Fault Tree Analysis: A survey of the state-of-the-art in modeling, analysis and tools

Enno Ruijters[†][*] and Mariëlle Stoelinga[†]
Formal Methods and Tools, University of Twente, The Netherlands
[†]E-mail: e.j.j.ruijters@utwente.nl (E. Ruijters), m.i.a.stoelinga@utwente.nl (M. I. A. Stoelinga)
[*]Corresponding author at: Universiteit Twente, t.a.v. Enno Ruijters, Vakgroep EWI-FMT, Zilverling, P.O. Box 217, 7500 AE Enschede

Abstract

Fault tree analysis (FTA) is a very prominent method to analyze the risks related to safety and economically critical assets, like power plants, airplanes, data centers and web shops. FTA methods comprise of a wide variety of modelling and analysis techniques, supported by a wide range of software tools. This paper surveys over 150 papers on fault tree analysis, providing an in-depth overview of the state-of-the-art in FTA. Concretely, we review standard fault trees, as well as extensions such as dynamic FT, repairable FT, and extended FT. For these models, we review both qualitative analysis methods, like cut sets and common cause failures, and quantitative techniques, including a wide variety of stochastic methods to compute failure probabilities. Numerous examples illustrate the various approaches, and tables present a quick overview of results.

Keywords: Fault Trees, Reliability, Risk analysis, Dynamic Fault Trees, Graphical models, Dependability Evaluation

Contents

1. Introduction

Risk analysis is an important activity to ensure that critical assets, like medical devices and nuclear power plants, operate in a safe and reliable way. Fault tree analysis (FTA) is one of the most prominent techniques here, used by a wide range of industries. Fault trees (FTs) are a graphical method that model how failures propagate through the system, i.e., how component failures lead to system failures. Due to redundancy and spare management, not all component failures lead to a system failure. FTA investigates whether the system design is dependable enough. It provides methods and tools to compute a wide range of properties and measures.

FTs are trees, or more generally directed acyclic graphs, whose leaves model component failures and whose gates failure propagation. Figure 1 shows a representative example, which is elaborated in Example 1.

Concerning analysis techniques, we distinguish between *qualitative* FTA, which considers the structure of the FT; and *quantitative* FTA, which computes values such as failure probabilities for FTs. In the qualitative realm, *cut sets* are an important measure, indicating which combinations of component failures lead to system failures. If a cut set contains too few elements, this may indicate a system vulnerability. Other qualitative measure we discuss are path sets and common cause failures.

Quantitative system measures mostly concern the computation of failure probabilities. If we assume that the failure of the system components are governed by a probability distribution, then quantitative FTA computes the failure probability for the system. Here, we distinguish between discrete and continuous probabilities. For both variants, the following FT measures are discussed. The *system reliability* yields the probability that the system fails with a given time horizon t; the *system availability* yields the percentage of time that the system is operational; the *mean time to failure* yields the average time before the first failure and the *mean time between failures* the average time between two subsequent failures. Such measures are vital to determine if a system meets its dependability requirements, or whether additional measures are needed. Furthermore, we discuss sensitivity analysis techniques, which determine how sensitive an analysis is with respect to the values (i.e., failure probabilities) in the leaves; we also discuss importance measures, which give means to determine how much different leaves contribute to the overall system dependability.

While SFTs (standard, or static, fault trees) provide a simple and informative formalism, it was soon realised that it lacks expressivity to model essential and often occurring dependability patterns. Therefore, several extensions to fault trees have been proposed, which are capable of expressing features that are not expressible in SFTs, like spare management, different operational modes, and dependent events. *Dynamic Fault Trees* are the best known, but extended fault trees, repairable fault trees, fuzzy fault trees, and state-event fault trees are popular as well. We discuss these extensions, as well as their analysis techniques.

In doing so, we have reviewed over 150 papers on fault tree analysis, providing an extensive overview of the state-of-the-art in fault tree analysis.

Organization of this paper As can be seen in the table of contents, this paper first discusses standard fault trees in Section 2, and then extensions that increase the expressiveness of the model. Dynamic fault trees, as the most widely used extension, is discussed in depth in Section 3, while other extensions are presented in Section 4.

For each of the models, we present the definition and structure of the models, then methods for qualitative analysis, and then methods for quantitative analysis (if applicable to the particular model). In each section, we discuss standard techniques is depth, while less common techniques are presented more briefly. Definitions of repeatedly used abbreviations and jargon can be found in Appendix A.

Note that all literature references in the electronic version are clickable, and that the reference list refers, for each paper, to the pages where that paper is cited.

1.1. Research Methodology

We intend for this paper to be as comprehensive as reasonable, but we cannot guarantee that we have found every relevant paper.

To obtain relevant papers, we searched for the keywords 'Fault tree' in the online databases
Google Scholar (`http://scholar.google.com`),
IEEExplore (`http://ieeexplore.ieee.org`),
ACM Digital Library (`http://dl.acm.org`),
Citeseer (`http://citeseerx.ist.psu.edu`),
ScienceDirect (`http://www.sciencedirect.com`),
SpringerLink (`http://link.springer.com`),
and SCOPUS (`http://www.scopus.com`). Further articles were obtained by following references from the papers found.

Articles were excluded that are not in English, or deemed of poor quality. Furthermore, to limit the scope of this survey, articles were excluded that present only applications of FTA, present only methods for constructing FTs, or only describe techniques for fault diagnosis based on FTs, unless the article also presents novel analysis or modeling techniques. Articles presenting implementations of existing algorithms were only included if they describe a concrete tool.

1.2. Related work

Apart from fault trees, there are a number of other formalisms for dependability analysis [37]. We list the most common ones below.

Failure Mode and Effects Analysis Failure Mode and Effects Analysis (FMEA) [144, 36] was one of the first systematic techniques for dependability analysis. FMEA, and in particular its extension with criticality FMECA (Failure Mode, Effects and Criticality Analysis), is still very popular today; users can be found throughout the safety-critical industry, including the nuclear, defence [174], avionics [73], automotive [11], and railroad domains. These analyses offer a structured way to list possible failures and the consequences of these failures. Possible countermeasures to the failures can also be included in the list.

If probabilities of the failures are known, quantitative analysis can also be performed to estimate system reliability and to assign numeric criticalities to potential failure modes and to system components [174].

Constructing an FME(C)A is often one of the first steps in constructing a fault tree, as it helps in determining the possible component failures, and thus the basic events [168].

HAZOP analysis A hazard and operability study (HAZOP) [105] systematically combines a number of guidewords (like *insufficient*, *no*, or *incorrect*) with parameters (like *coolant* or *reactant*), and evaluates the applicability of each combination to components of the system. This results in a list of possible hazards that the system is subject to. The approach is still used today, especially in industrial fields like the chemistry sector.

A HAZOP is similar to an FMEA in that both list possible causes of a failure. A major difference is that an FMEA considers failure modes of components of a system, while a HAZOP analysis considers abnormalities in a process.

Reliability block diagrams Similar to fault trees, reliability block diagrams (RBDs) [127] decompose systems into subsystems to show the effects of (combinations of) faults. Similar to FTs, RBDs are attractive to users because the blocks can often map directly to physical components, and because they allow quantitative analysis (computation of reliability and availability) and qualitative analysis (determination of cut sets).

To model more complex dependencies between components, Dynamic RBDs [61] include standby states where components fail at a lower rate, and triggers that allow the modeling of shared spare components and functional dependencies. This may improve the accuracy of the computed reliability and availability.

OpenSESAME The OpenSESAME modeling environment [182] extends RBDs by allowing more types of intercomponent dependencies, common cause failures, and limited repair resources. This is mostly an academic approach and sees little use in industry.

SAVE The system availability estimator (SAVE) [85] modeling language is developed by IBM, and allows the user to declare components and dependencies between them using predefined constructs. The resulting model is then analyzed to determine availability.

AADL The Architecture Analysis and Design Language (AADL) [165] is an industry standard for modeling safety-critical systems architectures. A complete AADL specification consists of a description of *nominal* behaviour, a description of *error* behaviour and a *fault injection* specification that describes how the error behaviour influences the nominal behaviour.

Such an AADL specification can be used to derive an FMEA table [90] in a systematic way. One can also automatically discover failure effects that may be caused by combinations of faults [72]. If failure rates are known, quantitative analysis can also determine the system reliability and availability [36].

UML Another industry standard for modeling computer programs, but also physical systems and processes, is the Unified Modeling Language (UML) [156]. UML provides various graphical models such as Statechart diagrams and Sequence diagrams to assist developers and analysts in describing the behaviours of a system.

It is possible to convert UML Statechart diagrams into Petri Nets, from which system reliability can be computed [25, 20]. Another approach combines several UML diagrams to model error propagation and obtain a more accurate reliability estimate [138].

Möbius The Möbius framework was developed by Sanders et al. [59, 158] as a multi-formalism approach to modeling. The tool allows components of a system to be specified using different techniques and combined into one model. The combined model can then be analyzed for reliability, availability, and expected cost using various techniques depending on the underlying models.

1.3. Legal background

FTA plays an important role in product certification, and to show conformance to legal requirements. In the European Union, legislature mandates that employers assess and mitigate the risks that workers face [2]. FTA can be applied in this context, e.g. to determine the conditions under which a particular machine is dangerous to workers [96]. The U.S. Department of Labor has also accepted the use of FTA for risk assessment in workplace environments [132].

Similarly, the EU Machine Directive [1] requires manufacturers to determine and document the risks posed by the machines they produce. FTA is one of the techniques that can be used for this documentation [93].

The transportation industry has also adopted risk analysis requirements, and FTA as a technique for performing such analysis. The Federal Aviation Administration adopted a policy in 1998 [74] requiring a formalized risk management policy for high-consequence decisions. Their System Safety Handbook [75] lists FTA as one of the tools for hazard analysis.

2. Standard Fault Trees

As discussed in the previous section, it can be necessary to analyze system dependability properties. A fault tree is a graphical model to do so: It describes the relevant failures that might occur in the system, and how these failures interact to possibly cause a failure of the system as a whole.

Standard, or static, fault trees (SFTs) are the most basic fault trees. They have been introduced in the 1960s at Bell Labs for the analysis of a ballistic missile [71]. The classical *Fault Tree Handbook* by Vesely et al. [177] provides a comprehensive introduction to SFTs. Below, we describe the most prominent modelling and analysis techniques for SFTs.

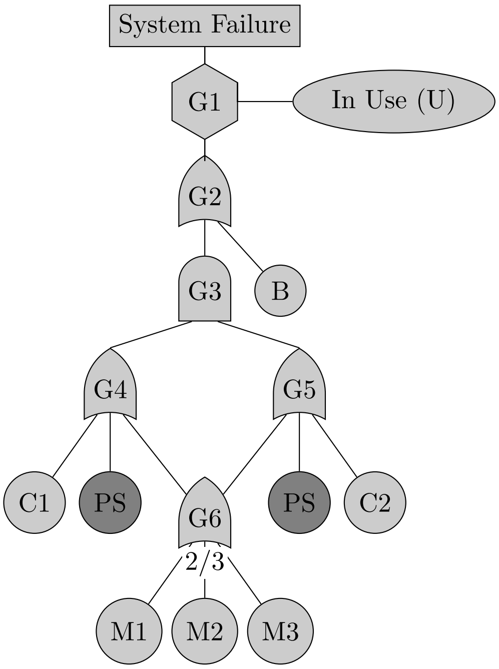
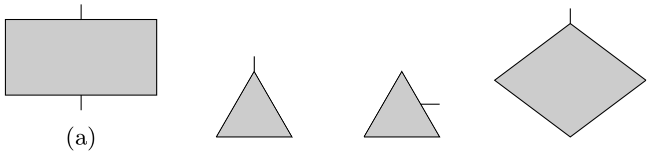


Figure 1: Example FT of a computer system with a non-redundant system bus (B), power supply (PS), redundant CPUs (C1 and C2) of which one can fail with causing problems, and redundant memory units (M1, M2, and M3) of which one is allowed to fail; failures are propagated by the gates (G1-G6). PS is somewhat darker to indicate that both leaves correspond to the same event.



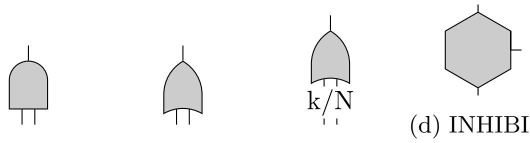
(a) Intermediate event (b) Transfer in (c) Transfer out (d) Undeveloped event

Figure 2: Images of non-basic events in fault trees

2.1. Fault Tree Structure

A fault tree is a directed acyclic graph (DAG) consisting of two types of nodes: *events* and *gates*. An event is an occurrence within the system, typically the failure of a subsystem down to an individual component. Events can be divided into *basic events (BEs)*, which occur spontaneously, and *intermediate events*, which are caused by one or more other events. The event at the top of the tree, called the *top event (TE)*, is the event being analyzed, modeling the failure of the (sub)system under consideration.

In addition to basic events depicted by circles, Figure 2 shows other symbols for events. An intermediate event is depicted by a rectangle. Intermediate events can be useful for documentation, but do not affect the analysis of the FT, and may therefore be omitted. If an FT is too large to fit on one page, triangles are used to *transfer* events between multiple FTs to act as one large FT. Finally, sometimes subsystems are not really BEs, but insufficient information is available or the event is not believed to be of sufficient importance to develop the subsystem into a subtree. Such an *undeveloped event* is denoted by a diamond.



(a) AND gate (b) OR gate (c) k/N gate (d) INHIBIT gate

Figure 3: Images of the gates types in a standard fault tree

2.1.1. Gates

Gates represent how failures propagate through the system, i.e. how failures in subsystems can combine to cause a system failure. Each gate has one output and one or more inputs. The following gates are commonly used in fault trees. Images of the gates are shown in Figure 3.

AND Output event occurs if all of the input events occur, e.g. gate G3 in the example.

OR Output event occurs if any of the input events occur, e.g. gate G2 in the example.

k/N a.k.a. VOTING, has N inputs. Output event occurs if at least k input events occur. This gate can be replaced by the OR of all sets of k inputs, but using one k/N gate is much clearer. Gate G6 in the example is a 2/3 gate.

INHIBIT Output event occurs if the input event occurs while the conditioning event drawn to the right of the gate also occurs. This gate behaves identically to an AND-gate with two inputs, and is therefore not treated in the rest of this paper. It is sometimes used to clarify the system behaviour to readers. Gate G1 in the example is an INHIBIT gate.

