An Analysis Strategy for Large Fault Trees

Prof. J.D. Andrews, PhD; Loughborough University; England

Keywords: fault tree analysis, binary decision diagrams

Abstract

In recent years considerable progress has been made on improving the efficiency and accuracy of the fault tree methodology. The majority of fault trees produced to model industrial systems can now be analysed very quickly on PC computers. However there can still be problems with very large fault tree structures such as those developed to model nuclear and aerospace systems. If the fault tree consists of a large number of basic events and gates and many of the events are repeated, possibly several times within the structure, then the processing of the full problem may not be possible. In such circumstances the problem has to be reduced to a manageable size by discarding the less significant failure modes in the qualitative evaluation to produce only the most relevant minimal cut sets and approximations used to obtain the top event probability or frequency.

The method proposed uses a combination of analysis options each of which reduces the complexity of the problem. A factorisation technique is first applied which is designed to reduce the 'noise' from the tree structure. Wherever possible, events which always appear together in the tree are combined together to create more complex, higher level events. A solution of the now reduced problem can always be expanded back out in terms of the original events. The second stage is to identify independent sections of the fault tree which can be analysed separately. Finally the Binary Decision Diagram (BDD) technique is used to perform the quantification. Careful selection of the ordering applied to the basic events (variables) will again aid the efficiency of the process.

Introduction

For systems with many components and a high degree of complexity, fault trees (ref. 1) generated to represent the causes of specified system failure modes may be too large to solve efficiently. Converting the fault tree logic structure to the alternative form of the binary decision diagram (BDD) (ref. 2) can enable an efficient and accurate solution to be obtained (refs. 3-4). However, the conversion process requires the basic events to be placed in an ordering. If the ordering is a good one the conversion process will result in a concise form of the BDD. With a poor ordering selection the size of the BDD may explode exponentially with the number of variables. Despite much work carried out to date (refs. 5-8) on methods to order the basic events there is no universally accepted approach to the ordering and the structure of some fault trees means that efficient orderings may not exist.

Generally it is true that the smaller the fault tree structure the less sensitive it is to the ordering selected with reasonable orderings producing a manageable BDD. Two methods exist which can be used to reduce the analysis of large fault trees to that of solving smaller structures. The first method is that of combining basic events together to form 'complex' events. The fault tree structure is then reduced in size by expressing the system failure mode in terms of the 'complex' events. This method was used in the FAUNET computer code developed in the 1970s (ref. 9). Reduction in this way removes the 'noise' from the fault tree but retains the relevant structure - this effectively forms very small modules of the fault tree. The second method identifies larger modules in the fault tree. Modules are independent sections of the failure logic diagram which

Originally published by the System Safety Society, in the Proceedings of the 21st International System Safety Conference, held at Ottawa, Canada, 2003

can be analysed separately and the results combined together to make predictions for the top event. Modularisation can provide very efficient solutions reducing large problems to the analysis of small manageable units. The most efficient means of identifying the modules is a method produced by Rauzy and Dutuit (ref. 10).

When analysing the modules a binary decision diagram method is used. This paper shows how the above approach is used to predict the system failure probability and system failure frequency when the failure mode is represented by a large, complex fault tree.

Binary Decision Diagrams

A binary decision diagram (BDD) is illustrated in figure 1.

