System Probability Distribution Modeling using MDDs

M. A. Thornton, T. W. Manikas, S. A. Szygenda
Southern Methodist University
Dallas, Texas USA
Email: {mitch,manikas,szygenda}@lyle.smu.edu

Shinobu Nagayama
Hiroshima City University
Hiroshima, JAPAN
Email: s_naga@hiroshima-cu.ac.jp

Abstract

Multi-Valued (MV) fault trees can be used to represent a variety of probability distributions characterizing system-related events. Representing MV fault trees in the form of multiple-valued decision diagrams (MDD) provides a means for representing overall system probability distributions and are constructed from structure functions. MDD edges are annotated with component probability values and allow for computation of overall system probability values. MDD 'phantom' vertices are presented to overcome inaccuracies introduced by the 'rare event approximation'. Additionally, a method that allows continuous probability distributions as MDD edge annotations is described. Experimental results are provided that illustrate the viability of the method.

1. Introduction

Modern systems and processes are becoming more complex and involve ever increasing numbers of subsystems and components. Design and analysis tasks include calculations of overall reliability, availability, threat vulnerability, and other system-level probabilistically distributed events. A past approach devised for these calculations uses "fault trees" that were developed at Bell Laboratories in 1961 [1]. Traditionally, fault trees are used to model failure or reliability distributions where subsystem components are assumed to have the binary states of "FAILURE" or "OPERATIONAL." Correspondingly, overall system analysis based on a fault tree model results in a composite estimate of the system being in one of these two states.

The idea of extending binary fault trees to multiple-valued fault trees for multi-state system reliability is described in [2]. In [3], the similar concept of a cyber threat tree is used for the purpose of analyzing the disaster tolerance of large systems in the presence of threats. Because some threats can render a system to be partially operational, the incorporation of intermediate states of operability that are not fully operational nor one of catastrophic failure motivates the use of multiple-valued ($p > 2$) states rather than only $p = 2$ states. The corresponding binary cyber threat tree is extended to a multiple-valued (MV) tree where symbolic AND and OR gates are replaced with multiple-valued MIN and MAX gates.

The use of binary decision diagrams (BDD) for representing fault trees was first proposed in [4] and was extended to use Multiple-valued Decision Diagrams (MDD) in [2]. This concept is used for disaster-tolerance analysis in [3] where a mark-up language, CyTML, represents cyber-threat trees and a companion parser converts the CyTML into a corresponding MDD with edge annotations. The cyber-threat tree approach is generalized for application to other analysis tasks in [5]. The work in [6] extends that of [5] by using an Edge-Valued MDD (EVMDD) software to provide additional experimental results and analysis of the computational memory resource requirements. New algorithms to minimize the number of edges in the EVMDDs are reported in [7]. Other applications of the MV fault tree analysis approach are reported in [8] and [9]. In [8], analysis of the fault tolerance of medical devices is described and includes the use of conditional probabilities. The work reported in [9] extends the MV fault tree approach to that of processes instead of systems.

In the work described here, we continue to use the term 'fault tree' since it is widely recognizable; however, the structure can represent any of a variety of probability distribution functions characterizing events within a system or process that may not necessarily be "faults." With this viewpoint, the fault tree structure is used in general as a means for describing composite system or process probability distribution functions in a compact manner. An important distinction between fault trees and switching circuit diagrams is that the variables are events with corresponding probability

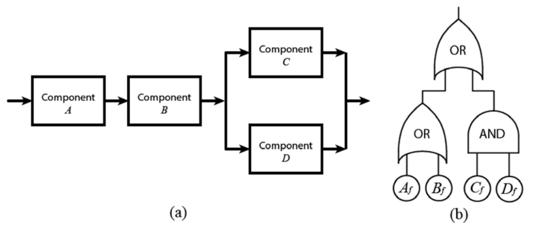


Figure 1. (a) System Diagram, (b) Fault Tree

distributions rather than circuit signal values.

The contribution of this paper is the introduction of methods to increase the accuracy of overall system probability values that are computed using MDDs. We also discuss the process of determining the "structure function" that indicates how events within a system are combined to form a fault tree. Two methods are introduced that increase the accuracy of system probability computations. First, we discuss the notion of a "phantom" vertex used to account for higher-ordered terms in probability computations. Secondly, a new method for annotating the edge values of an MDD with curve-fitting parameters obtained from system component probability distribution functions is included.