Several extensions of FTs introduce additional gates that allow the modelling of systems that can return to a functional state after failure. These 'Repairable Fault Trees' will be described in Section 4.3. Note that other formalisms (including standard FTs) include repairs, but do not model them with additional gates.

Other extensions include a NOT-gate or equivalent, so that a component failure can cause the system to go from failed to working again [110], or a functioning component can contribute to a system failure. Such a system is called noncoherent. It may indicate an error in modeling [177], however some systems naturally exhibit noncoherent behaviour: For example, the combination of a failed safety valve and a functioning pump can lead to an explosion, while a failed pump always prevents this.

Example 1. *Figure 1 (modified from Malhotra and Trivedi [120, 14]) shows a fault tree for a partially redundant computer system. The system consists of a bus, two CPUs 3 memory units, and a power supply. These components are represented as basic events in the leaves of the tree, B, C1, C2, M1, M2, M3, and PS respectively. The top of the tree (labeled System Failure here) represents the event of interest, namely a failure of the computer system.*

As stated, gates represent how failures propagate from through the system: Gate G1 is an Inhibit-gate indicating that a system failure is only considered when the system is in use, so that faults during intentional downtime do not affect dependability metrics.

The OR gate G2, just below G1, indicates that the failure of either the bus (basic event B) or the computing subsystem causes a system failure. The computing subsystem consists of two redundant units combined using an AND gate G3 so that both need to fail to cause an overall failure. Each unit can fail because either the CPU (C1 or C2) fails or the power supply (PS) fails. Note that the event PS is duplicated for each subtree, but still represents a single event.

A failure of the memory subsystem can also cause a unit to fail, but this requires a failure of two memory units. This is represented by the 2/3 gate G6. This gate is an input of both compute subsystems, making this a DAG, but the subtree could also have been duplicated if the method used required a tree but allowed repeated events.

2.1.2. Formal definition

To formalize an FT, we use $GateTypes = \{And, Or\} \cup \{\, VOT(k/N) \mid k, N \in \mathbb{N}^{>1}, k \leq N\}$. Following Codetta-Raiteri et al. [52], we formalize an FT as follows.

Definition 2. *An* FT *is a 4-tuple $F = \langle BE, G, T, I\rangle$, consisting of the following components.*

- *BE is the set of basic events.*
- *G is the set of gates, with $BE \cap G = \emptyset$. We write $E = BE \cup G$ for the set of* elements.
- *$T : G \mapsto GateTypes$ is a function that describes the type of each gate.*
- *$I : G \to \mathcal{P}(E)$ describes the inputs of each gate. We require that $I(g) \neq \emptyset$ and that $|I(g)| = N$ if $T(g) = VOT(k/N)$.*

Importantly, the graph formed by $\langle E, I\rangle$ should be a directed acyclic graph with a unique root TE which is reachable from all other nodes.

This description does not include the INHIBIT gate, since this gate can be replaced by an AND. The INHIBIT gate may, however, be useful for documentation purposes. Also, intermediate events are not explicitly represented, again because they do not affect analysis.

Some analysis methods described in Sections 2.2 and 2.3 require the undirected graph $\langle E, I\rangle$ to be a tree, i.e., forbid shared subtrees. In this paper, an FT will be considered a DAG. An element that is the input of multiple gates can be graphically depicted in two ways: The element (and its descendants) can be drawn multiple times, in which case the FT still looks like a tree, or the element can be drawn once with multiple lines connecting it to its parents. Since these depictions have the same semantics, we refer to these elements as *shared* subtrees or *shared* BEs regardless of graphical depiction.

2.1.3. Semantics

The semantics of an FT F describes, given a set S of BEs that have failed, for each element e, whether or not that element fails. We assume that all BEs not in S have not failed.

Definition 3. *The semantics of FT F is a function $\pi_F : \mathcal{P}(BE) \times E \mapsto \{0,1\}$ where $\pi_F(S, e)$ indicates whether e fails given the set S of failed BEs. It is defined as follows.*

- *For $e \in BE$, $\pi_F(S, e) = e \in S$.*
- *For $g \in G$ and $T(g) = \text{And}$, let* $\pi_F(S, g) = \bigwedge_{x \in I(g)} \pi_F(S, x)$.
- *For $g \in G$ and $T(g) = \text{Or}$, let* $\pi_F(S, g) = \bigvee_{x \in I(g)} \pi_F(S, x)$.
- *For $g \in G$ and $T(g) = \text{VOT}(k, N)$, let* $\pi_F(S, g) = \left(\sum_{x \in I(g)} \pi_F(S, x)\right) \geq k$.

Note that the AND gate with N inputs is semantically equivalent to an *VOT(N/N)* gate, and the OR gate with N inputs is semantically equivalent to a *VOT(1/N)* gate.

In the remainder of this paper, we abbreviate the interpretation of the top event t by stating $\pi_F(S, t) = \pi_F(S)$.

It follows easily that standard FT are *coherent*, i.e. if event set S leads to a failure, then every superset S' also leads to failure. Formally, $S \subseteq S' \wedge \pi_F(S, x) = 1 \Rightarrow \pi_F(S', x) = 1$.

2.2. Qualitative analysis of SFTs

Fault tree analysis techniques can be divided into quantitative and qualitative techniques. *Qualitative techniques* provide insight into the structure of the FT, and are used to detect system vulnerabilities. We discuss the most prominent qualitative techniques, being (minimal) cut sets, (minimal) path sets, and common cause failures. We recall the classic methods for quantitative and qualitative fault tree analysis presented by Lee et al. [110] as well as many newer techniques.

In Tables 1, 2, 3, and 4 (Pages 7, 9, 9, and 15 respectively), we have summarized the qualitative analysis techniques that we discuss in the current section.

Quantitative techniques are discussed in Section 2.3. These compute numerical values over the FT. Quantitative techniques can be further divided into *importance measures*, indicating how critical a certain component is, and *stochastic measures*, most notably failure probabilities. The stochastic measures are again divided into those handling single-time failure probabilities and continuous time ones; see Section 2.3.

2.2.1. Minimal cut sets

Cut sets and minimal cut sets provide important information about the vulnerabilities of a system. A *cut set* is a set of components that can together cause the system to fail. Thus, if an SFT contains cut sets with just a few elements, or elements whose failure is too likely, this could result in an unreliable system. Reducing the failure probabilities of these cut sets is usually a good way to improve overall reliability. Minimal cut sets are also used by some quantitative analysis techniques described in Section 2.3.

This section describes three important classes of cut set analysis: Classical methods which are based on manipulation of the boolean expression of the FT, methods based on Binary Decision Diagrams, and others. Table 1 summarizes these techniques.

Definition 4. *$C \subseteq BE$ is a cut set of FT F if $\pi_F(C) = 1$. A* minimal cut set *(MCS) is a cut set of which no subset is a cut set, i.e. formally $C \subseteq BE$ is an MCS if $\pi_F(C) = 1 \wedge \forall_{C' \subset C} : \pi_F(C') = 0$.*

Example 5. *In Figure 1, $\{U, B\}$ is an MCS. Another cut set is $\{U, M1, M2, M3\}$, but this is not an MCS since it contains the cut set $\{U, M1, M2\}$.*

Denoting the set of all MCS of an FT F as $MC(F)$, we can write an expression for the top event as $\bigvee_{C \in MC(F)} \bigwedge_{x \in C} x$. This property is useful for the analysis of the tree, as described below.

Boolean manipulation

The classical methods of determining minimal cut sets are the bottom-up and the top-down algorithms [177]. These represent each gate as a Boolean expression of BEs and/or other gates. These expressions are combined, expanded, and simplified into an expression that relates the top event to the BEs without any gates. This expression is called the *structure function.* At every step, the expressions are converted into disjunctive normal form (DNF), so that each conjunction is an MCS.

Example 6. *In Figure 1, the expression for the TE $G1$ is $U \wedge G2$, and that for $G2$ is $B \vee G3$. Substituting $G2$ into $G1$ gives $G1 = U \wedge (B \vee G3)$. Converting to DNF yields $G1 = (U \wedge B) \vee (U \wedge G3)$. Continuing in this fashion until all gates have been eliminated results in the minimal cut sets. This is the* top-down *method.*

The *bottom-up* method begins with the expressions for the gates at the bottom of the tree. This method usually produces larger intermediate results since fewer opportunities for simplification arise. As a result, it is often more computationally intense. However, it has the advantage of also providing the minimal cut sets for every gate.

Binary Decision Diagrams

An efficient way to find MCS is by converting the fault tree into a Binary Decision Diagram (BDD) [3]. A BDD is a directed acyclic graph that represents a boolean function $f : \{x_1, x_2, \dots x_n\} \to \{0, 1\}$. The leaves of a BDD are labeled with either 0 or 1. The other nodes are labeled with a variable x_i and have two children. The left child represents the function in case $x_i = 0$; the right child represents the function $x_i = 1$. BDDs are heavily used in model checking, to efficiently represent the state space and transition relation [55, 47].

To construct a BDD from a boolean formula, one can use the Shannon expansion formula [3] to construct the top node.

$$\begin{aligned} f(x_1, x_2, \cdots, x_n) = {} & (x_1 \wedge f(1, x_2, \cdots, x_n)) \\ & \vee (\neg x_1 \wedge f(0, x_2, \cdots, x_n)) \end{aligned}$$

We now let x_1 be the top node, and $f(0, x_2, \cdots, x_n)$ and $f(1, x_2, \cdots, x_n)$ the functions for its children. Recursively applying this expansion until all variables have been converted into BDD nodes yields a complete BDD.

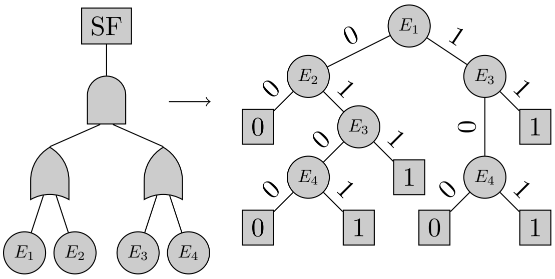


Figure 4: Example conversion of SFT to BDD

Example 7. *Figure 4 shows the conversion of an FT into a BDD. Each circle represents a BE, and has two children: a 0-child containing the sub-BDD that determines the system status if the BE has not failed, and a 1-child for if it has. The leaves of the BDD are squares containing 1 or 0 if the system has resp. has not failed. For example, if components E_1 and E_4 have failed, we begin traversing the BDD at its root, observe that E_1 has failed, and follow the 1-edge. From here, since E_3 is operational we follow the 0-edge. E_4 has failed, so here we follow the 1-edge to reach a leaf. This leaf contains a 1, so this combination results in a system failure.*

Author	Method	Remarks	Tool
Vesely et al. [177]	Top-down	Classic boolean method	MOCUS [83]
Vesely et al. [177]	Bottom-up	Produces MSC for gates	MICSUP [137]
Coudert and Madre [55]	BDD	Usually faster than classic methods	MetaPrime [56]
Rauzy [147]	BDD	Only for coherent FTs but faster than [55]	Aralia [146]
Dutuit and Rauzy [67]	Modular BDD	Faster for FTs with independent submodules	DIFTree [64]
Remenyte et al. [150, 151]	BDD	Comparison of BDD construction methods	-
Codetta-Raiteri [50]	BDD	Faster when FT has shared subtrees	-
Xiang et al. [187]	Minimal Cut Vote	Reduced complexity with large voting gates	CASSI [187]
Carrasco et al. [40]	CS-Monte Carlo	Less complex for FTs with few MCS	-
Vesely and Narum [178]	Monte Carlo	Low memory use, accuracy not guaranteed	PREP [178]

Table 1: Summary of methods to determine Minimal Cut Sets of SFTs

Cut Sets can be determined from the BDD by starting at all 1-leaves of the tree, and traversing upwards toward the root. The set of all BEs reached by traversing a 1-edge from a particular leaf forms one CS. The CS may not be minimal, depending on the algorithm used to construct the BDD. Rauzy and Dutuit [146] provide a method to construct BDDs encoding prime implicants, from which MCSs can be directly computed.

The BDD method was first coined by Coudert and Madre [55] as well as Rauzy [147]. Sinnamon et al. [164] improve this method by adding a minimization algorithm for the intermediate BDD. While the conversion to a BDD has exponential worst-case complexity, it has linear complexity in the best case. In practice, BDD methods are usually faster than boolean manipulation. This is strongly influenced by the fact that BDDs very compactly represent boolean functions with a high degree of symmetry [154], and fault trees exhibit this symmetry as the gates are symmetric in their inputs. A program that analyzes FTs using BDDs has been produced by Coudert and Madre [56].

The conversion of an FT to a BDD is not unique: Depending on the ordering of the BEs, different BDDs can be generated. Good variable ordering is important to reduce the size of the BDD. Unfortunately, even determining whether a given ordering of variables is optimal is an NP-complete problem [24]. Figure 5 shows how a different variable ordering affects the size of the resulting BDD.

Remenyte and Andrews [150, 151] have compared several different methods for constructing BDDs from FTs, and conclude that a hybrid of the if-then-else method [147] and the advanced component-connection method by Way and Hsia [185] is a good trade-off between processing time and size of the resulting BDD.

Improvements to BDD Tang and Dugan [172] propose the use of zero-suppressed BDDs to compute MCSs. This approach is more efficient than those based on classic BDDs in both time and memory use.

Dutuit and Rauzy [67] provide an algorithm for finding independent submodules of FTs, which can be converted separately to BDDs and analyzed, reducing the computational requirements for analyzing the entire tree.

If subtrees of an FT are shared, then the approach by Codetta-Raiteri [50] called 'Parametric Fault Trees' can be used. This method performs qualitative and quantitative analysis on such a tree without repeating the analysis for each repetition of a subtree.

Miao et al. [125] have developed an algorithm to determine minimal cut sets using a modified BDD, and claim its time complexity is linear in the number of BEs, although their paper does not seem to support this claim. Moreover, this result seems incorrect to us, since the number of MCSs is already exponential in the number of BEs.

Other methods For FTs with voting gates with many inputs, a combinatorial explosion can occur, since a k/N voting gate means each combination of k failed components results in a separate cut set. Xiang et al. [187] propose the concept of a Minimal Cut Vote as a term in an MCS to represent an arbitrary combination of k elements. This method is of linear complexity in the number of inputs to a voting gate, while the BDD approach has exponential complexity.