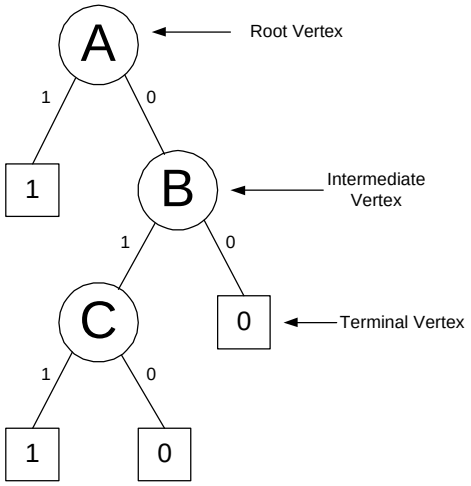

Figure 1- Binary Decision Diagram Structure

The diagram is entered at the root vertex. Each vertex or node represents a basic event from the fault tree and has two exit branches below it. If the event occurs then the node is left on the 1 branch. For the non-occurrence of the basic event the node is left on the 0 branch. When the set of component conditions is such that the top event is determined then a terminal-one node or a terminal-zero node occurs. A terminal-one vertex indicates top event occurrence and a terminal-zero vertex is the top event non-occurrence. This encodes the structure function of the fault tree. Each node in the BDD can be written as an ***ite*** format. This stands for ***if-then-else*** and is an ordered triple with a variable, a pointer to the one-branch and a pointer to the zero-branch, so ite(A,f1,f2) can be interpreted as:
if A
then consider function f1
else consider function f2.

The entire BDD in figure 1 can be expressed using this notation as:

ite(A, 1, ite(B, ite(C, 1, 0), 0))

Cut sets are combinations of component failures which cause the top event. On the BDD these can be tracked as component failure events which lead to a terminal-one vertex. So the BDD in figure 1 has cut sets A and BC. In this case the cut sets are minimal. However the significant

advantage to transforming the fault tree to the BDD form is gained when quantifying the top event probability and failure intensity. These can be obtained directly fro the BDD without need to produce a list of the minimal cut sets as an intermediate stage or resort to approximations.

The list of events contained on the N_p paths , through the BDD to a terminal-one, C_i, taking account of both success and failure states of the components, are disjoint (contain mutually exclusive sets of events). For the top event probability, Q_{sys}, then:

$$Q_{sys} = \sum_{i=1}^{N_p} P(C_i) \tag{1}$$

The system failure intensity, $w_{sys}(t)$, can be calculated from:

$$w_{sys}(t) = \sum_{i=1}^{n} G_i(\boldsymbol{q}(t)) w_i(t) \tag{2}$$

where $G_i(\boldsymbol{q}(t))$ is the criticality function (the probability that the system is in a critical state for component i such that the failure of i causes the system to pass from the working to the failed state) and $w_i(t)$ is the failure intensity for each of the n components.

$$G_i(\boldsymbol{q}(t)) = Q_{sys}(1_i, \boldsymbol{q}) - Q_{sys}(0_i, \boldsymbol{q}) \tag{3}$$

where $Q_{sys}(1_i, \boldsymbol{q})$ is the probability of system failure with q_i=1 and $Q_{sys}(0_i, \boldsymbol{q})$ is the probability of system failure with q_i=0. An efficient method to calculate the criticality function (equation 3) for each component is given in ref 11 and considers the probabilities of the path sections of the BDD up to and after the nodes of each variable x_i resulting in the following equation.

$$G_i(\boldsymbol{q}(t)) = \sum_{\substack{\textbf{nodes} \\ \boldsymbol{x_i}}} pr_{x_i}(\boldsymbol{q}(t))[po^1_{x_i}(\boldsymbol{q}(t)) - po^0_{x_i}(\boldsymbol{q}(t))] \tag{4}$$

where $pr_{x_i}(\boldsymbol{q}(t))$ is the probability of the path section from the root vertex to the node x_i; $po^1_{x_i}(\boldsymbol{q}(t))$ is the probability of the path section from the '1' branch of the node x_i to a terminal-one node; $po^0_{x_i}(\boldsymbol{q}(t))$ is the probability of the path section from the '0' branch of the node x_i to a terminal-one node. The summation is over all x_i nodes in the BDD.

Fault Tree to BDD Conversion Process

Details of the conversion process are well covered in other papers[2,3] and so will only be summarised here. The conversion is carried out pair-wise in a bottom-up procedure. Where a $\oplus$ gate is encountered, where $\oplus$ is either AND or OR, with inputs:

G = ite(X, G1, G2) and H = ite(Y, H1, H2)

then if X < Y then G $\oplus$ H = ite(X, G1 $\oplus$ H, G2 $\oplus$ H)
or if X = Y then G $\oplus$ H = ite(X, G1 $\oplus$ H1, G2 $\oplus$ H2)

Fault Tree Reduction

Stage 1 – Fault Tree Factorisation: The first stage of the fault tree reduction is to represent the basic logic structure with as little 'noise' as possible. This is achieved by a 3-stage process for which the basic elements of the procedure are given in reference 9. The stages are:

Contraction: Subsequent gates of the same type are contracted into a single gate. This structures the fault tree as an alternating sequence of *AND* and *OR* gates.

