$$

The first term of equation (7.52), $E[T_{i=1,j=1}]$ represents the probability that component is either failure relevant or irrelevant and component j is either failure relevant or irrelevant. The combinations in this term can be split into four distinct groups; those that represent the probability that:

i) i and j are failure relevant
ii) i is failure relevant and j is irrelevant
iii) i is irrelevant and j is failure relevant
iv) i and j are irrelevant

This can be represented in a more concise form shown below:

$$E[T_{i=1,j=1}] = P((i\,FR)\cdot(j\,FR)) + P((i\,FR)\cdot(j\,IR)) + P((i\,IR)\cdot(j\,FR)) + P((i\,IR)\cdot(j\,IR)) \tag{7.53}$$

Where, FR denotes failure relevant and IR denotes irrelevant.

Similar expressions can be obtained for the second, third and fourth term of equation (7.52). The second term of equation (7.52) represents the probability that component I is either failure relevant or irrelevant and component j is irrelevant.

$$E[T_{i=1,j='-'}] = P((i\,FR)\cdot(j\,IR)) + P((i\,IR)\cdot(j\,IR)) \tag{7.54}$$

Similarly, the third term of equation (7.52) represents the probability that component i is irrelevant and component j is either failure relevant or irrelevant.

$$E[T_{i='-',j=1}] = P((i\,IR)\cdot(j\,FR)) + P((i\,IR)\cdot(j\,IR)) \tag{7.55}$$

Finally, the fourth term of equation (7.52) represents the probability that components i and j are irrelevant.

$$E[T_{i='-',j='-'}] = P((i\,IR)\cdot(j\,IR)) \tag{7.56}$$

Hence from equations (7.53)-(7.56) it can be seen that, $G_{i,j}(\underline{q})$, represents the probability that components i and j are failure relevant to the system state:

$$\begin{aligned} G_{i,j}(\underline{q}) = {} & P((i\,FR)\cdot(j\,FR)) + P((i\,FR)\cdot(j\,IR)) + P((i\,IR)\cdot(j\,FR)) + P((i\,IR)\cdot(j\,IR)) - P((i\,FR)\cdot(j\,IR)) \\ & - P((i\,IR)\cdot(j\,IR)) - P((i\,IR)\cdot(j\,FR)) - P((i\,IR)\cdot(j\,IR)) + P((i\,IR)\cdot(j\,IR)) \end{aligned}$$

$$G_{i,j}(\underline{q}) = P((i\,FR)\cdot(j\,FR)) \tag{7.57}$$

Each of the terms in equation (7.52) can be calculated from just one pass of the consensus BDD. The first term in this equation, $E[T_{i=1,j=1}]$ is calculated by taking the sum of the probabilities of paths through the consensus BDD that pass through the one branch of a node x_i and the one branch of a node x_j, but do not pass through the consensus branch of any nodes. These will be known as paths of type (i).

The second term is calculated by taking the sum of the probabilities of paths through the consensus BDD that pass through the one branch of a node x_i and the consensus branch of a node x_j, but which do not pass through the consensus branch of any other nodes. These will be known as paths of type (ii).

The third and fourth terms in equation (7.52) are calculated by taking the sum the probabilities of paths of type (iii) and (iv) respectively. Where, paths of type (iii) are paths through the consensus BDD that pass through the consensus branch of a node x_i and the one branch of a node x_j, but which do not pass through the consensus branch of any others nodes. Finally paths of type (iv) are paths through the consensus BDD that pass through the consensus branch of a node x_i and the consensus branch of a node x_j, but do not pass through the consensus branch of any other nodes. The probability of each path type is calculated from the consensus BDD as follows:

The probability of paths of type (i) are calculated as follows:

$$E\left[T_{i=1,j=1}\right] = P(\text{paths of type (i)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j} \tag{7.58}$$

Where:

pr_{x_i} is the probability preceding node x_i.

$po^{1}_{x_i - x_j}$ is the probability from the one branch of node x_i to the node x_j, excluding the probability of node x_i.

$po^{1}_{x_j}$ is the probability from the one branch of node x_j to a terminal one node.

And, any paths passing through the consensus branch of a node are excluded from the calculation.

The probability of paths of type (ii) is calculated as follows:

$$E\left[T_{i=1,j=\text{'-'}}\right] = P(\text{paths of type (ii)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j} \tag{7.59}$$

Where:

$po^{c}_{x_j}$ is the probability from the consensus branch of node x_j to a terminal one node.

And, any paths passing through the consensus branch of a node not representing component j are excluded from the calculation.

The probability of paths of type (iii) is calculated using equation (7.60):

$$E\left[T_{i=\text{'-'},j=1}\right] = P(\text{paths of type (iii)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{1}_{x_j} \tag{7.60}$$

Where:

$po^{c}_{x_i - x_j}$ is the probability from the consensus branch of node x_i to the node x_j excluding the probability of x_i.

And, any paths passing through the consensus branch of a node not representing component i are excluded from the calculation.

Finally the probability of paths of type (iv) can be calculated using equation (7.61):

$$E\left[T_{i=\text{'-'},j=\text{'-'}}\right] = P(\text{paths of type (iv)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j} \tag{7.61}$$

Where, any paths passing through the consensus branch of a node not representing component i or j are excluded from the calculation.

Thus the following expression is obtained for $G_{i,j}(\underline{q})$:

$$G_{i,j}(\underline{q}) = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(i)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j} - \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(ii)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j}$$
$$- \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(iii)}}} pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{1}_{x_j} + \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(iv)}}} pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j} \qquad (7.62)$$

It is possible to derive a similar result for calculating $G_{i,\overline{j}}(\underline{q})$ from the consensus BDD. $G_{i,\overline{j}}(\underline{q})$ represents the probability that component i is failure critical to the system state and component j is repair critical to the system state. The notation introduced for calculating, $G_i^F(\underline{q})$ and $G_i^R(\underline{q})$ using the BDD technique can be used to form the following expression for $G_{i,\overline{j}}(\underline{q})$.

$$G_{i,\overline{j}}(\underline{q}) = E[T_{i=1,j=0}] - E[T_{i=1,j='-'}] - E[T_{i='-',j=0}] + E[T_{i='-',j='-'}] \qquad (7.63)$$

The first term in equation (7.63) $E[T_{i=1,j=0}]$ represents the probability that component i is failure relevant or irrelevant and component j is repair relevant or irrelevant. This probability is calculated from the consensus BDD by summing the probabilities of those paths through the consensus BDD that pass through the one branch of a node x_i and the zero branch of a node x_j, and does not pass through the consensus branch of any node. These paths will be known as paths of type (v). Hence the following expression is obtained for calculating $E[T_{i=1,j=0}]$ from the consensus BDD:

$$E[T_{i=1,j=0}] = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(v)}}} P(\text{Paths of type (v)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{0}_{x_j} \qquad (7.64)$$

Where:

$po^{0}_{x_j}$ is the probability from the zero branch of node x_j to a terminal one node.

Similar expressions are obtained for calculate the second, third and fourth term in equation (7.63). The second term represents the probability that component i is failure relevant or irrelevant and component j is irrelevant. This term is calculated by summing the probabilities of the paths of type (vi) through the consensus BDD.

$$E\left[T_{i=1,j='-'}\right] = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(vi)}}} P(\text{Paths of type (vi)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j} \qquad (7.65)$$

Where, a path of type (vi) is a path that passes through the one branch of a node x_i and the consensus branch of a node x_j, but which does not pass through the consensus branch of any other node.

The third term of equation (7.63) represents the probability that component i is irrelevant and component j is either repair relevant or irrelevant. It is calculated from the consensus BDD by taking the sum of the probabilities of all those paths passing through the consensus branch of a node x_i and the zero branch of a node x_j, but which do not pass through the consensus branch of any other node. These paths will be called type (vii):

$$E\left[T_{i='-',j=0}\right] = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(vi)}}} P(\text{Paths of type (vii)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{0}_{x_j} \qquad (7.66)$$

Where:

$po^{c}_{x_i - x_j}$ is the probability from the consensus branch of node x_j to a terminal one node.

Finally the fourth term of equation (7.63) denotes the probability that components i and j are irrelevant to the system state. This probability has already been calculated above:

$$E\left[T_{i='-',j='-'}\right] = P(\text{paths of type (iv)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j} \qquad (7.67)$$

The following expression is obtained for calculating $G_{i,\bar{j}}(q)$ from the consensus BDD.

$$G_{i,\bar{j}}(q) = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(v)}}} pr_{x_i} \cdot po^{1}_{x_i} \cdot po^{0}_{x_i} - \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(vi)}}} pr_{x_i} \cdot po^{1}_{x_i} \cdot po^{c}_{x_i} - \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(vii)}}} pr_{x_i} \cdot po^{c}_{x_i} \cdot po^{0}_{x_i} + \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(iv)}}} pr_{x_i} \cdot po^{c}_{x_i} \cdot po^{c}_{x_i} \qquad (7.68)$$

To calculate the correction terms, $G_{M_{i,j}}(\underline{q})$ and $G_{M_{i,\overline{j}}}(\underline{q})$ two modified system unavailability functions, $Q_{M_{i,j}}(\underline{q})$ and $G_{M_{i,\overline{j}}}(\underline{q})$ are formed. $Q_{M_{i,j}}(\underline{q})$ is formed by considering only a partial list of the prime implicant sets. All those prime implicant sets involving both i and j are ignored. Since the consensus BDD obtained for the system encodes the full list of the structure function of the fault tree it does not encode the modified system unavailability function. Thus to calculate $G_{M_{i,j}}(\underline{q})$, a new consensus BDD which only encodes the modified structure function must be computed. Both $Q_{M_{i,j}}(\underline{q})$ and $G_{M_{i,\overline{j}}}(\underline{q})$ can be calculated directly from this tree structure.

The **ite** procedure for calculating the consensus BDD for a non-coherent fault tree is used to calculate the modified consensus BDD. In order to use this procedure, a fault tree must be developed for the modified system. The fault tree structure will always take the same form. The top gate will be an OR gate with all gate inputs, the number of gate inputs will be equal to the number of prime implicant sets being considered in the modified system. Each gate input to the top gate will be an AND gate, with it inputs being the basic events of each prime implicant set.

Once the fault tree structure has been obtained the **ite** procedure is applied to compute the modified consensus BDD. It is then possible to calculate $G_{M_{i,j}}(\underline{q})$ using the method outlined previously for calculating $G_{i,j}(\underline{q})$.

$$G_{M_{i,j}}(\underline{q}) = \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(i)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j} - \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(ii)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j} - \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(iii)}}} pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{1}_{x_j} + \sum_{\substack{\text{paths}\\ \text{of type}\\ \text{(iv)}}} pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j} \tag{7.69}$$

The same procedure is then employed to calculate $G_{M_{i,\overline{j}}}(\underline{q})$, except this time all those prime implicant sets involving both i and $\overline{j}$ are ignored and the formula for calculating $G_{i,\overline{j}}(\underline{q})$ is used to calculate $G_{M_{i,\overline{j}}}(\underline{q})$.

$$G_{M_{i,\bar{j}}}(q) = \sum_{\substack{\text{paths} \\ \text{of type} \\ \text{(v)}}} pr_{x_i} \cdot po^1_{x_i} \cdot po^0_{x_i} - \sum_{\substack{\text{paths} \\ \text{of type} \\ \text{(vi)}}} pr_{x_i} \cdot po^1_{x_i} \cdot po^c_{x_i} - \sum_{\substack{\text{paths} \\ \text{of type} \\ \text{(vii)}}} pr_{x_i} \cdot po^c_{x_i} \cdot po^0_{x_i} + \sum_{\substack{\text{paths} \\ \text{of type} \\ \text{(iv)}}} pr_{x_i} \cdot po^c_{x_i} \cdot po^c_{x_i} \quad (7.70)$$

The technique for calculating the modified enabler measure of importance will now be illustrated by means of a worked example. Consider the fault tree in figure 7.7.

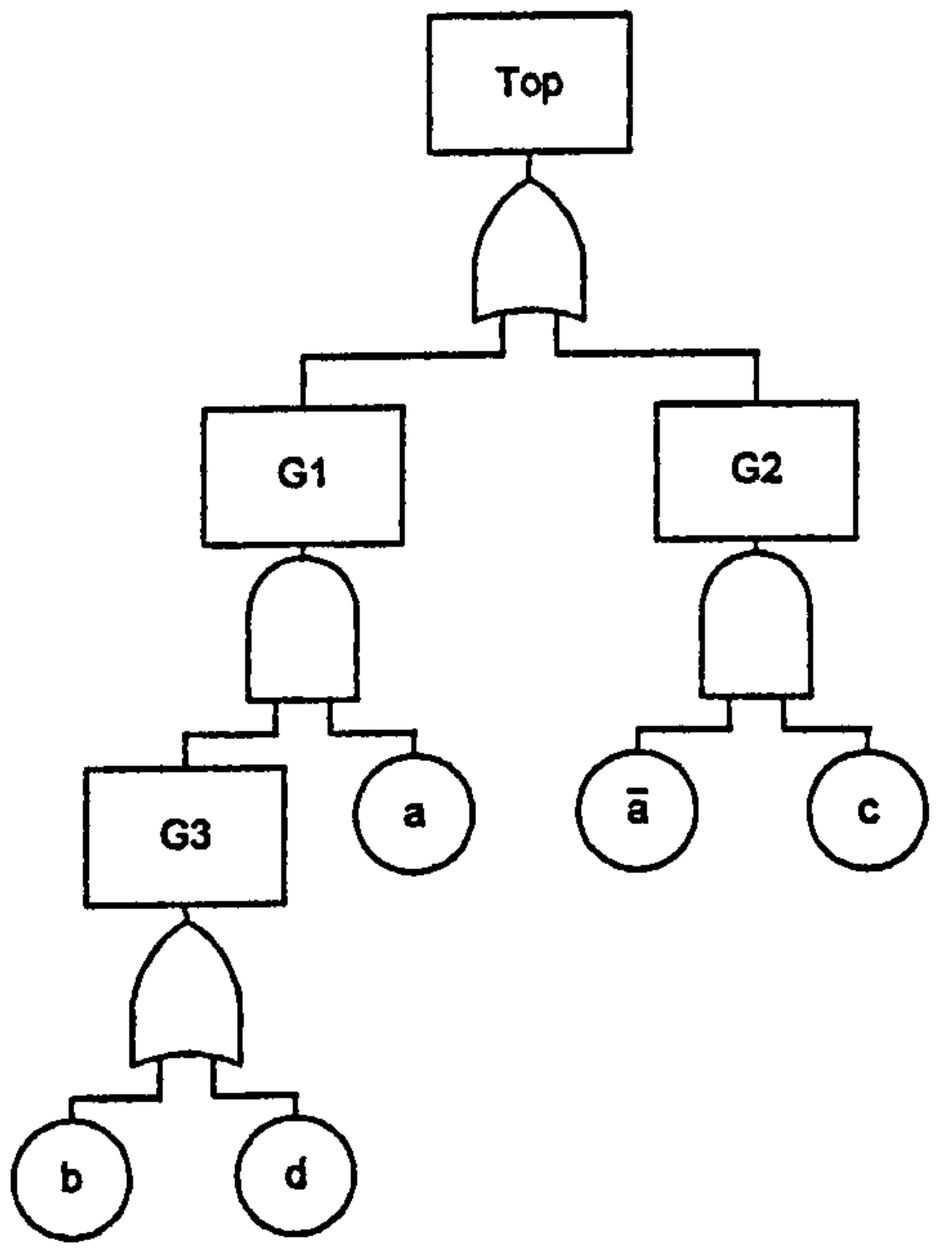


Figure 7.7: Non-coherent Fault Tree Diagram

Five prime implicant sets can be identified from the fault tree in figure 7.6:

$\{ab\}, \{\bar{a}\,c\}, \{bc\}, \{ad\}, \{cd\}$

Firstly the consensus BDD for the fault tree must be computed, if the basic events are assigned the ordering, $a < b < c < d$ the following consensus BDD is obtained:

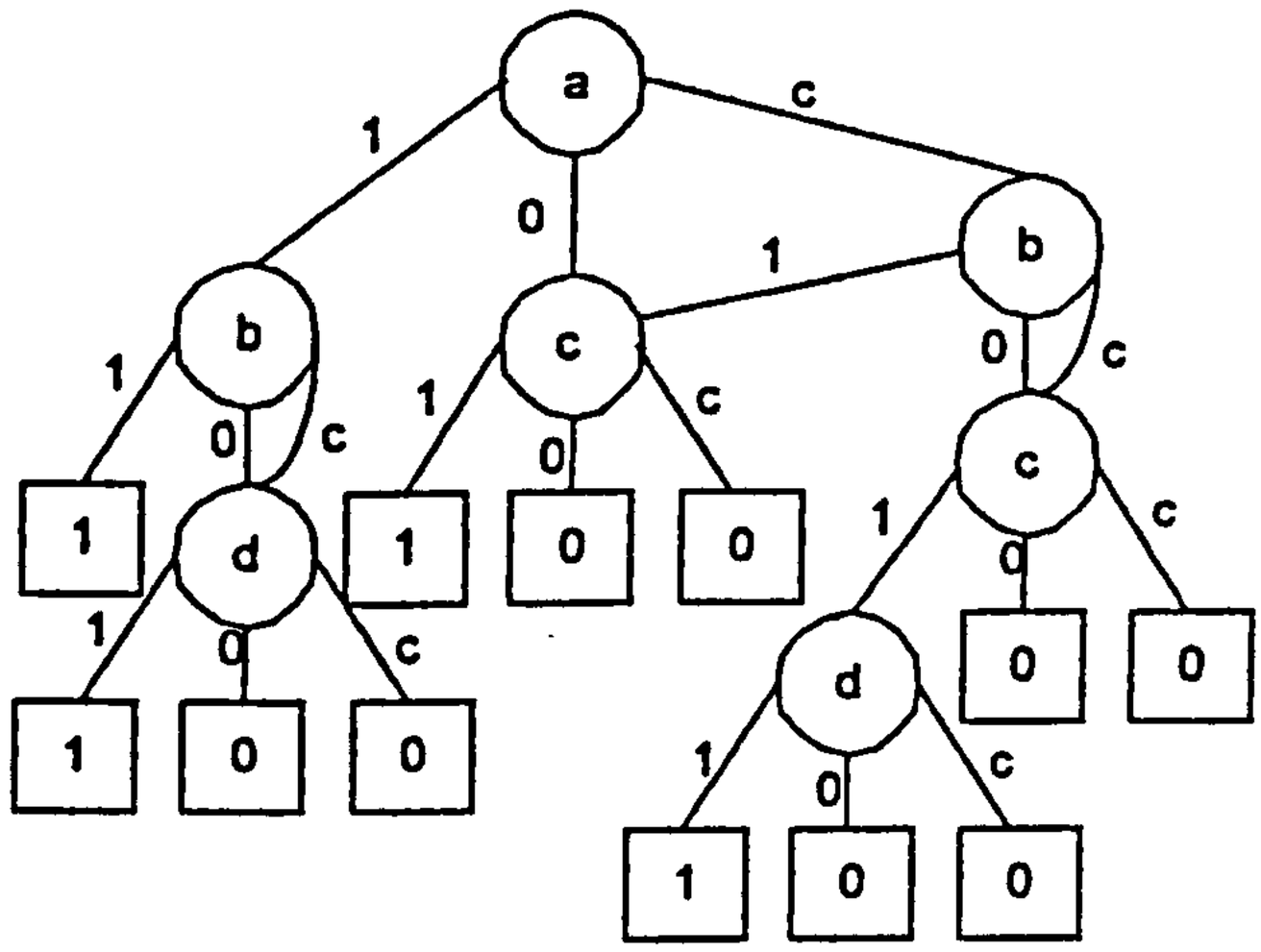
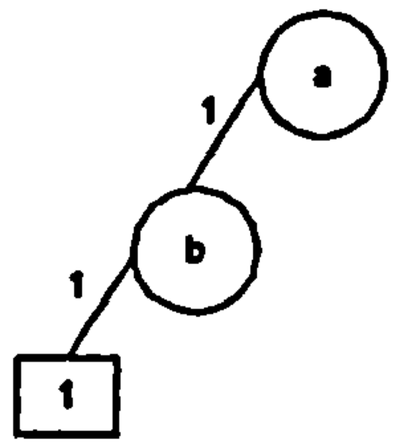


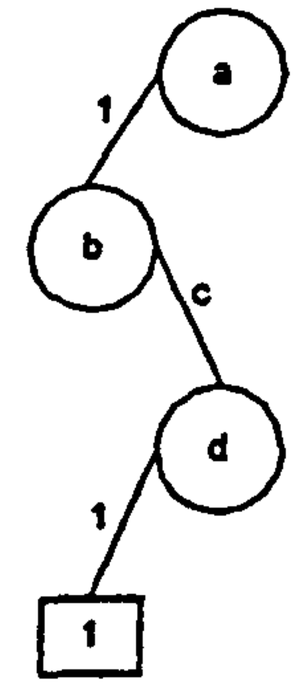
Figure7.8: Consensus BDD Obtained for the Non-coherent Fault Tree in Figure 7.7

The failure enabler importance of component a, when component b acts as an initiator will be calculated. Note, that since $\bar{b}$ does not appear in the prime implicant sets the repair of component b cannot act as an initiator, hence it is not necessary to calculate, either, $G_{a,\bar{b}}(q)$ or $G_{M_{a,\bar{b}}}(q)$.

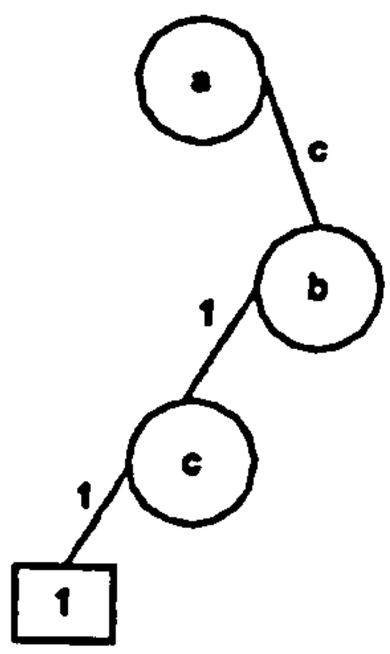
Firstly $G_{a,b}(q)$ must be calculated, from the consensus BDD in figure 7.8, thus paths which match one of the four types outlined above are identified from this BDD. From the consensus BDD in figure 7.8 it is clear that there is one path of type (i), one path of type (ii) one of type (iii) and one of type (iv) for components a and b. Each path is shown below:



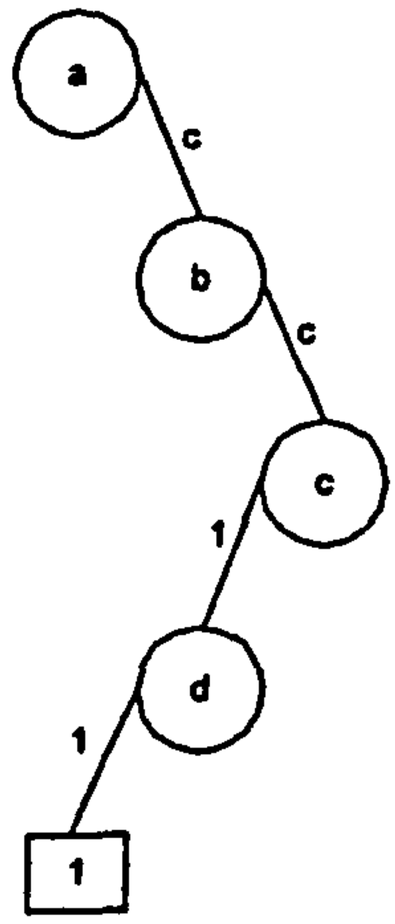
Path of type (i)



Path of type (ii)



Path of type (iii)



Path of type (iv)

Figure 7.9: Paths of Type (i), (ii) and (iii) through the Consensus BDD in Figure 7.8 for Components a and b

Thus from equations (7.58-7.61) the following probabilities are obtained for the four paths shown in figure 7.9:

$$P(\text{path of type (i)}) = \sum_{\substack{\text{paths of} \\ \text{type (i)}}} pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{1}_{x_j}$$

$$= 1 \cdot 1$$

$$= 1$$

$$P(\text{path of type (ii)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j}$$

$$= 1 \cdot 1 \cdot q_d$$

$$= q_d$$

$$P(\text{path of type (iii)}) = pr_{x_i} \cdot po^{c}_{x_q - x_j} \cdot po^{1}_{x_j}$$
$$= 1 \cdot 1 \cdot q_c$$
$$= q_c$$

$$P(\text{paths of type (iv)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j}$$
$$= 1 \cdot 1 \cdot q_c q_d$$
$$= q_c q_d$$

From equation (7.62) the following result is obtained for $G_{a,b}(\underline{q})$:

$$G_{a,b}(\underline{q}) = 1 - q_c - q_d + q_c q_d$$

To calculate the correction term all of the prime implicant sets except {ab} are considered. The fault tree shown in figure 7.10 is constructed for this system.