2. Fault Tree Concepts

As an example to describe basic fault tree concepts, we use a physical system as shown in Fig. 1a. We assume that each component has one of two basic states; either "OPERATIONAL" or "FAILURE." Due to the two-state nature of the system components, the probability distribution for each component consists of two point probabilities, $R = 1 - F$ is the reliability and F is the failure probability. The resulting fault tree shown in Fig. 1b uses binary operators depicted as binary switching circuit gates with inputs that are *events* representing component failure.

The *OR* and *AND* switching circuit gates represent additive and multiplicative probability relationships respectively. In the example system, components A and B are in series so the entire system will enter a state of failure when either component A *or* B fails and the corresponding probabilistic relationship is $F(A_f + B_f) = F(A_f) + F(B_f) - F(A_f \cap B_f)$. Depending on the nature of the events being modeled, it can be the case that the events are mutually exclusive resulting in $F(A_f + B_f) = F(A_f) + F(B_f)$ since $F(A_f \cap B_f) = 0$. It is common that failure analyses of system components assume the event of failure for component A is independent of that for component B. Using the independence assumption, the overall expression becomes $F(A_f + B_f) = F(A_f) + F(B_f) - F(A_f)F(B_f)$ since mutual exclusiveness does not hold but $F(A_f \cap B_f) = F(A_f)F(B_f)$. A further approximation is the so-called "rare event approximation" [1]. The rare event approximation states that since the magnitude of component failure probabilities is generally low, the joint failure term for non-mutually-exclusive events, $F(A_f)F(B_f)$, may be neglected, resulting in $F(A_f + B_f) \cong F(A_f) + F(B_f)$. Therefore, the overall failure probability for series components is additive in nature and is represented in the fault tree with the *OR* gate in the lower left position in Fig. 1b.

Components C and D are in parallel and for overall system failure to occur, both components C *and* D must fail. Assuming independence of the component failure events C_f and D_f, the overall expression for failure of both components C and D is $F(C_f \cap D_f) = F(C_f)F(D_f)$. This multiplicative relationship is represented in the fault tree through the use of the *AND* gate in the lower right of Fig. 1b. Finally, the intermediate series combination of A and B and the intermediate parallel combination of C and D are both themselves in series, therefore the topmost *OR* gate in Fig. 1b combines these component combinations.

2.1. System Structure Functions

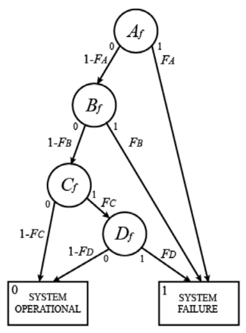
A *structure function* refers to the switching function represented by the fault tree. The determination of the structure function requires the states of each component to be mapped to discrete values appropriate for input to the logic gates followed by a synthesis operation. The topology of the system only partially dictates the structure function. Other important factors are the discrete value mappings and the nature of the events. In the current example, the fault tree is being used to represent system failure; however, if a fault tree were being constructed for system reliability, a different structure function would result.

In the example system, the structure function and resulting fault tree are constructed through a methodical reasoning about the failure events and knowledge of the system interconnections. Similar techniques for determination of the structure function are commonly used and prior methods have generally not viewed the fault tree construction process in terms of discrete value mappings and synthesis. When considered in terms of logic synthesis, the example system in Fig. 1a assumes that a failure is mapped to a logic 1 and the operational states are mapped to logic 0. After performing this mapping, a truth table is formulated as shown in Fig. 2a and represents the structure function $S_f = A_f + B_f + C_f D_f$.

Switching functions can be efficiently represented in the form of a decision diagram and this was first proposed for fault tree structure functions in [4]. Fig.

A_f	B_f	C_f	D_f	S_f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(a)



(b)

Figure 2. Example System Structure Function

2b depicts the edge-annotated BDD representation of the fault tree in Fig. 1b. The edges are annotated with the point probability of failure (F_i) or reliability $(1 - F_i)$ values and also with one of the two mapped switching constants. The terminal vertices of the BDD represent the overall system state of failure (mapped to value 1) or operation (mapped to value 0). A depth-first traversal can be used where the path probability for each path from the initial to root node of interest is computed by multiplying the edge weights along the path followed by summing the path probabilities together at the terminal vertex of interest. Details of this algorithm are described in more detail in [3] [5] [6].