For relatively large trees with few cut sets, the algorithm by Carrasco and Suñé [40] may be useful. Its space complexity is based on the MCSs, rather than the complexity of the tree like for BDDs. However, according to the article this method does seem to be slower than the BDD approach.

In practice, it is often not necessary to determine all of the MCSs: Cut sets with many components are usually unlikely to have all these components fail. It is often sufficient to only find MCSs with a few components. This may allow a substantial reduction in computation time by reducing the size of intermediate expressions [110].

Due to the potentially very large intermediate expressions, the earlier methods for finding MCSs can have large memory requirements. A Monte Carlo method can be used as an alternative. In the method by Vesely and Narum [178], random subsets of components are taken to be failed, according to the failure probabilities. If a subset causes a top event failure, it is a cut set. Additional simulations reduce these cut sets into MCSs. While the memory requirements of the Monte Carlo method are much smaller, the large number of simulations can greatly increase computation time. In addition, there is a chance that not all

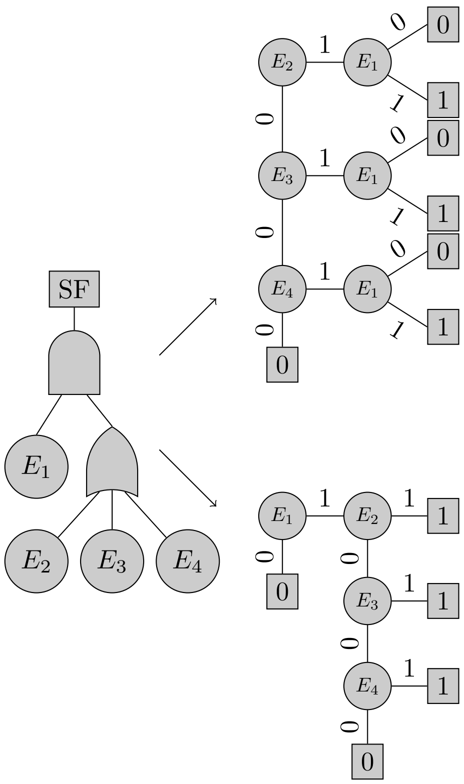


Figure 5: Example of how variable ordering affects BDD size. The upper BDD has 13 vertices, the lower BDD has 9. Other orderings are possible, but are not obvious.

MCSs are found.

2.2.2. Minimal path sets

A *minimal path set* (MPS) is essentially the opposite of an MCS: It is a minimal set of components such that, if they do not fail, the system remains operational.

Definition 8. *$P \subseteq BE$ is a path set of FT F if* $\pi(F, BE\backslash P) = 0$.

Example 9. *In Figure 1, an MPS is* $\{B, C1, M1, M2, PS\}$.

Similarly to MCSs, a fault tree has a finite number of MPSs. If we denote the set of all MPSs of a fault tree as

$$MP(F) = \left\{ P \subseteq BE \left| \begin{array}{c} \pi(F, BE\backslash P) = 0 \quad \wedge \\ \forall_{P' \subset P} : \pi(F, BE\backslash P') = 1 \end{array} \right. \right\}$$

then we can write a boolean expression for the TE as

$$TE = \bigwedge_{P \in MP(F)} \bigvee_{x \in P} x$$

Minimal Path Sets can, like MCSs, be used as a starting point for improving system reliability. Especially if the system has an MPS with few elements, improving such an MPS may improve the reliability of many MCSs.

Analysis Any algorithm to compute MCSs can also be used to compute MPSs. To do so, the FT is replaced by its dual: AND gates are replaced by OR gates, OR gates by AND gates, k/N voting gates by (N-k)/N voting gates, and BEs by their complement (i.e. 'component failure' by 'no component failure'). The MCSs of this dual tree are the MPSs of the original FT [15].

2.2.3. Common cause failures

Definition Another qualitative aspect is the analysis of probable common cause failures (CCF). These are separate failures that can occur due to a common cause that is not yet listed in the tree. For example, if a component can be replaced by a spare to avoid failure, both this component and its spare are in one cut set. If the spare is produced by the same manufacturer as the component, a shared manufacturing defect could cause both to fail at the same time. If such common causes are found to be too likely, they should be modeled explicitly to avoid overestimating the system reliability.

Analysis Although CCF analysis is not possible using automated methods from the FT alone, since CCF depend on external factors not modeled in the tree, experts may try to determine whether any cut sets have multiple components that are susceptible to a common cause failure. Such an analysis relies on expert insight, and is therefore quite informal.

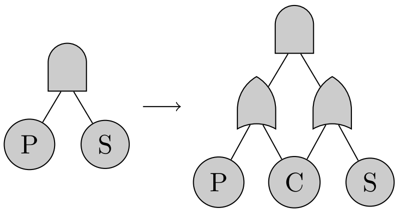


Figure 6: Example FT showing the addition of common cause C of events P and S.

Common causes can be added to an FT by inserting them as BEs and replacing the BEs they affect by OR-gates combining the CCF and the separate failure modes. An example is shown in Figure 6, where common cause C of event P and S is added.

2.3. Quantitative analysis of SFT: Single-time

Quantitative analysis methods derive relevant numerical values for fault trees. *Stochastic measures* are wide spread, as they provide useful information such as failure probabilities. *Importance measures* indicate how important a set of components is to the reliability of the system. Moreover, the *sensitivities* of these measures to variations in BE probabilities are important.

Model	Reliability	Availability	MTTFF	MTTF	MTBF	MTTR	ENF
Discrete-time	+						+
Continuous-time	+	+	+				+
Repairable cont.-time	+	+	+	+	+	+	+

Table 2: Applicability of stochastic measures to different FT types

Author	Measures	Remarks	Tool
Vesely et al. [177]	Reliability	Valid for infrequent failures	-
Barlow and Proschan [15]	Reliability	Exact calculation based on MCS	KTT [178]
Rauzy [147]	Reliability	Exact, Uses BDDs for efficiency	-
Stecher [169]	Reliability	Efficient for shared subtrees	-
Bobbio et al. [23]	Reliability	Allows dependent events	DBNet [130]
Durga Rao et al. [65]	Reliability	Monte Carlo, allows arbitrary distributions	DRSIM [65]
Aliee and Zarandi [5]	Reliability	Fast Monte Carlo, requires special hardware	-
Barlow and Proschan [15]	Availability	Translation to reliability problem	-
Durga Rao et al. [65]	Availability	Monte Carlo, allows arbitrary distributions	DRSIM [65]
Amari and Akers [7]	MTTF	Assumes exponential failure distributions	-
Schneeweiss [161]	MTBF	Exact method based on boolean expression	SyRePa [160]
Amari and Akers [7]	MTBF	Assumes exponential failure distributions	-

Table 3: Summary of qualitative analysis methods for SFTs

Moreover, stochastic measures can be used to decide whether it is safe to continue operating a system with certain component failures, or whether the entire system should be shut down for repairs.

The next section first describes some basic probability theory, and then provides definitions and analysis techniques for several measures applicable to single-time FTs.

2.3.1. Preliminaries on probability theory

A discrete random variable is a function $X : \Omega \to \mathbb{S}$ that assigns an outcome $s \in \mathbb{S}$ to each stochastic experiment. The function $\mathbb{P}[X = s]$ denotes the probability that X gets value s and is called the *probability density function.* We consider Boolean random variables, i.e. $s \in \{0, 1\}$ where $s = 1$ denotes a failure, and $s = 0$ a working FT element. If $X_1, X_2, \ldots X_n$ are random variables, and $f : \mathbb{S}^n \to \mathbb{S}$ is a function, then $f(X_1, X_2, \ldots X_n)$ is a random variable as well.

2.3.2. Modeling failure probabilities

The single-time approach does not consider the evolution of a system over time: a fixed time horizon is considered, during which each component can fail only once. We assume that the failures of the BEs are stochastically independent. If the FT has shared subtrees, then the failures of the gates are not independent.

The BE are equipped with a *failure probability function* $P : BE \to [0, 1]$ that assigns a failure probability $P(e)$ to each $e \in BE$, see Figure 7. Then, each BE e can be associated with random variable $X_e \sim \text{Alt}(P(e))$; that is $\mathbb{P}(X_e = 1) = P(e)$ and $\mathbb{P}(X_e = 0) = 1 - P(e)$. Given a fault tree F with BEs $\{e_1, e_2, \ldots e_n\}$, the semantics from Definition 3 yields a stochastic semantics for each gate $g \in G$, namely as the random variable $\pi_F(X_{e_1}, \ldots, X_{e_n}, g)$. We abbreviate the random variable for the top event of FT F as X_F.

Note that under these stochastic semantics, it holds for all $g \in G$ that

- $X_g = \min_{i \in I(g)} X_i$, if $T(g) = And$,
- $X_g = \max_{i \in I(g)} X_i$, if $T(g) = Or$,
- $X_g = \left(\sum\limits_{i \in I(g)} X_i \right) \geq k$, if $T(g) = VOT(k/N)$.

2.3.3. Reliability

The *reliability* of a single-time FT is the probability that the failure does not occur during the (modeled) life of the system [15].

Definition 10. *The reliability of a single-time FT F is defined as $Re(F) = \mathbb{P}(X_F = 0)$.*

The reliability of a fault tree F with BEs $e_1, \ldots e_n$ can be derived from the non-stochastic semantics by using the stochastic independence of the BE failures:

$$\begin{aligned}
&\mathbb{P}(X_F = 1) \\
&= \sum_{b_1,\dots,b_n \in \{0,1\}} \begin{array}{l} \mathbb{P}(X_F = 1 | X_{e_1} = b_1 \wedge \ldots \wedge X_{e_n} = b_n) \\ \cdot\, \mathbb{P}(X_{e_1} = b_1 \wedge X_{e_n} = b_n) \end{array} \\
&= \sum_{b_1,\dots,b_n \in \{0,1\}} \pi_F(b_1, \ldots, b_n) P_{b_1}(e_1) \cdot \ldots \cdot P_{b_n}(e_n) \qquad (*)
\end{aligned}$$

Here, $P_1(e) = P(e)$ and $P_0(e) = 1 - P(e)$. Computing (*) directly is complex. Below, we discuss several methods to speed up the reliability analysis.

Bottom up analysis For systems without shared BEs, failure probabilities can be easily propagated from the bottom up, by using standard probability laws. If the input distributions $X_1, X_2, \ldots X_n$ of a gate G are all stochastically independent (i.e., there are no shared subtrees), then we have

$$\begin{aligned}
&\mathbb{P}[X_{AND}(X_1, \ldots X_n) = 1] \\
&= \mathbb{P}[X_1 = 1 \wedge \ldots \wedge X_n = 1] \\
&= \mathbb{P}[X_1 = 1] \cdot \ldots \cdot \mathbb{P}[X_n = 1]
\end{aligned}$$

For the *OR*, we use

$$\begin{aligned}
&\mathbb{P}[X_{OR}(X_1, \ldots X_n) = 1] \\
&= 1 - \mathbb{P}[X_{OR}(X_1, \ldots X_n) = 0] \\
&= 1 - \mathbb{P}[X_1 = 0 \wedge \ldots \wedge X_n = 0] \\
&= 1 - (1 - \mathbb{P}[X_1 = 1]) \cdot \ldots \cdot (1 - \mathbb{P}[X_n = 1])
\end{aligned}$$

The *VOT(k/N)* gate is slightly more involved. It is possible to rewrite the gate into a disjunctions of all possible sets of k inputs, obtaining

$$\begin{aligned}
&\mathbb{P}[X_{VOT(k/N)}(X_1, \ldots X_n) = 1] \\
&= \mathbb{P}[(X_1 = 1 \wedge \ldots \wedge X_k = 1) \\
&\quad \vee (X_1 = 1 \wedge \ldots \wedge X_{k-1} = 1 \wedge X_{k+1} = 1) \\
&\quad \ldots \\
&\quad \vee (X_{n-k} = 1 \wedge \ldots \wedge X_n = 1)]
\end{aligned}$$

however, expanding this into an expression of simple probabilities requires the use of the inclusion-exclusion principle and results in very large expressions for gates with many inputs where k is neither very small nor close to N. It is more convenient to recursively define the voting gate:

$$\begin{aligned}
&\mathbb{P}[X_{VOT(0/N)}(X_1, \ldots X_n) = 1] = 1 \\
&\mathbb{P}[X_{VOT(N/N)}(X_1, \ldots X_n) = 1] = \mathbb{P}[X_{AND}(X_1, \ldots X_n) = 1] \\
&\mathbb{P}[X_{VOT(k/N)}(X_1, \ldots X_n) = 1] \\
&\quad = \mathbb{P}\left[(X_1 = 1 \wedge X_{VOT(k\text{-}1/N\text{-}1)}(X_2, \ldots X_n) = 1)\right. \\
&\qquad \left. \vee (X_1 = 0 \wedge X_{VOT(k/N\text{-}1)}(X_2, \ldots X_n) = 1)\right] \\
&\quad = \mathbb{P}[X_1 = 1] \cdot \mathbb{P}[X_{VOT(k\text{-}1/N\text{-}1)}(X_2, \ldots X_n) = 1] \\
&\qquad + \mathbb{P}[X_1 = 0] \cdot \mathbb{P}[X_{VOT(k/N\text{-}1)}(X_2, \ldots X_n) = 1)]
\end{aligned}$$

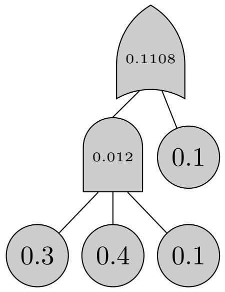


Figure 7: Example FT showing the propagation of failure probability in a single-time FT.

Example 11. *Figure 7 shows an example of how such probabilities propagate. Failure of the AND-gate requires all inputs to fail, which has a probability of* $0.3 \cdot 0.4 \cdot 0.1 = 0.012$*. The OR-gate fails if any input fails, i.e. remains operational only if all inputs do not fail. This has probability* $1 - (1 - 0.012)(1 - 0.1) = 0.1108$*.*

This approach does not work when BEs are shared, since the dependence between subtrees is not taken into account. To take an extreme example, consider an AND-gate with two children that are actually the same event with failure probability 0.1. Clearly, the unreliability of this gate is also 0.1, but propagating the probabilities as independent would give an incorrect unreliability of 0.01.