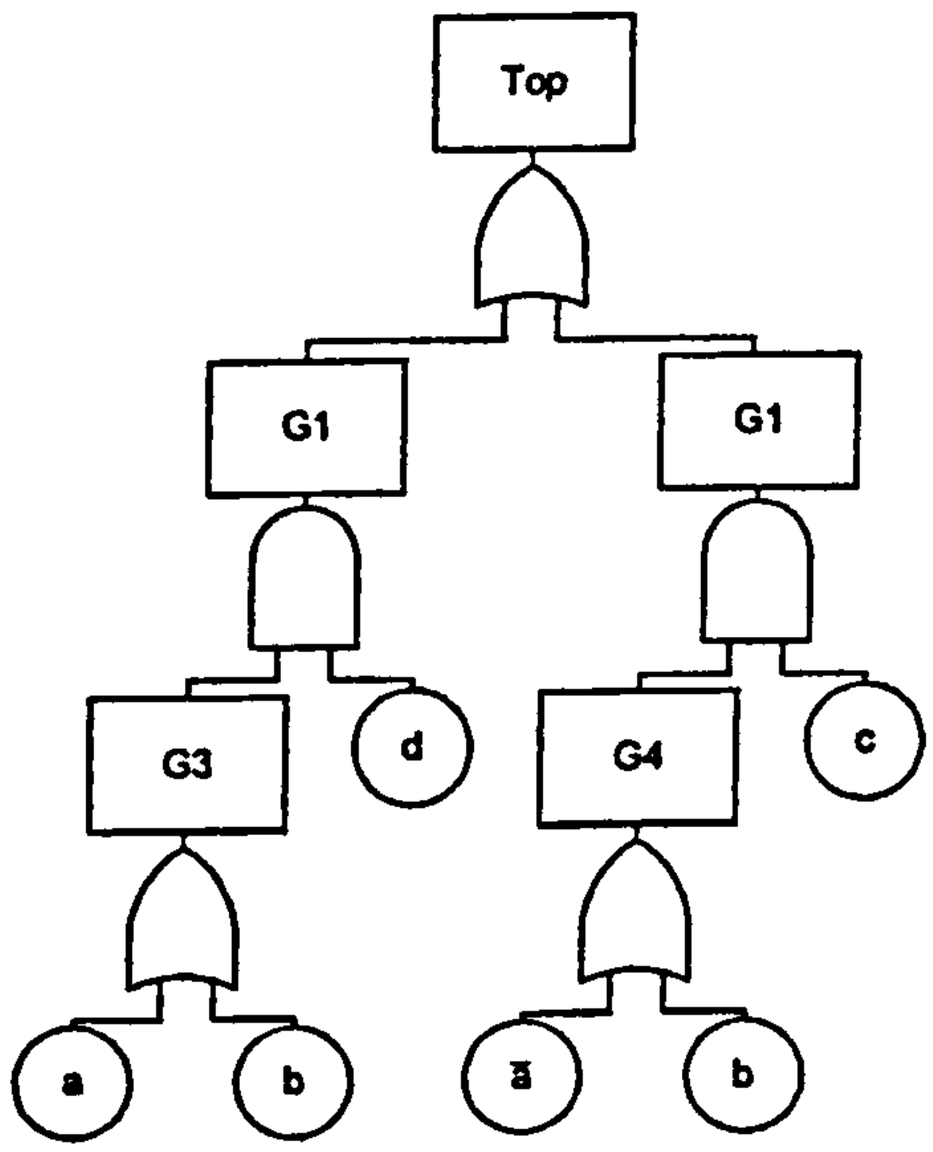


Figure 7.10: Modified Fault Tree Diagram

The consensus BDD shown in figure 7.11 is obtained when the basic events are assigned the ordering: $a < b < c < d$

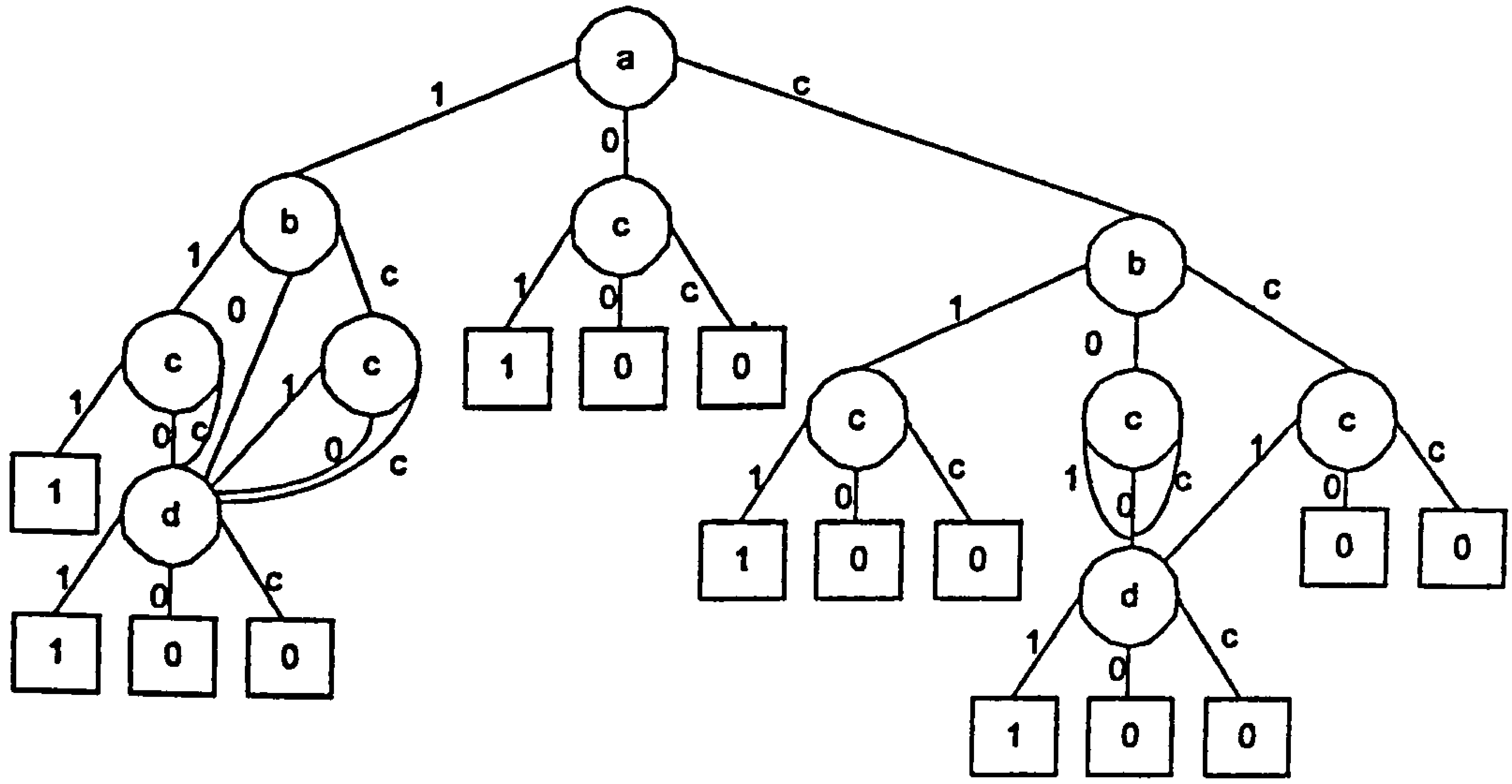


Figure 7.11: Modified Consensus BDD

To calculate $G_{M_{a,b}}(q)$ the paths of type (i), (ii), (iii) and (iv) are identified from the modified consensus BDD in figure 7.11. Two paths of type (i), two paths of type (ii), one path of type (iii) and one path of type (iv) are identified from this BDD, each path is shown in figure 7.12.

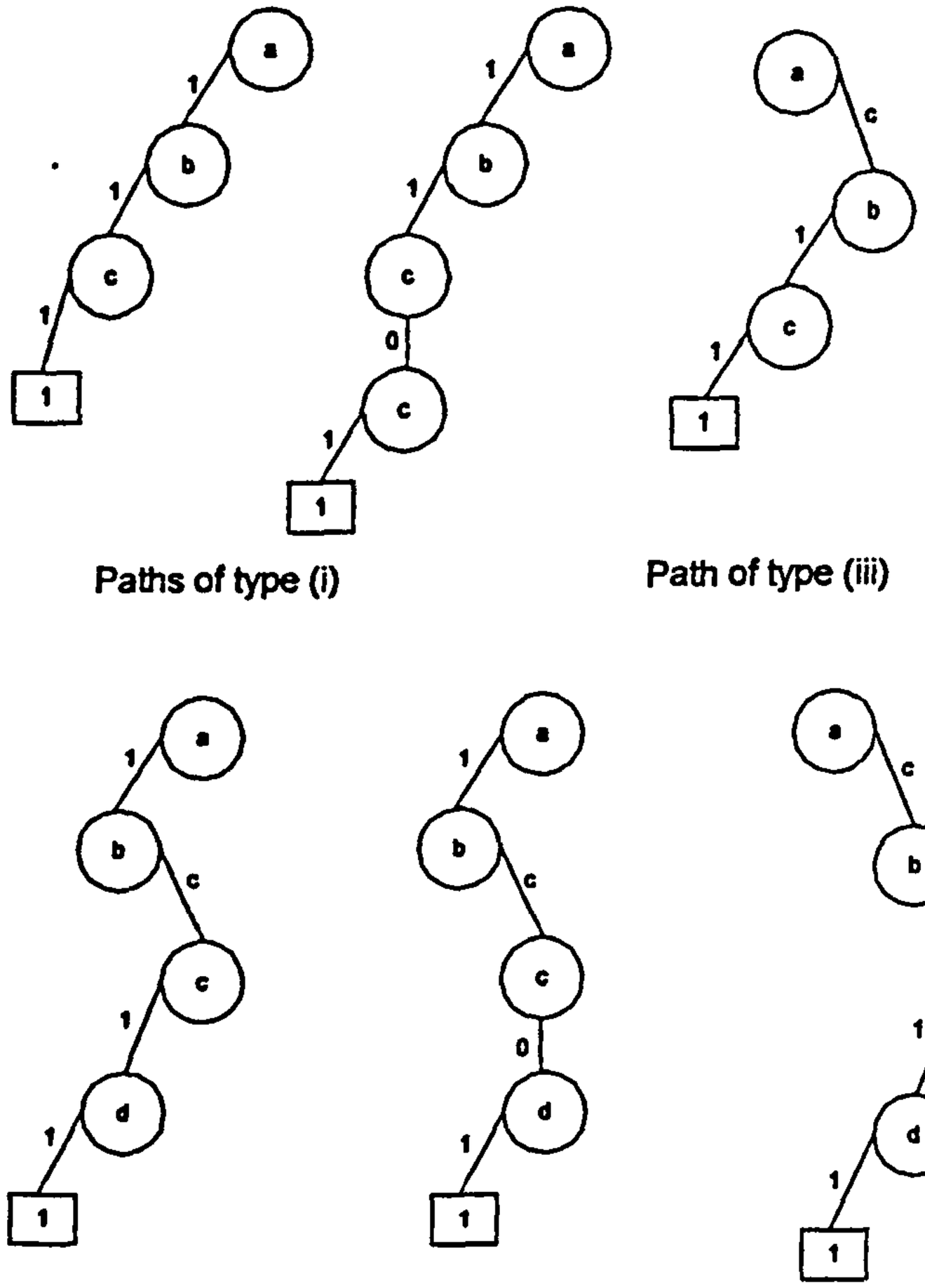


Figure 7.12: Path Types (i)-(iv) Identified from the Modified Consensus BDD in Figure 7.11

Thus from equations (7.58-7.61) the following probabilities are obtained for the six paths shown in figure 7.12:

$$P(\text{paths of type (i)}) = \sum_{\substack{\text{paths of} \\ \text{type (i)}}} pr_{x_i} \cdot po^1_{x_i - x_j} \cdot po^1_{x_j}$$

$$= 1 \cdot 1 \cdot q_c + 1 \cdot 1 \cdot p_c \cdot q_d$$

$$= q_c + p_c q_d$$

$$P(\text{path of type (ii)}) = pr_{x_i} \cdot po^{1}_{x_i - x_j} \cdot po^{c}_{x_j}$$
$$= 1 \cdot 1 \cdot q_c \cdot q_d + 1 \cdot 1 \cdot p_c \cdot q_d$$
$$= q_c q_d + p_c q_d$$

$$P(\text{path of type (iii)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{1}_{x_j}$$
$$= 1 \cdot 1 \cdot q_c$$
$$= q_c$$

$$P(\text{paths of type (iv)}) = pr_{x_i} \cdot po^{c}_{x_i - x_j} \cdot po^{c}_{x_j}$$
$$= 1 \cdot 1 \cdot q_c q_d$$
$$= q_c q_d$$

From equation (7.69) the following result is obtained for $G_{M_{a,b}}(\underline{q})$:

$$G_{M_{a,b}}(\underline{q}) = 0$$

Thus from equation (7.33) the following expression is obtained for the enabler failure importance of component a when component b causes system failure.

$$I^{F}_{E_{a,b}} = \frac{\int_0^t \left[\left[G_{a,b}(\underline{q}) - G_{M_{a,b}}(\underline{q}) \right] q_a w_b + \left[G_{a,\bar{b}}(\underline{q}) - G_{M_{a,\bar{b}}}(\underline{q}) \right] q_a v_b \right] du}{W_{SYS}(0,t)}$$
$$= \frac{\int_0^t \left[1 - q_c - q_d - q_c q_d \right] q_a w_b du}{W_{SYS}(0,t)}$$

7.8.2.3 Calculating the Measures of Prime implicant set Importance

A procedure for calculating the Extended Fussell-Vesely measure of prime implicant set importance exactly using either the consensus BDD or a combination of the SFBDD and the meta-products BDD is outlined in section 7.8.3.1. A procedure for calculating an approximation of the measure of frequency importance of a prime implicant set is also proposed in section 7.8.3.2.

7.8.2.3.1 Calculating The Extended Fussell-Vesely Measure of Prime Implicant Set Importance

It is not possible to calculate the extended Fussell-Vesely Measure of Prime Implicant set importance directly from the SFBDD or the consensus BDD. This measure requires knowledge of the prime implicant sets of the fault tree, hence, either the minimised consensus BDD or the meta-products BDD must be computed. Then, either the consensus BDD or the meta-products BDD is tracked to identify each prime implicant set. The probability of this implicant set is calculated and then divided by the value obtained for the system unavailability.

7.8.2.3.2 Calculating The Measure of Prime Implicant Set Frequency Importance

It is not possible to calculate the frequency importance of a prime implicant set from either the SFBDD or the consensus BDD exactly. This is because the inclusion of NOT logic makes the calculation of this measure using the BDD technique complex and time-consuming. Thus the approximation for minimal cut set importance outlined in chapter six can be extended for use with prime implicant sets. The frequency of the prime implicant set, ε_i^n is calculated:

$$I_F\left(\varepsilon_i^n\right) \approx \int_0^t \sum_{k=1}^{n} \prod_{\substack{j=1 \\ j \neq k}}^{n} q^{\alpha j} z^{\alpha k} du \qquad (7.71)$$

Knowledge of the prime implicant sets, the failure probabilities and failure and repair frequencies of each system component are essential to calculate this measure.

7.9 Importance Analysis a Worked Example

Chapter six highlighted that it is essential to consider the aims of the sensitivity analysis so that appropriate measures of importance can be chosen to analyse the importance of the components and the minimal cut sets of the system. The analyst must also have a clear understanding of each of the measures used, to ensure that the results obtained are interpreted correctly. To illustrate this the simple fault tree given in figure 7.2 will be analysed and the results will be interpreted.

Table 7.14 summarises the conditional failure rate, and the unconditional failure and repair rates for each component. These component parameters are used to quantify the expressions obtained for the five different measures of component importance and the extended Fussell-Vesely measure of cut set and the frequency measure of prime implicant set importance in section 7.8.1. The results are given in tables 7.15 and 7.16.

Component	Failure Rate $\lambda(t)$	Failure Probability $q(t)$	Failure Intensity $w(t)$	Repair Intensity $v(t)$
a	1.0×10^{-3}	1.0×10^{-2}	9.9×10^{-4}	6.7×10^{-3}
b	1.0×10^{-4}	2.0×10^{-4}	9.998×10^{-5}	8.9×10^{-4}
c	5.0×10^{-6}	2.5×10^{-5}	4.99875×10^{-4}	4.5×10^{-4}

Table 7.14 Summary of Component Parameters

Component / Measure	a	$\bar{a}$	b	c	Component Ranked 1st
Birnbaum	2.0×10^{-4}	2.45×10^{-5}	1.0×10^{-2}	0.99	c
Criticality	7.5×10^{-2}	0.925	7.5×10^{-2}	0.925	$\bar{a}$ and c
Fussell-Vesely	7.5×10^{-2}	0.925	7.5×10^{-2}	0.925	$\bar{a}$ and c
Initiator	3.99×10^{-4}	3.37×10^{-4}	2.01×10^{-3}	0.997	c
Enabler	2.01×10^{-3}	0.996	3.99×10^{-4}	3.37×10^{-4}	$\bar{a}$

Table 7.15: Results for the Various Measures of Component Importance

Prime Implicant Set	Fussell-Vesely's Importance	Frequency Importance
ab	7.5×10^{-2}	2.4×10^{-3}
$\bar{a}c$	0.925	0.9974
bc	1.87×10^{-4}	0

Table 7.16: Results for the Measures of Prime Implicant Set Importance

The five measures of component failure and repair importance were calculated for each component and the highest ranked component is recorded in the far right column of table 7.15. Notice, as in chapter six, that it is not always the same component that is ranked as "most important" for the different measures, i.e. Birnbaum's measure ranks the failure of component c as the most important, whereas the repair of component a is ranked highest for the enabler measure of importance. To interpret the results it is important to understand the definition of each measure.

From the results for Birnbaum's measure of importance it can be seen that the system is most likely to be in a working yet critical state for component c. Thus it could be concluded that if the system performance is considered inadequate extra resources should be allocated to reduce the existence of the necessary and sufficient conditions that make component c failure critical to the system state. Thus the reliability of b should be improved. It could also be concluded that a safety system should be incorporated to fail component a if component c is in a failed state. This would prevent prime implicant set $\bar{a}c$ causing system failure.

The results from both the Measure of Component failure and repair Criticality and Fussell-Vesely's measure of failure and repair importance are identical. The failure of component c and the repair of component a are ranked highest. Thus the previous conclusions are confirmed, but they can be further extended. If resources were to be allocated to improve the system availability efforts should be made to:

- Improve the reliability of component c.
- Reduce the existence of the necessary and sufficient conditions for component c to be failure critical
- Reduce the existence of the necessary and sufficient conditions for component a to be repair critical.
- Implement a safety system to fail component a if component c is in a failed state.

The initiator measure of importance indicates that it is the failure of component c that is most likely to actually *cause* the system to go from a working to a failed state. By considering the results for the enabler importance it is clear that the repair of component a is most likely to enable another component (more than likely component c) to act as an initiator causing system failure. From this it is clear that the most likely cause of system

failure is the existence of minimal cut set $\bar{a}c$ where the working state of component a acts as an enabler and the failure of component c acts as the initiator. This conclusion is confirmed by the results for both the extended Fussell-Vesely measure of prime implicant set importance and the frequency measure of prime implicant set importance, recorded in table 7.16.

If resources are to be allocated to improve the system efforts should be concentrated on improving the reliability of component c, and implementing a safety system to prevent the existence of the prime implicant set $\bar{a}c$. The failure of components a and b are least likely to cause or contribute to system failure, thus do not need attention at present.

7.10 Summary

Until now importance analysis of non-coherent systems has been extremely limited. If the measures developed for the analysis of coherent systems are used to analyse non-coherent systems the results obtained are inaccurate and misleading. This chapter has derived an extension of Birnbaum's measure of component reliability importance by considering both the failure and repair criticality of each component. This extension was then employed to extend the measure of component criticality and Barlow and Proschan's measure of initiator importance.

Extensions for the modified measure of enabler importance and the two measures of cut set importance introduced in chapter six have also been derived, enabling accurate importance analysis of non-coherent systems.

Conventional Fault Tree Analysis techniques for calculating these measures are inefficient. Hence, procedures have been developed for calculating the majority of these measures using either the SFBDD or the consensus BDD. These calculation procedures are extremely efficient in comparison to the conventional FTA techniques, eliminating the need for approximations during quantification.

Finally a worked example has been used to highlight the need to (i) choose the importance measures used carefully according to the aims of the analysis and (ii) interpret the results obtained correctly in order to draw meaningful conclusions from the analysis.

Chapter 8: Culling Techniques for Coherent Fault Tree Analysis

8.1 Introduction

With conventional techniques it is not always possible to identify a full list of minimal cut sets or to quantify the system exactly, especially for large fault trees with many repeated events. Even the most powerful computers cannot cope with the intensive analysis in a realistic time. Although the Binary Decision Diagram technique enables more efficient analysis, it can still prove too intensive for extremely large fault trees.

Under such circumstances it is necessary to analyse the fault tree approximately, rather than not at all. Culling techniques have been developed for conventional Fault Tree Analysis techniques, and more recently Rauzy developed a technique to cull the SFBDD in order to enable partial qualitative and quantitative analysis of coherent fault trees [4].

If only a partial list of minimal cut sets can be identified, then it would be desirable to identify those minimal cut sets that are most likely to contribute to system failure. Given that lower order minimal cut sets are generally more likely to contribute to system failure it intuitively makes sense to cull any minimal cut sets above a given order, k_{MAX}.

Whilst generally the lower the order of a minimal cut set the more likely it is to contribute to system failure. There are cases when some higher order minimal cut sets are more significant than those of a lower order. Thus if the minimal cut sets are culled according to their order, some significant combinations may be ignored. An alternative culling technique that can be used to identify the most significant minimal cut sets is the probabilistic or frequency technique, which culls any combinations with a probabilistic value or frequency below a pre-set limit, p_{MIN}, w_{MIN}.

Techniques have been developed to enable the minimal cut sets to be culled according to either their order or probability of existence for both conventional FTA methods and the BDD method.

The culling techniques for conventional FTA are concerned with identifying a partial list of minimal cut sets, from which approximations can be obtained for the system parameters including the system unavailability, the unconditional failure intensity and various measures of importance. These culling techniques can be used in conjunction with the conventional top-down and bottom-up methods for identifying the minimal cut sets of the fault tree.

The culling techniques for the BDD method are concerned with computing a culled SFBDD from which a partial list of minimal cut sets can be identified, and partial quantification can be performed. This chapter will consider these culling techniques in detail and illustrate how they are implemented by means of worked examples.

8.2 Culling Techniques for Conventional Fault Tree Analysis Methods

The three culling techniques developed for use with the conventional FTA methods are employed to obtain a culled Boolean expression for the top event, from which a partial list of minimal cut sets can be identified and approximations can be obtained during quantification.

The first culling technique produces a partial list of minimal cut sets according to a pre-set order. For example if the pre-set order of culling is four all minimal cut sets of order four or below are identified, and any minimal cut sets exceeding this order are eliminated as they are encountered in the development of the Boolean expression.

The second culling technique also produces a partial list of minimal cut sets, but it does this according to a pre-set probabilistic value, p_{MIN}. For example, if $p_{MIN}=1\times10^{-3}$ only those minimal cut sets with a probabilistic value of 1×10^{-3} and above are identified, any minimal cut sets that have a probabilistic value below 1×10^{-3} are disregarded as they are encountered. This culling technique can only be employed if the component failure probabilities are known.

The final culling technique again produces a partial list of minimal cut sets, but this time according to a pre-set frequency, f_{MIN}. This technique is similar to the probabilistic culling technique.

8.2.1 Culling Minimal Cut Sets Above a Given Order

The first stage of this culling technique is to set the order at which the minimal cut sets will be culled. Then the top-down or bottom-up approach is used to identify a partial list of minimal cut sets according to this pre-set culling order. At each stage of developing the Boolean expression for the top event, combinations that exceed the culling order are eliminated. Hence the final expression obtained for the top event only contains the minimal cut sets up to and including the pre-set culling order. To illustrate this technique consider the fault tree given in figure 8.1. The minimal cut sets of order three and below will be identified.

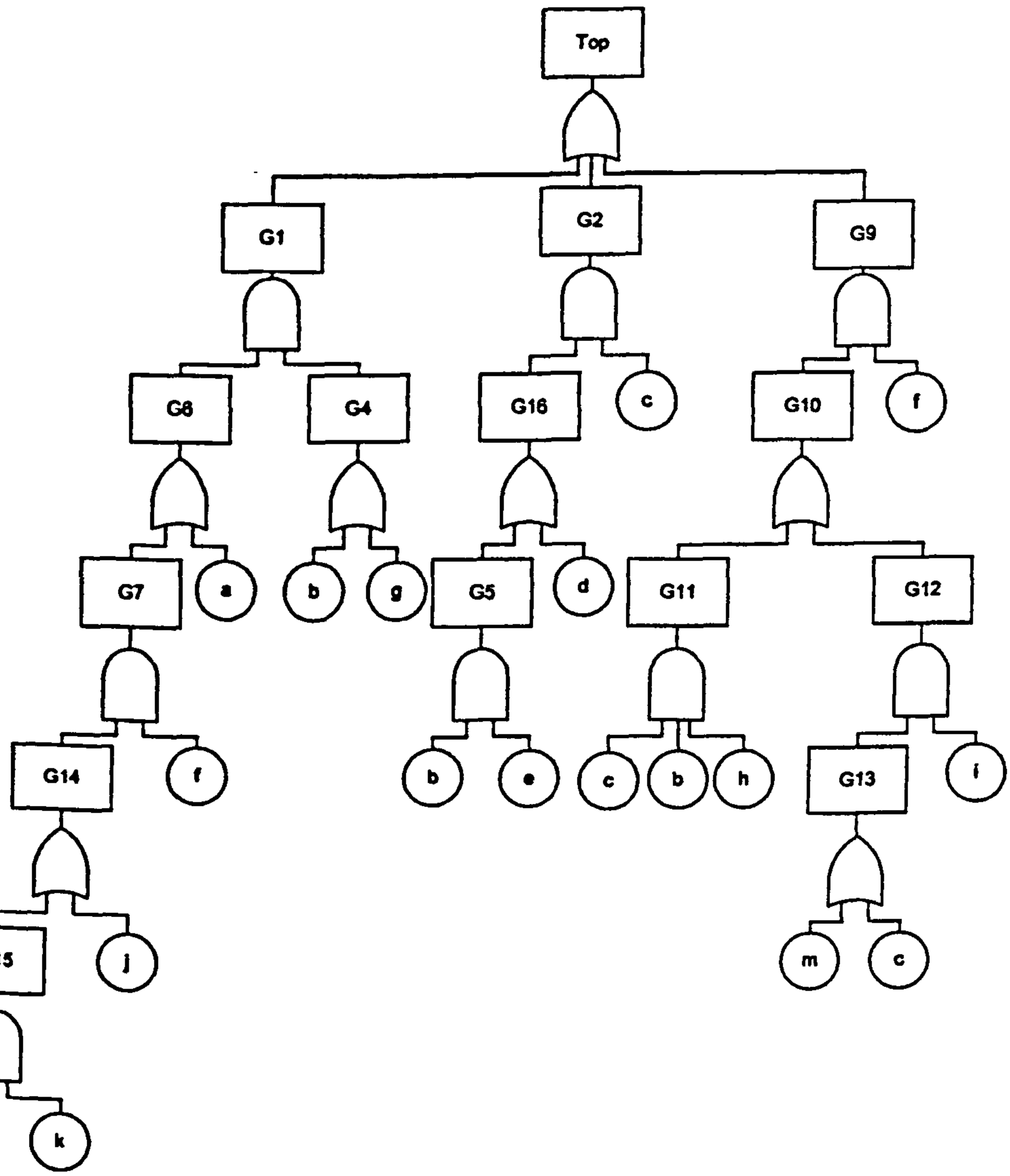


Figure 8.1: Fault Tree Diagram

A Boolean expression for the top event will be obtained using the bottom-up approach, from which all minimal cut sets of order three and below can be identified.

Beginning by developing an expression for gate G1, by dealing with gates, G15, G14, G7 G6 and G4:

$$G15 = e \cdot k$$
$$G14 = G15 + j = e \cdot k + j$$
$$G7 = G14 \cdot f = (e \cdot k + j) \cdot f = e \cdot f \cdot k + f \cdot j$$
$$G6 = G7 + a = e \cdot f \cdot k + f \cdot j + a$$
$$G4 = b + g$$

From this the following expression is obtained for gate G1:

$$G1 = G6 \cdot G4$$
$$= (e \cdot f \cdot k + f \cdot j + a)(b + g)$$
$$= b \cdot f \cdot j + a \cdot b + f \cdot g \cdot j + a \cdot g$$

The combinations $b \cdot e \cdot f \cdot k$ and $e \cdot f \cdot g \cdot k$ are eliminated from the Boolean expression for gate G1 because they exceed the culling order

Now obtaining an expression for gate G2 by dealing with gates, G16 and G5:

$$G5 = b \cdot e$$
$$G16 = G5 + d = b \cdot e + d$$

From these expressions an expression for gate G2 is obtained:

$$G2 = G16 \cdot c = b \cdot c \cdot e + c \cdot d$$

An expression for gate G9 is obtained by developing gates, G13, G12, G11 and G10:

$$G13 = c + m$$
$$G11 = b \cdot c \cdot h$$
$$G12 = G13 \cdot i = (c + m) \cdot i = c \cdot i + i \cdot m$$
$$G10 = G11 + G12 = c \cdot i + i \cdot m + b \cdot c \cdot h$$

The following expression is then obtained for gate G9:

$$G9 = G10 \cdot f$$
$$= (c \cdot i + i \cdot m + b \cdot c \cdot h) \cdot f$$
$$= c \cdot f \cdot i + f \cdot i \cdot m$$

The combination $b \cdot c \cdot h \cdot f$ is eliminated from the Boolean expression for gate G9 since it exceeds the culling order.