From the preceding discussion, the combination of the structure function and the probability distributions of system components result in a fault tree that represents a composite probability distribution function. Depending upon the particular types of events being considered, any of a variety of distribution functions may be represented. For this reason, the structure function is only partially related to the topology of the system and is not uniquely determined by the system interconnections alone.

In traditional fault trees representing failure or reliability, events are considered to have only two outcomes, either "FAILURE" or "OPERATIONAL" resulting in a binary structure function. The use of constant probability values is appropriate when the failure distribution curve is constant or nearly constant over the lifetime of the component such as the the "bathtub" curve. The nearly constant or "lifetime" portion of the bathtub curve allows for a single probability value to be used for the probability of failure.

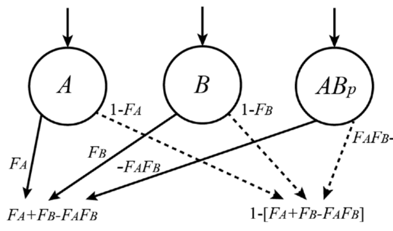


Figure 3. Phantom DD Vertex

2.2. Rare Event Approximation

The rare event approximation is applicable when individual probability values are sufficiently small. This is often the case for system failure computations since the probability of a component failure is typically small, thus the multiplicative term $F(A_f)F(B_f)$ becomes negligible. When the fault tree concept is used to represent more general distributions of events, the approximation may introduce too much error in the overall system probability value. Thus, we introduce the concept of a "phantom" vertex in the decision diagram representation of the structure function. The structure of the fault tree remains the same; however, the extraction of the structure function from the tree for the purpose of constructing a decision diagram changes in that when an *OR* or *MAX* gate is encountered, a decision is made as to whether a phantom node should be included or not. This decision is based upon the consideration of the size of the magnitude of the probability values. To illustrate the use of phantom nodes, we describe an example using a simple binary *OR* operator, although the principle is easily extended to higher-valued radices where the *MAX* gate is used.

Consider the portion of the system depicted in Fig. 1a comprised of the series components A and B. The expression for the probability of failure is $F_{AB} = F(A_f) + F(B_f) - F(A_f \cap B_f)$. Assuming the failure distributions for components A and B are independent and not mutually exclusive, the overall failure probability becomes $F_{AB} = F(A_f) + F(B_f) - F(A_f)F(B_f)$ and the "rare event approximation" neglects the $-F(A_f)F(B_f)$ term. When individual failure probabilities are sufficiently large, it may not be desirable to neglect this term. The inclusion of a phantom vertex in parallel with those representing components A and B allows this term to be included in the overall system probability computation. Fig. 3 illustrates the phantom node occurring in parallel to the vertices representing components A and B. The phantom node has exiting edges that are annotated with negative values corresponding to the term $-F(A_f)F(B_f)$.

2.3. MV Fault Trees

When system components or sub-processes do not have constant probability distributions or when more than two event outcomes are desired, the use of binary operators in the fault tree may not be adequate. This observation provides the motivation for the use of MV fault trees. for “multi-state” systems [2]. The analogous MVL operations for the binary *OR* and *AND* operations are the multi-valued *MAX* and *MIN* operators. The MVL operators use identical switching gate symbols as those used for the binary case with the understanding that they produce non-binary results.

The radix, or number of permissible discrete switching values of each MVL fault tree operator, depends upon the number of states being modeled. As an example, a ternary or three-valued switching system may be used to denote system failure states of “OPERATIONAL”, “DEGRADED”, or, “FAILURE”. Higher radices may be used to denote more intermediate states or degrees of degradation. Furthermore, a *mixed-radix* fault tree allows for each system component to be characterized with different outcomes. For example, some components may be more appropriately modeled with the binary outcome of “OPERATIONAL” or “FAILURE” while other components are modeled with a different set of outcomes such as “OPERATIONAL”, “DEGRADED”, and “FAILURE”.