Binary Decision Diagrams As discussed in Section 2.2.1, BDDs can be used to encode FTs very efficiently. In addition to the qualitative analysis already discussed, Efficient quantative analysis is also possible.

To construct a BDD for computing system reliability, one can use a method similar to Shannon decomposition [147]:

$$\begin{aligned}
\mathbb{P}(f(x_1, x_2, \cdots, x_n)) = {} & \mathbb{P}(x_1)\mathbb{P}(f(1, x_2, \cdots, x_n)) \\
& + \mathbb{P}(\neg x_1)\mathbb{P}(f(0, x_2, \cdots, x_n))
\end{aligned}$$

A caching mechanism is used to store intermediate results [145], as intermediate formulas often occur is more than one subdiagram. This algorithm can be applied even to non-coherent FTs, and has a complexity that is linear in the size of the BDD.

Rare event approximation For systems with shared events, the total unavailability of the system can also be approximated by summing the unavailabilities of all the MCSs. This *rare event approximation* [168] is reasonably accurate when failures are improbable. However, as failures become more common and the probability of multiple cut sets failure increases, the approximation deviates more from the true value. For example, a system with 10 independent MCSs, each with a probability 0.1, has an unreliability of 0.65, whereas the rare event approximation suggests an unreliability of 1.

Example 12. *Considering Figure 1 and assuming all basic events have an unavailability of 0.1, the probability of a failure of gate G6 can be approximated as* $P_{fail}(G6) \approx P_{fail}(\{M1, M2\}) + P_{fail}(\{M2, M3\}) + P_{fail}(\{M1, M3\}) = 0.03$. *As the actual probability is* 0.028*, the approximation has slightly overestimated the failure probability.*

If some cut sets have a relatively high probability, this rare event approximation is no longer accurate. If no component occurs in more than one cut set, the correct probability may be calculated as $P_{fail}(F) = 1 - \prod_{C \in MC(F)}(1 - P_{fail}(C))$.

If some components are present in many of the cut sets, more advanced analysis are needed. An exact solution may be obtained by using the inclusion-exclusion principle to avoid double-counting events. Alternative methods may be more efficient in special cases, such as the algorithm by Stecher [169] which reduces repeated work if the FT contains shared subtrees.

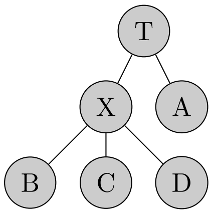
An algorithm using zero-suppressed BDDs [145] closely resembles the calculation of MCSs, but instead computes system reliability using the rare event approximation. This method has a complexity linear in the size of the BDD, and is more efficient than first computing the MCSs and then the reliability.

Bayesian Network analysis In order to accurately calculate the reliability of a fault tree in the presence of statistical dependencies between events, Bobbio et al. [23] present a conversion of SFT to Bayesian Networks. A *Bayesian Network* [19] is a sequence $X_1, X_2, \ldots, X_n$ of stochastically dependent random variables, where X_i can only depend on X_j if $j < i$. Indeed, the failure distribution of a gate in a FT only depends on the failure distributions of its children. Bayesian networks can be analyzed via conditional probability tables $\mathbb{P}[B|A_j]$ by using the law of total probability: for an event B, and a partition A_j of the event space, we have

$$\mathbb{P}[B] = \sum_j \mathbb{P}[B|A_j]\mathbb{P}[A_j]$$

For example, if X_4 depends on X_3 and X_2, then partitioning yields $\mathbb{P}[X_4 = 1] = \sum_{i,j \in \{0,1\}} \mathbb{P}[X_4 = 1|X_3 = i \wedge X_2 = j]\mathbb{P}[X_3 = i \wedge X_2 = j]$. The values $\mathbb{P}[X_4 = 1|X_3 = i \wedge X_2 = j]$ are given by conditional probability tables, and $\mathbb{P}[X_3 = i \wedge X_2 = j]$ are computed recursively.

Example 13. *Figure 8 shows the conversion of a simple FT into a Bayesian Network. The BEs A, B, and C are connected to top event T and assigned reliabilities. Gates have conditional probabilities dependent on the states of their inputs. All nodes can have only states 0 or 1 corresponding to operational and failed, respectively. Classic inference techniques [19] can be used to compute* $P(T = 1)$*, which corresponds to system unreliability. Alternatively, if it is known that the system has failed, the inference can provide probabilities of each of the BEs having failed.*



$$\mathbb{P}(T = 1|A = 1 \vee X = 1) = 1$$
$$\mathbb{P}(A = 1) = 0.1$$
$$\mathbb{P}(X = 1|B = C = D = 1) = 1$$
$$\mathbb{P}(B = 1) = 0.3$$
$$\mathbb{P}(C = 1) = 0.4$$
$$\mathbb{P}(D = 1) = 0.1$$

Figure 8: The BN obtained by converting the FT in Figure 7 to a Bayesian Network

In addition, Bobbio et al. [23] allow BEs with multiple states: Rather than being either up or failed, components can be in different failure modes, such as degraded operational modes, or a valve that is either stuck open or stuck closed. The Bayesian inference rules work the same for multiple-state fault trees, but lead to larger conditional probability tables. Also, Bobbio et al. [23] model common cause failures by adding a probability of a gate failing even when not enough of its inputs have failed, although this has the disadvantage of making the potential failure causes less explicit. Finally, gates can be 'noisy', meaning they have a chance of failure. For example, the failure of one element of a set of redundant components may have a small change of causing a system failure.

An important feature of Bayesian Network Analysis is that, not only can it compute the probability of the top event given the leaves, it can compute the probabilities of each of the leaves given the top event. This is very useful in fault diagnosis [109, 108], where one knows that a failure has occurred, and wants to find which leaves are the most like causes. Additional evidence can also be given, such as certain leaves that are known not to have failed.

Monte Carlo simulation Monte Carlo methods can also be used to compute the system reliability. Most techniques are designed for continuous-time models [57, 65] or qualitative analysis [178], but adaptation to single-time models is straightforward. Each component is randomly assigned a failure state based on its failure probability. The FT is then evaluated to determine whether the TE has failed. Given enough simulations, the fraction of simulations that does not result in failure is approximately the reliability.

2.3.4. Expected Number of Failures

Definition The *Expected Number of Failures* (ENF) describes the expected number of occurrences of the TE within a specified time limit. This measure is commonly used to evaluate systems where failures are particularly costly or dangerous, and where the system will operate for a known period of time.

A major advantage of the ENF is that the combined ENF of multiple independent systems over the same timespan can very easily be calculated, namely $ENF(S1, S2) = ENF(S1) + ENF(S2)$. For example, if a power company requests a number of 40-year licenses to operate nuclear power stations, it is easy to check that the combined ENF is sufficiently low.

Analysis Since a single-time system can fail at most once, it is easy to show that the ENF of such a system is equal to its unreliability. Let N_F denote the number of failures system F experiences during its mission time, so that

$$\begin{aligned}\mathbb{E}[N_F] &= \sum_i i \cdot \mathbb{P}[N_F = i] \\ &= 0 \cdot \mathbb{P}[N_F = 0] + 1 \cdot \mathbb{P}[N_F = 1] \\ &= 0 + \mathbb{P}[X_F = 1] \\ &= \mathit{Re}(F)\end{aligned}$$

2.4. Quantitative analysis of SFT: continuous-time

Where single-time systems treat the entire lifespan of a system as a single event, it is often more useful to consider dependability measures at different times. Provided adequate information is available, continuous-time fault trees provide techniques to obtain these measures. This section provides, after a description of the basic theory, definitions and analysis techniques for these measures.

2.4.1. Modeling failure probabilities

Continuous-time FTs consider the evolution of the system failures over time. The component failure behaviour is usually given by a probability function $D_e : \mathbb{R}^+ \mapsto [0,1]$, which yields for each BE e and time point t, the probability that e has not failed at time t. In practise, the failure distributions can often be adequately approximated by inverse exponential distributions, and BEs are specified with a failure rate $R : \mathit{BE} \mapsto \mathbb{R}^+$, such that $R(e) = \lambda \leftrightarrow D_e(t) = 1 - \exp(-\lambda t)$.

If components can be repaired without affecting the operations of other components, BEs have an additional repair distribution over time. Like failure distributions, repair distributions are often exponentially distributed and specified using a repair rate $\mathit{RR} : \mathit{BE} \mapsto \mathbb{R}^+$. More generally, BEs can be assigned repair distributions as $\mathit{RD}_e : \mathbb{R}^+ \mapsto [0,1]$. More complex and realistic models of repairs are discussed in section 4.3, this section does not consider such models.

Like for the single-time case, we can use random variables X_e to describe failures of basic events, and derive a stochastic semantics for the FT. However, due to the possibility of repair, it is helpful to introduce some additional variables. Consider a BE e with a failure distribution D_e and repair distribution RD_e. Now we take $F_{e,1}, F_{e,2}, \ldots$ as the relative failure times, and $Q_{e,1}, Q_{e,2}, \ldots$ as the relative repair times, with $Q_{e,1} = 0$ for convenience. It follows that $\mathbb{P}[F_{e,i} \leq t] = D_e(t)$ and $\mathbb{P}[Q_{e,i} \leq t] = \mathit{RD}_e(t)$ for $i > 1$. We can now define the random variables X_e and X_g.

For basic events, $X_e(t)$ is 1 if t is some time after a failure, and before the subsequent repair. We can rewrite this as follows:

$$\begin{aligned}&X_e(t) = 1 \text{ iff} \\ &\exists_i \left[\sum_{j<i}(Q_{e,j} + F_{e,j}) \leq t \wedge Q_{e,i} + \sum_{j<i}(Q_{e,j} + F_{e,j}) > t \right] \\ \Leftrightarrow\ &\exists_i \left[\sum_{j<i}(Q_{e,j} + F_{e,j}) \leq t \wedge t - Q_{e,i} < \sum_{j<i}(Q_{e,j} + F_{e,j}) \right] \\ \Leftrightarrow\ &\exists_i \left[t - Q_{e,i} \leq \sum_{j<i}(Q_{e,j} + F_{e,j}) \leq t \right]\end{aligned}$$

For gates, $X_g(t)$ is defined analogously to the single-time case. To summarize, we have the following definition:

Definition 14.

$$X_e(t) = \begin{cases} 1 & \mathit{if}\ \exists_i : t - Q_{e,i} < \sum\limits_{j<i}(Q_{e,j} + F_{e,j}) \leq t \\ 0 & \mathit{otherwise} \end{cases}$$

$$X_g(t) = \begin{cases} \min_{i \in I(g)} X_i(t) & \mathit{if}\ T(g) = \mathit{And} \\ \max_{i \in I(g)} X_i(t) & \mathit{if}\ T(g) = \mathit{Or} \\ \left(\sum\limits_{i \in I(g)} X_i(t) \right) \geq k & \mathit{if}\ T(g) = \mathit{Vote}(k/N) \end{cases}$$

Depending on the failure distributions, the random variables of the BEs can have relatively easy distributions. For example, a BE with exponentially distributed failures with rate λ has probability $\mathbb{P}(X_e(t) = 0) = 1 - \exp(-\lambda t)$. The distributions of the gates typically do not follow convenient distributions.

Given the definition of X_i, classic statistical methods may be used to analyze the FT. For example, the *availability* of an FT F is described as $A(F) = \lim_{t \to \infty} \mathbb{E}(X_F(t))$, as explained in section 2.4.3.

This method of analysis can be applied to FTs with arbitrary failure distributions, even if the BEs are statistically dependent on each other. Unfortunately, the algebraic expressions for the probability distributions often become too large and complex to calculate, so other techniques have to be used for larger FTs.

2.4.2. Reliability

Definition The *reliability* of a continuous-time FT F is the probability that the system it represents operates for a certain amount of time without failing. Formally, we define a random variable $Y_F = \max\{t | \forall_{s<t} X_F(s) = 0\}$ to denote the time of the first failure of the tree. The reliability of the system up to time t is then defined as $\mathit{Re}_F(t) = \mathbb{P}(Y_F > t)$.

Analysis In continuous-time systems, the reliability in a certain time period can be calculated by conversion into a single-time system, taking BE probabilities as the probability of failure within the specified timeframe.

Monte Carlo methods can also be used to compute system reliability. In the method by Durga Rao et al. [65], random failure times and, if applicable, repair times are generated according to the BE distributions. The system is simulated with these failures, and the system reliability and availability recorded. Given enough simulations, reasonable approximations can be obtained. Modifying the method to record other failure measures is trivial.

For higher performance than conventional computer simulation, Aliee and Zarandi [5] have developed a method for programming a model of an FT into a special hardware chip called a Field Programmable Gate Array, which can perform each MC simulation very quickly.

2.4.3. Availability

Definition The *availability* of a system is the probability that the system is functioning at a given time. Availability can also be calculated over an interval, where it denotes the fraction of that interval in which the system is operational [15]. Availability is particularly relevant for repairable systems, as it includes the fact that the system can become functional again after failure. For non-repairable systems, the availability in a given duration may still be useful. The long-run availability always tends to 0 for nontrivial non-repairable systems, as eventually some cut set will fail and remain nonfunctional.

Definition 15. *The availability of FT F at time t is defined as $A_F(t) = \mathbb{E}(X_F(t))$. The availability over the interval $[a, b]$ is defined as $A_F([a, b]) = \frac{1}{b-a}\int_a^b X_F(t)dt$. The long-run availability is $A_F = \lim_{t\to\infty} A_F([0, t])$ or equivalently, $A_F = \lim_{t\to\infty} A_F(t)$ when this limit exists.*

Analysis As the availability at a specific time is a simple probability, it is possible to treat the FT as a single-time FT, by replacing the BE failure distribution with the probability of being in a failed state at the desired time. The single-time reliability of the resulting FT is then the availability of the original. Failure probabilities of the BE are usually easy to calculate, also for repairable FTs [15].

Long-term availability of a system can be calculated the same way, provided the limiting availability of each BE exists. This is the case for most systems.

Availability over an interval cannot be calculated so easily. Since this availability is defined as an integral over an arbitrary expression, no closed-form expression exists in the general case. Numerical integration techniques can be used should this availability be needed.

2.4.4. Mean Time To Failure

Definition The Mean Time To Failure (MTTF) describes the expected time from the moment the system becomes operational, to the moment the system subsequently fails.