Finally the following Boolean expression is obtained for the top gate, Top:

$$\begin{aligned} Top &= G1 + G2 + G9 \\ &= b \cdot f \cdot j + f \cdot g \cdot j + a \cdot b + a \cdot g + b \cdot c \cdot e + c \cdot d + f \cdot i \cdot m + c \cdot f \cdot i \end{aligned} \tag{8.1}$$

Eight minimal cut sets are identified from equation (8.1):

{bfj}, {fgj}, {ab}, {ag}, {bce}, {cd}, {fim}, {cfi}

Once a partial list of minimal cut sets has been obtained, it is possible to use these to quantify the system approximately using the conventional FTA techniques outlined in chapter two.

8.2.2 Culling Minimal Cut Sets Below a Given Probabilistic Value

The first stage of this culling technique is to set the probabilistic value according to which the minimal cut sets will be culled. Then the top-down or bottom-up approach is used to identify a partial list of minimal cut sets. As the Boolean expression for the top event is developed, any combinations that have a probabilistic value below the pre-set value are eliminated from the expression. The final expression obtained for the top event only contains the minimal cut sets with a probabilistic value above the pre-set probabilistic value. Component failure probabilities are essential to employ this technique.

To illustrate this technique consider the fault tree given in figure 8.2. The probabilistic culling limit will be set to 1×10^{-7}, thus any minimal cut sets with a probabilistic value below this culling limit will be eliminated from the Boolean expression for the top event. The component failure probabilities for each component in this example are given in table 8.1.

Component	Failure Probability q(t)	Failure Frequency w(t)
a	1×10^{-2}	1×10^{-5}
b	2.5×10^{-4}	2×10^{-4}
c	2×10^{-3}	4.5×10^{-3}
d	6.5×10^{-2}	2.3×10^{-2}
e	5×10^{-3}	5×10^{-3}
f	2×10^{-2}	6.5×10^{-2}
g	1×10^{-2}	3×10^{-3}

Table 8.1: Summary of the Component Failure Probabilities

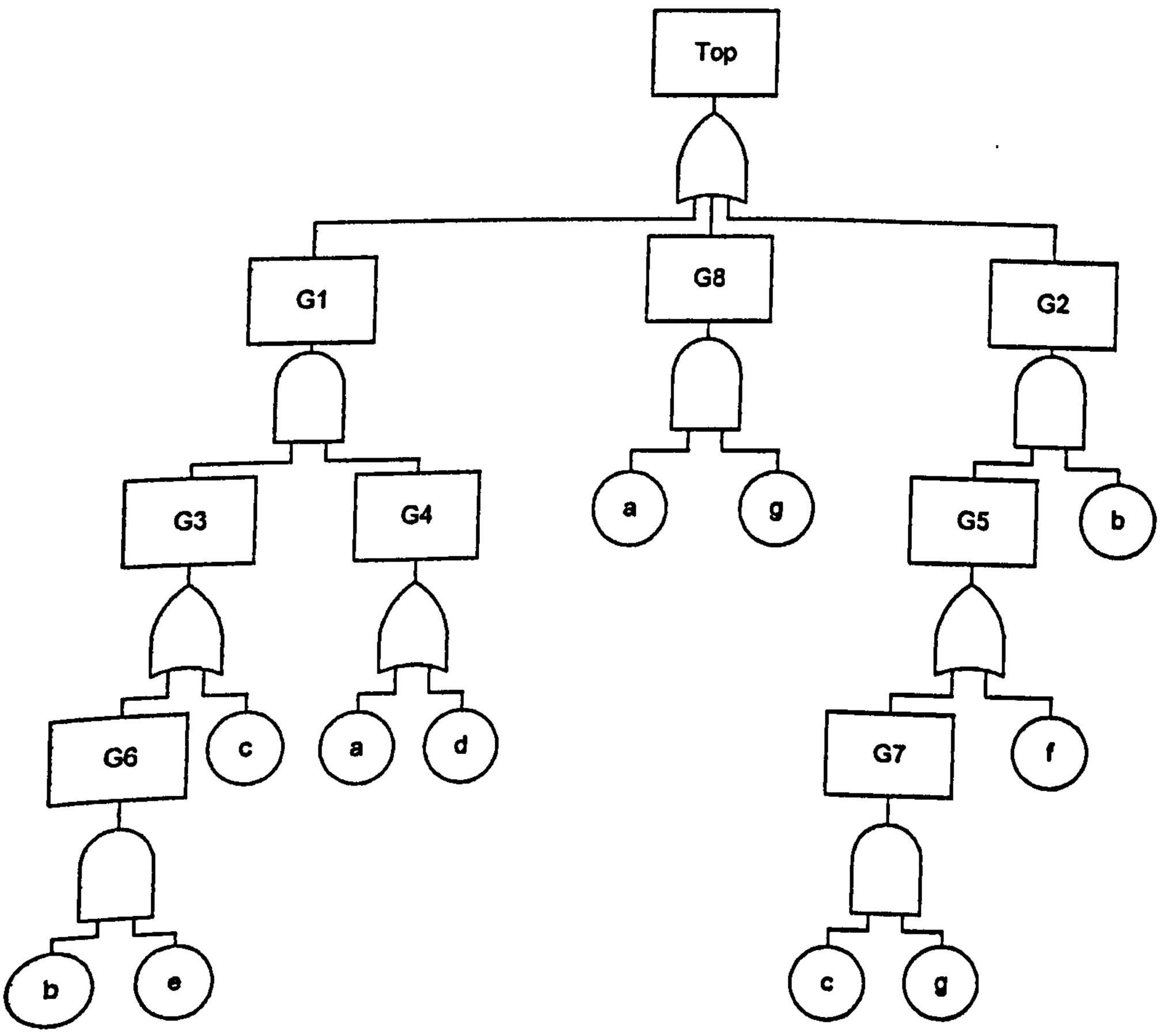


Figure 8.2: Fault Tree Diagram

Beginning by developing an expression for G1 by evaluating, G3, G4 and G6:

$G6 = b \cdot e$

$P(b) \cdot P(e) = (2.5 \times 10^{-4})(5 \times 10^{-3})$ The probability of this combination is higher

$= 1.25 \times 10^{-7}$ than the culling limit of 1×10^{-7}.

$G4 = a + d$

$P(a) = 1 \times 10^{-2}$ The probabilities of both of these combinations

$P(d) = 6.5 \times 10^{-2}$ are higher than the culling limit.

$G3 = G6 + c = b \cdot e + c$

$P(b \cdot e) = 1.25 \times 10^{-7}$ The probabilities of both of these combinations

$P(c) = 2 \times 10^{-3}$ are higher than the culling limit.

Now an expression can be obtained for gate G1:

$G1 = G3 \cdot G4 = (b \cdot e + c)(a + d) = a \cdot b \cdot e + b \cdot d \cdot e + a \cdot c + a \cdot d$

$P(a \cdot b \cdot e) = (1 \times 10^{-2})(1.25 \times 10^{-7})$

$= 1.25 \times 10^{-9}$

$P(b \cdot d \cdot e) = (6.5 \times 10^{-2})(1.25 \times 10^{-7})$

$= 8.125 \times 10^{-9}$

$P(a \cdot c) = (1 \times 10^{-2})(2 \times 10^{-3})$

$= 2 \times 10^{-5}$

$P(a \cdot d) = (1 \times 10^{-2})(6.5 \times 10^{-2})$

$= 1.3 \times 10^{-4}$

The probability of the combinations, $a \cdot b \cdot e$ and $b \cdot d \cdot e$ are below the probabilistic culling limit, 1×10^{-7}, hence they are eliminated from the expression for gate G1, the culled expression for this gate is given below:

$$G1 = a \cdot c + a \cdot d$$

Now developing an expression for gate G8:

$G8 = a \cdot g$

$$P(a \cdot g) = (1\times10^{-2})(1\times10^{-2}) \qquad \text{The probability of this combination is higher}$$
$$= 1\times10^{-4} \qquad \text{than the culling limit.}$$

By evaluating gates G5 and G7 it is possible to obtain an expression for gate G2:

$$G7 = c \cdot g$$

$$P(c \cdot g) = (2\times10^{-3})(1\times10^{-2}) \qquad \text{The probability of this combination is higher}$$
$$= 2\times10^{-5} \qquad \text{than the culling limit}$$

$$G5 = G7 + f = c \cdot g + f$$

$$P(c \cdot g) = 2\times10^{-5} \qquad \text{The probabilities of both of these combinations}$$
$$P(f) = 2\times10^{-5} \qquad \text{are higher than the culling limit}$$

From these results the following expression is obtained for gate G2:

$$G2 = G5 \cdot b = (c \cdot g + f) \cdot b = b \cdot c \cdot g + b \cdot f$$

$$P(b \cdot c \cdot g) = (2.5\times10^{-4})(2\times10^{-5})$$
$$= 5\times10^{-9}$$

$$P(b \cdot f) = (2.5\times10^{-4})(2\times10^{-2})$$
$$= 5\times10^{-9} + 5\times10^{-6}$$

The probability of combination $b \cdot c \cdot g$ is below the culling limit; hence it is eliminated from the expression for gate G2, the final expression for gate G2 is given below:

$$G2 = b \cdot f$$

Finally an expression for the top event is obtained:

$$Top = G1 + G2 + G8 = a \cdot c + a \cdot d + a \cdot g + b \cdot f$$

The probability of all of the combinations in the Boolean expression for the top event are higher than the culling limit of 1×10^{-7}. Hence no further culling is required, the final Boolean expression for the top event is given below:

$$Top = a \cdot c + a \cdot d + a \cdot g + b \cdot f \qquad (8.2)$$

Four minimal cut sets are identified from equation (8.2):

{ac},{ad},{ag},{bf}

The minimal cut sets identified can be used to quantify the system approximately. The conventional quantification techniques outlined in chapter two are employed in exactly the same way.

8.2.3 Culling Minimal Cut Sets Below a Given Frequency

This culling technique is very similar to the probabilistic culling procedure outlined in section 8.2.2; the only difference is that the minimal cut sets are culled according to a pre-set frequency rather than probability. Once again the frequency value must be set according to which the minimal cut sets will be culled. Then the top-down or bottom-up approach is used to identify a partial list of minimal cut sets. As the Boolean expression for the top event is developed, any combinations that have a frequency below the pre-set value are eliminated from the expression. The final expression obtained for the top event only contains the minimal cut sets with a frequency above the pre-set frequency value. Component failure probabilities and frequencies are essential to employ this technique.

To illustrate this technique consider again the fault tree given in figure 8.2. The frequency culling limit will be set to 5×10^{-7}, thus any minimal cut sets with a frequency below this culling limit will be eliminated from the Boolean expression for the top event. The component failure probabilities and frequencies for each component in this example are given in table 8.1.

Beginning by developing an expression for G1 by evaluating, G3, G4 and G6:

$G6 = b \cdot e$

$$q_b w_e + q_e w_b = (2.5\times10^{-4})(3\times10^{-3}) + (1\times10^{-2})(2\times10^{-4})$$

$$q_b w_e + q_e w_b = 7.5\times10^{-7} + 2\times10^{-6}$$

$$q_b w_e + q_e w_b = 2.75\times10^{-6}$$

The probability of this combination is higher than the culling limit of 5×10^{-7}.

$$G4 = a + d$$

$w_a = 1\times10^{-5}$

$w_d = 2.3\times10^{-2}$

The frequencies of these combinations are higher than the culling limit.

$$G3 = G6 + c = b\cdot e + c$$

$q_b w_e + q_e w_b = 2.75\times10^{-6}$

$w_c = 4.5\times10^{-3}$

The frequencies of these combinations are higher than the culling limit.

Now an expression can be obtained for gate G1:

$$G1 = G3\cdot G4 = (b\cdot e + c)(a + d) = a\cdot b\cdot e + b\cdot d\cdot e + a\cdot c + a\cdot d$$

$$\begin{aligned} q_a q_b w_e + q_a q_e w_b + q_b q_e w_a &= (1\times10^{-2})(2.5\times10^{-4})(5\times10^{-6}) + (1\times10^{-2})(5\times10^{-3})(2\times10^{-5}) \\ &\quad + (2.5\times10^{-4})(5\times10^{-3})(1\times10^{-5}) \\ &= 1.025\times10^{-9} \end{aligned}$$

$$\begin{aligned} q_b q_d w_e + q_b q_e w_d + q_d q_e w_b &= (2.5\times10^{-4})(6.5\times10^{-2})(5\times10^{-6}) + (2.5\times10^{-4})(5\times10^{-3})(2.3\times10^{-2}) \\ &\quad + (6.5\times10^{-2})(5\times10^{-3})(2\times10^{-5}) \\ &= 3.533\times10^{-8} \end{aligned}$$

$$\begin{aligned} q_a w_c + q_c w_a &= (1\times10^{-2})(4.5\times10^{-3}) + (2\times10^{-3})(1\times10^{-5}) \\ &= 4.52\times10^{-5} \end{aligned}$$

$$\begin{aligned} q_c w_d + q_d w_c &= (2\times10^{-3})(2.3\times10^{-2}) + (6.5\times10^{-2})(4.5\times10^{-3}) \\ &= 3.39\times10^{-4} \end{aligned}$$

The frequency of the combinations $a\cdot b\cdot e$ and $b\cdot d\cdot e$ are below the culling limit, 5×10^{-7} hence they are eliminated from the expression for gate G1, the culled expression for this gate is given below:

$$G1 = a\cdot c + a\cdot d$$

Now developing an expression for gate G8:

$$G8 = a\cdot g$$

$$q_a w_g + q_g w_a = (1\times10^{-2})(3\times10^{-3}) + (1\times10^{-2})(1\times10^{-5})$$

$$q_a w_g + q_g w_a = 3\times10^{-5} + 1\times10^{-7}$$

$$q_a w_g + q_g w_a = 3.01\times10^{-5}$$

The frequency of this combination is higher than the culling limit.

By evaluating gates G5 and G7 it is possible to obtained an expression for gate G2:

$$G7 = c \cdot g$$

$$q_c w_g + q_g w_c = \left(2\times10^{-3}\right)\left(3\times10^{-3}\right) + \left(1\times10^{-2}\right)\left(4.5\times10^{-3}\right)$$

$$q_c w_g + q_g w_c = 6\times10^{-6} + 4.5\times10^{-5}$$

$$q_c w_g + q_g w_c = 5.1\times10^{-5}$$

The frequency of this combination is higher than the culling limit.

$$G5 = G7 + f = c \cdot g + f$$

$$q_c w_g + q_g w_c = 5.1\times10^{-5}$$

$$w_f = 6.5\times10^{-2}$$

The frequencies of these combinations are higher than the culling limit

From these results the following expression is obtained for gate G2:

$$G2 = G5 \cdot b = (c \cdot g + f) \cdot b = b \cdot c \cdot g + b \cdot f$$

$$q_b q_c w_g + q_b q_g w_c + q_c q_g w_b = \left(2.5\times10^{-4}\right)\left(2\times10^{-3}\right)\left(3\times10^{-3}\right) + \left(2.5\times10^{-4}\right)\left(1\times10^{-2}\right)\left(4.5\times10^{-3}\right)$$
$$+ \left(2\times10^{-3}\right)\left(1\times10^{-2}\right)\left(2\times10^{-5}\right)$$
$$= 1.315\times10^{-8}$$

$$q_b w_f + q_f w_b = \left(2.5\times10^{-4}\right)\left(6.5\times10^{-2}\right) + \left(2\times10^{-2}\right)\left(2\times10^{-5}\right)$$
$$= 1.665\times10^{-5}$$

The frequency of combination $b \cdot c \cdot g$ is below the culling limit; hence it is eliminated from the expression for gate G2, the final expression for gate G2 is given below:

$$G2 = b \cdot f$$

Finally an expression for the top event is obtained:

$$Top = G1 + G2 + G8 = a \cdot c + a \cdot d + a \cdot g + b \cdot f$$

The frequency of all of the combinations in the Boolean expression for the top event are higher than the culling limit of 5×10^{-7}. Hence no further culling is required the final Boolean expression for the top event is given below:

$$Top = a \cdot c + a \cdot d + a \cdot g + b \cdot f \qquad (8.3)$$

Four minimal cut sets are identified from equation (8.3):

$$\{ac\},\{ad\},\{ag\},\{bf\}$$

8.3 Culling Techniques for the Binary Decision Diagram Method

Rauzy [4] developed two techniques that can be used to produce a culled BDD, which encodes a partial list of minimal cut sets. The first technique is concerned with culling any minimal cut sets above a given order, k_{MAX}. The second technique produces a list of minimal cut sets with a probabilistic value $\geq p_{MIN}$. These culling techniques employ a modified form the **Ite** procedure introduced in chapter three, and will be considered in detail in sections 8.3.1 and 8.3.2. An alternative technique has been developed as part of the research for this thesis, which can be used to compute a frequency culled BDD. This will be considered in section 8.3.3.

8.3.1 Rauzy's Technique for Computing an Order Culled Binary Decision Diagram

It is first necessary to set the culling limit, k_{MAX}. If k_{MAX} is set to four, then cut sets up to and including order four will be encoded in the BDD. Any paths through the BDD that encode cut sets exceeding order four are terminated with a terminal 0 vertex.

Once k_{MAX} has been set and the system components have been assigned an ordering, the culled BDD can be computed. The procedure for computing the SFBDD that is outlined in chapter three is modified to ensure that any cut sets exceeding k_{MAX} are culled. The modified procedure is outlined below:

1. Assign each basic event x_i in the fault tree an **Ite** structure, $ite(x_i,1,0)$.
2. Modify the fault tree structure so that each gate has only two inputs.
3. Consider each gate in a bottom-up fashion.
4. If the two gate inputs are J and H such that:

$$J = ite(x,F1,F2) \qquad H = ite(y,G1,G2)$$

To compute J<op>H the same rules are applied as outlined in chapter three. However an additional check is required before the relevant rule is applied to ensure that the culling limit will not be exceeded by a new calculation.

The variable k_{x_i} tracks the number of basic events encoded in a particular path from the root vertex down to and including the node x_i.

If $k_{x_i} < k_{MAX}$ then a new calculation can be performed on the one branch, if however $k_{x_i} \geq k_{MAX}$ then the culling limit has been reached, thus the one branch of node x_i must be terminated by a terminal 0 vertex. When dealing with the zero branch of node x_i, component i itself is not included in the path, hence, if $k_{x_i} - 1 < k_{MAX}$ then a new calculation can be performed on the zero branch, but if $k_{x_i} - 1 \geq k_{MAX}$ the zero branch of node x_i is terminated. The algorithm for computing J<op>H for a culled BDD is outlined in figure 8.3.

```
Computing J<op> H
if(x<y)
        if(k_xi <k_MAX)
                J<op>H=ite(x,F1<op>H,F0<op>H)
        else if(k_xi ≥ k_MAX and  k_xi -1<k_MAX)
                J<op>H=ite(x,0,F0<op>H)
        else if(k_xi -1 ≥ k_MAX)
                J<op>H=0
else if(x=y)
        if(k_xi <k_MAX)
                J<op>H=ite(x,F1<op>G1,F0<op>G0)
        else if(k_xi ≥ k_MAX and  k_xi -1<k_MAX)
                J<op>H=ite(x,0,F0<op>G0)
        else if(k_xi -1 ≥ k_MAX)
                J<op>H=0
```

Figure 8.3: The Algorithm for Computing J<op>H for an Order Culled BDD

These rules are applied in conjunction with the four identities outlined in chapter three. Although the identities, $0 \cdot H = 0$ and $1 + H = 1$ can be applied directly during the computation process, since they reduce to terminal 1 or 0 vertices. The remaining identities, $1 \cdot H = H$ and $0 + H = H$ can increase the number of basic events encoded in the paths through the BDD. Hence additional work is required to ensure that the culling limit is not exceeded when these identities are applied.

Suppose that the result of the one branch of node x_i is of the form, $1 \cdot H$, which would usually reduce to H. This identity cannot be employed directly, since if H is encoded on the one branch of node x_i, it could lead to cut sets, which exceed the culling limit being encoded. Thus before H can be encoded it must be checked and where necessary modified to ensure only cut sets within the culling limit are encoded.

The value k_{x_i} records the number of basic events encoded in the path from the root vertex down to and including the node x_i, and k_H represents the maximum number of basic events encoded in the paths through H. If $k_{x_i} + K_H$ exceeds k_{MAX}, then, if H is retained as a solution on the one branch of node x_i it will result in cut sets which exceed the culling order being encoded. In such cases further processing is required to check through H for any paths which when combined with the path preceding node x_i encodes cut sets that exceed the culling limit. Any such paths are terminated by a terminal 0 vertex. Once H has been fully modified the resulting Ite structure is taken as the solution to the one branch of node x_i.

Similarly when dealing with this type of calculation on the zero branch of node x_i, checks must be made to ensure that $k_{x_i} - 1 + k_H \leq k_{MAX}$. If $k_{x_i} - 1 + k_H > k_{MAX}$, H cannot be encoded exactly. Again further processing is required to modify H such that it does not encode any paths that exceed the culling limit, k_{MAX}.

The same procedure is employed when the result of a one or zero branch is of the form 0+H. The algorithm for implementing the four identities is outlined in figure 8.4.

Employing the four identities for the one branch of a node

```
if(<op>=AND)
        if(J=1)
            if(k_xi +k_H ≤ k_MAX)
                J·H=H
            else if(k_xi +k_H > k_MAX)
                check and modify H
        else if(j=0)
                J·H=0
else if(<op>=OR)
        if(j=1)
                J+H=1
        else if(j=0)
            if(k_xi +k_H ≤ k_MAX)
                J+H=H
            else if(k_xi +k_H>k_MAX)
                check and
                modify H
```

Employing the four identities for the zero branch of a node

```
if(<op>=AND)
        if(J=1)
            if(k_xi −1+k_H ≤ k_MAX)
                J·H=H
            else if(k_xi −1+k_H>k_MAX)
                check and modify H
        else if(j=0)
                J·H=0
else if(<op>=OR)
        if(j=1)
                J+H=1
        else if(j=0)
            if(k_xi −1+k_H ≤ k_MAX)
                J+H=H
            else if(k_xi −1+k_H>k_MAX)
                check and
                modify H
```

Figure 8.4: The Algorithm for Implementing the Four Identities for Computing an Order Culled BDD

To illustrate this technique consider the fault tree in figure 8.5. The culling limit, k_{MAX} will be set to two. Thus any paths encoding cut sets of order three or above will be terminated by a terminal 0 vertex.

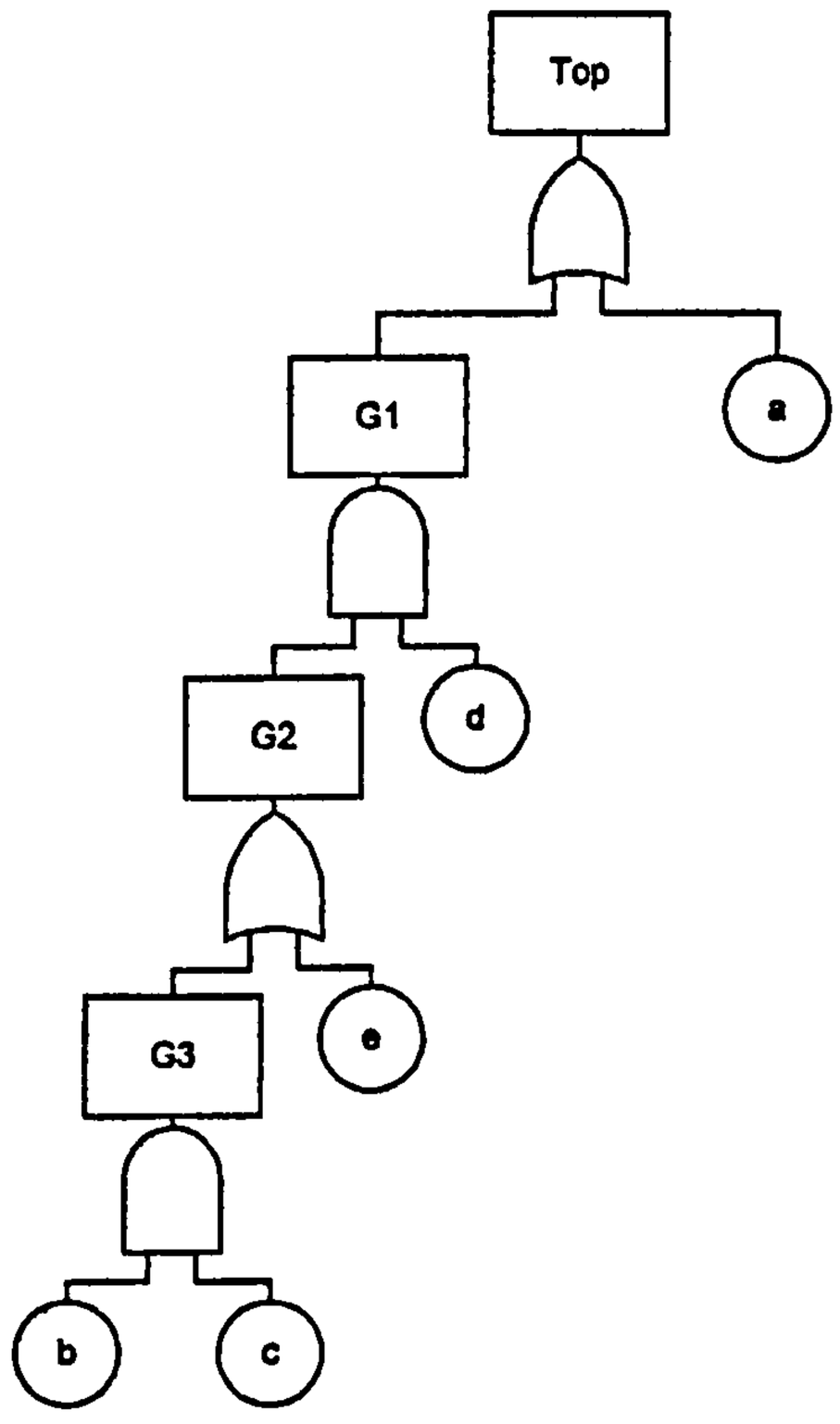


Figure 8.5: Fault Tree Diagram

Beginning with gate G3:

$$G3 = ite(b,1,0) \cdot ite(c,1,0)$$
$$G3 = ite(b,[1 \cdot ite(c,1,0)],[0 \cdot ite(c,1,0)]) \quad (8.4)$$

Evaluating the one branch of equation (8.4) $k_{x_b} = 1$ and $k_H = 1$:

$1 \cdot ite(c,1,0) = ite(c,1,0)$ $\quad$ $\boxed{k_{x_b} + k_H \leq k_{MAX} \quad 1 \cdot H = H}$

Evaluating the zero branch of equation (8.4) $k_{x_b} - 1 = 0$ and $k_H = 1$:

$0 \cdot ite(c,1,0) = 0$ $\quad$ $\boxed{0 \cdot H = 0}$

Therefore the final expression for G3 is given below:

G3=ite(b,ite(c,1,0),0)

Now dealing with gate G2:

$$G2 = G3 + ite(e,1,0) \quad (8.5)$$
$$G2 = ite(b,ite(c,1,0),0) + ite(e,1,0)$$
$$G2 = ite(b,[ite(c,1,0) + ite(e,1,0)],[0 + ite(e,1,0)])$$

Evaluating the one branch of equation (8.5) $k_{x_b}=1$ and $k_H=1$:

$$\text{ite}(c,1,0)+\text{ite}(e,1,0)=\text{ite}(c,[1+\text{ite}(e,1,0)],[0+\text{ite}(e,1,0)]) \tag{8.6}$$

In equation (8.6) the one branch is of the form 1+H, which reduces to 1. The zero branch of this equation is of the form 0+H since $k_{x_c}-1+k_H \le k_{MAX}$ this reduces to H as normal.