Factorisation: Pairs of events which always occur together in the same gate type are replaced with a 'complex event'. For identification purposes this is given a numerical label from 2000 upwards.

Extraction: Structures of the type shown in figure 2 are identified and restructured as indicated.

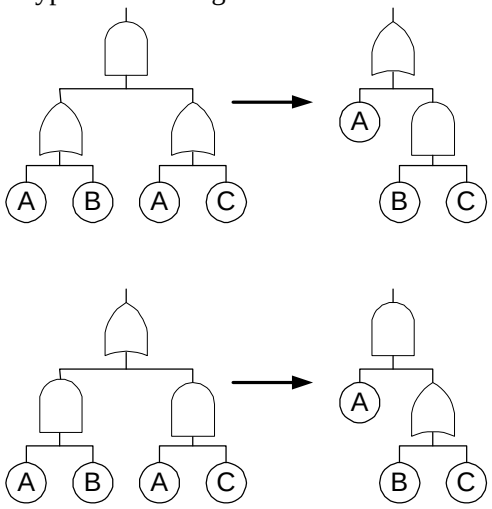

Figure 2 - Extraction structures

The three stages are applied repeatedly until no further reduction in the fault tree structure can be achieved.

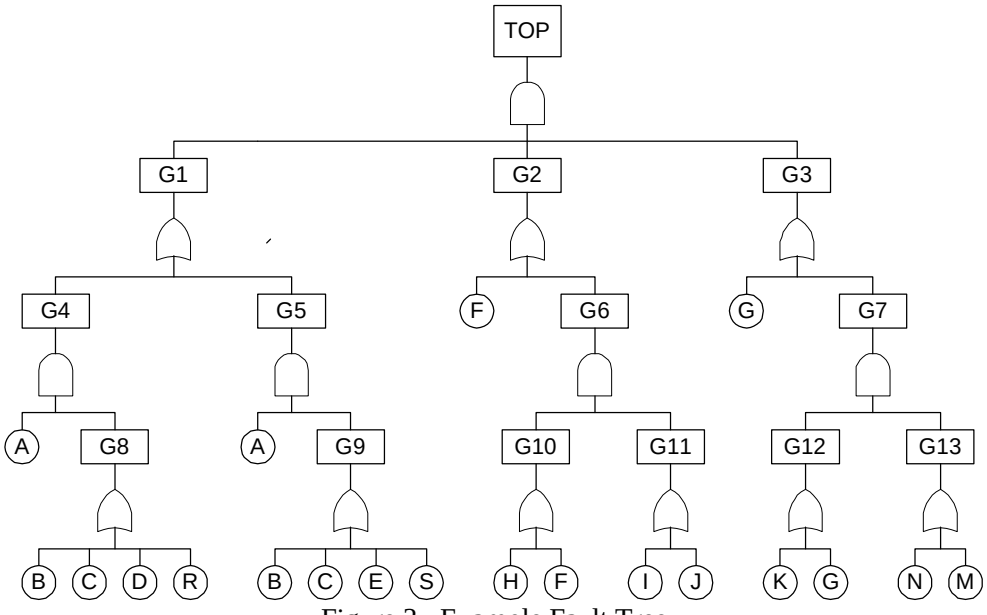

Figure 3 - Example Fault Tree

As an example of the methodology presented it will be applied to the fault tree shown in figure 3. The fault tree has 15 basic events many of which are repeated in the tree structure. This fault tree is small by real standards but can be used to demonstrate the method for analysis. As an indication of the size of the problem this fault tree has 54 minimal cut sets. The top down, left to right ordering of the basic events considers the fault tree one level at a time and passes left to right placing any events, not previously encountered, in the ordering list. As such the first level at which basic events are encountered features basic events F and G. Ordering from left to right places F and G as the first two events in the ordering. Progressing to the next level adds event A to the list. Finally the bottom row of the fault tree produces an ordering of the basic events:

F<G<A<B<C<D<R<E<S<H<I<J<K<N<M

The BDD for the fault tree produced with this ordering is shown in figure 4.