Formally, we introduce an additional random variable $Z_F(t)$ denoting the number of times the system has failed up to time t.

Definition 16. *To define $Z_F(t)$, we first define the failure and repair times of the gate:*

$$\begin{aligned} Q_{g,1} &= 0 \\ F_{g,i} &= \min\{t > Q_{g,i} | X_g(t) = 1\} - Q_{g,i} \\ Q_{g,i} &= \min\{t > F_{g,i-1} | X_g(t) = 0\} - F_{g,i-1} \end{aligned}$$

We then define $Z_g(t)$ of a gate as:

$$Z_g(t) = \max\left\{ i \in \mathbb{N} \middle| \sum_{j\leq i} (Q_{g,j} + F_{g,j}) \leq t \right\}$$

Now $Z_F(t) = Z_T(t)$ with T being the TE of FT F.

The MTTF up to time t is then $MTTF_F(t) = \frac{A_F(t)\cdot t}{Z_F(t)}$. The long-run MTTF is $MTTF_F = \lim_{t\to\infty} MTTF_F(t)$.

In repairable systems the time to failure depends on the system state when it becomes operational. The first time, all components are operational, but when the system becomes operational due to a repair, some components may still be non-functioning. This difference is made explicit by distinguishing between *Mean Time To First Failure* (MTTFF) and MTTF.

To illustrate this difference, consider the FT in Figure 9. Here, failures will initially be caused primarily by component 3, resulting in an MTTFF slightly less than $\frac{1}{10}$. In the long run, however, component 1 will mostly be in a failed state, and component 2 will cause most failures. This results in a long-run MTTF of approximately 1.

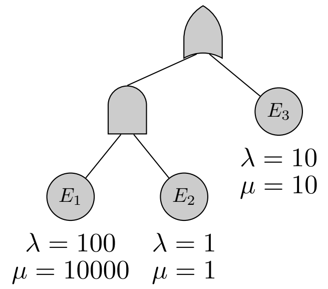


Figure 9: Example FT of a repairable system where MTTF and MTTFF differ significantly. Failure rates are denoted by λ, repair rates by μ.

While MTTF and availability are often correlated in practise, only the MTTF can distinguish between frequent, short failures and rare, long failures.

Analysis Many failure distributions have expressions to immediately calculate the MTTF of components. For example, a component with exponential failure distribution with rate λ has MTTF $\frac{1}{\lambda}$. For gates, however, the combination of multiple BE often does not have a failure distribution of a standard type, and algebraic calculations produce very large equations as the FTs become more complex.

Amari and Akers [7] have shown that the Vesely failure rate [176] can be used to approximate the MTTF, and can do so efficiently even for larger trees.

2.4.5. Mean Time Between Failures

Definition For repairable systems, the *Mean Time Between Failures* (MTBF) denotes the mean time between two successive failures. It consists of the MTTF and the *Mean Time To Repair* (MTTR). In general, it holds that MTBF = MTTR + MTTF.

The MTBF is defined similarly to the MTTF except ignoring the unavailable times. Formally, $MTBF_F(t) = \frac{t}{Z_F(t)}$, and in the long run $MTBF_F = \lim_{t\to\infty} MTBF_F(T)$.

The MTBF is useful in systems where failures are particularly costly or dangerous, unlike availability which focuses more on total downtime. For example, if a railroad switch failure causes a train to derail, the fact that an accident occurs is much more important than the duration of the subsequent downtime.

The MTTR is often less useful, but may be of interest if the system is used in some time-critical process. For example, even frequent failures of a power supply may not be very important if a battery backup can take over long enough for the repair, while infrequent failures that outlast the battery backup are more important.

Analysis An exact value for the MTBF may be obtained using the polynomial form of the FT's boolean expression, as described by Schneeweiss [161]. The Vesely failure rate approximation by Amari and Akers [7] can also be used.

2.4.6. Expected Number of Failures

Definition Like in a single-time FT, the ENF denotes the expected number of times the top event occurs within a given timespan. For repairable systems, it is possible for more than one failure to be expected.

Analysis The ENF of a non-repairable system is equal to its unreliability. The ENF of a repairable system can be calculated from the MTBF using the equation $ENF(t) = \frac{t}{MTBF(t)}$, or using simulation.

2.5. Sensitivity analysis

Quantitative techniques produce values for a given FT, but it is often useful to know how sensitive these values are to the input data. For example, if small changes in BE probabilities result in a large variation in system reliability, the calculated reliability may not be useful if the probabilities are based on rough estimates. On the other hand, if the reliability is very sensitive to one particular component's failure rate, this component may be a good candidate for improvement.

If the quantitative analysis method used gives an algebraic expression for the failure probability, it may be possible to analyze this expression to determine the sensitivity to a particular variable. One method of doing so is provided by Rushdi [157].

In many cases, however, sensitivity analysis is performed by running multiple analysis with slightly different values for the variables of interest.

If the uncertainty of the BE probabilities is bounded, an extension to FT called a *Fuzzy Fault Tree* can be used to analyze system sensitivity. This method is explained in Section 4.1.

2.6. Importance measures

In addition to computing reliability measures of a system, it is often useful to determine which parts of a system are the biggest contributors to the measure. These parts are often good candidates for improving system reliability.

In FTs, it is natural to compute the relative importances of the cut sets, and of the individual components. Several measures are described below, and the applicability of these measures is summarized in Table 4.

MCS size An ordering of minimal cut sets can be made based on the number of components in the set. This ordering approximately corresponds to ordering by probability, since a cut set with many components is generally less likely to have all of its elements fail than one with fewer components. Small Cut sets are therefore good starting points for improving system reliability.

Stochastic measures For a more exact ordering, the stochastic measures described above can also be calculated for each cut set, and used to order them.

For systems specified using exponential failure distributions, the probability $W(C,t)\Delta t$ of cut set C causing a system failure between time t and Δt is approximately the probability that all but one BE of C have failed at time t and that the final component fails within the interval Δt. If we write the failure rate of a component x as λ_x, and we write $Re_x(t)$ for the reliability of x up to time t, the probability of cut set C causing a failure in a small interval can be approximated as

$$W(C,t)\Delta t \approx \sum_{x\in C} \left(\lambda_x \Delta t \prod_{y\in(C\setminus\{x\})} Re_y(t) \right)$$

Cancelling the Δt on both sides gives

$$W(C,t) \approx \sum_{x\in C} \left(\lambda_x \prod_{y\in(C\setminus\{x\})} Re_y(t) \right)$$

This approximation is only valid if the other cut sets have low failure probabilities, but can then be used to order cut sets by the rate with which they cause system failures. The full derivation of this approximation is provided by Vesely et al. [177].

Author	Measure	Remarks
Various	Cut set size	Very rough approximation
Various	Cut set failure measure	Specific to each failure measure
Vesely et al. [176]	Cut set failure rate	Applicable to exponential distributions
Birnbaum [21]	Structural importance	Based only on FT structure
Jackson [98]	Structural importance	Also for noncoherent systems
Andrews at al. [8]	Structural importance	Also includes repairs
Contini et al. [54]	Init. & Enab. importance	For FTs with initiating and enabling events
Hong and Lie [92]	Joint Reliability Importance	Interaction between pairs of events
Armstrong [9]	Joint Reliability Importance	Also for dependent events
Lu [118]	Joint Reliability Importance	Also for noncoherent systems
Vesely-Fussell [82]	Primary Event Importance	BE contribution to unavailability
Dutuit et al. [66]	Risk Reduction Factor	Maximal improvement of reliability by BE

Table 4: Summary of importance measures for cut sets and components

Structural importance Other than ranking by failure probability, several other measures of component importance have been proposed. Birnbaum [21] defines a system state as the combination of all the states (failed or not) of the components. A component is now defined as critical to a state if changing the component state also changes the TE state. The fraction of states in which a component is critical is now the Birnbaum importance of that component.

Formally, an FT with n components has 2^n possible states, corresponding to different sets χ of failed components. A component e is considered critical in a state χ of FT F if $\pi(F, \chi \cup \{c\}) \neq \pi(F, \chi \backslash \{c\})$.

Jackson [98] extended this notion to noncoherent systems, in a way that does not lead to negative importances when component failure leads to system repair. An additional refinement was made by Andrews and Beeson [8], to also consider the criticality of a component being repaired.

The *Vesely-Fussell importance factor* $VF_F(e)$ is defined as the fraction of system unavailability in which component e has failed [82]. Formally, $VF_F(e) = P(e \in S | \pi_F(S) = 1)$. An algorithm to compute this measure is given by Dutuit and Rauzy [66].

The *Risk Reduction Worth* $RRF_F(e)$ is the highest increase in system reliability that can be achieved by increasing the reliability of component e. It may be calculated using the algorithm by Dutuit and Rauzy [66].

Initiating and enabling importance In systems where some components have a failure rate and others have a failure probability, Contini and Matuzas [54] introduce a new importance measure that separately measures the importance of *initiating events* that actively cause for the TE, and *enabling events* that can only fail to prevent the TE.

To illustrate this distinction, consider an oil platform. If the event of interest is an oil spill, the event 'burst pipe' would be an initiating event, since this event leads to an oil spill unless something else prevents it. The event 'emergency valve stuck open' is an enabling event. It does not by itself cause an oil spill, it only fails to prevent the burst pipe causing one. The distinction is not usually explicit in the FT, since both these events would simply be connected by an AND gate.

Initiating events often occur only briefly, and either cause the TE or are quickly 'repaired'. Repair in this case can also include the shutdown of the system, since that would also prevent the catastrophic TE. In contrast, enabling events may remain in a failed state for along time.

Due to this difference, overall reliability of such a system can be improved by reducing the failure frequency of initiating events, or by reducing the frequency or increasing the repair rate of enabling events. This is one reason for the distinction between the two in the analysis.

Joint importance To quantify the interactions between components, Hong and Lie [92] developed the *Joint Reliability Importance* and its dual, the *Joint Failure Importance.* These measures place greater weight on pairs of components that occur together in many cut sets, such as a component and its only spare, than on two relatively independent components. This may be useful to identify components for which common cause failures are particularly important.

Armstrong [9] extends this notion of the Joint Reliability Importance to include statistical dependence between the component failures, and proves that the JRI is always nonzero for certain classes of systems. Later, Lu [118] determines that the JFI can also be used for noncoherent systems.

2.7. Commercial tools

In addition to the academic methods described in this section, commercial tools exist for FTA. The algorithms used in these tools are usually well documented. Several of these programs also allow the analysis of dynamic FTs, which will be explained in Section 3.

This subsection describes several commonly used commercial FTA tools. This list is not exhaustive, nor intended as a comparison between the tools, but rather to give an overview of the capabilities and limitations of such tools in general.

A.L.D. RAM Commander A.L.D. produces an FTA program as part of its RAM Commander toolkit [60]. This program can automatically generate FTs from FMECAs, FMEAs, or RBDs, and allows the user to generate a new FTA. It supports continuous and single-time FT, and can combine different failure distributions in one FT. Repairs are also supported.

The only supported qualitative analysis is the generation of minimal cut sets.

For qualitative analysis, the tool can compute reliability and expected number of failures up to a specified time bound, and availability at specific times as well as long-run mean availability. Failure frequency up to a given time is also supported. Moreover, the program can compute the importances and sensitivities of the BEs.

EPRI CAFTA CAFTA (Computer Aided Fault Tree Analysis) [70] is a tool developed by EPRI for FTA. It supports single- and continuous-time FTs, including non-coherent FTs and the PAND gate from dynamic FTs. Continuous-time BEs can have various probability distributions, including normal and uniform distributions. Several models of CCF are also included.

CAFTA can compute cut sets. For quantitative analysis, the program can compute reliability, and several importance and sensitivity measures.

Isograph FaultTree+ The Isograph FaultTree+ program [97] is one of the most popular FTA tools on the market. It performs quantitative and qualitative fault tree analysis. It can analyze FTs with various failure distributions, and can replace BEs by Markov Chains to allow the user to arbitrarily closely approximate any distribution [99]. Dynamic FTs and Non-coherent FTs including NOT gates can also be analyzed.

Qualitatively, the program supports minimal cut set determination and the analysis of common cause failures. A static analysis is also supported for errors such as circular dependencies.

All the quantitative measures described in Section 2.4 can be calculated by FaultTree+. The program can also determine confidence intervals if uncertainties in the BE data are known. Without such information, sensitivity analysis can still be performed by automatic variation of the failure and repair rates. Importance measures that can be computed over the BE are the Fussell-Vesely, Birnbaum, Balow-Proschan, and Sequential importances.

ITEM ToolKit The ITEM ToolKit by ITEM software [166] supports FTA, as well as other reliability and safety analyses, such as Reliability Block Diagrams [61].

This program uses Binary Decision Diagrams for its analysis, but can also perform an approximation method. The analysis supports non-coherent FTs, and several different failure models for BEs.

Qualitative analysis can determine minimal cut sets, and has four methods for common cause failure analysis.

Quantitative analysis supports reliability and availability computation. Uncertainty analysis of the results can be performed if input uncertainties are known, and sensitivity analysis even if they are not. The program can also compute importance measures, although for which measures is not specified.

OpenFTA The open-source tool OpenFTA [133] can perform basic FTA. It only supports non-repairable FTs, and allows only single-time BEs and BEs with exponentially distributed failure times.

OpenFTA supports minimal cut set generation, deterministic analysis of system reliability, and Monte Carlo simulation to determine reliability.

ReliaSoft BlockSim ReliaSoft's BlockSim program [149] can analyze Reliability Block Diagrams [61] and FTs.

Quantitative analysis can determine exact reliability of the system, including the changes in reliability over time. If information about possible reliability improvements is available, the program can compute the most cost-effective improvement strategy to obtain a given reliability.

Availability of repairable systems can be approximated using discrete event simulation. Given information about repair costs and spare part availability, the analysis can determine the most effective maintenance strategy for a cost or availability requirement, as well as the optimal spare parts inventory.

BlockSim supports the determination of minimal cut sets, but does not appear to offer other quantitative analysis options.

PTC Windchill FTA The Windchill FTA program by PTC [141] allows the design and analysis of fault trees and event trees, including dynamic FTs. The program supports non-coherent FTs, as well as different failure distributions for the BEs.

Windchill FTA can compute minimal cut sets, as well as several methods for determining common cause failures.