Evaluating the zero branch of equation (8.5) $k_{x_b}-1=0$ and $k_H=1$:

$$0+\text{ite}(e,1,0)=\text{ite}(e,1,0) \qquad \boxed{k_{x_b}-1+k_H \le k_{MAX}\ 1\cdot H=H}$$

The final ite structure for gate G2 is given below:

G2=ite(b,ite(c,1,ite(e,1,0)),ite(e,1,0))

Now evaluating gate G1:

$$G1=G2\cdot\text{ite}(d,1,0)$$
$$G1=\text{ite}(b,\text{ite}(c,1,\text{ite}(e,1,0)),\text{ite}(e,1,0))\cdot\text{ite}(d,1,0) \tag{8.7}$$
$$=\text{ite}(b,[\text{ite}(c,1,\text{ite}(e,1,0))\cdot\text{ite}(d,1,0)],[\text{ite}(e,1,0)\cdot\text{ite}(d,1,0)])$$

Evaluating the one branch of equation (8.7) $k_{x_b}=1$ and $k_H=1$:

$$=\text{ite}(c,1,\text{ite}(e,1,0))\cdot\text{ite}(d,1,0)$$
$$=\text{ite}(c,[1\cdot\text{ite}(d,1,0)],[\text{ite}(e,1,0)\cdot\text{ite}(d,1,0)]) \tag{8.8}$$

In equation (8.8), $k_{x_c}=2$ and $K_H=1$, hence, $k_{x_c}+k_H>k_{MAX}$ the one branch of equation (8.8) is terminated by a terminal zero vertex. The zero branch is also terminated by a terminal zero vertex since if the calculation is performed exactly the culling order will be exceeded. The final result obtained for the one branch for equation (8.7) is given below:

ite(c,0,0)=0

Evaluating the zero branch of equation (8.7) $k_{x_b}-1=0$ and $k_H=1$:

$$\text{ite}(e,1,0)\cdot\text{ite}(d,1,0)=\text{ite}(d,[1\cdot\text{ite}(e,1,0)],[0\cdot\text{ite}(e,1,0)]) \tag{8.9}$$

In equation (8.9) $k_{x_d} = 1$ and $k_H = 1$, thus since k_{MAX} is not exceeded H, i.e. ite(e,1,0) in this case is taken as the solution to the one branch of equation (8.9). The zero branch of this equation is of the form $0 \cdot H$, which simply reduces to 0.

The final ite structure for gate G1 is given below:

G1 = ite(b,0,ite(d,ite(e,1,0),0))

Finally computing the ite structure of the top gate, Top:

$$\begin{aligned} &Top = G1 + ite(a,1,0) \\ &Top = ite(b,0,ite(d,ite(e,1,0),0)) + ite(a,1,0) \\ &Top = ite(a,[1 + ite(b,0,ite(d,ite(e,1,0),0))],[0 + ite(b,0,ite(d,ite(e,1,0),0))]) \end{aligned} \quad (8.10)$$

Evaluating the one branch of equation (8.10) $k_{x_a} = 1$ and $k_H = 2$:

$1 + ite(d,ite(e,1,0),0) = 1$ $\quad$ $\boxed{1+H=1}$

Evaluating the zero branch of equation (8.10) $k_{x_a} - 1 = 0$ and $k_H = 2$:

$0 + ite(b,0,ite(d,ite(e,1,0),0)) = ite(b,0,ite(d,ite(e,1,0),0))$ $\quad$ $\boxed{k_{x_a} - 1 + k_H \leq k_{MAX}\ 0+H=H}$

The ite structure for the top gate, Top is given in equation (8.11):

$$Top = ite(a,1,ite(b,0,ite(d,ite(e,1,0),0))) \quad (8.11)$$

The culled BDD obtained for the fault tree in figure 8.5 is given in figure 8.6:

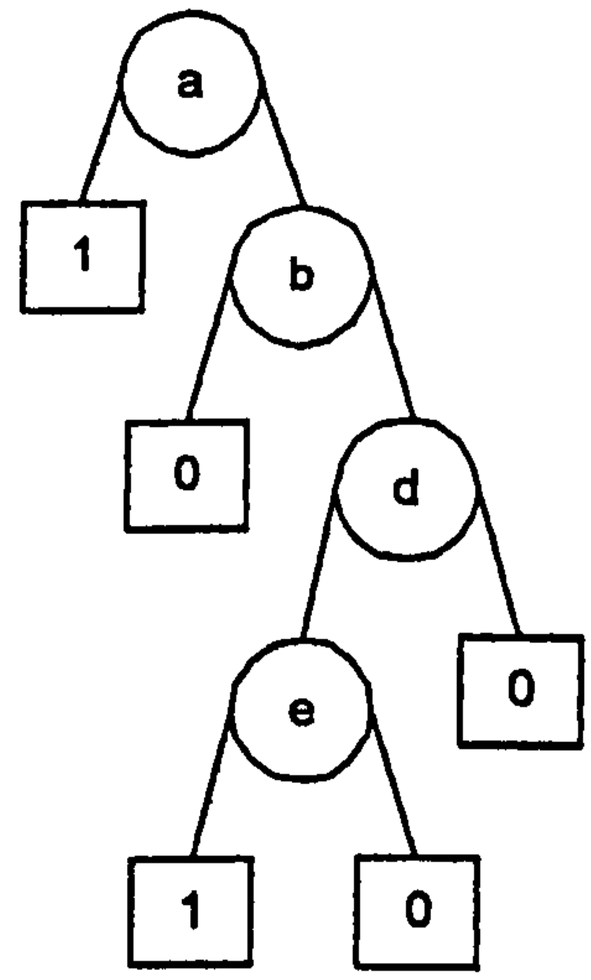


Figure 8.6: The Order Culled BDD Obtained for the Fault Tree in Figure 8.5, k_{MAX}=2

8.3.2 Rauzy's Technique for Computing a Probability Culled Binary Decision Diagram

It is first necessary to set the culling limit, p_{MIN}. If p_{MIN} is set to 1×10^{-5}, then cut sets with a probability below this value are culled. Any paths through the BDD encoding cut sets with a probability below p_{MIN} are terminated with a terminal 0 vertex.

Once p_{MIN} has been set, the culled BDD can be computed. The procedure for computing the SFBDD that is outlined in chapter three is again modified, as shown below. The first three steps outlined in the previous section remain the same. The fourth step, computing the **ite** structure of a gate with inputs J and H has to be modified such that the BDD computed is culled according to p_{MIN}. The procedure for computing J<op>H is outlined below:

If the two gate inputs are J and H such that:

$$J = \text{ite}(x, F1, F2) \qquad H = \text{ite}(y, G1, G2)$$

To compute J<op>H rules outlined in chapter three are applied. However, an additional check is required before the relevant rule is applied to ensure that the minimal cut sets below the probabilistic culling limit are not encoded in the BDD. The variable p_{x_i} tracks the probability of basic events encoded in the path from the root vertex to the current node x_i.

$$p_{x_i} = \prod_{\substack{j=1 \\ j \text{ exists on} \\ \text{the 1 branch}}}^{i} q_j \tag{8.12}$$

If $p_{x_i} > p_{MIN}$, then a calculation can be performed on the one branch, if however $p_{x_i} \leq p_{MIN}$, then the culling limit has been reached, thus the one branch of node x_i must be terminated by a terminal 0 vertex. Similarly if, $\frac{p_{x_i}}{q_i} > p_{MIN}$, then a new calculation can be performed on the zero branch, but if, $\frac{p_{x_i}}{q_i} \leq p_{MIN}$, the zero branch of node x_i is terminated. The algorithm for computing J<op>H is outlined in figure 8.7.

Computing J<op> H

if(x<y)

if(p_{x_i} >p_{MIN})

J<op>H=ite(x,F1<op>H,F0<op>H)

else if($p_{x_i} \leq p_{MIN}$ and $\frac{p_{x_i}}{q_i} > p_{MIN}$)

J<op>H=ite(x,0,F0<op>H)

else if($\frac{p_{x_i}}{q_i} \leq p_{MIN}$)

J<op>H=0

else if(x=y)

if(p_{x_i} >p_{MIN})

J<op>H=ite(x,F1<op>G1,F0<op>G0)

else if($p_{x_i} \leq p_{MIN}$ and $\frac{p_{x_i}}{q_i} > p_{MIN}$)

J<op>H=ite(x,0,F0<op>G0)

else if($\frac{p_{x_i}}{q_i} \leq p_{MIN}$)

J<op>H=0

Figure 8.7: The Algorithm for Computing J<op>H for a Probability Culled BDD

These rules are again applied in conjunction with the four identities outlined in chapter three. The identities $1 \cdot H = H$ and $0 + H = H$ cannot be applied directly, instead H must be checked and modified where necessary to ensure any minimal cut sets with a probability below p_{MIN} are eliminated.

For this culling technique if the one branch of a node x_i is of the form $1 \cdot H$ or 1+H, $p_{x_i} \cdot p_H$ is calculated, where p_H represents the minimum probability of all the paths through H. If this probability is below p_{MIN}, then H must be checked and modified such that any paths through H that when combined with the path preceding node x_i encode cut sets with a probability below the culling limit are eliminated from H.

Similarly if the zero branch is of the form $1 \cdot H$ or 1+H, $\frac{p_{x_i}}{q_i} \cdot p_H$ is calculated and if it is below p_{MIN}, H must be checked and modified accordingly.

Employing the four identities for the one branch of a node

```
if(<op>=AND)
        if(J=1)
            if(p_xi · p_H ≥ p_MIN)
                J·H=H
            else if(p_xi · p_H < p_MIN)
                check and modify H
        else if(j=0)
                J·H=0
else if(<op>=OR)
        if(j=1)
                J+H=1
        else if(j=0)
            if(p_xi · p_H ≥ p_MIN)
                J+H=H
            else if(p_xi · p_H < p_MIN)
                check and
                modify H
```

Employing the four identities for the zero branch of a node

```
if(<op>=AND)
        if(J=1)
            if(p_xi/q_i · p_H ≥ p_MIN)
                J·H=H
            else if(p_xi/q_i · p_H < p_MIN)
                check and modify H
        else if(j=0)
                J·H=0
else if(<op>=OR)
        if(j=1)
                J+H=1
        else if(j=0)
            if(p_xi/q_i · p_H ≥ p_MIN)
                J+H=H
            else if(p_xi/q_i · p_H < p_MIN)
                check and modify H
```

Figure 8.8: The Algorithm for Implementing the Four Identities for Computing a Probability Culled BDD

To illustrate this technique consider again the fault tree given in figure 8.5. The culling limit, p_{MIN} will be set to 1×10^{-8}. Hence, any paths through the BDD that would encode cut sets with a probability below 1×10^{-8} will be terminated by a terminal 0

vertex. The failure probabilities and failure frequencies for each of the components in the fault tree are summarised in table 8.2:

Component	Failure Probability	Failure Frequency
a	1×10^{-5}	6×10^{-3}
b	2×10^{-3}	7×10^{-4}
c	1×10^{-2}	2×10^{-3}
d	2.1×10^{-4}	3.1×10^{-5}
e	2×10^{-7}	2×10^{-6}

Table 8.2: Summary of Component Failure Probabilities and Frequencies

Beginning with gate G1:

$$G1 = ite(b,1,0)\cdot ite(c,1,0)\cdot ite(d,1,0)$$
$$G1 = ite(b,[1\cdot ite(c,1,0)],[0\cdot ite(c,1,0)])\cdot ite(d,1,0) \quad (8.13)$$

Evaluating the one branch of equation (8.13), $p_{x_b} = 2\times10^{-3}$ and $p_H = 1\times10^{-2}$:

$1\cdot ite(c,1,0) = ite(c,1,0)$ $\quad$ $\boxed{p_{x_b}+p_H>p_{MIN}\ \ 1\cdot H = H}$

Evaluating the zero branch of equation (8.13), $\frac{p_{x_b}}{q_{x_b}} = 1$ and $p_H = 1\times10^{-2}$:

$0\cdot ite(c,1,0) = 0$ $\quad$ $\boxed{0\cdot H = 0}$

Gate G1 needs further development:

$$G1 = ite(b,ite(c,1,0),0)\cdot ite(d,1,0) \quad (8.14)$$
$$G1 = ite(b,[ite(c,1,0)\cdot ite(d,1,0)],[0\cdot ite(d,1,0)])$$

Evaluating the one branch of equation (8.14):

$$ite(c,1,0)\cdot ite(d,1,0) = ite(c,[1\cdot ite(d,1,0)],[0\cdot ite(d,1,0)]) \quad (8.15)$$

In equation (8.15), $p_{x_c} = 2\times10^{-5}$ and $p_H = 2.1\times10^{-3}$, thus $p_{x_c}\cdot p_H > p_{MIN}$, $1\cdot H = H$, so the one branch of (8.15) has solution H, i.e., ite(d,1,0) . The zero branch of equation (8.15) is of the form $0\cdot H$, which reduces to 0.

Evaluating the zero branch of equation (8.14), $\frac{p_{x_c}}{q_{x_c}} = 1$ and $p_H = 2.1\times10^{-3}$:

$0 \cdot ite(d,1,0) = 0$ $\boxed{0 \cdot H = 0}$

The final **ite** structure for gate G1 is given below:

G1=ite(b,ite(c,ite(d,1,0),0),0)

Now evaluating gate G2:

$G2 = ite(d,1,0) \cdot ite(e,1,0)$ (8.16)

$G2 = ite(d,[1 \cdot ite(e,1,0)],[0 \cdot ite(e,1,0)])$

Evaluating the one branch of equation (8.16), $p_{x_d} = 2.1\times10^{-3}$ and $p_H = 2\times10^{-7}$:

$1 \cdot ite(e,1,0) = 0$ $\boxed{p_{x_d} \cdot p_H \le p_{MIN}\ 1 \cdot H = 0}$

Evaluating the zero branch of equation (8.16), $\frac{p_{x_d}}{q_{x_d}} = 1$ and $p_H = 2\times10^{-7}$:

$0 \cdot ite(e,1,0) = 0$ $\boxed{0 \cdot H = 0}$

The final **ite** structure for gate G2 is given below:

$G2 = ite(d,0,0)$

G2=0

Finally computing the **ite** structure of the top gate, Top:

$Top = G1 + G2 + ite(a,1,0)$

$Top = ite(b,ite(c,ite(d,1,0),0),0) + 0 + ite(a,1,0)$

$Top = ite(b,ite(c,ite(d,1,0),0),0) + ite(a,1,0)$

$Top = ite(a,[1 + ite(b,ite(c,ite(d,1,0),0),0)],[0 + ite(b,ite(c,ite(d,1,0),0),0)])$

(8.17)

Evaluating the one branch of equation (8.17), $p_{x_a} = 1\times10^{-5}$ and $p_H = 4.2\times10^{-8}$:

$1+ite(b,ite(c,ite(d,1,0),0),0) = 1$ $\boxed{1+H=H}$

Evaluating the zero branch of equation (8.17), $\frac{p_{x_a}}{q_{x_a}} = 1$ and $p_H = 4.2 \times 10^{-8}$:

$0+ite(b,ite(c,ite(d,1,0),0),0) = ite(b,ite(c,ite(d,1,0),0),0)$ $\boxed{\frac{p_{x_a}}{q_{x_a}} \cdot p_H \geq p_{MIN}\ 0+H=H}$

The **ite** structure for the top gate, Top is given in equation (8.18):

$$Top = ite(a,1,ite(b,ite(c,ite(d,1,0),0),0)) \quad (8.18)$$

The culled BDD obtained for the fault tree in figure 8.5 is given in figure 8.9.

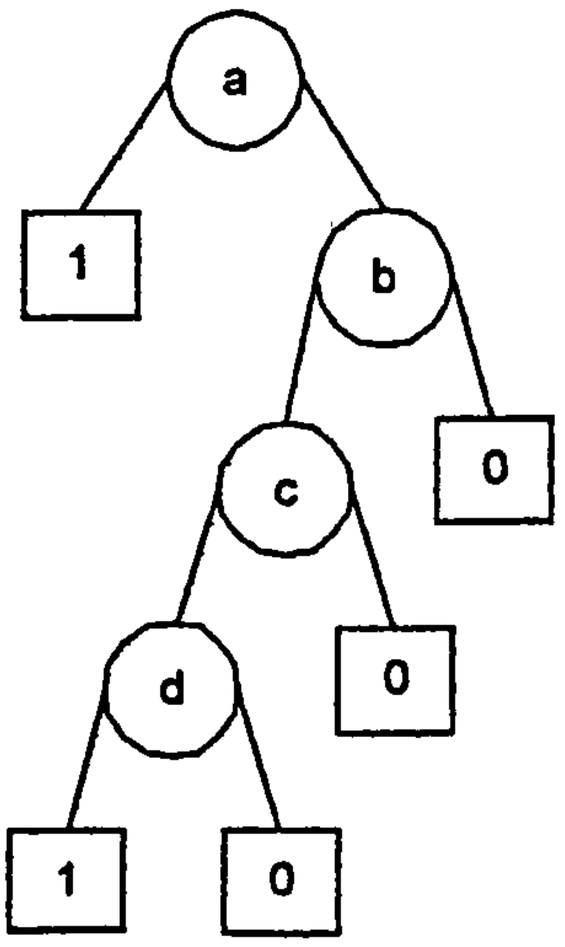


Figure 8.9: Probability Culled BDD Obtained for the Fault Tree in Figure 8.5,

$p_{MIN} = 1 \times 10^{-8}$

8.3.3 Beeson and Andrews Technique for Computing a Frequency Culled Binary Decision Diagram

An alternative technique has been developed as part of this research project, which can be used to produce a frequency culled BDD. To cull the SFBDD according to the frequency of the minimal cut sets it is necessary to calculate the frequency of each path through the SFBDD as it is computed. The culling limit f_{MIN} is set, and then if the frequency of the path being computed is below this limit no further development is carried out, instead the path is terminated with a terminal zero vertex. To calculate the frequency of a given path the failure probability and frequency of each system

component must be known. The frequency of a cut set is calculated using equation (8.19).

$$w_C = \sum_{i=1}^{n} w_i \prod_{\substack{j=1 \\ j \neq i}}^{n} q_j \qquad (8.19)$$

The equation for calculating the failure frequency of a cut set is more complex that the equation for calculating the failure probability of a cut set; hence two variables must be used during the BDD construction. The first variable denoted by p_{x_i} tracks the probability of the path from the root vertex down to and including the node x_i. The second denoted by f_{x_i} tracks the frequency of the path from the root vertex down to and including the node x_i. Since the probability and frequency of the cut sets encoded is being calculated, only components represented by those vertices that lie on the 1 branch through the current path are included in the calculation.

For example if the path from the root vertex to node x_i is of the form:

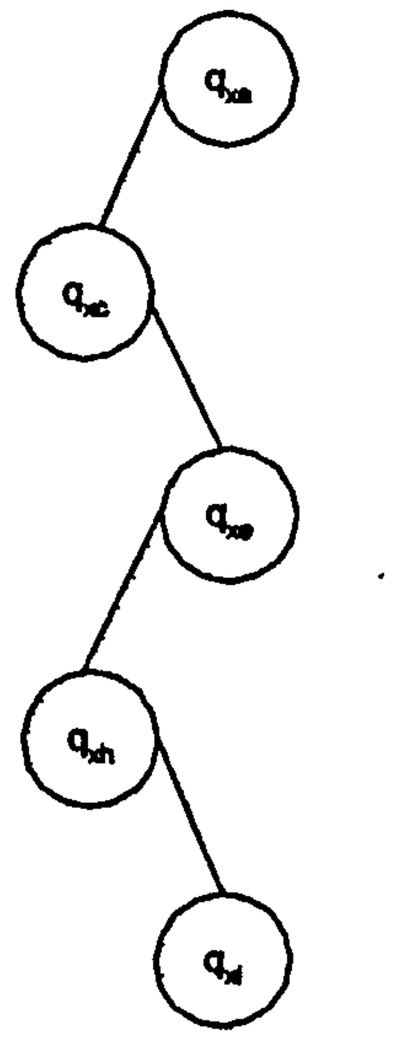


Figure 8.10: Path through a General BDD

Then the following results are obtained for p_{x_i} and f_{x_i} :

$$p_{x_i} = q_{x_a} q_{x_e} q_{x_i}$$

$$f_{x_i} = q_{x_a} q_{x_e} w_{x_i} + q_{x_a} q_{x_i} w_{x_e} + q_{x_e} q_{x_i} w_{x_a}$$

At each stage p_{x_i} and f_{x_i} are recorded for each node, hence the probability and frequency of the path up to x_i in figure 8.10 can be calculated as follows:

$$p_{x_i} = p_{x_h} q_{x_i}$$

$$f_{x_i} = p_{x_h} w_{x_i} + f_{x_h} q_{x_i}$$

To implement this technique it is first necessary to set the culling limit, f_{MIN}. If f_{MIN} is set to 1×10^{-5}, then cut sets with a frequency below this value are culled. Any paths through the BDD encoding cut sets with a probability below f_{MIN} are terminated with a terminal 0 vertex.

Once f_{MIN} has been set, the culled BDD can be computed. The procedure for computing the frequency culled BDD is very similar to that for computing the probability culled BDD, except that it is the frequency of the cut sets encoded that is of interest. The algorithm for computing J<op>H is outlined below in figure 8.11:

```
Computing J<op> H
if(x<y)
        if( f_xi >f_MIN)
                J<op>H=ite(x,F1<op>H,F0<op>H)
        else if( f_xi ≤ f_MIN and f_PR,xi >f_MIN)
                J<op>H=ite(x,0,F0<op>H)
        else if( f_PR,xi ≤ f_MIN)
                J<op>H=0
else if(x=y)
        if( f_xi >f_MIN)
                J<op>H=ite(x,F1<op>G1,F0<op>G0)
        else if( f_xi ≤ f_MIN and  f_PR,xi >f_MIN)
                J<op>H=ite(x,0,F0<op>G0)
        else if( f_PR,xi ≤ f_MIN)
                J<op>H=0
```

Figure 8.11: The Algorithm for Computing J<op>H for a Frequency Culled BDD

Where, f_{PR,x_i} denotes the frequency of the path from the root vertex to the node preceding x_i.

These rules are again applied in conjunction with the four identities outlined in chapter 3. As before it is not possible to apply the identities $1 \cdot H = H$ and $0 + H = H$ directly, instead H must be checked and modified where necessary to ensure any minimal cut sets with a frequency below f_{MIN} are eliminated.

The algorithms for applying these two identities are similar to those outlined for computing the probability culled BDD and are outlined below in figure 8.12.

Employing the four identities for the
one branch of a node

if(<op>=AND)
    if(J=1)
        if$\left(\left(p_{x_i} \cdot f_H + f_{x_i} + p_H\right) \geq f_{MIN}\right)$
            $J \cdot H = H$
        else if$\left(\left(p_{x_i} \cdot f_H + f_{x_i} + p_H\right) < f_{MIN}\right)$
            check and modify H
    else if(j=0)
        $J \cdot H = 0$
else if(<op>=OR)
    if(j=1)
        $J + H = 1$
    else if(j=0)
        if$\left(\left(p_{x_i} \cdot f_H + f_{x_i} + p_H\right) \geq f_{MIN}\right)$
            $J + H = H$
        else if$\left(\left(p_{x_i} \cdot f_H + f_{x_i} + p_H\right) < f_{MIN}\right)$
            check and modify H

Figure 8.12a: The Algorithm for Implementing the Four Identities on the one Branch of a Node for Computing a Frequency Culled BDD

Employing the four identities for the zero branch of a node

if(<op>=AND)

if(J=1)

if $\left(\left(p_{PR,x_i} \cdot f_H + f_{PR} + p_H\right) \geq f_{MIN}\right)$

$J \cdot H = H$

else if $\left(\left(p_{PR,x_i} \cdot f_H + f_{PR} + p_H\right) < f_{MIN}\right)$

check and modify H

else if(j=0)

$J \cdot H = 0$

else if(<op>=OR)

if(j=1)

$J + H = 1$

else if(j=0)

if $\left(\left(p_{PR,x_i} \cdot f_H + f_{PR} + p_H\right) \geq f_{MIN}\right)$

$J + H = H$

else if $\left(\left(p_{PR,x_i} \cdot f_H + f_{PR} + p_H\right) < f_{MIN}\right)$

check and modify H

Figure 8.12b: The Algorithm for Implementing the Four Identities on the Zero Branch of a Node for Computing a Frequency Culled BDD

To illustrate this technique consider again the fault tree in figure 8.5. A frequency culled BDD will be computed for this fault tree and f_{MIN} will be set to 1×10^{-8}.

Beginning by developing an **ite** structure for gate G3:

$$G3 = ite(b,1,0) \cdot ite(c,1,0)$$

$$G3 = ite(b,[1 \cdot ite(c,10)],[0 \cdot ite(c,1,0)]) \qquad (8.20)$$

Evaluating the one branch of equation (8.20), $p_b = q_b$, $f_b = w_b$, $H = ite(c,1,0)$, $p_H = q_c$, $f_H = w_c$:

The one branch of equation (8.20) is of the form $1 \cdot H$ thus in order to obtain a solution, the frequency of all the combinations that would be encoded if H is taken as the solution to the one branch must be checked to ensure they are not below the culling limit.

$$p_b \cdot f_H = 4 \times 10^{-6} \qquad p_H \cdot f_b = 1.4 \times 10^{-6},$$

Since, $p_b \cdot f_H + p_H \cdot f_b \geq f_{MIN}$ the one branch of equation (8.20) takes the solution H.

Evaluating the zero branch of equation (8.20):

$$0 \cdot ite(c,1,0) = 0 \qquad \boxed{0 \cdot H = 0}$$

The ite structure computed for gate G3 is given below:

$$G3 = ite(b, ite(c,1,0), 0)$$

Now dealing with gate G2:

$$G2 = G3 + ite(e,1,0)$$

$$G2 = ite(b, ite(c,1,0), 0) + ite(e,1,0)$$

$$G2 = ite(b, [ite(c,1,0) + ite(e,1,0)], [0 + ite(e,1,0)]) \qquad (8.21)$$

Evaluating the one branch of equation (8.21):

$$ite(c,1,0) + ite(e,1,0) = ite(c, [1 + ite(e,1,0)], [0 + ite(e,1,0)]) \qquad (8.22)$$

Evaluating the one branch of equation (8.22):

$$1 + ite(e,1,0) = 1 \qquad \boxed{1 + H = 1}$$

Evaluating the zero branch of equation (8.22), $p_{PR,c} = q_b$, $f_{PR,c} = w_b$, $H = ite(e,1,0)$, $p_H = q_e$, $f_H = w_e$:

The zero branch of equation (8.22) is of the form $0 + H$ thus in order to obtain a solution the frequency of all the combinations that would be encoded if H is taken as the solution to the one branch must be checked to ensure they are not below the culling limit.

$$p_{PR,c} \cdot f_H = 4 \times 10^{-9} \qquad p_H \cdot f_{PR,c} = 4 \times 10^{-10},$$

Since, $p_{PR,c} \cdot f_H + p_H \cdot f_{PR,c} < f_{MIN}$ the zero branch of equation (8.22) is terminated by a terminal zero vertex.