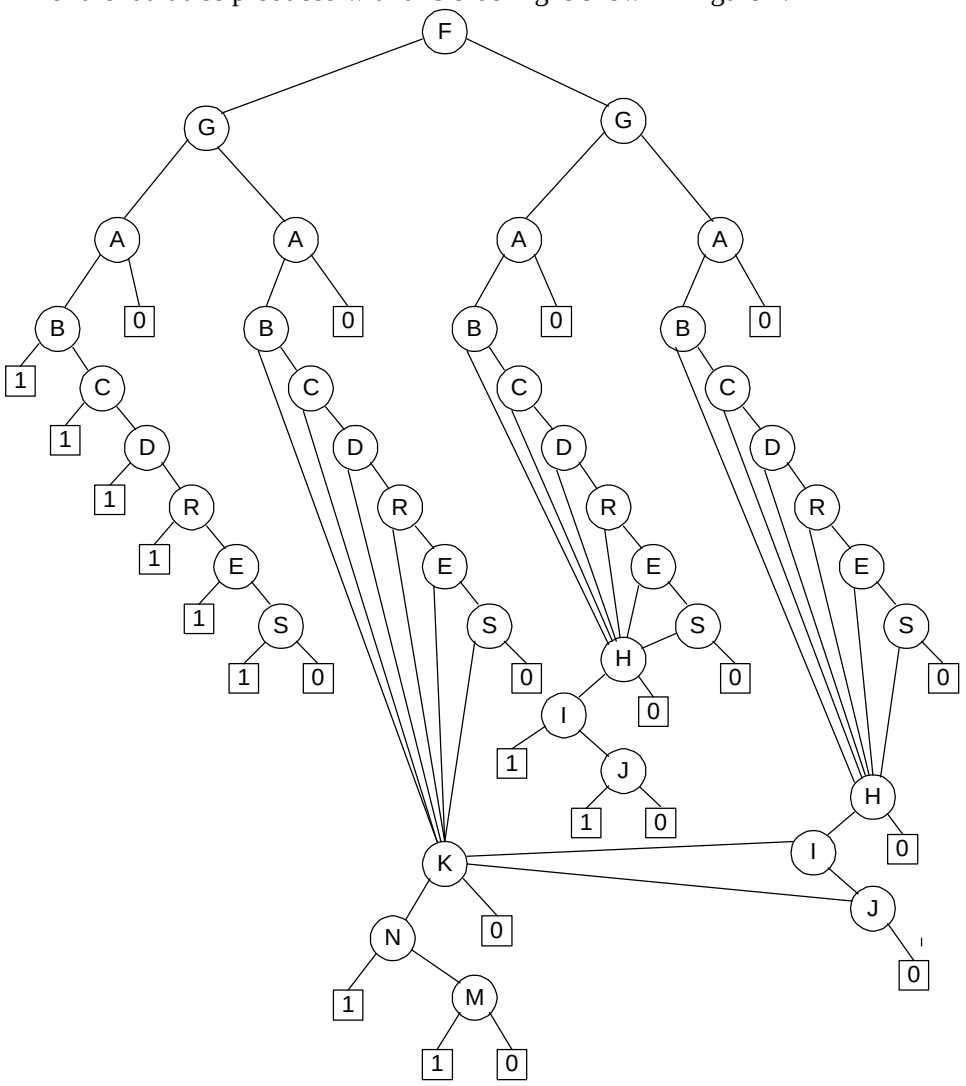

Figure 4 - BDD for example fault tree

The BDD has a total of 40 non-terminal nodes. Each branch leaving an intermediate node on the left is a one-branch and the right branch is a zero-branch.