Qualitative measures than can be computed include reliability, availability, and failure frequency. These can be determined using exact computations or by Monte Carlo simulation. The Birnbaum, Fussell-Vesely, and Criticaly importances of BEs can also be computed.

RiskSpectrum FTA The software suite RiskSpectrum [53] by Lloyd's Register Consulting includes an FTA tool. The overall suite is designed for probabilistic safety assessment, and includes tools for FMEA and human reliability analysis. RiskSpectrum FTA supports static fault trees with CCFs.

The analysis tool can perform qualitative analysis producing MCSs, and quantitative analysis including reliability and availability, as well as sensitivity and importance measures and time-dependent analysis.

3. Dynamic Fault Trees

Traditional FT can only model systems in which a combination of failed components results in a system failure,

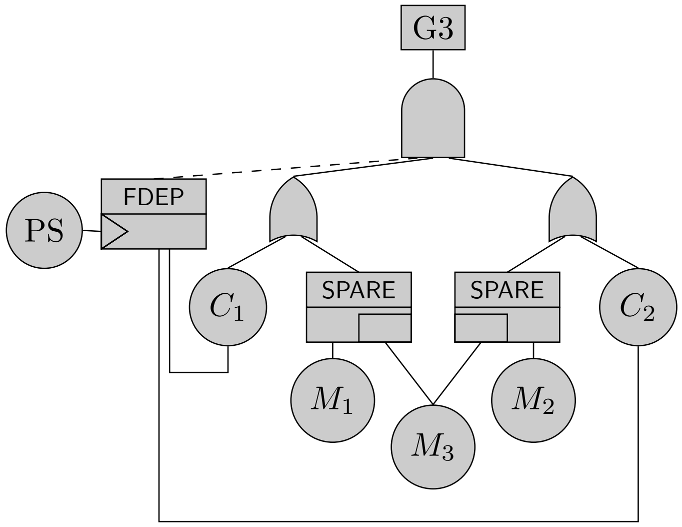
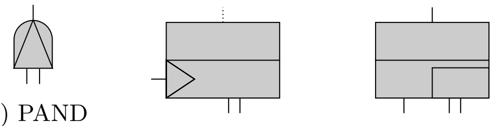


Figure 10: Example of a DFT, equivalent to subtree G3 in Figure 1



(a) PAND gate (b) FDEP gate (c) SPARE gate

Figure 11: Images of the new gates types in a DFT

regardless of when each of those component failures occurred. In reality, many systems can survive certain failure sequences, while failing if the same components fail in a different order. For example, if a system contains a switch to alternate between a component and its spare, the failure of this switch after it has already activated the spare does not cause a failure.

The most widely used way of including temporal sequence information in FT is the *dynamic fault tree* or DFT [62]. The next subsection explains the DFT formalism in detail.

Since a DFT considers temporal behaviour, the methods used for the analysis of static FT cannot be directly used to analyze DFT. An overview of the various quantitative methods is shown in Table 5. The qualitative methods are listed in Table 6. Details of qualitative and quantitative analysis methods are given in Sections 3.3 and 3.4.

3.1. DFT Structure

The structure of a DFT is very similar to an FT, with the addition of several gate types shown in Figure 11. The new gates are:

PAND (Priority AND) Output event occurs if all inputs occur from left to right.

FDEP (Function DEPendency) Output is a dummy and never occurs, but when the trigger event on the left occurs, all the other input events also occur.

SPARE Represents a component that can be replaced by one or more spares. When the primary unit fails, the first spare is activated. When this spare fails, the next is activated, and so on until no more spares are available. Each spare can be connected to multiple Spare gates, but once activated by one it cannot be used by another. By convention, spares components are ordered from left to right.

Example 17. *An example of a DFT is shown in Figure 10. This DFT has the same cut sets as the subtree rooted at G3 of Figure 1, but has a more intuitive informal description: M_3 is clearly shown as a shared spare for M_1 and M_2. Also, the system does not directly depend on the power supply PS. Instead, the failure of PS triggers a failure of both CPUs, which more accurately describes the system and eliminates the shared event.*

BEs can have an additional parameter α called the *dormancy factor*. This parameter is a value between 0 and 1, and reduces the failure rate of the BE to that fraction of its normal failure rate if the BE is an inactive input to a SPARE gate [29]. For example, a spare tire will not wear out as fast as one that is in operation. For BEs that are not inputs to a SPARE gate, α has no effect.

The introduction of the PAND gate means that a DFT is not generally coherent: An increase in the failure rate of the right input to a PAND can increase the reliability of the gate. Since the inputs to PAND gates are commonly also inputs to other subtrees, non-coherence is often indicative of a modeling error or suboptimal system design.

In non-repairable DFTs the FDEP gate can be removed by replacing its children by an OR gate of the child and the FDEP trigger. In repairable DFT the applicability of this approach depends on the definition of the FDEP gate: If failures triggered by the FDEP require separate repairs, the transformation is not correct. If repair of the FDEP trigger also restores the triggered components to operation, the transformation does preserve the behaviour.

Definition 18. *A* DFT *is a tuple $DF = \langle BE, G, T, I\rangle$, where BE and G are the same as in a static FT (and we still write $E = BE \cup G$). The function T still denotes the gate type, but now $T : G \mapsto DGT$, with the set of dynamic gates $DGT = GateTypes \cup \{FDEP, PAND, SPAR\}$. I is replaced by an input function: $I : G \mapsto E^*$ yielding an ordered sequence of inputs to each gate.*

Since the output of the FDEP gate is a dummy output and not relevant to the behaviour of the FT, it is often useful to use a *pruned input function* which does not include FDEP inputs [30].

Some types of DFT have additional gates, which are not included in the rest of this paper. Such gates are:

Hot spare Special case of SPARE gate, where the dormancy factor of the spares is 1, i.e. the spare failure rate is the same as the normal failure rate [62].

Cold Spare Special case of SPARE, with a dormancy factor of 0, i.e. spares cannot fail before activated [62].

Priority OR Fails when the leftmost input fails before the others [180]. Can be replaced by a PAND and an FDEP.

Sequence enforcing Prohibits failures of inputs until all inputs to the left have failed [27]. Can be replaced by (cold) SPARE provided the inputs are not shared with other gates.

One-shot PDEP Special case of the FDEP gate, where the occurrence of the trigger event has some probability of causing a failure of the dependent events [140].

Persistent PDEP Special case of the FDEP gate, where the occurrence of the trigger event causes an increase in the failure rates of the dependent events [140].

3.1.1. Stochastic Semantics

This section presents the formal semantics of DFTs in terms of random variables, pinning down the stochastic behaviour of a DFT model in a mathematically precise way. Such a semantics did not exist yet, which is surprising, since it forms the basis of the analysis methods.

We focus on non-repairable DFTs where all children of FDEP and SPARE gates are BEs: extensions with repair or general FDEPs and SPAREs require novel research and fall outside the scope of this survey. In particular, repairable PAND gates can be interpreted in several ways [22, 87]: If the second input to a PAND is repaired but fails again it is unclear from the informal description if the PAND should fail. Similar discussions exist for other dynamic gates [51]. Also, the semantics is not clearly defined if the children of a SPARE or FDEP gate are (potentially shared) subtrees.

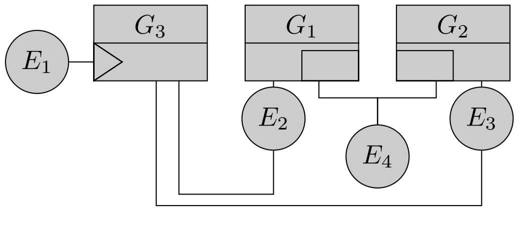


Figure 12: Example of a dynamic fault tree, failure of E_1 may cause nondeterministic allocation of E_4.

We decorate every BE e with a failure distribution $D_e : \mathbb{R}^+ \mapsto [0,1]$ such that $D_e(t)$ yields the probability that BE e fails within time t. Additionally each BE has a dormancy factor α_e which determines how much slower the component degrades when it is an inactive spare. We now define the independent event failure times just like for SFTs, namely $F_e \sim D_e$. Later, we will define F_e^D to be the actual failure time, which includes corrections for time spent as a dormant space, and for failures caused by functional dependencies.

If BEs simultaneously fail for multiple SPARE gates, these gates may attempt to claim the same spare. In this case, the activation order of the SPARE gates is nondeterministic. In Figure 12, the failure of E_1 causes the failure of either G_1 or G_2, but does not specify which. Analysis tools often provide some resolution of this nondeterminacy, but different tools make different choices, possibly resulting in unexpected differences in results between different analyses of the same system.

First we define the *claiming* semantics of the SPARE gates. The goal here is twofold: (1) we need to determine which set of inputs needs to fail for the gate to fail, since the gate may fail when other inputs have not failed but are claimed by other gates, and (2) we need to determine the times at which each spare component is claimed, to compute the correct failure times including dormancy factors.

Later, we will define $\mathit{Suc}(e)$ to be the set of all BEs that are claimed as an immediate result of the failure of e. First, we define $C : \mathbb{N} \times G \mapsto \mathit{BE} \cup \{\bot\}$ to be either the BE claimed by a specific SPARE gate due to the failure of one of its inputs, or $\bot$ if the failure of this input causes the gate to fail, and therefore not claim any other BE. $C(i,g)$ is a strategy that fixes a particular activation order and claiming gate.

Intuitively, we distinguish tree cases resulting from the failure of a spare BE e:

- If e is the rightmost input, no other BE can be activated
- If all BEs to the right of e are already claimed, i.e. activated as a result of another BE that failed before e, the gate cannot claim any BEs.
- Otherwise, there is a leftmost spare f that has not yet been claimed, and this spare will be claimed when e fails. Note that the failure time of f may be before e, in which case yet another BE may be claimed immediately upon claiming f.

Formally, $C(i,g) =$

$$\begin{cases} \bot & \text{if } i = |I(g)| - 1 \\ \bot & \text{if for } e = I(g)_i, \\ & \quad \forall_{j>i} \exists_{f \neq e} I(g)_j \in \mathit{Suc}(f) \wedge F_f^D \leq F_e^D \\ I(g)_j & \text{if for } e = I(g)_i, \\ & \quad j = \underset{j>i}{\arg\min}\, \nexists_{f \neq e} I(g)_j \in \mathit{Suc}(f) \wedge F_f^D \leq F_e^D \end{cases}$$

We now define the successor set $\mathit{Suc}(e)$ and predecessor $\mathit{Pre}(e)$ of an event. Every spare component e has exactly one predecessor, which is the BE whose failure immediately causes e to be claimed and activated by one of its parent gates. For notational convenience, let us denote the set of SPARE parent gates as $\mathit{PSP}(e) = \{g \in G | T(g) =$

$\mathit{SPARE} \wedge e \in I(g)\}$. Now

$$\begin{aligned} &\mathit{Suc}(e) = \{C(i,g) \mid e = I(g)_i \wedge g \in \mathit{PSP}(e) \wedge C(i,g) \neq \perp\} \\ &\mathit{Pre}(e) = f \text{ where } e \in \mathit{Suc}(f) \end{aligned}$$

The actual failure time of a BE, possibly delayed from the time predicted from its failure distribution due to dormancy, can be computed depending on the failure time of its predecessor as

$$F_e^S = \begin{cases} F_e & \text{if } \forall_{g \in \mathit{PSP}(e)} : e = I(g)_0 \\ F^D_{\mathit{Pre}(e)} & \text{if } \frac{F_e}{\alpha} \leq F^D_{\mathit{Pre}(e)} \\ F_e + (1-\alpha)F^D_{\mathit{Pre}(e)} & \text{otherwise} \end{cases}$$

Moreover, the effect of possible early failures as a result of FDEP gates needs to be considered:

$$F_e^D = \min\left(\{F_e^S\} \cup \left\{ F_t \middle| \exists_{g,t} : \begin{array}{l} T(g) = \mathit{FDEP} \\ \wedge\, t = I(g)_0 \wedge e \in I(g) \end{array} \right\}\right)$$

For the sake of clarity, we do not consider FTs where BEs are functionally dependent on themselves, directly or indirectly.

For notational convenience, let $C(g) = \{i | i = I(g)_0 \vee \nexists_e : i \in C(e,g)\}$ denote the set of events that are claimed by SPARE gate g at any time. Also, let $\mathit{Ord}(s) = \forall_{n<|s|-1} : F^D_{s_n} \leq F^D_{s_{n+1}}$ denote whether the failures of all events in s occur in the order they are listed, with $|s|$ denoting the length of sequence s.

Finally, we can determine the failure times of the gates

$$F_g = \begin{cases} \max\{F_i^D \in \mathbb{R} | i \in I(g)\} & \text{if } T(g) = \mathit{AND} \\ \min\{F_i^D \in \mathbb{R} | i \in I(g)\} & \text{if } T(g) = \mathit{OR} \\ \min\left\{ t \in \mathbb{R} \middle| \sum\limits_{i \in I(g)} X_i(t) \geq k \right\} & \text{if } T(g) = \mathit{VOT(k/N)} \\ \infty & \text{if } T(g) = \mathit{FDEP} \\ \max\{F_i^D \in \mathbb{R} | i \in S(g)\} & \text{if } T(g) = \mathit{SPARE} \\ \max\{F_i^D \in \mathbb{R} | i \in I(g)\} & \begin{array}{l}\text{if } T(g) = \mathit{PAND} \\ \text{and } \mathit{Ord}(I(g))\end{array} \\ \infty & \text{otherwise} \end{cases}$$

and $F_g^D = F_g^S = F_g$. The state of an element can be described as

$$X_x(t) = \begin{cases} 1 & \text{if } F_x^D \leq t \\ 0 & \text{otherwise} \end{cases}$$

3.2. Analysis of DFT

The remainder of this section explains the various analysis techniques applicable to DFTs, and the measures they compute.

Measures of interest Most of the values that can be computed for classic FT can still be used in the analysis of DFT; the reliability, availability, and MTTF are still of interest.

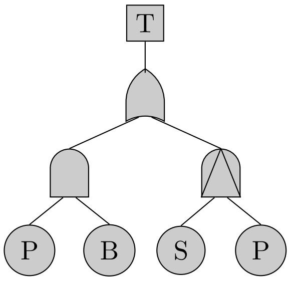


Figure 13: Example of a DFT with temporal sequence requirements. The system fails if both the primary (P) and backup (B) fail, or if the primary fails when the switch (S) to enable the backup has already failed.