As an example, consider the system shown in Fig. 1a where component A is modeled as having a ternary (radix-3) discrete point probability distribution for the states “OPERATIONAL”, “PARTIAL FAILURE”, and “COMPLETE FAILURE”. In this case, a mixed-radix MV fault tree is required to model the overall system failure distribution function. The structure function is determined by first assigning logic values to the component states and then synthesizing the resulting function. While the actual assignment of switching values is arbitrary, it does affect the structure of the resulting fault tree. We use the arbitrary discrete value mappings of 2 for “FAILURE”, 1 for “PARTIAL FAILURE”, and 0 for “OPERATIONAL”. Although system components B, C, and D continue to have binary states, their corresponding switching values are 0 for “OPERATIONAL” and 2 for “FAILURE” to maintain consistency with the ternary mapping for component A. The determination of optimal switching value assignments in terms of producing a compact decision diagram structure is left as an area of future research. The MV fault tree and corresponding decision diagram are shown in Fig 4. The MDD is obtained by synthesizing the fault tree using a standard MDD *APPLY* algorithm.

The MDD edges are annotated with the probabilities

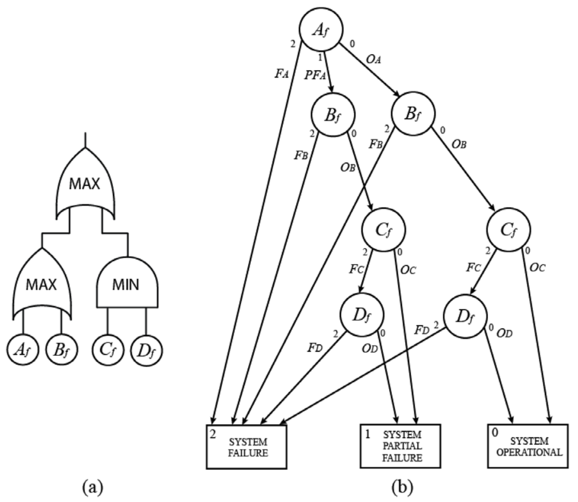


Figure 4. (a) MV Fault Tree, (b) MDD

of failure, F_i, partial failure, PF_i, and operational, O_i as well as the mapped switching values. For the binary-state components B, C, and D, $O_i = 1 - F_i$ and for the ternary-state component A, $F_A + PF_A + O_A = 1$. Overall system failure probabilities are computed using the algorithms in the previously cited work [3] [5] [6]. The EVMDD can be optimized through the use of a variety of optimization methods such as sifting resulting in a more compact structure.

3. Probability Distribution Representation

Many existing analysis methods for multi-state systems, including the methods described previously, utilize discrete random variables and their associated probability distributions. In practical applications, continuous random variables and their distributions are often required. More accurate analysis results can be achieved with a new extension to previous analysis methods that incorporate continuous distributions. Continuous probability distributions are usually given as PDFs or CDFs. Thus, we formulate the system analysis problem addressed in this section as follows:

Problem: Given a structure function S of a multi-state system and probability distributions for components as probability density functions (PDFs) or cumulative distribution functions (CDFs), compute the probability distribution of states in the multi-state system.

Each state of the components represents an interval of continuous values such as performance or reliability of multi-state systems. That is, each component state represents a range of continuous values. Probability density functions (PDFs) and cumulative distribution functions (CDFs) are continuous functions, and the