Considering the factorisation process for the fault tree in figure 1 requires the stages of *contraction*, *factorisation* and *extraction* to be repeatedly and sequentially applied. The first application of contraction does not change the fault tree structure as it is already and alternating sequence of AND and OR gates. Factorisation produces the following pairs of events which always occur under the same gate type:

2000=B+C 2002=E+S 2004=N+M
2001=D+R 2003=I+J

By then applying extraction to gate G1, followed by a contraction stage and then factorisation to produce:

2005=2000+2001 2006=2005+2002 2007=A.2006

No further reduction is possible and the fault tree shown in figure 5 results.

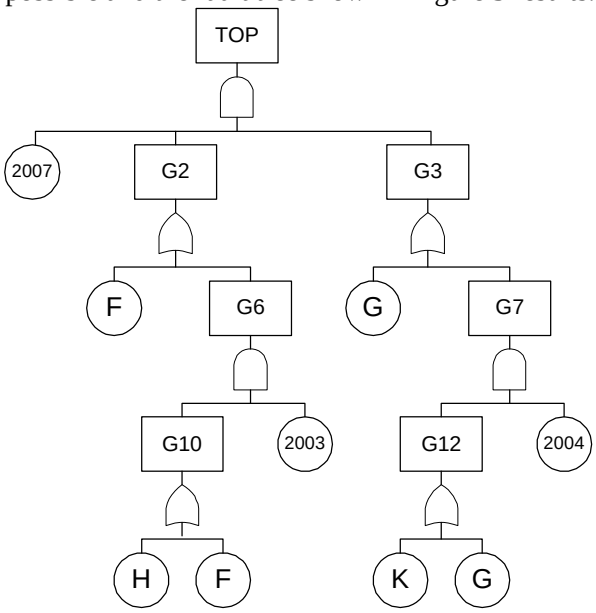

Figure 5 - Reduced Fault Tree Structure

Ordering the variables in a top-down, left-right manner provides an ordering:

2007 < F < G < 2003 < 2004 < H < K

The fault tree – BDD conversion process yields the BDD illustrated in figure 6. A reduction in the magnitude of the problem has already been produced as the BDD has only 11 intermediate nodes compared to 40 in the original BDD.

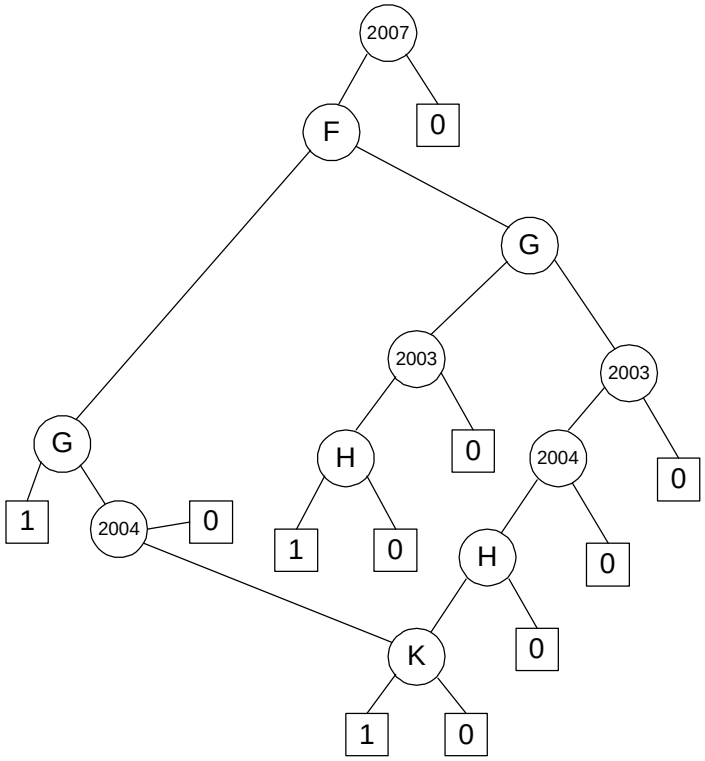

Figure 6 - BDD for the Factorised Fault Tree

Stage-2 Modularisation: This second stage of the reduction identifies subtrees which are completely independent of the rest of the structure. The method for identifying the modules was developed by Rauzy and Dutuit[10]. Once identified the modules can be analysed separately and the results substituted back into the remaining structure for the top event to make predictions on the system performance.