DFT are generally non-repairable, so measures that are only applicable to repairable systems are not generally applicable to DFT. Some extensions to DFT, such as that by Boudali et al. [27], do allow repairs, and then measures such as MTBF become useful.

Cut set analysis is less useful for DFT, as CS do not include sequence information. A variant of cut sets, called cut sequences and explained below, can be used, but importance measures over these are not well developed.

3.3. Qualitative analysis

Cut sets and sequences A simple form of qualitative analysis of a DFT can be performed by employing the same techniques as used for SFT; namely by replacing the PAND and SPARE gates by AND gates, and the FDEP gates by OR gates. This analysis will not capture the temporal requirements of the tree. Nonetheless, the cut sets can be used to improve system reliability, since at least one cut set must completely fail for a system failure to occur.

Example 19. *In Figure 13, this method replaces the PAND gate on the right by an AND gate. The resulting cut sets are $\{P, B\}$ and $\{S, P\}$. These cut sets can be useful, as preventing the failures of every cut set still prevents system failure. However, unlike in the SFT, the failure of $\{S, P\}$ does not necessarily cause a system failure, depending on the ordering of the failures.*

To capture these temporal requirements, Tang et al. [172] introduced the notion of 'cut sequences' as the dynamic counterpart to cut sets. A cut sequence is a sequence of failures which cause a system failure. Formally, a sequence $\langle e_1, e_2, \ldots, e_n \rangle$ is a cut sequence of the DFT D if, given failure times $F_{e_1} < F_{e_2} < \cdots < F_{e_n}$, $X_D(F_{e_n}) = 1$ according to the semantics of Section 3.1.1.

Tang et al. [172] also showed that these cut sequences can be determined by replacing the dynamic gates by static gates, determining the minimal cut sets, and then adding any sequencing requirements to the cut sets.

For example, the DFT in Figure 13 has cut sequence set (CSS) $\{\langle S, P \rangle, \langle P, B \rangle, \langle B, P \rangle\}$. The sequence $\langle P, S \rangle$ is not a cut sequence since the failure of S after P does not trigger the PAND gate.

Zhang et al. [192] offer a more compact way of representing cut sequences, by adding temporal ordering requirements to cut sets. This allows one representation to cover multiple cut sequences at once, where some events are ordered independently of other events. This method would represent the CSS of Figure 13 as
$\{\{S, P, S < P\}, \{P, B\}\}$.

Liu et al. [114] provide an alternative method to determine cut sequences by composition of the cut sequences of the subtrees. This method reduces the amount of repeated work if the same components are present in multiple cut sets. Additionally, they show [116] that the cut sequences can be used to perform quantitative analysis.

A different definition of qualitative analysis for repairable DFT is provided by Chaux et al. [45]. The complexity of this method is based on the length of the longest non-looped sequence of failures and repairs in the system. This definition defines a language of failure and repair sequences, and provides a means for constructing a finite automaton that generates all sequences of failures and repairs in which the final state in a system failure. To keep the language finite, only the sequence up to the first system failure is considered.

Another algebraic method for determining and expressing cut sequences was developed by Merle et al., by extending the structure function used for static FTA (described in section 2.2.1) to first include the Priority-AND gate [122] by allowing a 'before' relation as a boolean primitive. This method is subsequently developed to include the other DFT gates [124, 121, 123]. The structure function can subsequently be used to perform quantitative analysis [121].

Considering again Figure 13, the FT has the boolean expression $(P \wedge B) \vee (S \wedge P \wedge (S < P))$. This expression can be simplified using the law $A \wedge (A < B) = (A < B)$ into $(P \wedge B) \vee (P \wedge (S < P))$. This is the minimal disjunctive normal form, showing that $P \wedge B$ and $P \wedge (S < P)$ are the minimal sets of failures and sequence dependencies that yield a top event failure.

More recently, Rauzy [148] proposed a variant of Minato's Zero-Suppressed BDD [95] to include ordering information. This variant can be used to find the minimal cut sequences of DFT, and the author believes that more efficient algorithms for other analyses can be based on this representation.

3.4. Quantitative analysis

This section describes analysis techniques for quantitative measures of DFTs. The definitions of the measures have already been explained in sections 2.3 and 2.4, so we will only state which measures can be computed by each technique.

Algebraic analysis The structure function obtained by qualitative analysis can also be used for quantitative analysis. Applying the inclusion-exclusion principle to the cut sets, we obtain

$$\mathbb{P}(T) = \mathbb{P}(P \wedge B) + \mathbb{P}(P \wedge (S < P)) - \mathbb{P}(P \wedge B \wedge (S < P))$$

Now, expressions for the probabilities can be substituted [124], giving the failure probabilities at time t in terms of the BE failure distributions $D_e(t)$ and failure probability density functions $d_e(t)$:

$$\begin{aligned}\mathbb{P}(T)(t) &= D_P(t) \cdot D_B(t) \\ &\quad + \int_0^t d_P(u) D_S(u) du - D_B(t) \cdot \int_0^t d_P(u) D_S(u) du\end{aligned}$$

For larger DFTs, many repeated integrations make this approach computationally impractical.

Analysis by Markov Chains

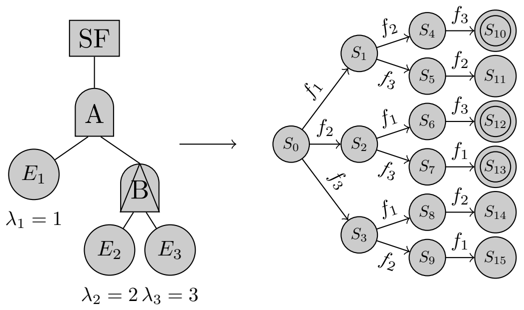


Figure 14: Example conversion of DFT to a Continuous Time Markov Chain. States corresponding to system failures (goal states) are indicated by a double circle. Transition f_i denotes the failure of BE E_i, and occurs with rate λ_i.

The first method proposed to analyze DFT was by Dugan et al. [62, 63], and computes the unreliability of the system during a time window $[0, t]$. This method converts the DFT into a Markov Chain, in which the states represent the history of the DFT in terms of what components have failed and, where needed, in what order. Since the number of failed subsets grows exponentially in the number of BEs, this method is not practical for very complex systems.

Example 20. *Figure 14 shows a simple DFT converted into a Markov Chain. From the starting state S_0, in which all components are operational, three transitions are possible representing the failures of the three BEs. After the failure of the first BE, two more BEs can fail, and finally the last BE fails. If all three BEs have failed, and E_2 failed before E_3, system failure occurs, which corresponds to the circled (goal) states in the MC. In the other states the system is still operational. Existing tools such as PRISM [107] can be used to compute the probability of* *reaching a goal state within a certain time, corresponding to system unreliability.*

The MC in Figure 14 could be reduced without affecting the computed probabilities. For example, from S_3 no goal state can ever be reached. It is therefore acceptable to replace S_3 by an absorbing state to reduce the complexity of further analysis. A full discussion of minimization techniques is beyond the scope of this paper, but several are listed in [12].

Codetta-Raiteri [48] presents a transformation of DFTs to Stochastic Petri Nets [46], which are in turn analyzed by conversion to Markov Chains. Although this method still suffers from a combinatorial explosion when constucting the Markov Chain, the Petri Nets are much smaller and easier to understand and extend.

Compositional analysis of DFT Boudali et al. [27, 29] use a different method to calculate the reliability of a DFT, which reduces the combinatorial explosion in many common cases. They provide a *compositional* semantics for DFT, i.e. each DFT element is interpreted as an Interactive Markov Chain [91] and the semantics of the DFT is the parallel composition of the elements. The papers provide several reduction techniques to minimize the resulting Markov Chain. In addition, it allows DFT to be extended with repairable components and mutually exclusive events.

The analysis is performed by converting a DFT into an *Input/Output Interactive Markov Chain* for analysis. This model is constructed by computing the parallel composition of the I/O IMCs for parts of the tree, down to individual gates and events. Since intermediate models can be analyzed to remove unnecessary states, the total I/O IMC can be much smaller than the Markov Chain produced by earlier methods, and the combinatorial explosion is reduced.

The program DFTCalc was developed by Arnold et al. [10] to analyze reliability and availability of DFT using the I/O IMC methodology.

Example 21. *Figure 15 shows the I/O IMC equivalents of the basic event E_1 and the gate A of the DFT in Figure 14. Below that, the parallel composition of the two elements are shown. This composition behaves as if the two separate elements are ran in parallel, with the output signal of the BE (f_{E_1}!) permitting the transition with input signal f_{E_1}? in the gate's IMC.*

Observe that input signal f_B? is still present in the composition, allowing this IMC to be composed with gate B later. Similarly, output action F_{E_1}! allows the later composition with other gates in which E_1 is an input. If no such gates exist, the IMC can be minimized by removing these output transitions.

Unlike traditional Markov Chains, I/O IMC are capable of modeling nondeterminism between actions. Guck et al. [87] use this approach to model maintenance strategies where it is not specified which of multiple failed components to repair first.

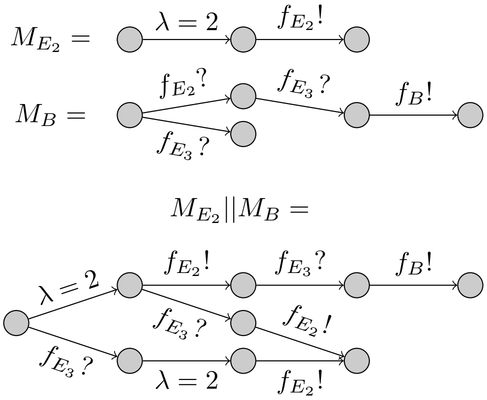


Figure 15: Example conversion of part elements E_1 and A of the DFT in figure 14 to an I/O Interactive Markov Chain. Input signals are denoted by a question mark, output signals by an exclamation mark.

Pullum and Dugan [142] developed a program to divide a DFT into independent submodules for computing reliability. Submodules containing only static gates can then be solved using a traditional BDD method, while submodules containing dynamic gates can be solved using Markov Chain analysis.

Example 22. *Suppose we are computing the availability at time t of the DFT in figure 14. We can convert the entire DFT into a Markov Chain such as the figure shows, but only the subtree rooted at B is dynamic. We can therefore replace this subtree by a fictional node B^* and use a BDD to determine the minimal cut sets of the tree, which is only $\{E_1, B^*\}$. Following section 2.4.3, the availability of the tree is given by $A_{SF}(t) = A_{E_1}(t) \cdot A_{B^*}(t)$. Markov chain analysis can now be used to compute the value $A_{B^*}(t)$, and $A_{E_1}(t)$ is the same as for a static fault tree.*

An algebraic method for quantitative analysis is introduced by Long et al. [117], which can compute availability at a specific time and ENF per unit time. It uses a system of logic called 'Sequential Failure Logic' to describe the temporal restrictions within cut sets. Unfortunately, the equations produced are difficult to solve due to many multiple integrals, and only a special case where all failure and repair rates are identical is presented.

Han et al. [89] also modularize a DFT and use BDD for the static submodules, but use the approximation by Amari et al [6] to solve the dynamic submodules. This avoids the state-space explosion problem of analysis by conversion to Markov Chain, while retaining a reasonable degree of accuracy.

Later, Liu et al. [115] proposed a method to modularize DFT further, by also collapsing static subtrees of a dynamic gate, but keeping additional information about the probability distribution of these subtrees.

Yevkin [189] provides additional modularization techniques, which can convert static subtrees and some dynamic subtrees into equivalent BEs, reducing the complexity of further analysis.

Analysis using Dynamic Bayesian Networks The method by Bobbio et al. [23] of converting an SFT into a Bayesian Network (described in Section 2.3.3) was later improved by Montani et al. [128] by using a Dynamic Bayesian Network (DBN) to analyze DFTs.

In this approach, the DBN is evaluated at many points in time, with the state probability distributions carried over from each timestep to the next. By also allowing nodes to have probabilities conditional on their own state in the previous timestep, dynamic behaviour can be included in the analysis. Due to the discretization, results from this method are not exact. Results can be made arbitrarily accurate, but at the cost of a sharp increase in computation time required. Only non-repairable FTs are analyzed by this method, however Portinale et al. [140] propose a similar method for repairable FTs. Other extensions from the earlier BN work such as noisy gates remain applicable.

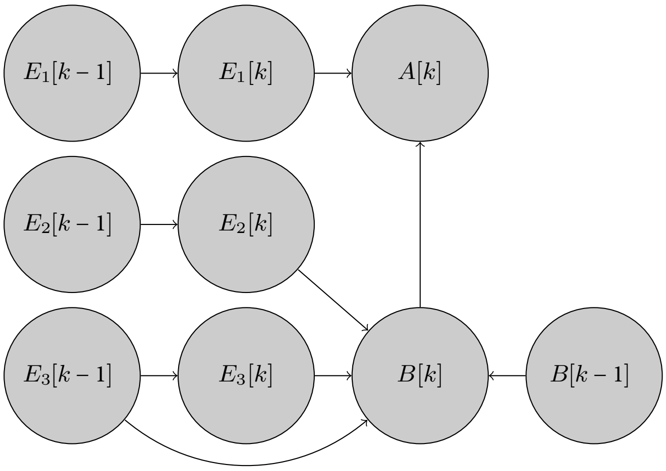
The Bayesian Network method has been extended by Boudali and Dugan [31] to model DFT gates. This method can produce results equivalent to solving a discretized version of the Markov Chain corresponding to the DFT, but can also be extended with dependent component failures and multi-state components by changing the produced DBN. No comparison between this method and the method by Montani et al. [128] is presently available.

Example 23. *Figure 16 shows the dynamic bayesian network of the DFT in figure 14. Gates $\{A, B\}$ and basic events $\{E_1, E_2, E_3\}$ form the nodes of the network, while input relations in the DFT form one-way conditional probabilities. Basic events are not repairable, and thus remain failed if they were failed in the previous timestep. Otherwise, the probability of their failure in the current timestep depends on their failure rate. This explains the first two conditional probability rules.*

The next two rules give the behaviour of the PAND gate B. If it was failed in the previous timestep (i.e. $B[k-1] = 1$, it remains failed (i.e. $B[k] = 1$). Otherwise, it fails if both inputs are failed, and E_3 was not failed earlier. Note that behaviour on simultaneous failure is deterministic in this model (namely the PAND gate fails on simultaneous failure of its inputs).