Evaluating the zero branch of equation (8.21) $p_{PR,b} = 1$, $f_{PR,b} = 1$, $H = \text{ite}(e,1,0)$, $p_H = q_e$, $f_H = w_e$:

The zero branch of equation (8.21) is of the form $0+H$ thus in order to obtain a solution the frequency of all the combinations that would be encoded if H is taken as the solution to the one branch must be checked to ensure they are not below the culling limit.

$$p_{PR,b} \cdot f_H = 2\times10^{-6} \quad p_H \cdot f_{PR,b} = 2\times10^{-7},$$

Since, $p_{PR,b} \cdot f_H + p_H \cdot f_{PR,b} \geq f_{MIN}$ the zero branch of equation (8.21) takes the solution H.

The **ite** structure computed for gate G2 is given below:

$$G2 = \text{ite}(b,\text{ite}(c,1,0),\text{ite}(e,1,0))$$

Dealing with gate G1:

$$G1 = G2 \cdot \text{ite}(d,1,0)$$

$$G1 = \text{ite}(b,\text{ite}(c,1,0),\text{ite}(e,1,0)) \cdot \text{ite}(d,1,0)$$

$$G1 = \text{ite}\big(b,[\text{ite}(c,1,0)\cdot\text{ite}(d,1,0)][\text{ite}(e,1,0)\cdot\text{ite}(d,1,0)]\big) \qquad (8.23)$$

Evaluating the one branch of equation (8.23):

$$\text{ite}(c,1,0)\cdot\text{ite}(d,1,0) = \text{ite}\big(c,[1\cdot\text{ite}(d,1,0)][0\cdot\text{ite}(d,1,0)]\big) \qquad (8.24)$$

Evaluating the one branch of equation (8.24), $p_c = q_b q_c$, $f_c = q_b w_c + q_c w_b$, $H = \text{ite}(d,1,0)$, $p_H = q_d$, $f_H = w_d$:

The one branch of equation (8.24) is of the form $1\cdot H$ thus in order to obtain a solution, the frequency of all the combinations that would be encoded if H is taken as the solution to the one branch must be checked to ensure they are not below the culling limit.

$$p_c \cdot f_H = 6.2\times10^{-10} \quad p_H \cdot f_c = 2.4\times10^{-9},$$

Since, $p_{PR,b} \cdot f_H + p_H \cdot f_{PR,b} < f_{MIN}$ the one branch of equation (8.24) is terminated with a terminal zero vertex.

Evaluating the zero branch of equation (8.24):

$$0 \cdot ite(d,1,0) = 0 \qquad \boxed{0 \cdot H = 0}$$

Evaluating the zero branch of equation (8.23):

$$ite(e,1,0) \cdot ite(d,1,0) = ite(d,[1 \cdot ite(e,1,0)],[0 \cdot ite(e,1,0)]) \qquad (8.25)$$

Evaluating the one branch of equation (8.25), $p_d = q_d$, $f_d = w_d$, $H = ite(e,1,0)$, $p_H = q_e$, $f_H = w_e$:

The one branch of equation (8.25) is of the form $1 \cdot H$ thus in order to obtain a solution, the frequency of all the combinations that would be encoded if H is taken as the solution to the one branch must be checked to ensure they are not below the culling limit.

$$p_d \cdot f_H = 4.2 \times 10^{-11} \quad p_H \cdot f_d = 6.2 \times 10^{-12},$$

Since, $p_d \cdot f_H + p_H \cdot f_d < f_{MIN}$ the one branch of equation (8.25) is terminated with a terminal zero vertex.

Evaluating the zero branch of equation (8.25):

$$0 \cdot ite(e,1,0) = 0 \qquad \boxed{0 \cdot H = 0}$$

The **ite** structure computed for gate G1 is given below:

$$G1 = ite(b,ite(c,0,0),ite(d,0,0)) = 0$$

Finally dealing with the top gate, Top:

$$Top = 0 + ite(a,1,0)$$

$$Top = ite(a,1,0)$$

Since, $f_a = 6 \times 10^{-3}$ and is thus higher than the culling limit f_{MIN}, the **ite** structure obtained for the top gate does not need culling. The **ite** structure obtained for the frequency culled BDD is given in equation (8.26) and the BDD itself is shown in figure 8.13.

$$Top = ite(a,1,0) \qquad (8.26)$$

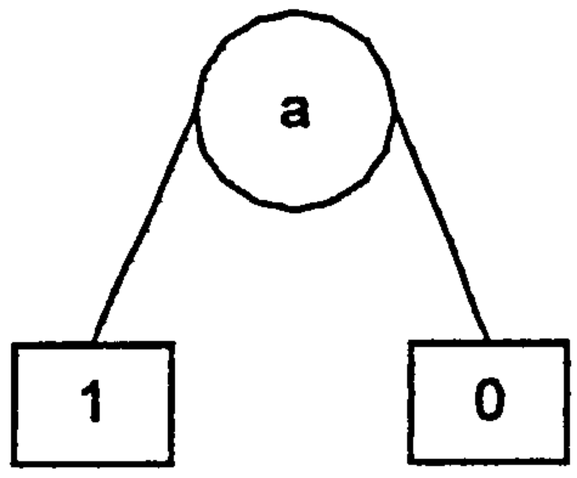


Figure 8.13: Frequency Culled BDD Obtained for the Fault Tree in Figure 8.5 where,

$$f_{MIN} = 1 \times 10^{-8}$$

8.4 Analysing the Culled Binary Decision Diagram

Once the culled BDD has been computed it can be used to identify a partial list of minimal cut sets and also quantify the fault tree approximately. The culled BDD can be used directly for quantification; the techniques outlined in chapter three are applied to calculate the system unavailability, unreliability and the various measures of component importance.

In order to identify the minimal cut sets a minimisation procedure must be applied to the culled BDD to remove any redundancies. The minimal cut sets are then identified by tracing all the terminal 1 paths through the minimised BDD. Each path begins at the root vertex and proceeds through the BDD until a terminal vertex is reached. Only those vertices that lie on the one branch on the way to a terminal 1 vertex are included in the minimal cut set.

Thus consider the culled BDD obtained using the probabilistic culling technique, shown in figure 8.9. This BDD is already minimal so it is possible to obtain a partial list of minimal cut sets exactly. By tracing all the terminal one paths through the culled BDD two minimal cut sets are identified: {a}, {bcd}. It is also possible to quantify the system approximately using the techniques outlined in chapter three.

8.5 Summary

For many larger fault trees with repeated events, full and exact qualitative and quantitative analysis cannot always be performed. This is especially true when employing conventional FTA techniques, which are exhaustive and thus

computationally intensive. Whilst the BDD technique significantly improves the efficiency of analysis it may still not be possible to compute the SFBDD for some very large fault trees.

Culling techniques that can be used in conjunction with conventional FTA techniques have been developed to enable approximate analysis of such fault trees. These techniques are concerned with producing a partial list of minimal cut sets from which approximations can be obtained for the system unavailability and unreliability and the various measures of component and cut set importance. These techniques cull the minimal cut sets according to either:

- A pre-set order
- A pre-set probabilistic value
- A pre-set frequency

Whilst these techniques can be useful, they are still quite inefficient and have the same disadvantages as the conventional FTA techniques. Until recently there were no culling techniques available for the BDD technique, however, in 1998 Rauzy developed two techniques for computing a culled BDD. These techniques again cull the minimal cut sets of a fault tree either according to a pre-set order or probabilistic value. A culled BDD is computed from which a partial list of minimal cut sets can be identified and approximations for the system performance measures can be calculated. An alternative technique was developed by Beeson and Andrews which can be used to compute a frequency culled BDD.

The techniques developed for computing a culled SFBDD are significantly more efficient than the conventional FTA culling techniques, enabling extremely efficient qualitative and quantitative analysis [4]. Furthermore, quantification can be performed using the culled BDD without knowledge of the minimal cut sets.

Chapter 9: Reduction and Culling Techniques for Non-coherent Fault Tree Analysis

9.1 Introduction

The analysis of non-coherent fault tree structures is more complex than the analysis of coherent fault tree structures; this is due to the inclusion of NOT logic. The techniques for analysing such structures are also more computationally intensive, and consequently the application of either reduction or culling techniques to non-coherent fault tree structures may provide the only means of analysing large diagrams with many repeated events.

Reduction techniques are concerned with approximating the structure function of non-coherent fault trees by eliminating all negated literals encountered, thus producing a coherent structure function for the fault tree. Culling techniques on the other hand are concerned with reducing the work required to analyse a fault tree structure by only considering "the most significant" minimal cut sets or prime implicant sets of the system.

Whilst knowledge of the prime implicant sets can be useful, it may be the case that the analyst is concerned only with quantifying the system. Under such circumstances, the inclusion of working states may merely complicate analysis without providing any additional information. This is because the reliability or availability of components tends to be quite high and thus close to 1, hence provided this assumption holds the exclusion of negated literals has little impact during quantification. In such cases a reduction technique that computes a coherent approximation for the system can provide a useful means of partial analysis.

Conventional FTA techniques can be employed in conjunction with a reduction technique to identify a full list of minimal p-cuts. The combinations identified can then be used to quantify the system approximately. The BDD method is significantly more efficient than conventional FTA methods and lends itself to reduction techniques. If knowledge of the prime implicant sets is not vital then it is possible to obtain a full list of minimal p-cuts from the minimised SFBDD, it is also possible to use the

unminimised BDD to quantify the system exactly using the techniques outlined in chapters five and seven.

If however a partial list of the prime implicant sets is required additional work must be carried out. Culling techniques can be employed in conjunction with conventional FTA methods to cull the prime implicant sets according to order, probability, or frequency. Culling techniques can also be applied in conjunction with the BDD method. It is not possible to cull the SFBDD according to order or probability and then compute the meta-products BDD to produce a partial list of prime implicant sets. Instead the SFBDD must be calculated as normal and then a culling operation can be applied to compute a culled meta-products BDD. Rauzy and Dutuit developed an order culling technique, to produce a meta-products BDD, which encodes a partial list of prime implicant sets according to a pre-set order [6]. An alternative technique has been developed as part of this research project, which can be employed to produce a culled meta-products BDD encoding all prime implicant sets with a probability or frequency greater than a pre-set value.

This chapter will consider each of the reduction and culling techniques outlined above in detail, and a worked example will be used to illustrate how they are implemented.

9.2 Reduction Techniques for Conventional Fault Tree Analysis Methods

If knowledge of the prime implicant sets is not required then obtaining a coherent approximation during qualitative analysis can significantly reduce the work required to analyse the system. This involves eliminating all negated literals from the Boolean expression obtained for the top event. A full list of minimal p-cuts can be identified from this expression and subsequently used to quantify the system approximately. Since, the reliability and availability of components tends to be close to 1 the results obtained during quantification are generally accurate. This technique will not be illustrated here, since it was considered in detail in chapter four, section 4.3.3.

9.3 Reduction Technique for the Binary Decision Diagram Method

Chapter five outlined how the minimised SFBDD computed for a non-coherent fault tree can be used to compute a full list of minimal p-cuts. This technique is significantly more efficient than the equivalent FTA technique outlined above in section 9.2. Furthermore, since the computed SFBDD encodes the structure function of the system, it is still possible to use it to perform full and exact quantification, using the techniques outlined in chapters five and seven. This culling technique is particularly useful if knowledge of the prime implicant sets is not required. This technique will not be illustrated again here.

9.4 Culling Techniques for Conventional Fault Tree Analysis Methods

The culling techniques introduced in chapter eight for application with conventional FTA methods can be employed in the same way for analysing non-coherent fault tree structures, to produce a partial list of prime implicant sets according to:

- A pre-set order
- A pre-set probability
- A pre-set frequency

Each of the three culling techniques will be considered in detail in sections 9.4.1-9.4.3 and illustrated by means of a worked example.

9.4.1 Culling Prime Implicant Sets Above a Given Order

This technique produces a partial list of prime implicant sets according to a pre-set order, k_{MAX}. The Boolean expression for the top gate is developed using a traditional top-down or bottom up approach, however, at each stage in the development any combination whose order exceeds k_{MAX} is culled from the expression. Once the culled Boolean expression for the top gate has been obtained, the consensus theorem can be applied to obtain a more complete list of prime implicant sets. However, hidden prime implicant sets are not a requirement for exact quantification.

To illustrate this technique consider the fault tree in figure 9.1, prime implicant sets of order three and below will be identified.

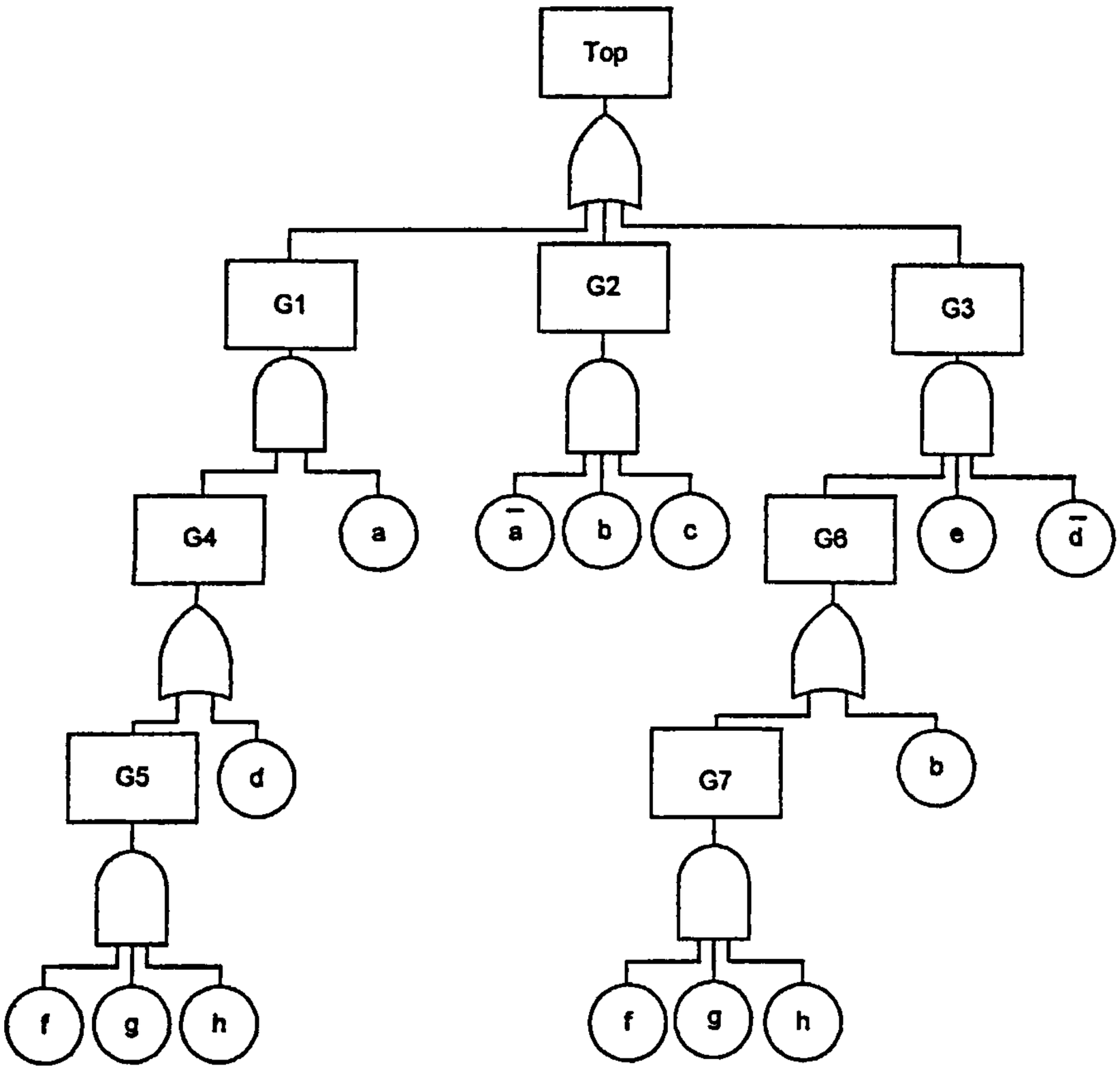


Figure 9.1: Non-coherent Fault Tree Structure

A Boolean expression for the top event will be obtained using the bottom-up approach outlined in chapter two. From this a partial list of prime implicant set of order three and below can be identified.

Beginning by developing an expression for gate G1, by dealing with gates, G5 and G4:

$$G5 = f \cdot g \cdot h$$

$$G4 = G5 + d = f \cdot g \cdot h + d$$

From this the following expression is obtained for gate G1:

$$G1 = G4 \cdot a$$

$$= (f \cdot g \cdot h + d) \cdot a$$

$$= a \cdot d$$

The combination $a \cdot f \cdot g \cdot h$ is eliminated from the Boolean expression for gate G1 because it exceeds the culling order.

Now obtaining an expression for gate G2:

$$G2 = \bar{a} \cdot b \cdot c$$

Next an expression for gate G3 is obtained by developing gates, G7 and G6:

$$G7 = f \cdot g \cdot h$$
$$G6 = G7 + b = f \cdot g \cdot h + b$$

The following expression is obtained for gate G3:

$$G3 = G6 \cdot \bar{d} \cdot e$$
$$= (f \cdot g \cdot h + b) \cdot \bar{d} \cdot e$$
$$= b \cdot d \cdot e$$

The combination $\bar{d} \cdot e \cdot f \cdot g \cdot h$ is eliminated from the Boolean expression for gate G3 since it exceeds the culling order.

Finally the following Boolean expression is obtained for the top gate, Top:

$$\begin{aligned} Top &= G1 + G2 + G3 \\ &= a \cdot d + \bar{a} \cdot b \cdot c + b \cdot \bar{d} \cdot e \end{aligned} \tag{9.1}$$

Three prime implicant sets are identified from equation (9.1):

$$\{ad\}, \{\bar{a}\,bc\}, \{b\,\bar{d}\,e\}$$

Culling techniques aim to identify those combinations that are most likely to contribute to system failure. In general for coherent fault trees the lower the order of a minimal cut set the more likely it is to contribute to system failure. However, this is not the case for prime implicant sets. For example, suppose that a system has two prime implicant sets, {abc}, $\{\bar{a}\,\bar{c}\,\bar{d}\,e\}$, assigning the following component failure probabilities,

$$q_a=1\times10^{-3},\ q_b=2.5\times10^{-4},\ q_c=3\times10^{-3},\ q_d=2\times10^{-5},\ q_e=1.5\times10^{-3}$$

The probability of existence of each prime implicant set is given below:

$$P(abc)=q_a q_b q_c=7.5\times10^{-10}$$

$$P(\bar{a}\,\bar{c}\,\bar{d}\,e)=p_a p_c p_d q_e=1.49\times10^{-3}$$

The probability of existence of the higher order prime implicant set is significantly greater than that of the lower order set, this is due to the inclusion of working states, whose probabilities are generally close to 1. Hence if the prime implicant sets are culled according to their order, it is highly likely that some significant combinations will be ignored. When dealing with non-coherent structures, techniques that cull the

prime implicant sets according to their probability or frequency tend to identify the most significant causes of system failure more successfully than order culling techniques.

9.4.2 Culling Prime Implicant Sets Below a Given probability

The prime implicant sets can only be culled according to their probability if the component failure probabilities are known. The first stage of this culling technique is to set the probabilistic value, p_{MIN}, according to which the prime implicant sets will be culled. Then the top-down or bottom-up approach is used to identify a partial list of prime implicant sets. As the Boolean expression for the top event is developed, any combinations that have a probabilistic value below the pre-set value are eliminated from the expression. The final expression obtained for the top event only contains the prime implicant sets with a probabilistic value above p_{MIN}.

To illustrate this technique consider the fault tree given in figure 9.1. The probabilistic culling limit will be set to $1x10^{-10}$, thus any prime implicant sets with a probabilistic value below this culling limit will be eliminated from the Boolean expression for the top event. The component failure probabilities for each component in this example are given in table 9.1.

Component	Failure Probability	Failure Frequency	Repair Frequency
a	$1x10^{-7}$	$1x10^{-8}$	$1x10^{-4}$
b	$1x10^{-4}$	$2x10^{-5}$	$2x10^{-3}$
c	$2x10^{-3}$	$5x10^{-5}$	$3x10^{-3}$
d	$2.5x10^{-6}$	$3x10^{-8}$	$2x10^{-5}$
e	$1x10^{-2}$	$2x10^{-3}$	$1x10^{-2}$
f	$2x10^{-2}$	$1x10^{-4}$	$2x10^{-3}$
g	$3x10^{-3}$	$1x10^{-2}$	$3x10^{-3}$
h	$2.5x10^{-3}$	$2x10^{-3}$	$3x10^{-4}$

Table 9.1: Component Failure Probabilities and Failure and Repair Frequencies

Beginning by developing an expression for G1 by evaluating, G5, G4:

$G5 = f \cdot g \cdot h$

$P(f \cdot g \cdot h) = (2 \times 10^{-2})(3 \times 10^{-3})(2.5 \times 10^{-3})$

$= 1.5 \times 10^{-7}$

The probability of this combination is higher than the culling limit of $1x10^{-10}$.

$$G4 = G5 + d = f \cdot g \cdot h + d$$

$$P(f \cdot g \cdot h) = 1.5 \times 10^{-7}$$

$$P(d) = 2.5 \times 10^{-6}$$

The probabilities of both of these combinations are higher than the culling limit.

Now an expression can be obtained for gate G1:

$$G1 = G4 \cdot a = (f \cdot g \cdot h + d)a = a \cdot f \cdot g \cdot h + a \cdot d$$

$$P(a \cdot f \cdot g \cdot h) = (1 \times 10^{-7})(1.5 \times 10^{-7})$$
$$= 1.5 \times 10^{-14}$$

$$P(a \cdot d) = (1 \times 10^{-7})(2.5 \times 10^{-6})$$
$$= 2.5 \times 10^{-13}$$

The probability of both of these combinations are below the probabilistic culling limit, $1x10^{-10}$ hence they are eliminated from the expression for gate G1, the culled expression for this gate is given below:

$$G1 = 0$$

Now developing an expression for gate G2:

$$G2 = \bar{a} \cdot b \cdot c$$

$$P(\bar{a} \cdot b \cdot c) = (1 - 1 \times 10^{-7})(1 \times 10^{-4})(2 \times 10^{-3})$$
$$= 2 \times 10^{-7}$$

The probability of this combination is higher than the culling limit.

By evaluating gates, G7 and G6 it is possible to obtain an expression for gate G3:

$$G7 = f \cdot g \cdot h$$

$$P(f \cdot g \cdot h) = 1.5 \times 10^{-7}$$

The probability of this combination is higher than the culling limit.

$$G6 = G7 + b = f \cdot g \cdot h + b$$

$$P(f \cdot g \cdot h) = 1.5 \times 10^{-7}$$

$$P(b) = 1 \times 10^{-4}$$

The probabilities of both of these combinations are higher than the culling limit.

From these results the following expression is obtained for gate G3:

$$G3 = G6 \cdot \bar{d} \cdot e = (f \cdot g \cdot h + b) \cdot \bar{d} \cdot e = \bar{d} \cdot e \cdot f \cdot g \cdot h + b \cdot \bar{d} \cdot e$$

$$P(\bar{d}\cdot e\cdot f\cdot g\cdot h)=(1-2.5\times10^{-6})(1\times10^{-2})(1.5\times10^{-7})$$
$$=1.5\times10^{-9}$$
$$P(b\cdot\bar{d}\cdot e)=(1\times10^{-4})(1-2.5\times10^{-6})(1\times10^{-2})$$
$$=1\times10^{-6}$$

The probability of both of these combinations is above the culling limit; hence the expression for gate G3 is unchanged.

Finally an expression for the top event is obtained:

$$Top=G1+G2+G3=\bar{a}\cdot b\cdot c+\bar{d}\cdot e\cdot f\cdot g\cdot h+b\cdot\bar{d}\cdot e$$

The probabilities of all of the combinations in the Boolean expression for the top event are higher than the culling limit of $1x10^{-10}$. Hence no further culling is required the final Boolean expression for the top event is given below:

$$Top=\bar{a}\cdot b\cdot c+\bar{d}\cdot e\cdot f\cdot g\cdot h+b\cdot\bar{d}\cdot e \qquad (9.2)$$

Three prime implicant sets are identified from equation (9.2):

{a bc}, {defgh}, {bde}

The prime implicant sets identified can be used to quantify the system approximately. The conventional quantification techniques outlined in chapter four and seven are employed in exactly the same way.

9.4.3 Culling Prime Implicant Sets Below a Given Frequency

This culling technique is very similar to the probabilistic culling procedure outlined in section 9.4.2; the only difference is that the prime implicant sets are culled according to a pre-set frequency rather than probability. Once again the frequency value must be set according to which the prime implicant sets will be culled. Then the top-down or bottom-up approach is used to identify a partial list of prime implicant sets. As the Boolean expression for the top event is developed, any combinations that have a frequency below the pre-set value are eliminated from the expression. The final expression obtained for the top event only contains the prime implicant sets with a frequency above the pre-set frequency value. Component failure probabilities and frequencies are essential to employ this technique.

To illustrate this technique consider again the fault tree given in figure 9.1. The frequency culling limit will be set to 1×10^{-8}, thus any prime implicant sets with a frequency below this culling limit will be eliminated from the Boolean expression for the top event. The component failure probabilities and frequencies for each component in this example are given in table 9.1.

Beginning by developing an expression for G1 by evaluating, G5, G4:

$$G5 = f \cdot g \cdot h$$

$$q_f q_g w_h + q_f q_h w_g + q_g q_h w_f = 1.53\times10^{-7}$$

The frequency of this combination is higher than the culling limit of 1×10^{-8}.

$$G4 = G5 + d = f \cdot g \cdot h + d$$

$$q_f q_g w_h + q_f q_h w_g + q_g q_h w_f = 1.53\times10^{-7}$$

$$w_d = 3\times10^{-8}$$

The frequencies of these combinations are higher than the culling limit.

Now an expression can be obtained for gate G1:

$$G1 = G4 \cdot a = (f \cdot g \cdot h + d) \cdot a = a \cdot f \cdot g \cdot h + a \cdot d$$

$$q_a q_f q_g w_h + q_a q_f q_h w_g + q_a q_g q_h w_f + q_f q_g q_h w_a = 6.4\times10^{-14}$$

$$q_a w_d + q_d w_a = 2.8\times10^{-14}$$

The frequency of both of these combinations are below the culling limit, 1×10^{-8} hence they are eliminated from the expression for gate G1, the culled expression for this gate is given below:

$$G1 = 0$$

Now developing an expression for gate G2:

$$G2 = \bar{a} \cdot b \cdot c$$

$$p_a q_b w_c + p_a q_c w_b + q_b q_c v_a = 4.5\times10^{-8}$$

The frequency of this combination higher than the culling limit.