To find modules two depth-first traversals of the fault tree are made. The first performs a step-by-step visit to each gate and event recording the step number for the first, second and final visits. The step number for the second visit to each event is always the same as the first visit. To illustrate the method consider the fault tree shown in figure 5. Starting at the top gate and progressing through the structure in a depth-first manner means that the gates and events are encountered in the order shown in table 1. Where a gate has both basic events and other gates as inputs the basic events are always considered first.

step	Gate/event	step	Gate/event	step	Gate/event
1	TOP	9	F	17	G12
2	2007	10	G10	18	K
3	G2	11	G6	19	G
4	F	12	G2	20	G12
5	G6	13	G3	21	G7
6	2003	14	G	22	G3
7	G10	15	G7	23	TOP
8	H	16	2004		

Table 1 - Steps through the fault tree shown in figure 5

Each gate is visited at least twice, once on the way down and again on the way back up the tree. Once a gate has been visited it can be visited again, but the depth first traversal beneath that gate is not repeated. The step numbers for visits for the gates are shown in table 2 and the events in table 3.

	TOP	G2	G3	G6	G7	G10	G12
First visit	1	3	13	5	15	7	17
Second visit	23	12	22	11	21	10	20
Final visit	23	12	22	11	21	10	20
Min of first visits of inputs	2	4	14	4	14	4	14
Max of final visits of inputs	22	11	21	10	20	9	19
Module	Yes	Yes	Yes	No	No	No	No

Table 2 - Gate visit summary

	2007	F	2003	H	G	2004	K
First visit	2	4	6	8	14	16	18
Second visit	2	4	6	8	14	16	18
Final visit	2	9	6	8	19	16	18

Table 3 - Basic Event visit summary

The second pass of the fault tree finds the maximum of the final visits and the minimum of the first visits of any gates or basic events which appear at any level below the gate (see table 2).

The principal of the algorithm is that if any descendent of a gate has a first visit step number smaller than the first visit step number of the gate then it must have appeared before it in the tree structure. Conversely if any descendent of a gate has a final visit number larger than the final visit number of the gate then it must have appeared after the gate in the traversal. Therefore a gate can be identified as heading a module only if:

- the first visit to each of its descendents is after the first visit to the gate and
- the last visit to each of its descendents is before the last visit to the gate.

From table 2 it can be seen that the modules are headed by the events: TOP, G2 and G3.

The fault tree in figure 5 then reduces to solving the fault tree modules shown in figure 7. The BDDs for these modules, when the basic events are ordered in a top-down, left-right manner, are illustrated in figure 8.