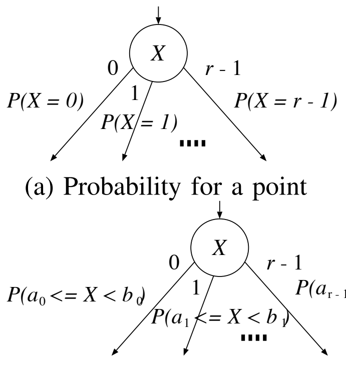


Figure 5. Edges Annotated with Probabilities

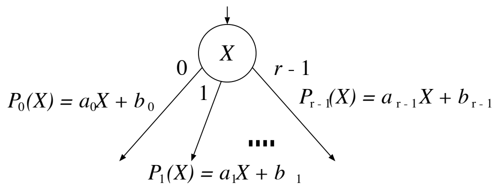


Figure 6. Edges Annotated with Polynomials

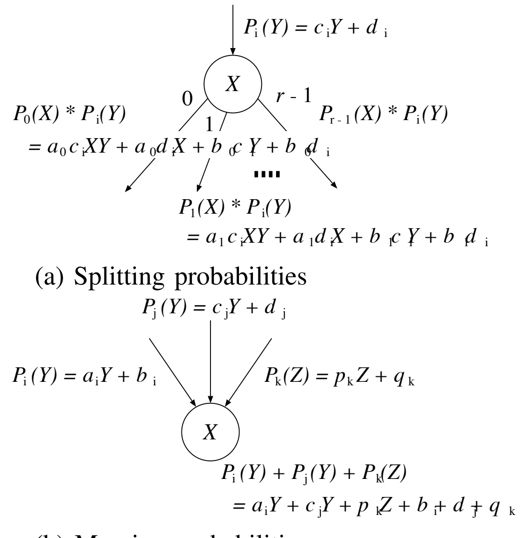

Pi(Y) = ciY + di
X
0
1
r - 1
P0(X) * Pi(Y)
= a0ciXY + a0diX + b0ciY + b0di
Pr-1(X) * Pi(Y)
P1(X) * Pi(Y)
= a1ciXY + a1diX + b1ciY + b1di
(a) Splitting probabilities
Pj(Y) = cjY + dj
Pi(Y) = aiY + bi
Pk(Z) = pkZ + qk
X
Pi(Y) + Pj(Y) + Pk(Z)
= aiY + cjY + pkZ + bi + dj + qk
(b) Merging probabilities

Figure 7. Computation of Probabilities at a Vertex

probability for a range $[a, b)$ can be computed by using PDFs or CDFs as shown in the following equations,

$$P(a \leq X < b) = \int_a^b PDF(x)dx \quad (1)$$

$$P(a \leq X < b) = CDF(b) - CDF(a) \quad (2)$$

where $P(a \leq X < b)$ is the probability for the range $[a, b)$, $PDF(x)$ and $CDF(x)$ are given probability density or cumulative distribution functions, respectively. In this way, the probability for each range (i.e., the probability for each component state) can be computed by Equations 1 and 2 even if probability distributions are given as continuous functions. Since the obtained probability for each component state is a single value, we can analyze multi-state systems using the same method as the analysis method for discrete probability distributions described in the previous work.

Fig. 5a shows an MDD vertex whose edges are annotated with *point probability values*, and Fig. 5b shows an MDD vertex whose edges are annotated with *probabilities obtained by the equation 1 or 2* for ranges. As shown in this figure, we can easily incorporate continuous probability distributions into decision diagrams. However, in the following subsections, we describe other methods to incorporate continuous probability distributions into decision diagrams.

3.1. Curve-fitting Method

The objective of this section is to represent continuous probability distributions using discrete MV structure functions so that the overall composite probability distribution can be represented using the MDD data structure. This requires determination of appropriate partitions of the distribution curve followed by mapping or assigning a logic value to each partition. The number of identified partitions becomes the radix value for the MDD vertex. The edge value(s) contain the information required to approximate the corresponding portion of the distribution curve. A variety of curve-fitting techniques may be used to represent the various portions of the distribution curve. One method is to determine a linear equation for each partition and then to annotate each edge with the slope of the fitted line m and the vertical axis with the intercept point b. In this manner, a linear interpolation can be performed during the traversal of the MDD to determine a more accurate value from the probability distribution curve. Any of a variety of curve-fitting techniques is possible, the key factors are the degree of accuracy required in approximating the distribution function and the number of parameters that are required for storing the curve-fit information. The identified parameters are then stored in the MDD data structure as edge value annotations.

Using a piecewise polynomial approximation, we can analyze multi-state systems similarly to the previous MDD-based analysis methods [5] [8] [6] [7] that use a single value for each probability. Instead of a single value, we assign a polynomial to each edge of an MDD, as shown in Fig. 6. Then, by multiplying and adding the polynomials, we obtain probability distributions of system states that are given

Table 1. Practical Systems Results

NAME	MDD SIZE	TIME
Redundant fire pumps [10]	1.6 KB	$<$ 1msec.
Engine starter [10]	41.1 KB	$<$ 1msec.
Overrun of motor [1]	44.3 KB	$<$ 1msec.