By evaluating gates, G7 and G6 it is possible to obtain an expression for gate G3:

$$G7 = f \cdot g \cdot h$$

$$q_f q_g w_h + q_f q_h w_g + q_g q_h w_f = 1.53 \times 10^{-7}$$

The frequency of this combination is higher than the culling limit.

$$G6 = G7 + b = f \cdot g \cdot h + b$$

$$q_f q_g w_h + q_f q_h w_g + q_g q_h w_f = 1.53 \times 10^{-7}$$

$$w_b = 2 \times 10^{-5}$$

The frequencies of these combinations are higher than the culling limit.

From these results the following expression is obtained for gate G3:

$$G3 = G6 \cdot \bar{d} \cdot e = (f \cdot g \cdot h + b) \cdot \bar{d} \cdot e = \bar{d} \cdot e \cdot f \cdot g \cdot h + b \cdot \bar{d} \cdot e$$

$$p_d q_e q_f q_g w_h + p_d q_e q_f q_h w_g + p_d q_e q_g q_h w_f + p_d q_f q_g q_h w_e + q_e q_f q_g q_h v_d = 1.22 \times 10^{-8}$$

$$q_b p_d w_e + q_b q_e v_d + p_d q_e w_b = 2.2 \times 10^{-7}$$

The frequencies of these combinations are higher than the culling limit.

Finally an expression for the top event is obtained:

$$Top = G1 + G2 + G3 = \bar{a} \cdot b \cdot c + \bar{d} \cdot e \cdot f \cdot g \cdot h + b \cdot \bar{d} \cdot e$$

The frequency of all of the combinations in the Boolean expression for the top event are higher than the culling limit of 1×10^{-8}. Hence no further culling is required, the final Boolean expression for the top event is given below:

$$Top = \bar{a} \cdot b \cdot c + \bar{d} \cdot e \cdot f \cdot g \cdot h + b \cdot \bar{d} \cdot e \qquad (9.3)$$

Three prime implicant sets are identified from equation (9.3):

$$\{\bar{a}\, bc\}, \{\bar{d}\, efgh\}, \{b\, \bar{d}\, e\}$$

The prime implicant sets identified can be used to quantify the system approximately. The conventional quantification techniques outlined in chapter four and seven are employed in exactly the same way.

Although these culling techniques can significantly reduce the work required during quantification, they still provide an inefficient means of analysis. Consequently the prime implicant sets may need to be culled with greater severity than required, which can diminish the accuracy of the results obtained during quantification.

9.5 Culling Techniques for The Binary Decision Diagram Method

The BDD technique is significantly more efficient than conventional FTA methods; hence, it would be beneficial to employ the BDD technique to produce a partial list of prime implicant sets.

Under certain circumstances knowledge of the prime implicant sets may not be required, if this is the case it is possible to obtain a *full list of minimal p-cuts from the* minimised SFBDD. It is also possible to use this BDD to quantify the system exactly and efficiently, using the techniques outlined in chapters five and seven.

However, knowledge of the prime implicant sets can be helpful when trying to design safety systems in order to reduce the risk of system failure. Rauzy and Dutuit developed a culling technique, which can be employed in conjunction with the meta-products algorithm to produce an order culled meta-products BDD. An alternative technique, which is again employed in conjunction with the meta-products algorithm, has been developed. This technique produces a culled meta-products BDD according to a pre-set probability or frequency. This BDD encodes all those prime implicant sets with a probability or frequency equal to or above the pre-set value.

9.5.1 Rauzy and Dutuit's Order Culling Technique for the Meta-products Binary Decision Diagram

Rauzy and Dutuit developed an algorithm to compute a culled meta-products BDD, which encodes all the prime implicant sets below a given pre-set order [6]. To identify a partial list of prime implicant sets according to a pre-set culling order, it is necessary to compute the SFBDD of the non-coherent fault tree exactly and then compute a culled meta-products BDD. If the SFBDD is culled some hidden prime implicant sets within the culling order may be missed.

The algorithm for encoding the culled meta-products BDD is similar to that employed to compute the exact meta-products BDD, however, it is modified slightly to ensure that prime implicant sets exceeding the culling order are not encoded in the resulting BDD. The pre-set culling order is denoted by k_{MAX}, if k_{MAX} is set to three, only prime implicant sets of order three and below are encoded. The algorithm for computing the order culled meta-products BDD is outlined below.

The variable k is used to track the remaining number of basic events that can be encoded in each path through the meta-products BDD before the culling order is exceeded. Initially k is set to k_{MAX}, then given a node with the **ite** structure:

$$ite(x_i,f1,f0)$$

The meta-products structure of this node is denoted as follows:

$$PI[ite(x_i,f1,f0),\ L,\ k]$$

Where:

L denotes the list of all ordered basic events except for those that appear on the current path from the root node to the node under consideration.

And k denotes the number of basic events that can be encoded on the current path before the culling order is exceeded.

$PI[ite(x_i,f1,f0),\ L,\ k]$ is evaluated according to the following rules:

- If x_i is the first basic event in L:

$$PI[ite(x_i,f1,f0),\ L,\ k] = ite(P_{x_i}, ite(S_{x_i},P1,P0), P2)$$

Where:

$$P2 = PI[f1 \cdot f0,\ L',\ k]$$

$$P1 = PI[f1,\ L',\ k-1]$$

$$P0 = PI[f0,\ L',\ k-1]$$

Note: k is reduced by 1 for both P1 and P0 this is because the basic event x_i is already encoded in these paths through the meta-products BDD. However, k remains the same for P2 since x_i is not encoded in this path through the meta-products BDD. And $L' = x_{i+1}, x_{i+2}, \ldots, x_n$.

- If x_i is not the first basic event in L, i.e. $L = x_j, x_{j+1}, \ldots, x_n$ such that $i > j$:

$$PI[ite(x_i, f1, f0),\ L,\ k] = ite(P_{x_j}, 0, PI[ite(x_{j+1}, f1, f0),\ L,\ k])$$

Note: k is not reduced since the basic event x_i is not actually encoded in the current path.

These rules are applied in conjunction with the following identities:

$PI[0,\ L,\ k] = 0$

$PI[1,\ x \cdot L,\ k] = ite(Px, 0, PI[1,\ L,\ k])$

$PI[f,\ L,\ 0] = 0$

Notice the additional identity, $PI[f,\ L,\ 0] = 0$, in this case, k is equal to 0 signifying that the culling limit has been reached, hence if any more basic events are encoded on the current path, prime implicant sets exceeding the culling order will be encoded in the meta-products BDD.

To illustrate how this algorithm is applied in practise consider the non-coherent fault tree given in figure 9.2, only those prime implicants of order two or below will be identified.

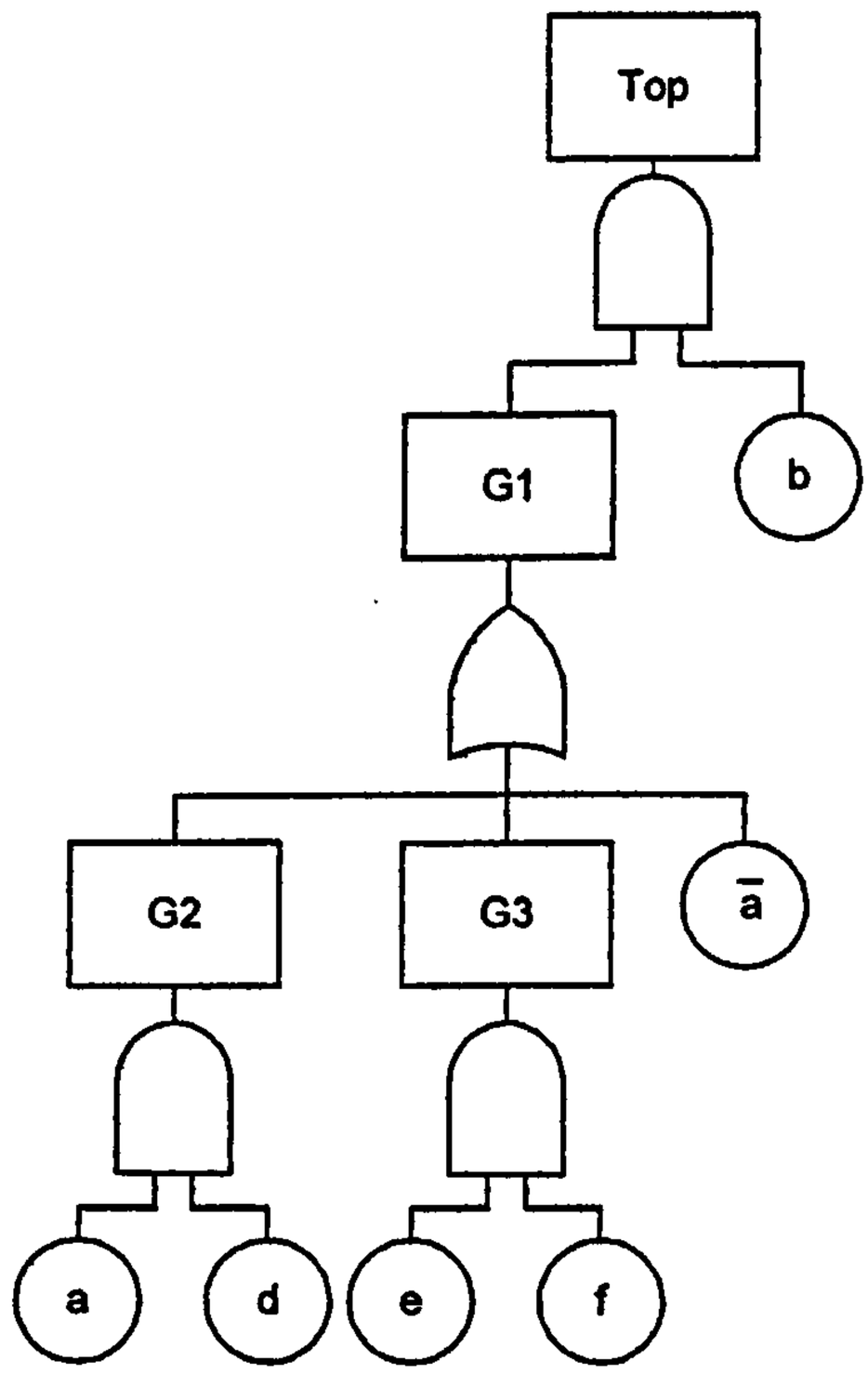


Figure 9.2: Non-coherent Fault Tree Structure

If the order $b < a < d < e < f$ is adopted for the basic events of this fault tree the following ite structure is obtained for the top gate:

$$Top = ite(b, ite(a, ite(d, 1, ite(e, ite(f, 1, 0), 0)), 1), 0)$$

To encode the order two culled meta-products BDD the following meta-products structure must be computed:

$$PI[ite(b, ite(a, ite(d, 1, ite(e, ite(f, 1, 0), 0)), 1), 0)\ , badef,\ 2] = ite(P_b, ite(S_b, P1, P0), P2)$$

Where:

$P2 = PI[0,\ \ adef, 2] = 0 \quad \therefore \overline{P2} = 1$

$P1 = PI[ite(a, ite(d, 1, ite(e, ite(f, 1, 0), 0)), 1),\ \ adef,\ 1] \cdot \overline{P2}$

$P0 = PI[0,\ \ adef,\ 2] \cdot \overline{P2} = 0$

Evaluating P1

$P1 = PI[ite(a, ite(d, 1, ite(e, ite(f, 1, 0), 0)), 1),\ \ adef,\ 1] \cdot \overline{P2}$

$= ite(P_a, ite(S_a, P1.1, P0.1), P2.1)$

$= ite(P_a, ite(S_a, 0, ite(P_d, 0, ite(P_e, 0, ite(P_f, 0, 1)))), ite(P_d, ite(S_d, ite(P_e, 0, ite(P_f, 0, 1)), 0), 0))$

Where:

$P2.1 = PI[ite(d, 1, ite(e, ite(f, 1, 0), 0)),\ def,\ 1]$

$P1.1 = PI[ite(d, 1, ite(e, ite(f, 1, 0), 0)),\ def,\ 0] \cdot \overline{P2.1} = 0$ This is terminated because further development would result in combinations that would exceed the culling limit.

$P0.1 = PI[1,\ def,\ 0] \cdot \overline{P2.1}$

Evaluating P2.1

$P2.1 = PI[ite(d, 1, ite(e, ite(f, 1, 0), 0)),\ def,\ 1]$

$= ite(P_d, ite(S_d, P1.2, P0.2), P2.2)$

$= ite(P_d, ite(S_d, ite(P_e, 0, ite(P_f, 0, 1)), 0), 0)$

Where:

$P2.2 = PI[ite(e,ite(f,1,0),0),\ ef,\ 1]$

$P1.2 = PI[1,\ ef,\ 0] \cdot \overline{P2.2}$

$P0.2 = PI[ite(e,ite(f,1,0),0),\ ef,\ 0] \cdot \overline{P2.2} = 0$

This is terminated because further development would result in combinations that would exceed the culling limit.

Evaluating P2.2

$$P2.2 = PI[ite(e,ite(f,1,0),0),\ ef,\ 1]$$
$$= ite(P_e, ite(S_e, P1.3, P0.3), P2.3)$$
$$= ite(P_e, ite(S_e, 0, 0), 0)$$
$$= 0$$

Where:

$P2.3 = PI[0,\ f,\ 0] = 0 \quad \therefore\ \overline{P2.3} = 1$

$P1.3 = PI[ite(f,1,0),\ f,\ 0] \cdot \overline{P2.3} = 0$

This is terminated because further development would result in combinations that would exceed the culling limit.

$P0.3 = PI[0,\ f,\ 0] \cdot \overline{P2.3} = 0$

Evaluating P1.2

$$P1.2 = PI[1,\ ef,\ 0] \cdot \overline{P2.2}$$
$$= ite(P_e, 0, ite(P_f, 0, 1)) \cdot 1$$
$$= ite(P_e, 0, ite(P_f, 0, 1))$$

Evaluating P0.1

$$P0.1 = PI[1,\ def,\ 0] \cdot \overline{P2.1}$$
$$= ite(P_d, 0, ite(P_e, 0, ite(P_f, 0, 1))) \cdot ite(P_d, ite(S_d, ite(P_e, 1, ite(P_f, 1, 0)), 1), 1)$$
$$= ite(P_d, 0, ite(P_e, 0, ite(P_f, 0, 1)))$$

The final meta-products structure obtained is given below in equation (9.4) and the meta-products BDD is shown in figure 9.3:

$$\begin{aligned}\text{Top} = \text{ite}(P_b, \text{ite}(S_b, (P_a, \text{ite}(S_a, 0, \text{ite}(P_d, 0, \text{ite}(P_e, 0, \text{ite}(P_f, 0, 1))))) \\ , \text{ite}(P_d, \text{ite}(S_d, \text{ite}(P_e, 0, \text{ite}(P_f, 0, 1)), 0), 0)), 0), 0)\end{aligned} \tag{9.4}$$

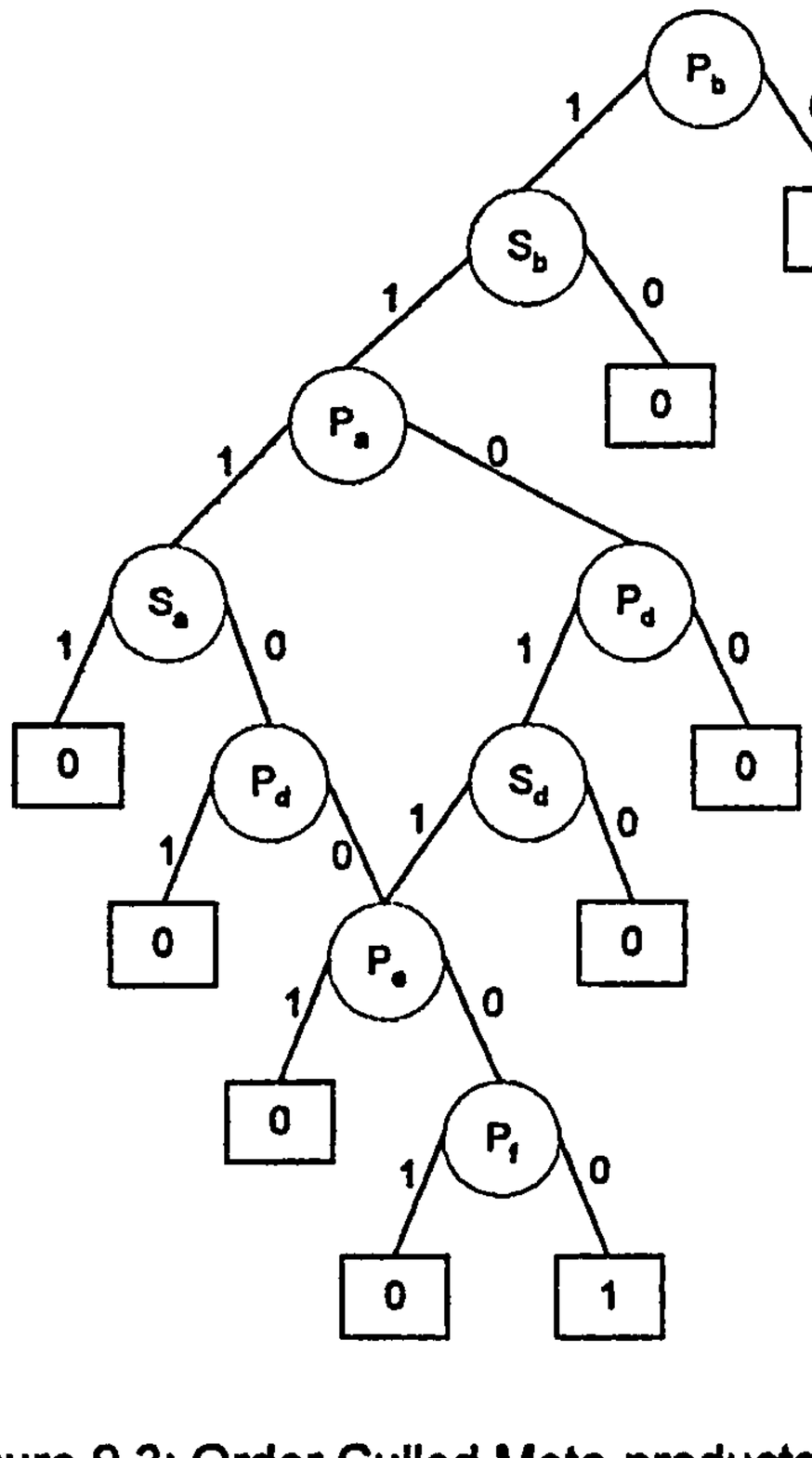


Figure 9.3: Order Culled Meta-products BDD

From the meta-products BDD shown in figure 9.3 it is possible to obtain the meta-products, by tracing the terminal one paths through this BDD, the prime implicant sets are subsequently identified from the meta-products as shown below.

$$P_b \wedge S_b \wedge P_a \wedge \bar{S}_a \wedge \bar{P}_d \wedge \bar{P}_e \wedge \bar{P}_f = \bar{a}b$$

$$P_b \wedge S_b \wedge \bar{P}_a \wedge P_d \wedge S_d \wedge \bar{P}_e \wedge \bar{P}_f = bd$$

9.5.2 Probability or Frequency Culling Techniques for the Meta-products Binary Decision Diagram

In section 9.4.1 It was shown that culling the prime implicant sets according can order can result in some significant combinations being ignored. When dealing with non-coherent fault tree structures it is better to cull the prime implicant sets according

to probability or frequency, since higher order prime implicant sets containing negated literals can be significantly more likely to contribute to system failure than other lower order prime implicant sets.

An alternative culling technique has been developed as part of this research project, which encodes a probability or frequency, culled BDD. From which a partial list of prime implicant sets can be identified.

The algorithm for encoding a probability or frequency culled meta-products BDD is similar to the algorithm outlined above for computing an order-culled meta-products BDD. The order algorithm is modified such that prime implicant sets are culled according to a pre-set probability, p_{MIN} or frequency, f_{MIN}. The algorithms for probability and frequency are identical except when computing a probability (frequency) culled BDD p denotes the current probability (frequency) of the combinations of variables included in the current path from the root vertex to the current node. The algorithm is outlined below:

Initially p is set to 1, then given a node with **ite** structure:

$$ite(x_i, f1, f0)$$

The meta-products structure of this node is denoted as follows:

$$PI[ite(x_i, f1, f0),\ L,\ p]$$

Where:

L denotes the list of all ordered basic events except for those that appear on the current path from the root node to the node under consideration.

And p denotes the probability or frequency of the combinations of variables encoded in the path from the root vertex to the current node.

$PI[ite(x_i, f1, f0),\ L,\ p]$ is evaluated according to the following rules:

- If x_i is the first basic event in L:

$$PI[ite(x_i, f1, f0),\ L,\ p] = ite(P_{x_i}, ite(S_{x_i}, P1, P0), P2)$$

Where:

$$P2 = PI[f1 \cdot f0,\ L',\ p]$$

$$P1 = PI[f1,\ L',\ p \cdot q_{x_i} \text{ or } p \cdot w_{x_i} + q_{x_i} \cdot w_p]$$

$$P0 = PI[f0,\ L',\ p\cdot(1-q_{x_i})\ \text{or}\ p\cdot v_{x_i} + p_{x_i}\cdot w_p]$$

Note: The probability or frequency value for P1 becomes $p\cdot q_{x_i}$ or $p\cdot w_{x_i} + q_{x_i}\cdot w_p$ since x_i belongs to the prime implicant set encoded in the current path through the meta-products BDD. Similarly the probability or frequency value for P0 becomes $p\cdot(1-q_{x_i})$or $p\cdot v_{x_i} + p_{x_i}\cdot w_p$ since $\bar{x}_i$ belongs to the prime implicant set encoded in the current path through the meta-products BDD.

- If x_i is not the first basic event in L, i.e. $L=x_j,x_{j+1},\ldots,x_n$ such that $i > j$:

$$PI[ite(x_i, f1, f0),\ L,\ p] = ite(P_{x_j}, 0, PI[ite(x_{j+1}, f1, f0)\ \ L,\ p])$$

Note: p is not changed because the basic event x_j is not actually encoded in the current path.

These rules are applied in conjunction with the following identities:

$PI[0,\ L,\ p] = 0$

$PI[1,\ x\cdot L,\ p] = ite(Px, 0, PI[1,\ L,\ p])$

$PI[f,\ L,\ p] = 0 \ \text{ if } \ p < p_{MIN} \ \text{ or } \ p < f_{MIN}$

Notice the additional identity, $PI[f,\ L,\ p] = 0 \ \text{ if } \ p < p_{MIN} \ \text{ or } \ p < f_{MIN}$, in this case, the probability or frequency of the prime implicant set encoded in the current path through the meta-products BDD is below the pre-set probability or frequency. Hence, if any more basic events are encoded on the current path, prime implicant sets with a probability or frequency below the pre-set limit will be encoded in the meta-products BDD.

To illustrate how this algorithm is applied in practise consider the non-coherent fault tree given in figure 9.4, only those prime implicant sets with a probability above 5×10^{-8} will be identified. Table 9.2 records the failure probability of each basic event in the fault tree.

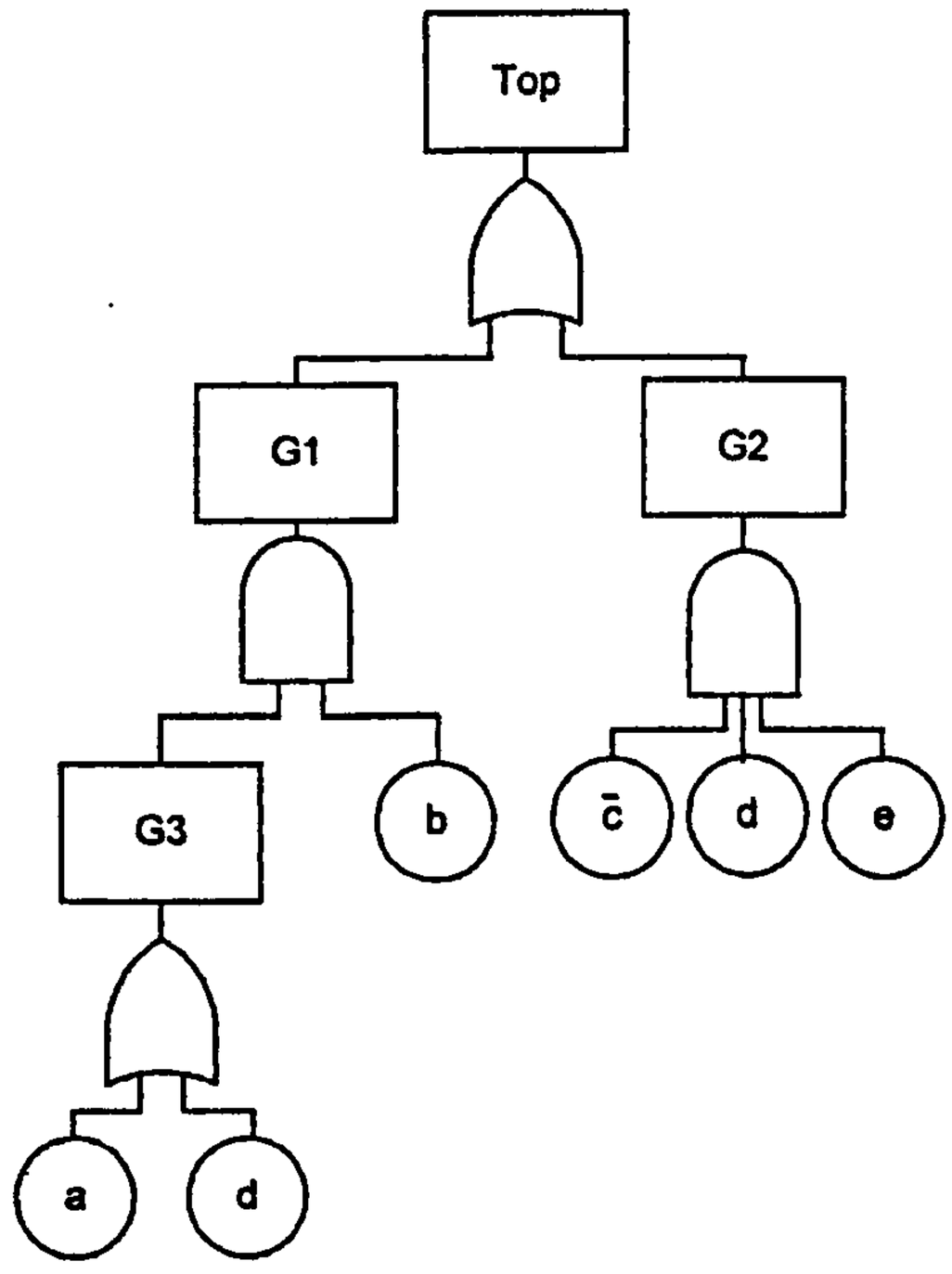


Figure 9.4: Non-coherent Fault Tree Structure

Basic Event	Failure Probability
a	1×10^{-3}
b	2×10^{-4}
c	2×10^{-7}
d	2×10^{-5}
e	1×10^{-2}