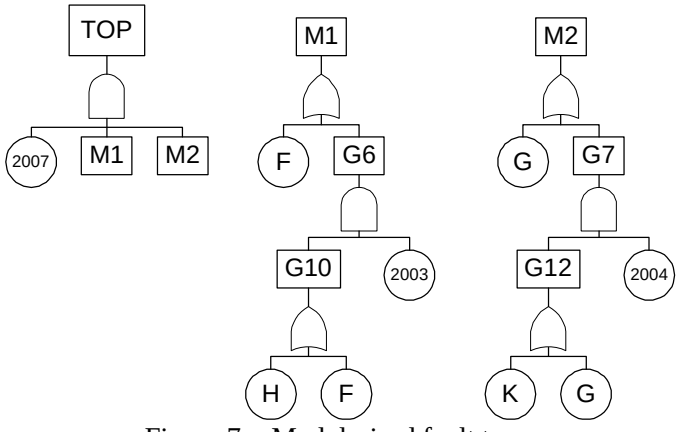

Figure 7 - Modularised fault trees

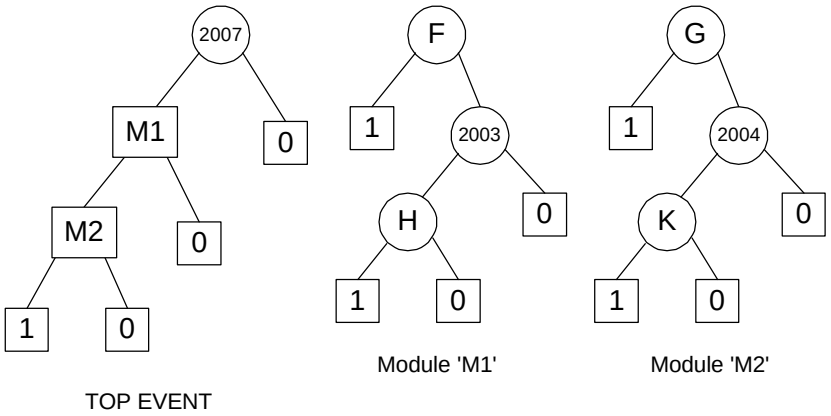

Figure 8 - BDDs for Fault Tree Modules

The problem has now reduced to solving BDDs with a combined total of 9 intermediate nodes. Much smaller than the original problem. In this small example the reduction in complexity is not critical for larger fault tree structures this can be essential for a solution.

Top Event Quantification

For any BDD consisting of only basic events the method for calculating the top event probability or failure intensity was described earlier. This section describes how this methodology can be extended to account for BDDs which feature complex events or modules.

Top Event probability

First the failure probability of the complex events and modular events is calculated. For complex event X_C with inputs X_1 and X_2, then:

For an AND gate $$q_C = q_1 q_2$$

For an OR gate $$q_C = q_1 + q_2 - q_1 q_2$$

The calculation for the modular events is the same as that described for calculating the probability of a complete BDD. Probabilities of failure for complex events and other modular events contained in the structure being analysed are necessarily evaluated first. When all complex events and modular events have been quantified then the BDD representing the top event can then be evaluated.

Top Event Frequency

For a BDD containing only basic events, one pass through the BDD calculating the parameters contained in equation 4 enables the failure intensity to be calculated using equation 2.

For BDDs with complex events and modules the criticality function, $G_i(\mathbf{q})$ for each basic event still needs to be calculated in order to use equation 2.

Criticality function of basic events within complex events: The criticality function can still be calculated using equation 4. For this we need to know $pr_C(\boldsymbol{q}(t))$ $po_C^1(\boldsymbol{q}(t))$ $po_C^0(\boldsymbol{q}(t))$ for the complex events. Since complex events can only be one of two forms, the BDDs are simply that of an AND gate or an OR gate. When these are inserted into the original BDD they have the structures illustrated in figure 9 where the terminal 1 nodes are replaced by $po_C^1(\boldsymbol{q}(t))$ and the terminal o nodes by $po_C^0(\boldsymbol{q}(t))$. The probability of the paths before the root node of the complex event BDD is $pr_C(\boldsymbol{q}(t))$ rather than 1.

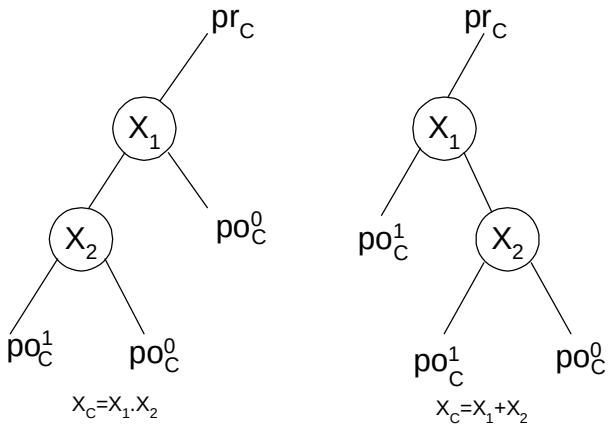

Figure 9 - Complex event BDDs

Using Figure 9 the values of $pr_{x_i}(\boldsymbol{q}(t))$ $po_{x_i}^1(\boldsymbol{q}(t))$ $po_{x_i}^0(\boldsymbol{q}(t))$ can be calculated for the variables X_1 and X_2 from:

AND

$$pr_{X_1} = pr_C$$

X_1

$$po_{X_1}^1 = q_{X_2} po_C^1 + (1 - q_{X_2}) po_C^0$$

$$po_{X_1}^0 = po_C^0$$

$$pr_{X_2} = pr_C q_{X_1}$$

X_2

$$po_{X_2}^1 = po_C^1$$

$$po_{X_2}^0 = po_C^0$$

OR

$$pr_{X_1} = pr_C$$

X_1

$$po_{X_1}^1 = po_C^1$$

$$po_{X_1}^0 = q_{X_2} po_C^1 + (1 - q_{X_2}) po_C^0$$

$$pr_{X_2} = pr_C(1 - q_{X_2})$$

X_2 $$po^1_{X_2} = po^1_C$$

$$po^0_{X_2} = po^0_C$$

Events X_1 and X_2 may be either basic events or other complex events, this process is repeated until values have been calculated for all the basic events. The criticality functions of the basic events are calculated using equation 3. Complex events can occur more than once in the BDD in which case the criticality functions must be summed.

Criticality function of basic events within modules: Modules are treated in a similar way to the complex events. The modular event has pr_m as the prior probability to the root of the module. po^1_m is the probability which occurs at any terminal-one node on the module and po^0_m the probability of any terminal-zero node on the module. The contributions for each component X_i: pr_{x_i} $po^1_{x_i}$ and $po^0_{x_i}$ are then derived in terms of the similar quantities for the module. This differs from the approach taken for the complex events only in that the structures of the modules are not predetermined. Where modules appear more than once in the fault tree structure the calculations are repeated for each occurrence and the criticality contributions for the basic events summed.

Conclusions

A strategy has been produced which enables the analysis for large fault tree structures. The approach makes use of the Binary Decision Diagram and its inherent accuracy and efficiency. In breaking the analysis down into smaller sections by factorisation and modularisation, the selection of an appropriate variable ordering system for the BDDs becomes less critical.

References

1. Andrews J.D. and Moss T.R., Reliability and Risk Assessment, 2nd Edition, Professional Engineering Publishing Ltd, 2002.
2. Rauzy A, New Algorithms for fault tree analysis, Reliab Engng Syst Safety, 1993, 40, pp203-211.
3. Sinnamon R.M. and Andrews J.D., Improved Efficiency in Qualitative Fault Tree Analysis, Quality and Reliability Engineering International, Vol 13, 1997, pp285-292.
4. Sinnamon R.M. and Andrews J.D., Increased accuracy in Quantitative Fault Tree Analysis, , Quality and Reliability Engineering International, Vol 13, 1997, pp299-309.
5. Nikolskaia, M, Binary Decision Diagrams and Applications to Reliability Analysis, Doctoral Thesis, University of Bordeaux, 1999.
6. Bartlett L.M. and Andrews J.D., Efficient Basic Event Ordering Schemes for Fault Tree Analysis, Quality and Reliability Engineering International, 15, 1999, pp95-101.
7. Bartlett L.M. and Andrews J.D., Selecting an Ordering Heuristic for the Fault Tree-Binary Decision Diagram Conversion Process using Neural Networks, IEEE Trans Reliability, 51, No 3 2002, pp344-349
8. Bouissou M, An Ordering Heuristic for Building Binary Decision Diagrams from Faul Trees, Proc of Reliability and Maintainability Symposium , ARMS96, Jan 1996, pp208-214.

9. Platz O and Olsen J.V., FAUNET: a Program Package for Evaluation of Fault Tress and Networks, Research Establishment Risk Report No 348, DK-4000 Roskilde, Denmark, Sept 1976.
10. Dutuit Y and Rauzy A, A Linear-time Algorithm to find Modules of Fault Trees, IEEE Trans Reliab, 1996, 45(3).
11. Sinnamon R.M. and Andrews J.D., Quantitative Fault Tree Analysis using Binary Decision Diagrams, Eur J Automation, 1996, 30(8).

Biography

Prof. John Andrews, PhD., Department of Systems Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, England., telephone +44 (0)1509 227286, e-mail – J.D.Andrews@lboro.ac.uk

John Andrews joined Loughborough University, Department of Mathematical Sciences in 1989 having spend 10 years performing industrial research. Recently he transferred to the newly formed Department of Systems Engineering. He has numerous publications on risk and reliability methods including the jointly authored text Reliability and Risk Assessment now in its second edition.