Table 2. Random Systems Results

Number of Components n	MDD SIZE	TIME
4	0.2 MB	0.02 sec.
5	0.6 MB	0.19 sec.
6	1.5 MB	1.19 sec.
7	3.5 MB	7.01 sec.
8	8.3 MB	72.42 sec.

as continuous distribution functions (i.e., polynomials). Multiplications and additions of polynomials can be realized by multiplications and additions of the corresponding coefficients obtained from the series expansions as shown in Figs. 7. Note that coefficients cannot be summed up when ranges for polynomial approximations are different even if the variable is the same. This is shown in Fig. 7b. In this figure, the Ys' ranges (i and j) are different from each other. Multiplications and additions of polynomials are performed at each MDD vertex in a top-down manner, and polynomials obtained at terminal vertices are probability distributions of the entire system state.

4. Experimental Results

To illustrate the usage of the technique described here, we implemented the curve-fitting method using MDDs to represent the structure function on the following computer environment: CPU: Intel Core2 Quad Q6600 2.4GHz; memory: 4GB, OS: CentOS 5.7; and C-compiler: gcc -O2 (version 4.1.2). The results of the experiments are given in Tables 1 and 2. For all these example systems, computer runtimes were less than 1 msec. To show the effectiveness of our method for larger systems, we randomly generated multi-state systems consisting of n 6-valued components as in [6]. From this table, it is shown that our method is a practical and viable approach for large systems.

5. Conclusions

The use of the fault tree as a means to represent probability distributions is shown to be viable for continuous and empirical distribution data through the use of MVL structure functions. The formulation of the structure function is described as a process of discrete switching value assignment followed by MVL logic synthesis. We have described two techniques for improving the accuracy of overall system or process probability representations using MDDs. The technique of including phantom vertices allows higher-ordered terms in additive probability relationships to be included in the calculations. The second enhancement involves the representation of continuous probability distributions when decision diagrams are used to represent fault trees. Experimental results indicate that our method is a practical approach for large system analysis tasks.

References

[1] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, *Fault Tree Handbook, NUREG-0492.* U.S. Nuclear Regulatory Commision, 1981.

[2] S. Garribba, E. Guagnono, and P. Mussio, "Multiple-valued logic trees: Meaning and prime implicants," *IEEE Trans. Rel.*, vol. 34, no. 5, pp. 463–472, Dec. 1985.

[3] P. Ongsakorn, K. Turney, M. A. Thornton, S. Nair, S. A. Szygenda, and T. W. Manikas, "Cyber threat trees for large system threat cataloging and analysis," in *Proceedings. IEEE Systems Conference*, 2010, pp. 610–15.

[4] J. B. Dugan and S. Doyle, "New results in fault-tree analysis," in *Tutorial Notes. Annual Reliability and Maintainability Symposium*, 1997.

[5] T. W. Manikas, M. A. Thornton, and D. Y. Feinstein, "Using multiple-valued logic decision diagrams to model system threat probabilities," in *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2011, pp. 263–7.

[6] S. Nagayama, T. Sasao, and J. T. Butler, "Analysis of multi-state systems with multi-state components using EVMDDs," in *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2012, pp. 122–7.

[7] ——, "Minimization of the number of edges in an evmdd by variable grouping for fast analysis of multi-state systems," in *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2013, pp. 284–9.

[8] T. W. Manikas, D. Y. Feinstein, and M. A. Thornton, "Modeling medical system threats with conditional probabilities using multiple-valued logic decision diagrams," in *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2012, pp. 244–9.

[9] T. W. Manikas, M. A. Thornton, and F. R. Chang, "Mission planning analysis using decision diagrams," in *Proceedings. Reed-Muller Workshop*, 2013, pp. 61–5.

[10] J. Marshall. (2012, accessed: 29 October, 2013) An introduction to fault tree analysis (FTA), University of Warwick. [Online]. Available: http://www2.warwick.ac.uk/fac/sci/wmg/ftmsc/modules/modulelist/peuss/slides/section_11b_fta_lecture_slides_compatibility_mode.pdf