Table 9.2: Failure Probabilities for Each Basic Event in Figure 9.4

If the order $b < d < a < c < e$ is adopted for the basic events of this fault tree the following **ite** structure is obtained:

$$Top = ite(b, ite(d,1, ite(a,1,0)), ite(d, ite(c,0, ite(e,1,0)),0))$$

To encode the probability culled meta-products BDD the following meta-products structure must be computed:

$$PI[ite(b, ite(d,1, ite(a,1,0)), ite(d, ite(c,0, ite(e,1,0)),0)),\ bdace, 1] = ite(P_b, ite(S_b, P1, P0), P2)$$

Where:

$$P2 = PI[ite(d, ite(c,0, ite(e,1,0)),0),\ dace,\ 1]$$

$$P1 = PI[ite(d,1, ite(a,1,0)),\ dace,\ 2\times10^{-4}]\cdot\overline{P2}$$

$$P0 = PI[ite(d, ite(c,0, ite(e,1,0)),0),\ dace,\ 0.9998]\cdot\overline{P2}$$

Evaluating P2

$$P2 = PI[ite(d, ite(c,0, ite(e,1,0)),0),\ dace,\ 1]$$
$$= ite(P_d, ite(S_d, P1.1, P0.1), P2.1)$$
$$= ite(P_d, ite(S_d, ite(P_c, ite(S_c, 0, ite(P_e, ite(S_e,1,0),0)),0),0),0),0)$$

Where:

$$P2.1 = PI[0,\ ace,\ 1] = 0 \quad \therefore\ \overline{P2.1} = 1$$

$$P1.1 = PI[ite(c,0, ite(e,1,0)),\ ace,\ 2\times10^{-5}]\cdot\overline{P2.1}$$

$$P0.1 = PI[0,\ ace,\ 0.99998]\cdot\overline{P2.1} = 0$$

Evaluating P1.1

$$P1.1 = PI[ite(c,0, ite(e,1,0)),\ ace,\ 2\times10^{-5}]\cdot\overline{P2.1}$$
$$= ite(P_a, 0, P2.2)$$
$$= ite(P_a, 0, ite(P_c, ite(S_c, 0, ite(P_e, ite(S_e,1,0),0)),0))$$

Where:

$$P2.2 = PI[ite(c,0, ite(e,1,0)),\ ce,\ 2\times10^{-5}]$$

Evaluating P2.2

$$P2.2 = PI[ite(c,0, ite(e,1,0)),\ ce,\ 2\times10^{-5}]$$
$$= ite(P_c, ite(S_c, P1.3, P0.3), P2.2)$$
$$= ite(P_c, ite(S_c, 0, ite(P_e, ite(S_e,1,0),0)),0)$$

Where:

$$P2.3 = PI[0,\ e,\ 2\times10^{-5}] = 0 \quad \therefore\ \overline{P2.3} = 1$$

$$P1.3 = PI[0,\ e,\ 4\times10^{-12}]\cdot\overline{P2.3} = 0$$

$$P0.3 = PI[ite(e,1,0),\ e,\ 2\times10^{-5}]\cdot\overline{P2.3} = ite(P_e, ite(S_e,1,0),0)$$

Evaluating P1

$$P1 = PI[ite(d,1,ite(a,1,0)),\ dace,\ 2\times10^{-4}]\cdot\overline{P2}$$
$$= ite(P_d, ite(S_d,P1.4,P0.4),P2.4)\cdot ite(P_d, ite(S_d, ite(P_a,0, ite(P_c, ite(S_c,1, ite(P_e, ite(S_e,0,1),1)),1),1)),1)$$
$$= ite(P_d, ite(S_d,0,0), ite(P_a, ite(S_a, ite(P_c,0, ite(P_e,0,1)),0),0))$$

Where:

$$P2.4 = PI[ite(a,1,0),\ ace,\ 2\times10^{-4}]$$

$$P1.4 = PI[1,\ ace,\ 2\times10^{-9}]\cdot\overline{P2.4} = 0$$

This combination is terminated because the probability of the basic events encoded in the current path have a probability below the culling limit.

$$P0.4 = PI[ite(a,1,0),\ ace,\ 2\times10^{-4}]\cdot\overline{P2.4}$$

Evaluating P2.4

$$P2.4 = PI[ite(a,1,0),\ ace,\ 2\times10^{-4}]$$
$$= ite(P_a, ite(S_a,P1.5,P0.5),P2.5)$$
$$= ite(P_a, ite(S_a, ite(P_c,0, ite(P_e,0,1)),0),0)$$

Where:

$$P2.5 = PI[0,\ ce,\ 2\times10^{-4}] = 0 \ \therefore \overline{P2.5} = 1$$

$$P1.5 = PI[1,\ ce,\ 2\times10^{-7}]\cdot\overline{P2.5} = ite(P_c,0, ite(P_e,0,1))$$

$$P0.5 = PI[0,\ ce,\ 2\times10^{-4}]\cdot\overline{P2.5} = 0$$

Evaluating P0.4

$$P0.4 = PI[ite(a,1,0),\ ace,\ 2\times10^{-4}]\cdot\overline{P2.4}$$
$$= ite(P_a, ite(S_a,P1.6,P0.6),P2.6)\cdot ite(P_a, ite(S_a, ite(P_c,1, ite(P_e,1,0)),1),1)$$
$$= ite(P_a, ite(S_a, ite(P_c,0, ite(P_e,0,1)),0),0)\cdot ite(P_a, ite(S_a, ite(P_c,1, ite(P_e,1,0)),1),1)$$
$$= 0$$

Where:

$$P2.6 = PI[0,\ ce,\ 2\times10^{-4}] = 0 \ \ \therefore\ \overline{P2.6} = 1$$

$$P1.6 = PI[1,\ ce,\ 2\times10^{-7}]\cdot\overline{P2.6} = ite(P_c,0,ite(P_e,0,1))$$

$$P2.6 = PI[0,\ ce,\ 2\times10^{-4}] = 0 \ \ \therefore\ \overline{P2.6} = 1$$

Evaluating P0

$$\begin{aligned} P0 &= PI[ite(d,ite(c,0,ite(e,1,0)),0),\ dace,\ 0.9998]\cdot\overline{P2} \\ &= ite(P_d,ite(S_d,P1.7,P0.7),P2.7)\cdot ite(P_d,ite(S_d,ite(P_a,0,ite(P_c,ite(S_c,0,ite(P_e,ite(S_e,1,0),0)),0)),0),0) \\ &= 0 \end{aligned}$$

Where:

$$P2.7 = PI[0,\ \ ace,\ 0.9998] = 0 \ \ \therefore \overline{P2.7} = 1$$

$$P1.7 = PI[ite(c,0,ite(e,1,0)),\ \ ace,\ 2\times10^{-5}]\cdot\overline{P2.7}$$

Evaluating P1.7

$$\begin{aligned} P1.7 &= PI[ite(c,0,ite(e,1,0)),\ \ ace,\ 2\times10^{-5}]\cdot\overline{P2.7} \\ &= ite(P_a,0,P2.8)\cdot 1 \\ &= ite(P_a,0,ite(P_c,ite(S_c,0,ite(P_e,ite(S_e,1,0),0)),0)) \end{aligned}$$

Where:

$$P2.8 = PI[ite(c,0,ite(e,1,0)),\ ce,\ 2\times10^{-5}]$$

Evaluating P2.8

$$\begin{aligned} P2.8 &= PI[ite(c,0,ite(e,1,0)),\ ce,\ 2\times10^{-5}] \\ &= ite(P_c,ite(S_c,P1.9,P0.9),P2.9) \\ &= ite(P_c,ite(S_c,0,ite(P_e,ite(S_e,1,0,),0)),0) \end{aligned}$$

Where:

$$P2.9 = PI[0,\ e,\ 2\times10^{-5}] = 0 \ \ \therefore \overline{P2.9} = 1$$

$$P1.9 = PI[0,\ e,\ 4\times10^{-12}]\cdot\overline{P2.9} = 0$$

$$P0.9 = P[\text{ite}(e,1,0),\ e,\ 2\times10^{-5}]\cdot\overline{P2.9} = \text{ite}(P_e,\text{ite}(S_e,1,0),0)$$

The final meta-products structure obtained is given below in equation (9.5) and the meta-products BDD is shown in figure 9.5:

$$\begin{aligned}\text{Top} = \text{ite}(P_b,\text{ite}(S_b,\text{ite}(P_d,0,\text{ite}(P_a,\text{ite}(S_a,\text{ite}(P_c,0,\text{ite}(P_e,0,1)),0),0),0),0),\\ \text{ite}(P_d,\text{ite}(S_d,\text{ite}(P_c,\text{ite}(S_c,0,\text{ite}(P_e,\text{ite}(S_e,1,0),0)),0),0),0))\end{aligned} \tag{9.5}$$

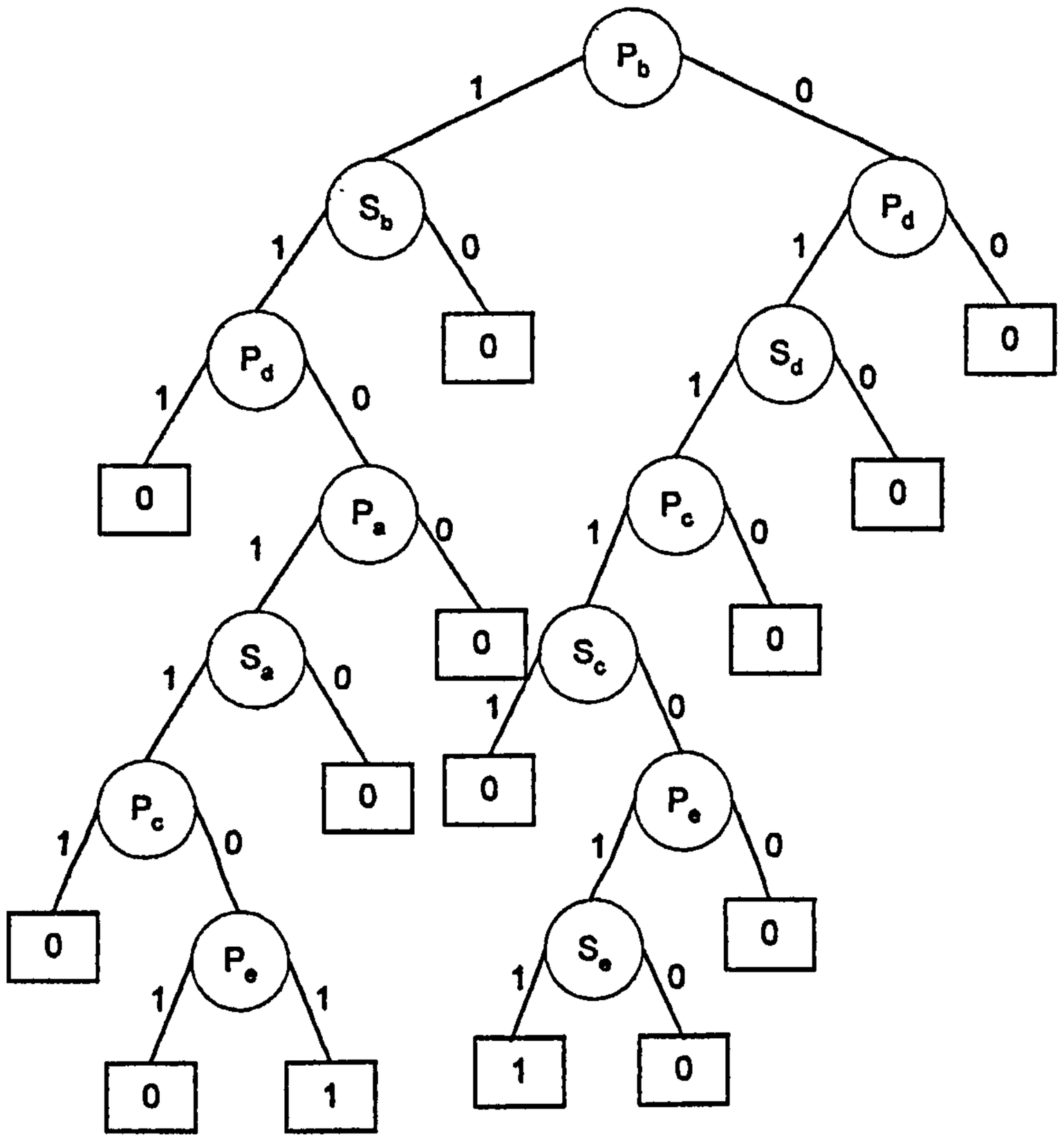


Figure 9.5: Probability Culled Meta-products BDD

Two prime implicant sets can be identified from the probability culled meta-products BDD in figure 9.5: {ab}, {d $\bar{c}$ e}.

9.6 Summary

The analysis of non-coherent fault trees can be computationally intensive; hence, it may not be possible to perform full and exact analysis, even for moderate sized trees. If this is the case, it is critical to be able to perform partial analysis in order to assess the reliability of the system and its components, and thus identify most significant causes of system failure.

Reduction and culling techniques provide a means of performing partial analysis. Reduction techniques are concerned with approximating the structure function of a non-coherent fault tree, whereas culling techniques can be used to identify the most significant prime implicant sets of the fault tree.

Reduction techniques can be employed in conjunction with either conventional FTA methods or the BDD method. Although both techniques reduce the work required to analyse a fault tree, the BDD technique is significantly more efficient than the equivalent FTA technique. It also has the advantage of enabling exact quantification.

If knowledge of the most significant prime implicant sets is required, culling techniques should be employed. Conventional FTA culling techniques can be employed to cull the prime implicant sets according to order, probability or frequency. Again, whilst these techniques reduce the work required to analyse the fault tree, they still provide an inefficient means of analysis.

To overcome the shortfalls of the FTA culling techniques, Rauzy and Dutuit developed a technique to produce an order culled meta-products BDD. Although this technique more efficient than conventional FTA culling techniques, culling the prime implicant sets according to order can result in some significant combinations being ignored. An alternative technique was developed as part of this research project, which produces a probability, or frequency culled meta-products BDD. The BDD culling techniques are more efficient than those employed during conventional FTA. Furthermore, since the SFBDD is computed as part of these techniques, full and exact quantitative analysis can be performed efficiently.

Chapter 10: Conclusions and Future Work

10.1 Summary of Work

Extensive reviews of the current techniques available for the analysis of non-coherent fault trees structures has highlighted that various aspects of these techniques require further investigation. The three main areas identified, were; the application of the BDD technique for both qualitative and quantitative analysis of non-coherent structures, the importance analysis of such structures, and the development of suitable culling techniques for the partial analysis of and non-coherent fault tree structures. The research carried out into each of these areas will be considered below in detail.

10.1.1 Qualitative and Quantitative Analysis of Non-coherent Systems

Conventional techniques for analysing non-coherent structures are inefficient and often inaccurate because approximations prove unavoidable even for moderate sized fault trees. The BDD method was initially introduced for the purposes of fault tree analysis by Rauzy in 1993, who developed a technique for performing qualitative analysis of coherent fault trees [3]. Sinnamon and Andrews subsequently developed procedures for quantifying the system, which employ the BDD method [13]. The BDD method overcomes some of the shortfalls of conventional FTA techniques, enabling efficient qualitative analysis and accurate quantitative analysis.

In 1998, Rauzy and Dutuit developed a technique for computing the prime implicant sets of a non-coherent fault tree using the BDD method [4,6]. This technique involves computing the meta-products BDD, which encodes the prime implicant sets of the system exactly, eliminating the need to perform a minimisation operation. However, the meta-products BDD is encoded from the SFBDD, hence it is necessary to compute two BDD's to perform this analysis.

An alternative technique was developed as part of the research reported in this thesis, which involves computing only one BDD, known as the consensus BDD. This BDD is then minimised to encode the prime implicant sets of the system exactly.

Procedures for quantifying the system using the consensus BDD and the SFBDD were developed. These procedures are extremely efficient in comparison to conventional FTA techniques, eliminating the need for approximations.

The procedures for quantification of the system using the consensus structure are significantly less complex than those developed for the SFBDD. However, the meta-products technique enables significantly more efficient qualitative analysis than the consensus technique. This is because a minimisation procedure must be applied to the consensus BDD before a full list of prime implicant sets can be identified, which can be an intensive process for large BDD with many redundancies.

10.1.2 Importance Analysis of Non-coherent Systems

Initial investigations into the application of current measures of importance for the analysis of non-coherent systems demonstrated that these measures were not suitable, producing inaccurate and misleading results. Furthermore Jackson's proposed extension of Birnbaum's measure of component reliability importance was shown to be inconsistent.

An alternative extension of Birnbaum's measure was developed, by considering the failure and repair criticality of each system component [22]. This extension was subsequently employed to extend other commonly used measures of importance *including the measure of component criticality* and Barlow and Proschan's measure of component initiator importance.

A detailed examination of the most commonly used measures of component and minimal cut set importance revealed that both Lambert's measure of component enabler importance, [19], and Barlow and Proschan's measure of cut set importance, [20], did not produce the required results. Modifications were made to both of these measures ensuring that the results obtained were consistent with each of their definitions.

These modified measures, along with Fussel-Vesely's measures of component and cut set importance were then extended for use with non-coherent systems. The extensions developed for the seven measures of importance outlined above enable

both the reliability and availability importance of components and prime implicant sets of a non-coherent system to be assessed.

Having developed these extensions it was necessary to develop calculation procedures for each of the extended measures. Initially techniques employing conventional Fault Tree Analysis methods were developed, however, these techniques require full and exact qualitative analysis to be performed before quantification can be undertaken.

Conventional FTA techniques for qualitative analysis are inefficient; since they rely on Boolean reduction methods. Consequently approximations are often unavoidable, furthermore, because the quantification techniques involve series expansions whose lengths are dependent on the number of prime implicant sets, for large fault trees with repeated events it is not always possible to obtain exact results during quantification.

Sinnamon and Andrews developed procedures for quantifying a coherent system using the SFBDD [13]. These procedures were compared to the Kinetic Tree Theory approach developed by Vesely [8] and shown to be superior in terms of both accuracy and efficiency.

Procedures were developed for calculating the system unavailability, unreliability and the seven extended measures of *importance from the consensus* BDD. These procedures are applied to the non-minimal consensus BDD providing an efficient and accurate means of quantification. Procedures for calculating all of these system parameters except the measure of component failure and repair enabler importance from the SFBDD have also been developed. The procedures developed for quantification using the SFBDD are more complex than those developed for the consensus BDD. However, they still provide an efficient and accurate means of analysing a non-coherent fault tree, eliminating the need for approximations that are often unavoidable when employing conventional FTA techniques.

10.1.3 Reduction and Culling Techniques for the Partial Analysis of Non-coherent Systems

For large fault trees with many repeated events *it is not always possible to analyse* the system exactly, in such circumstances approximate analysis must be performed. There are two main methods that can be employed in such circumstances, reduction techniques and culling techniques. Reduction techniques are used to approximate the structure function of a non-coherent fault tree, and culling techniques are used to produce a partial list of minimal cut sets or prime implicant sets thus reducing the work required to analyse the fault tree.

A reduction technique can be employed in conjunction with conventional FTA methods to produce a full list of minimal p-cuts, which can then be used to quantify the system approximately. Although this technique can significantly reduce the work required to analyse the system, it is still an inefficient means of analysis. The ite structure of the BDD lends itself well to reduction techniques, enabling the minimal p-cuts to be identified efficiently. In addition full and exact quantification can be performed using the computed SFBDD.

Although reduction techniques provide a useful means of partial analysis, in certain circumstances knowledge of the prime implicant sets can be advantageous. Culling techniques can be used to produce a partial list of prime implicant sets according to order, probability or frequency.

The culling techniques for analysing non-coherent systems are considerably more involved than those for analysing coherent systems. The conventional FTA culling technique is concerned with computing a partial list of prime implicant sets during qualitative analysis. The prime implicant sets identified can then be used to quantify the system approximately. Although this technique can considerably reduce the work required to quantify the system, it is still inefficient. Hence, for large fault trees with many prime implicant sets, it may not be possible to perform even approximate analysis.

To overcome the shortfalls of conventional FTA culling techniques, Rauzy and Dutuit developed a culling technique that is employed with the meta-products algorithm to produce an order culled meta-products BDD, from which a partial list of prime implicant sets can be obtained [6,7].

For non-coherent systems the order of prime implicant sets bares no relation to its probability of existence, this is due to the inclusion of working states, which tend to have a probability close to 1. Hence an alternative culling technique was developed to compute a probability or frequency culled meta-products BDD, from which a partial list of prime implicant sets can be identified. In order to compute a culled meta-products BDD it is necessary to encode the SFBDD exactly, hence full and exact quantification can be performed using the BDD.

10.2 Conclusions

1. Rauzy developed the meta-products algorithm, which can be used to compute the meta-products BDD; this BDD encodes a full list of prime implicant sets exactly. An alternative technique was developed for analysing non-coherent systems, which involves computing the consensus BDD. Whilst this BDD lends itself to quantification, a minimisation procedure must be applied to the BDD before the prime implicant sets can be identified.

2. The BDD method can be used to quantify a non-coherent fault tree structure accurately. This technique is significantly more efficient that the conventional FTA techniques reducing the need to employ approximations.

3. The extensions developed for the seven most commonly used measures of component and cut set importance provide a solid foundation for assessing the component and prime implicant set importance of a non-coherent system.

4. A number of reduction and culling techniques can be employed to partially analyse both coherent and non-coherent systems, for which full and exact analysis proves too intensive. Those techniques applied in conjunction with the BDD method provide the most efficient means of analysis.

10.3 Future Work

10.3.1 Calculating the Enabler Measure of Failure and Repair Importance from the SFBDD

The procedure for calculating the measure of component enabler failure and repair importance from the SFBDD is extremely time consuming and inefficient. Hence at present it must either be approximated by the coherent measure, or conventional FTA techniques must be used to calculate it exactly. Whilst the coherent measure can give an indication of the significance of each component in contributing to system failure it does not ranking both the failure and repair importance of each component. Furthermore, the conventional FTA techniques for calculating this measure are inefficient and for large fault trees with many repeated events approximations are unavoidable. The BDD technique lends itself to quantification, and eliminates the need for lengthy series expansions and approximations. The development of an efficient procedure for calculating this measure would increase the accuracy of analysis.

10.3.2 Extending Other measures of Importance

Importance analysis is a key part of the quantification process, although seven of the most commonly used measures of importance have been extended for use with non-coherent fault trees. There are many other measures of importance that have not yet been extended for the purpose of non-coherent FTA, for example, Xie and Shen's measures for parallel redundancies importance and standby redundancy importance [24]. The extension of such measures for non-coherent analysis would enable a more detailed sensitivity analysis of such structures.

10.3.3 Reduction of Non-coherent Fault Trees

The consensus and meta-products methods for analysing non-coherent fault trees are significantly more efficient than the conventional FTA techniques. However, for large fault trees with many repeated events computing the BDD's required for

analysis can be time consuming. Furthermore if the BDD's obtained are complex, the subsequent analysis can be intensive making approximations unavoidable.

Reay and Andrews demonstrated that the application of the extended reduction technique to coherent *fault trees prior to employing the BDD technique can* significantly reduce the complexity of the BDD computed, [25,26]. The smaller (more minimal) the BDD obtained, the more efficient the conversion process and the more efficient the subsequent analysis.

This technique could be applied to non-coherent fault trees before either the consensus method or the meta-products method is employed. If a more minimal consensus BDD is obtained, the minimisation process could be significantly less intensive, making the consensus method a serious option for non-coherent fault tree analysis. Similarly, the application of this technique should reduce the size and complexity of the SFBDD obtained. This would in turn reduce the work required to compute the meta-products BDD, thus increasing the efficiency of this method and reducing the need to employ culling techniques.

References

[1] Z. W. Birnbaum. "On the importance of Different Components in a Multi-component System" Multivariate *Analysis II, PR Krishnaiah, ed., Academic* Press, 1969.

[2] A. Bendall and J. Ansell. "The incoherency of Multistate Coherent Systems". Reliability Engineering, VOL. 8, 1984, PP165-178

[3] A. Rauzy. "New Algorithms for Fault Tree Analysis". Reliability Engineering and System Safety, vol. 40, 1993, p203-211.

[4] A. Rauzy. "Mathematical Foundations of Minimal Cut Sets", in press

[5] Groupe Aralia (LaBRI-LADS), Universite Bordeaux, "Computation of Prime Implicants of a Fault Tree within Aralia", Proceedings of ESREL'95 Conference, Bournemouth, June, p190-202.

[6] A. Rauzy and Y. Dutuit. "Exact and Truncated Computations of Prime Implicants of Coherent and Non-coherent Fault Trees within Aralia". Reliability Engineering and System Safety, vol. 58, 1997, p127-144

[7] Y. Dutuit and A. Rauzy. "Polynomial Approximations of Boolean Functions by means of Positive Binary Decision Diagrams", Safety and Reliability, Lydersen, Hansen and Sandtorv (eds), 1998.

[8] W. E. Vesely. "A Time Dependent Methodology for Fault Tree Evaluation", Nuclear Design and Engineering, vol. 13, 1970, p337-360.

[9] J. D. Andrews and T. R. Moss. "Reliability and Risk Assessment", Longman Scientific and Technical ,UK 1993.

[10] W. G. Schneeweiss. "Boolean Functions with Engineering Applications and Computer Programs", Springer-Verlag, Berlin, 1989.

[11] L. M. Bartlett. "Variable Ordering Heuristics for Binary Decision Diagrams". Doctoral Thesis, Loughborough University, Mar 2000.

[12] Freidman and Supowit. "Finding the Optimal Variable Ordering for Binary Decision Diagrams", IEEE Transactions on Computers, Vol. 39, No. 5, May 1990, pp710-713.

[13] R. M. Sinnamon and J. D. Andrews. "Quantitative Fault Tree Analysis Using Binary Decision Diagrams". European Journal of Automation, Vol. 30, No. 8, 1996.

[14] J. D. Andrews. "To Not or Not to Not". Proceedings of the International System Safety Conference, Forte Worth, Sept 2000, pp267-275.

[15] S. Dunnett and J. D. Andrews. "Improving Accuracy in Event Tree Analysis". Fore sight and Precaution. Cottam, Harvey, Pape and Tate (eds). Proceedings of ESREL 2000, SARS ad SRA-EUROPE annual Conference, 15-17 May 2000.

[16] Inagaki & E. J. Henley. "Probabilistic Evaluation of Prime Implicants and Top Events for Non-Coherent Systems". IEEE Transactions on Reliability, vol. R-29 No. 5, Dec 1980.

[17] G. Becker & L. Camarinopoulos. "Failure Frequencies of Non-Coherent Structures". Reliability Engineering and System Safety, Vol. 41, 1993, pp209-215.

[18] R. E. Barlow and F. Proschan. " Importance of System Components and Fault Tree Events", Stochastic Processes and their Applications vol. 3, 1975 pp153-173.

[19] H. E. Lambert. "Fault Trees for Decision Making in Systems Analysis", Doctoral Thesis, University of California, Livermore, 1975.

[20] J. Fussell. "How to Hand Calculate System Reliability Characteristics", IEEE Transactions on Reliability, Vol. R-24, Aug 1975, pp 169-174.

[21] P. S. Jackson. "On the S-Importance of Elements and Implicants of Non-Coherent Systems", IEEE Transactions on Reliability, vol. R-32, No. 1, APR 1983.

[22] S. Beeson and J. D. Andrews. "Birnbaum's Measure of Component Importance for Non-coherent Systems", IEEE Transactions on Reliability, in press.

[23] S. Beeson and J. D. Andrews. Importance Measures for Non-coherent System Analysis, IEEE Transactions on Reliability, 2002.

[24] M. Xie, and K. Shen, "Some New Aspects on Component Importance Measures", 11th Advances in Relibaility Technology Symposium, Apr, 1990

[25] K. A. Reay. "Efficient Fault Tree Analysis Using Binary Decision Diagrams", Doctoral Thesis, Loughborough University, 2002.

[26] K. A. Reay and J. D. Andrews. "A Fault Tree Analysis Strategy Using Binary Decision Diagrams", Journal of Reliability Engineering and System Safety, 2002.

Appendix (I): Minimisation Procedure

Computing minsol(F)

```
/* Terminal 1 or 0 nodes are already minimal */
if(F=0) or (F=1)
        minsol(F)=F
/* Non-terminal nodes of the form F=ite(x, G, H) If calculation
  has already been performed, simply copy the result across */
else if (Computation table has entry {minsol, F, R})
        minsol(F)=R
/* If calculation has not been performed, a new calculation
  required to calculate, minsol(G) without(Gmin, H) & minsol(H) */
else
        Compute minsol(G)
        Compute without(Gmin, H)
        Compute minsol(H)
        minsol(F)=ite(x, without(Gmin, H), Hmin)
```

Computing without(Gmin, H)

```
/* Terminal nodes are simple to deal with */
if(Gmin=0) or (H=1)
        without(Gmin, H)=0
else if(H=0) or (Gmin=1)
        without(Gmin, H)=Gmin
/* Non-terminal nodes: Gmin=ite(x, G1, G2) and H=ite(y, H1, H2). Additional work
  required to compute without(Gmin, H) depending on positions of x & y in ordering */
else if(x<y)
        Compute without(G1, H)
        Compute without(G2, H)
        without(Gmin, H)=ite(x, without(G1,H),without(G2,H))
else if(x>y)
        Compute without(G, H2)
        without(Gmin, H)=without(G, H2)
else
        Compute without(G1, H1)
        Compute without(G2, H2)
        without(Gmin, H)=ite(x, without(G1, H1), without(G2, H2))
```

Appendix(II): Fault Tree Structures

Below is a list of the non-coherent fault trees used to compare the mate-products and consensus technique in chapter five. The fault tree data is as follows:

Gate number, gate type (1=OR, 2=AND, 3=NOT), number of gate inputs, number of event inputs, gate inputs (listed), event inputs (listed).

Lisa102

```
1000  2  3  0  1001  1054  1003
1001  1  1  2  1004  1       2
1002  1  3  4  1005  1055  1007  3  4  5  6
1003  1  2  2  1008  1009  7  8
1004  2  4  3  1010  1011  1012  1056  10  11  12
1005  2  1  1  1013  13
1006  2  0  4  14  15  16  17
1007  2  0  3  18  19  16
1008  2  3  2  1014  1015  1016  20  21
1009  2  1  2  1017  22  23
1010  1  1  1  1018  24
1011  1  1  4  1019  25  26  27  9
1012  1  3  5  1020  1021  1022  28  29  30  19  31
1013  1  2  0  1023  1024
1014  1  1  1  1025  32
1015  1  3  1057  33  34  36
1016  1  3  5  1026  1027  1028  37  31  38  39  40
1017  1  3  4  1029  1030  1031  41  42  43  44
1018  2  1  1  1032  45
1019  2  1  1  1033  46
1020  2  0  5  47  48  49  50  51
1021  2  3  0  1034  1035  1036
1022  2  1  1  1037  52
1023  2  3  3  1058  1039  1040  53  54  55
1024  2  2  0  1041  1042
1025  2  2  0  1043  1044
1026  2  3  4  1045  1046  1047  56  57  58  59
1027  2  1  3  1048  60  13  61
1028  2  2  0  1049  1050
1029  2  0  4  36  62  63  64
1030  2  2  2  1051  1059  65  67
1031  2  2  0  1052  1053
1032  1  0  2  68  69
1033  1  0  3  25  70  71
1034  1  0  4  72  73  74  7
1035  1  0  5  75  76  77  2  71
1036  1  1  1  1060  78
1037  1  0  3  76  79  75
1038  1  1  2  1061  80  81
1039  1  0  2  82  83
1040  1  0  3  84  85  86
1041  1  0  2  87  29
1042  1  0  2  88  89
1043  1  0  2  11  81
```

1044 1 0 2 90 91
1045 1 0 5 92 93 94 66 95
1046 1 0 2 95 96
1047 1 0 4 56 97 58 98
1048 1 0 3 99 21 25
1049 1 1 1 1062 99
1050 1 0 3 101 3 102
1051 1 0 3 103 104 105
1052 1 0 2 4 106
1053 1 0 5 107 18 108 109 110
1054 3 1 0 1002
1055 3 1 0 1006
1056 3 0 1 14
1057 3 0 1 35
1058 3 1 0 1038
1059 3 0 1 69
1060 3 0 1 28
1061 3 0 1 24
1062 3 0 1 100

Lisa106
1000 2 2 4 1001 1002 1 2 3 4
1001 1 1 1 1003 5
1002 1 2 4 1004 1005 6 7 8 9
1003 2 2 3 1128 1007 1 10 11
1004 2 3 0 1008 1009 1010
1005 2 3 0 1011 1012 1013
1006 1 1 1 1014 9
1007 1 0 5 12 13 14 15 16
1008 1 2 4 1015 1016 17 18 19 20
1009 1 2 4 1017 1129 21 22 23 24
1010 1 0 2 25 26
1011 1 2 3 1019 1020 27 28 29
1012 1 3 4 1021 1022 1023 30 31 32 33
1013 1 3 0 1024 1025 1026
1014 2 2 5 1027 1028 34 35 36 37 2
1015 2 1 5 1029 38 3 39 17 40
1016 2 0 5 41 42 43 44 45
1017 2 0 2 46 47
1018 2 1 1 1130 48
1019 2 3 2 1031 1032 1033 49 2
1020 2 1 3 1034 50 51 52
1021 2 2 4 1035 1131 53 22 56 55
1022 2 3 3 1133 1037 1132 7 25 26
1023 2 2 3 1039 1134 57 58 64
1024 2 3 3 1040 1041 1042 61 62 63
1025 2 2 2 1135 1136 25 69
1026 2 3 4 1043 1044 1045 67 68 69 70
1027 1 3 4 1046 1047 1048 71 72 73 50
1028 1 1 5 1049 74 26 75 76 77
1029 1 1 1 1050 31
1030 1 2 1 1051 1052 78
1031 1 1 3 1053 79 80 81
1032 1 2 0 1054 1055

1033 1 1 1 1056 56
1034 1 1 3 1057 82 83 84
1035 1 2 0 1058 1059
1036 1 2 2 1060 1061 85 86
1037 1 2 4 1137 1138 83 88 82 90
1038 1 3 5 1063 1064 1139 91 92 93 94 95
1039 1 1 4 1066 23 96 58 97
1040 1 3 0 1067 1068 1069
1041 1 0 4 98 99 100 101
1042 1 0 2 47 102
1043 1 1 1 1070 103
1044 1 0 4 61 104 105 106
1045 1 3 3 1071 1072 1073 107 9 108
1046 2 2 5 1074 1075 46 109 110 111 9
1047 2 3 1 1076 1077 1078 9
1048 2 3 3 1079 1080 1081 111 42 4
1049 2 3 5 1082 1083 1084 56 112 113 69 114
1050 2 1 1 1085 115
1051 2 3 2 1086 1087 1088 42 5
1052 2 0 5 99 116 113 117 118
1053 2 1 4 1089 34 52 119 120
1054 2 1 1 1090 121
1055 2 2 5 1091 1092 122 123 66 124 125
1056 2 3 5 1093 1094 1095 87 106 126 14 127
1057 2 0 2 34 88
1058 2 3 3 1096 1097 1098 128 129 130
1059 2 2 1 1099 1100 104
1060 2 3 3 1101 1102 1103 8 115 131
1061 2 3 0 1104 1140 1106
1062 2 2 2 1142 1108 136 87
1063 2 3 0 1109 1143 1111
1064 2 1 2 1112 114 133
1065 2 1 1 1113 94
1066 2 3 3 1114 1115 1144 119 134 135
1067 2 1 2 1145 137 53
1068 2 0 2 83 138
1069 2 3 0 1117 1118 1119
1070 2 2 4 1120 1121 139 124 90 140
1071 2 2 4 1122 1146 10 141 52 142
1072 2 2 0 1123 1124
1073 2 3 5 1125 1126 1127 143 144 72 145 146
1074 1 0 3 147 148 149
1075 1 0 2 123 150
1076 1 0 4 151 152 121 30
1077 1 0 5 120 72 85 153 154
1078 1 0 2 128 99
1079 1 0 2 155 156
1080 1 0 5 157 158 159 160 161
1081 1 0 2 72 79
1082 1 0 2 21 162
1083 1 0 2 141 163
1084 1 0 5 164 165 43 96 166
1085 1 0 2 167 13
1086 1 0 2 19 168
1087 1 0 3 42 69 169

1088 1 1 3 1147 170 171 76
1089 1 0 5 172 173 116 174 137
1090 1 0 2 18 92
1091 1 0 2 175 153
1092 1 0 2 33 70
1093 1 1 3 1148 101 49 176
1094 1 0 2 148 177
1095 1 0 2 178 31
1096 1 0 2 41 103
1097 1 0 2 102 56
1098 1 0 2 91 116
1099 1 0 2 179 19
1100 1 0 2 180 157
1101 1 0 2 84 115
1102 1 1 3 1149 118 165 135
1103 1 0 5 69 182 91 183 103
1104 1 0 2 104 184
1105 1 0 3 185 61 54
1106 1 0 2 186 187
1107 1 0 2 100 181
1108 1 0 5 188 184 1 189 190
1109 1 0 2 191 150
1110 1 0 2 28 160
1111 1 0 2 18 192
1112 1 0 2 167 76
1113 1 0 2 62 186
1114 1 1 1 1150 193
1115 1 0 2 194 69
1116 1 0 3 136 68 195
1117 1 0 3 81 117 196
1118 1 0 2 79 18
1119 1 1 1 1151 198
1120 1 0 2 199 200
1121 1 0 4 169 201 202 203
1122 1 0 2 165 53
1123 1 0 2 204 205
1124 1 0 2 121 191
1125 1 0 4 201 156 162 117
1126 1 0 2 154 192
1127 1 1 1 1152 206
1128 3 1 0 1006
1129 3 1 0 1018
1130 3 0 1 49
1131 3 0 1 54
1132 3 1 0 1038
1133 3 0 1 1
1134 3 0 1 60
1135 3 0 1 64
1136 3 0 1 66
1137 3 0 1 87
1138 3 0 1 89
1139 3 1 0 1065
1140 3 1 0 1105
1141 3 1 0 1107
1142 3 0 1 132

1143 3 1 0 1110
1144 3 1 0 1116
1145 3 0 1 136
1146 3 0 1 48
1147 3 0 1 123
1148 3 0 1 107
1149 3 0 1 181
1150 3 0 1 166
1151 3 0 1 197
1152 3 0 1 207

Dre1058

1000 2 6 0 1001 1002 1003 1004 1005 1006
1002 1 1 3 1013 2 3 4
1003 1 0 7 5 6 7 8 9 10 1
1001 1 1 2 1007 12 13
1007 2 2 0 1008 1009
1008 1 0 7 14 15 16 17 18 19 20
1009 1 1 10 1014 21 22 23 24 17 25 26 28 29 30
1005 1 0 4 1 2 31 32
1006 1 0 7 7 8 9 10 11 33 34
1004 1 1 2 1010 35 36
1010 2 2 0 1011 1012
1011 1 0 13 37 22 23 24 17 25 26 27 28 29 30 38 39
1012 1 0 7 16 17 18 19 20 40 41
1013 3 0 1 1
1014 3 0 1 27

Bpfsw02

1000 2 2 0 1001 1002
1001 1 1 1 1003 1
1002 1 1 1 1004 7
1003 2 2 0 1005 1006
1004 2 2 0 1007 1008
1009 1 0 4 2 3 4 5
1010 1 0 4 6 7 8 9
1011 1 0 4 10 11 12 13
1012 1 0 4 14 15 16 17
1013 1 0 4 18 19 20 21
1014 1 0 4 22 23 24 25
1015 1 0 4 26 27 28 29
1016 1 1 3 1021 30 31 32
1005 1 1 2 1017 34 35
1007 1 1 3 1018 34 36 37
1006 1 1 2 1019 38 39
1008 1 1 3 1020 38 40 37
1019 2 2 0 1009 1010
1017 2 2 0 1013 1014
1020 2 2 0 1011 1012
1018 2 2 0 1015 1016
1021 3 0 1 3

Bpfeg03

```
1000 1 5 15 1020 1002 1003 1021 1022 1 2 3 4 5 6 7 9 11 12
13 14 15 16 17
1001 2 6 0 1004 1005 1006 1007 1008 1009
1010 1 2 0 1011 1012
1013 1 1 1 1023 18
1014 1 0 5 20 21 22 23 24
1002 2 0 2 25 26
1003 2 0 2 27 28
1015 2 0 2 29 30
1016 2 0 2 31 32
1004 1 0 2 33 34
1005 1 0 2 35 36
1006 1 0 2 37 38
1007 1 0 2 39 40
1008 1 0 2 41 42
1009 1 0 2 43 44
1011 2 3 0 1000 1014 1017
1012 2 0 3 45 46 47
1017 1 3 13 1018 1015 1016 48 49 50 51 52 53 54 55 56 57 58
59 60
1018 2 2 0 1013 1019
1019 1 0 3 61 62 63
1020 3 1 0 1001
1021 3 0 1 19
1022 3 0 1 18
1023 3 0 1 19
```

Nakashi

```
1000 2 2 0 1001 1002
1001 1 4 0 1003 1004 1005 1006
1002 1 4 0 1007 1008 1009 1010
1003 2 1 1 1011 1
1004 2 1 1 1012 2
1011 1 0 2 3 4
1012 1 1 1 1013 5
1013 2 1 1 1014 3
1014 1 1 1 1021 7
1005 2 1 1 1015 6
1006 2 0 3 8 9 10
1015 1 1 1 1016 11
1016 2 0 2 4 12
1007 2 1 1 1017 3
1008 2 1 1 1018 13
1017 1 0 2 1 2
1018 1 2 0 1019 1020
1019 2 0 2 8 10
1020 2 0 2 14 15
1009 2 0 2 16 14
1010 2 0 4 2 7 15 5
1021 3 0 1 6
```

Lisab13

```
1000  1  3  2  1001  1002  1003  1  2
1001  2  0  4  3  4  5  6
1002  2  1  1  1004  7
1003  2  1  1  1005  8
1004  1  0  2  9  3
1005  1  2  4  1006  1007  10  11  2  12
1006  2  3  2  1008  1009  1010  3  12
1007  2  0  2  11  13
1008  1  2  2  1011  1012  14  15
1009  1  0  5  16  9  11  17  18
1010  1  3  2  1013  1014  1015  19  20
1011  2  1  3  1016  10  7  8
1012  2  1  1  1017  18
1013  2  4  3  1018  1019  1020  1024  21  22  10
1014  2  1  1  1021  23
1015  2  2  3  1022  1023  15  24  11
1016  1  0  2  23  7
1017  1  0  3  25  20  26
1018  1  0  2  18  9
1019  1  0  2  27  28
1020  1  0  2  1  19
1021  1  0  5  13  30  8  26  14
1022  1  0  2  31  3
1023  1  0  4  5  21  23  10
1024  3  0  1  10
```

Lisa118

```
1000  1  1  1  1001  1
1001  2  3  0  1002  1003  1004
1002  1  2  1  1005  1006  2
1003  1  3  2  1007  1008  1009  3  4
1004  1  2  4  1010  1011  5  6  7  8
1006  2  1  1  1037  13
1007  2  2  4  1014  1015  14  15  16  17
1008  2  0  5  18  19  20  21  22
1009  2  0  3  9  23  24
1010  2  3  2  1016  1038  1018  25  26
1011  2  0  2  13  27
1012  1  0  4  28  29  30  9
1013  1  2  0  1019  1020
1014  1  0  2  31  32
1015  1  2  0  1021  1022
1016  1  0  4  33  34  35  8
1017  1  0  5  36  37  38  39  40
1018  1  3  3  1023  1024  1025  41  42  43
1019  2  2  2  1026  1027  44  45
1020  2  0  2  46  29
1021  2  2  0  1028  1029
1022  2  2  0  1030  1031
1023  2  2  1  1032  1033  47
1024  2  3  3  1039  1035  1036  47  49  50
1025  2  0  4  51  52  53  54
1026  1  0  2  55  56
```

1027 1 0 4 34 57 58 3
1028 1 1 1 1040 59
1029 1 0 4 61 62 12 63
1030 1 0 5 64 38 28 44 65
1031 1 0 2 66 56
1032 1 0 4 67 68 69 70
1033 1 0 5 71 57 72 24 42
1034 1 0 2 14 73
1035 1 0 2 74 75
1036 1 1 5 1041 4 76 57 15 77
1037 3 0 1 15
1038 3 0 1 29
1039 3 0 1 48
1040 3 0 1 60
1041 3 0 1 64

Lisa117
1000 2 3 2 1001 1002 1003 1 2
1001 1 2 0 1004 1005
1002 1 2 1 1006 1007 3
1003 1 2 4 1008 1009 4 5 6 7
1004 2 3 1 1010 1011 1012 8
1005 2 2 3 1013 1014 9 10 1
1006 2 1 5 1015 12 13 3 14 15
1007 2 2 0 1016 1017
1008 2 2 0 1018 1019
1009 2 2 0 1020 1021
1010 1 3 0 1022 1023 1080
1011 1 1 3 1025 16 17 18
1012 1 3 4 1026 1027 1081 19 20 22 23
1013 1 3 0 1028 1029 1030
1014 1 3 0 1031 1032 1033
1015 1 3 5 1034 1082 1036 24 1 25 26 27
1016 1 1 1 1037 28
1017 1 1 1 1038 29
1018 1 2 1 1039 1040 30
1019 1 0 4 31 1 32 33
1020 1 2 0 1041 1042
1021 1 1 5 1043 34 35 36 37 26
1022 2 3 3 1083 1045 1046 38 39 40
1023 2 2 5 1047 1048 41 42 43 44 45
1024 2 0 2 46 6
1025 2 1 3 1084 46 47 48
1026 2 2 3 1049 1085 50 51 52
1027 2 2 2 1050 1051 53 54
1028 2 0 5 51 55 56 57 50
1029 2 2 2 1052 1053 58 59
1030 2 0 2 46 60
1031 2 2 0 1054 1055
1032 2 3 0 1056 1057 1058
1033 2 2 5 1059 1060 61 62 63 64 65
1034 2 1 4 1061 66 67 68 69
1035 2 2 1 1062 1063 70

1036 2 2 2 1064 1065 71 36
1037 2 1 1 1066 36
1038 2 0 3 3 72 17
1039 2 3 0 1067 1068 1069
1040 2 2 0 1070 1071
1041 2 3 4 1072 1086 1074 7 75 74 72
1042 2 3 3 1087 1076 1077 77 41 76
1043 2 2 2 1078 1079 77 78
1044 1 0 4 79 19 80 81
1045 1 0 2 81 5
1046 1 0 4 62 83 84 85
1047 1 0 5 78 86 87 88 89
1048 1 0 4 34 90 91 7
1049 1 0 5 92 90 3 93 94
1050 1 0 5 95 96 97 98 61
1051 1 0 4 58 76 99 100
1052 1 0 4 7 101 102 103
1053 1 0 4 104 105 13 106
1054 1 0 5 107 108 109 110 111
1055 1 0 3 41 112 113
1056 1 0 3 114 115 77
1057 1 0 2 110 67
1058 1 0 5 116 38 117 118 119
1059 1 1 3 1088 120 122 108
1060 1 0 3 59 26 123
1061 1 0 2 124 125
1062 1 0 2 126 127
1063 1 0 3 102 128 129
1064 1 0 2 108 76
1065 1 0 2 130 65
1066 1 0 2 94 22
1067 1 0 4 131 132 126 127
1068 1 0 2 127 133
1069 1 0 2 134 20
1070 1 0 2 100 28
1071 1 0 2 21 57
1072 1 0 2 135 136
1073 1 0 2 137 92
1074 1 0 2 73 3
1075 1 0 4 75 59 106 138
1076 1 0 2 20 139
1077 1 1 4 1089 27 140 141 142
1078 1 0 5 143 144 137 145 146
1079 1 0 2 147 81
1080 3 1 0 1024
1081 3 0 1 24
1082 3 1 0 1035
1083 3 1 0 1044
1084 3 0 1 49
1085 3 0 1 53
1086 3 0 1 73
1087 3 0 1 75
1088 3 0 1 121
1089 3 0 1 103

Lisab15

```
1000  2  2  1  1001  1002  1
1001  1  2  3  1003  1004  2  3  4
1002  1  2  2  1005  1006  5  6
1003  2  0  5  7  8  9  10  5
1004  2  2  1  1007  1008  11
1005  2  0  2  12  4
1006  2  1  3  1009  13  14  15
1007  1  1  1  1010  16
1008  1  1  4  1011  17  18  19  20
1009  1  3  1  1012  1013  1014  21
1010  2  3  3  1015  1016  1017  22  23  24
1011  2  1  3  1018  25  26  27
1012  2  2  0  1019  1020
1013  2  2  3  1021  1049  28  29  30
1014  2  2  2  1022  1023  31  32
1015  1  1  5  1024  33  34  35  36  37
1016  1  1  1  1025  38
1017  1  2  1  1026  1027  39
1018  1  3  5  1028  1029  1030  24  40  41  42  43
1019  1  1  3  1031  44  10  24
1020  1  1  5  1032  45  46  47  48  49
1021  1  2  4  1033  1034  19  35  50  51
1022  1  0  2  52  53
1023  1  1  1  1035  54
1024  2  1  1  1036  55
1025  2  1  4  1037  23  56  57  58
1026  2  0  2  59  60
1027  2  1  4  1050  61  62  63  64
1028  2  3  0  1038  1039  1040
1029  2  0  2  66  29
1030  2  0  3  67  68  69
1031  2  1  1  1041  70
1032  2  2  5  1042  1043  71  2  72  73  41
1033  2  2  0  1044  1045
1034  2  3  1  1046  1047  1051  74
1035  2  0  3  75  70  76
1036  1  0  2  77  78
1037  1  0  4  79  80  81  82
1038  1  0  2  83  84
1039  1  0  4  85  43  86  87
1040  1  0  2  60  88
1041  1  1  1  1052  23
1042  1  0  2  90  91
1043  1  0  2  22  92
1044  1  0  4  26  21  93  47
1045  1  0  2  94  39
1046  1  0  3  68  95  96
1047  1  0  2  50  97
1048  1  0  2  43  98
1049  3  0  1  29
1050  3  0  1  65
1051  3  1  0  1048
1052  3  0  1  89
```

jdtree5

```
1000  1  1  2  1001  1  2
1002  1  2  1  1003  1011  4
1004  1  2  1  1005  1006  3
1005  2  1  2  1002  5  6
1006  2  1  2  1000  7  8
1003  2  1  2  1007  9  10
1001  2  1  2  1008  11  12
1008  1  1  2  1009  13  14
1007  1  1  2  1010  15  16
1009  2  0  2  17  18
1010  2  0  2  19  20
1011  3  0  1  4
```

jdtree3

```
1000  2  1  2  1001  1  2
1002  2  1  2  1003  3  4
1004  2  2  1  1005  1006  5
1005  1  1  2  1002  6  7
1006  1  1  2  1000  8  9
1003  1  2  1  1007  1011  11
1001  1  1  2  1008  12  13
1008  2  1  2  1009  14  15
1007  2  1  2  1010  16  17
1009  1  0  2  18  19
1010  1  0  2  20  21
1011  3  0  1  11
```

Lisab56

```
1000  1  1  3  1011  1  2  3
1001  2  1  1  1002  1
1002  1  3  5  1003  1004  1005  3  4  5  6  7
1003  2  0  2  8  9
1004  2  2  3  1006  1007  2  10  11
1005  2  3  1  1012  1009  1010  6
1006  1  0  4  12  13  14  15
1007  1  0  2  10  9
1008  1  0  4  3  6  10  5
1009  1  0  2  16  17
1010  1  0  2  15  1
1011  3  0  1  1
1012  3  1  0  1008
```

Sjdtree

```
1000  2  3  0  1001  1002  1003
1001  1  2  0  1004  1005
1002  1  1  1  1006  1
1003  1  1  1  1007  2
1004  2  1  1  1015  3
```

```
1005  2  1  1  1009  3
1006  2  2  0  1010  1011
1007  2  2  0  1012  1013
1009  1  1  3  1017  4  5  9
1010  1  0  2  10  1
1011  1  0  2  11  12
1012  1  0  2  13  2
1013  1  0  2  14  15
1015  3  0  1  3
1017  3  0  1  8
```

Lisa103

```
1000  1  1  1  1001  1
1001  2  3  0  1002  1003  1004
1002  1  0  4  2  3  4  5
1003  1  0  2  6  7
1004  1  1  2  1005  4  9
1005  3  0  1  8
```

Fatram2

```
1000  2  2  1  1001  1002  1
1002  1  1  2  1003  2  3
1001  1  1  2  1005  4  5
1003  2  1  1  1004  6
1004  1  0  3  7  4  8
1005  3  0  1  9
```

Rando15

```
1000  1  1  5  1005  1  2  3  4  5
1001  2  3  5  1002  1003  1004  7  8  9  10  11
1002  1  1  2  1006  7  9
1003  1  0  2  8  9
1004  1  0  3  10  9  7
1005  3  1  0  1001
1006  3  0  1  6
```

Rando83

```
1000  2  3  0  1001  1002  1003
1001  1  3  3  1004  1005  1006  1  2  3
1002  1  1  1  1007  4
1003  1  0  2  5  6
1004  2  0  3  7  8  9
1005  2  0  3  2  10  11
1006  2  1  2  1014  12  13
1007  2  1  1  1019  10
1008  1  3  0  1010  1011  1012
1009  1  1  2  1013  14  8
1010  2  1  3  1015  15  16  17
1011  2  0  2  3  13
1012  2  0  3  13  19  18
```

```
1013  2  1  3  1016  7  20  6
1014  3  1  0  1008
1015  3  0  1  18
1016  3  0  1  21
```

Rand146

```
1000  2  2  3  1001  1002  1  2  3
1001  1  2  3  1003  1010  4  6  7
1002  1  2  0  1004  1005
1003  2  0  2  8  9
1004  2  0  5  10  4  11  7  12
1005  2  1  5  1006  13  14  15  16  17
1006  1  1  1  1007  18
1007  2  2  1  1008  1009  19
1008  1  0  2  20  13
1009  1  0  3  5  21  12
1010  3  0  1  6
```

Rand117

```
1000  1  2  0  1001  1002
1001  2  2  0  1003  1010
1002  2  2  5  1011  1006  1  2  3  4  5
1003  1  0  2  6  1
1004  1  0  3  7  8  9
1005  1  3  1  1007  1008  1009  10
1006  1  0  2  11  6
1007  2  0  5  12  13  14  10  15
1008  2  0  2  16  17
1009  2  0  3  15  6  14
1010  3  1  0  1004
1011  3  1  0  1005
```