Finally, the state of and AND gate A is determined purely by its inputs.

As for other analysis methods, computational requirements can be reduced by modularizing the FT and using more efficient methods for the static subtrees. Such an approach combining BDD and DBN was proposed by Rongxing et al. [153].



$$\mathbb{P}\big[E_i[k] = 1 | E_i[k-1] = 1\big] = 1$$
$$\mathbb{P}\big[E_i[k] = 1 | E_i[k-1] = 0\big] = \delta\lambda_i$$
$$\mathbb{P}\big[B[k] = 1 | B[k-1] = 1\big] = 1$$
$$\mathbb{P}\left[\begin{array}{l} B[k] = 1 | E_2[k] = E_3[k] = 1 \\ \wedge E_3[k-1] = 0 \end{array}\right] = 1$$
$$\mathbb{P}\big[A[k] = 1 | E_1[k] = 1 \wedge B[k] = 1\big] = 1$$

Figure 16: Dynamic Bayesian Network corresponding to the DFT in Figure 14 with timestep δ. Default rules with probability 0 have been omitted.

Since a BN allows arbitrary conditional probabilities to be specified, it is possible to include failure rates of gates in addition to that implied by the tree structure. This improves accuracy and reduces the effect of modeling errors. Such an approach was described by Graves et al. [86]. This is useful, since many real-life systems record component failures at an intermediate level, rather than diagnosing every fault to the level of the BE.

Other approaches Mo [126] described a method for converting a DFT into a multiple-valued decision diagram (MDD) to compute the reliability of non-repairable systems. In this approach, subtrees containing only static gates are directly converted into MDDs, while subtrees with dynamic gates are solved by conversion into a CTMC before the results are included in the MDD. This approach reduces the state-space explosion problem in many common cases, but in the worst case of a dynamic gate as the TE a full CTMC still needs to be solved.

A purely algebraic approach is suggested by Amari et al. [6], which calculates the probability distribution at every gate by appropriately combining the distributions of the inputs. While this approach gives exact results and does not suffer from the state-space explosion effect common when using Markov Chains, only a subset of trees satisfying particular rules can be analyzed this way.

Ni et al. [131] propose a different algebraic method for describing the DFT structure, which produces a boolean-like expression of the DFT. This method allows minimal cut sequence determination as well as quantitative analysis.

Simulation Quantitative analysis can be performed by Monte Carlo simulation. Failures and/or failure times are sampled from their respective distributions, and the effect these failures have on the system are calculated.

Quantitative Monte Carlo analysis can be performed using the method by Durga Rao et al. [65], which can also be applied if the components are individually independently repairable.

Boudali et al. [26] developed a program to analyze DFT using Monte Carlo simulation. It allows BE failure distributions to change over time, and even based on different clocks for different BE, resulting in non-Markovian models. This is useful when, for example, a system takes time to warm up and this affects the failure rates.

If the minimal cut sets have already been determined, Liang et al. [112] propose a Monte Carlo method for computing the unreliability of an RFT. This approach allows the failure and repair rates to follow arbitrary distributions, but still does not allow repair policies other than independent component repair.

Zhang et al. [193] showed that it is possible to convert a DFT to a Petri Net, on which quantitative analysis can be performed by simulation. Exact analysis on Petri Nets is normally done by conversion into Markov Chains, still resulting in a state-space explosion. Simulation, however, can be performed directly on the Petri Net, although the benefits compared to simulation of the untransformed DFT are not stated.

If very high performance is required, it is possible to construct a hardware circuit to perform Monte Carlo Simulations much faster than normal computer simulation. Such an approach is described by Aliee and Zarandi [4].

Rajabzadeh and Jahangiry [143] propose a conversion of a DFT into an analogue electronic circuit, which outputs a voltage corresponding to the system failure probability. This approach does require an approximation for some of the gates, and the accuracy on larger models is not demonstrated.

A method for the analysis of the sensitivity of various model parameters is provided by Ou and Dugan [134].

4. Other Fault Tree extensions

While dynamic fault trees are the most popular extension to static fault trees, several other ways of extending FTs have been proposed. The extensions can be approximately divided into several categories. (1) fault trees using *fuzzy numbers* can be used in cases where failure probabilities or behaviour are not known exactly. (2) Several extensions allow fault trees to model systems where basic events are *stochastically dependent,* such as when a failure of one component increases the failure rate of another component. (3) *Repairable Fault Trees* can represent more complex repairable systems than the simple repair rates in classic FT. (4) The temporal relations between events are important. Dynamic fault trees include certain temporal dependencies, but other extensions have been proposed as well. (5) In particular, State/Event Fault Trees were introduced to model systems and components with a state that varies over time, and where this state affects the consequences of component failures or the failure rates. (6) Miscellaneous extensions, e.g. integrating Attack Trees with FTs.

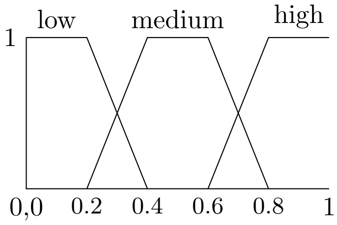


Figure 17: Example of fuzzy membership functions of the sets 'low', 'medium', and 'high'

These extensions are discussed in sections 4.1 through 4.6, respectively. An overview of the extensions can be found in Tables 7 (page 31) and 8 (page 32).

4.1. FTA with fuzzy numbers

Fault trees using fuzzy numbers were introduced by Tanaka et al. [171] as a way to reduce the problem that failure probabilities of components are often not exactly known. Fuzzy numbers represent uncertainty by not specifying an exact number, but rather a range which contains the true value. Alternatively, they can be used as input to the FT, in which case they specify categories to which a probability belongs, to a greater or lesser degree.

Example 24. *For example, suppose we would like experts to specify a failure probability using the categories 'high', 'medium', and 'low'. It is possible to set exact endpoints and ask the experts to rate any value between 0 and 0.2 as low, this has two disadvantages: First, linguistic descriptions are commonly used so that the expert does not need to estimate an exact probability, and giving endpoints reintroduces that requirement. Second, if the expert estimates a probability to be approximately 0.2, the expert must decide whether this is low or medium, and the model does not capture the uncertainty that the expert may have.*

Alternatively, we can describe the categories as fuzzy subsets of the interval $[0, 1]$*. Figure 17 shows possible membership functions for the categories. Here, for example, the value 0.1 is said to be fully a member of 'low' and no member of either other category. Thus experts are assumed to always classify 0.1 as low. The value 0.3 is partly a member of 'low' with membership 0.5, signifying that half of the experts would classify 0.3 as low.*

Author	Method	Repairs	Reliability	Availability	Other	Remarks	Tool support
Dugan et al. [62]	Markov Chain		+	+	+	Suffers from state-space explosion	
Boudali et al. [27]	I/O IMC	+	+	+	+	Less state-space explosion for most models	CORAL [28], DFTCalc [10]
Codetta-Raiteri [52]	Petri Nets		+	+	+	Intermediate model smaller than Markov Chain	DFT2GSPN [48]
Pullum and Dugan [142]	Modularization		+			Fast when FT has small dynamic subtrees	SHADE Tree [142], DIFTree [64]
Long et al. [117]	SFL			+	+	No practical algorithm for realistic DFT	
Han et al. [89]	Approximation		+			Reasonable accuracy based on experiments	
Liu et al. [115]	Prob. distr.		+			For DFT with large static subtrees, approx.	
Yevkin [189]	Modularization		+			Reduces complexity of some specific subtrees	
Amari et al. [6]	Approximation		+			Requires tree following certain rules	
Montani et al. [128]	DBN	+	+			Not exact, allows dependent BE	Radyban [128, 139, 129]
Boudali and Dugan [31]	DBN	+	+			Not exact, allows multi-state, dependent BE	
Rongxing et al. [153]	BDD & DBN		+			Efficient for DFT with static subtrees	
Graves et al. [86]	DBN		+			Incorporates gate failure data	
Mo [126]	MDD		+			Reduces state-space explosion	
Ni et al. [131]	Algebraic		+			Finds MCS and performs quantitative analysis	
Durga Rao et al. [65]	Monte Carlo	+	+	+	+	Allows independently repairable components	
Boudali et al. [26]	Monte Carlo	+	+	+	+	Allows non-Markovian systems	DFTSim [26]
Liant et al. [112]	Monte Carlo	+	+	+	+	Requires cut sets, allows repairs	
Zhang et al. [193]	Monte Carlo	+	+	+	+	Transforms to Petri Net	DRSIM [65]
Aliee et al. [4]	Monte Carlo	+	+	+	+	Hardware method for fast simulations	
Rajabzadeh et al. [143]	Hardware		+	+	+	Not exact, untested for large models	

Table 5: Overview of DFT quantitative analysis methods

Author	**Method**	**Remarks**
Tang et al. [172]	Cut sets	Postprocessing to convert cut sets to cut sequences
Liu et al. [114, 116]	Composition	Reduces work for shared components
Zhang et al. [192]	Cut sequences	More compact representation of CSS
Chaux et al. [45]	Language theory	Allows repairs up to first TE occurrence
Merle et al. [121]	Algebraic	Also allows quantitative analysis
Rauzy [148]	ZBDD	Starting point for other analyses

Table 6: Overview of DFT qualitative analysis methods

Mahmood et al. [119] have conducted a literature review exploring different variations of Fuzzy Fault Trees, and various methods for their analysis. A brief overview is provided below.

FTs are often specified using fuzzy numbers for the probabilities or possibilities of basic events. A common method is to use fuzzy set theory: A fuzzy set has a membership function which gives, for any argument, the degree to which that argument is a member of the given fuzzy set. In this context, BE probabilities are given as a fuzzy subset of the interval [0, 1].

The membership function of a fuzzy subset of the real numbers is similar to the probability density function of a probability distribution. The difference is that where a PDF gives the probability of a variable having a value given the distribution this variable belongs to, the membership function gives the degree to which a value belongs to a fuzzy set, without making a claim regarding the likely values of variables given the fuzzy set.

If a fuzzy number contains only one possible value, it is the same as a conventional or *crisp* number.

Singer [163] provides a method for computing the TE fuzzy probability if the membership function can be specified in a special form called an L-R type. This is a function that is symmetric about some point on the probability axis except for a scaling factor, and can be represented by a function of the form

$$m(p) = \begin{cases} L(p) = f\left(\frac{c-p}{l}\right) & \text{if } p < c \\ R(p) = f\left(\frac{p-c}{r}\right) & \text{if } p >= c \end{cases}$$

Where $f : \mathbb{R} \mapsto \mathbb{R}$ is some function, c is the point of symmetry, and l and r are scaling factors.

This method is frequently applicable since many common probability distributions (including the normal, uniform, and triangular distributions) can be described in this form.

An alternative method is described by Lin et al. [113], in which some of the BEs are described by multiple fuzzy numbers obtained from different experts. These fuzzy numbers could, for example, be derived from natural language expressions describing the events from 'very probable' to 'very improbable'. This method combines these multiple fuzzy probabilities into one crisp probability for the BE, and then analyses the FT as normal.

When multiple probability estimates are available, Kim et al. [104] offer a method to use these to calculate 'optimistic' and 'pessimistic' fuzzy probabilities for the TE. This approach may be useful when each expert gives only small uncertainties due to natural variation in components, but different experts give these uncertainties over different ranges, for example due to different opinions of the likelihood of human error.

If the membership functions for the BE probabilities are themselves uncertain, this may be included in the model using 'intuitionistic fuzzy set theory', as described by Shu et al. [162, 111]. In this model, two membership functions describe an upper and lower bound on the membership. This can be used if, for example, a probability is believed to lie between 0.4 and 0.6, but it is not known what value in this range is the most likely.

Ren and Kong [152] provide a means for analyzing an FT when not only the BE probabilities are uncertain, but also the effects of component failure on the rest of the system. In this framework, components can have multiple states rather than only operational and failed. Each gate can also have multiple states, and these states can be triggered by various combinations of input states. This can model a system which can continue operating after certain component failures, but only in a degraded way. Such a degradation can have other effects on the gates above it.

An alternative approach to uncertain network structure is the introduction of *noisy* gates [23]. These gates have some probability of failing when the standard gate would not, or vice versa. For example, a computer with redundant hard drives may fail to detect and correct certain errors, leading to a system failure even though the backup drive is perfectly functional.

In repairable FTs, uncertainty can exist not only in the BE failure rate but also in the repair rate. A system for accounting for this uncertainty in calculating the overall system availability is given by El-Iraki and Odoom [69].

If the failure probabilities are very uncertain, Huang et al. [94] offer a method based on possibility measures that may offer better results than probability-based fuzzy number approaches. In this method, basic events are specified with possibilities representing estimated lower bounds on the failure probabilities. In this context, the possibility of the TE can be calculated quite efficiently.

It is also possible to model the probabilities as themselves being random variables with a normal distribution. As Page and Perry [135] showed, this allows a better quantification of the uncertainty in the result, although it may require more assumptions on the part of the FT designer.

More generally, Forster and Trapp [79] suggest that BE probabilities can be specified as intervals, within which the actual probabilities are sure to lie. Their method uses Monte Carlo simulation treating these intervals as bounds on a uniform distribution (although they mention that arbitrary distributions may be used) to compute the second-order probability mass function for the TE probability.

Importance measures for fault trees with fuzzy numbers Aside from the TE probability, it can also be useful to determine which components have the greatest effect on this probability. Several methods for determining this have been developed.

Furuta et al. [81] suggested to extend the structural importance to be calculated using fuzzy probabilities, and named the resulting value the *fuzzy importance*.

Alternative measures were suggested by Suresh et al. [170], which also include the amount of uncertainty contributed by each component. The *Fuzzy Importance Measure* of a component i is defined as
$FIM(i) = ED\,[Q_{qi=1}, Q_{qi=0}]$, where ED denotes the Euclidean distance between two fuzzy numbers, $Q_{qi=1}$ is the TE probability if event i has an occurrence probability of 1, and $Q_{qi=0}$ is the TE probability if event i has a probability of 0.

Similarly, the *Fuzzy Uncertainty Importance Measure* is defined as $FUIM(i) = ED\,[Q, Q_{qi=0}]$, where Q is the TE probability. This measure ranks a component as more important if its probability is less certain.

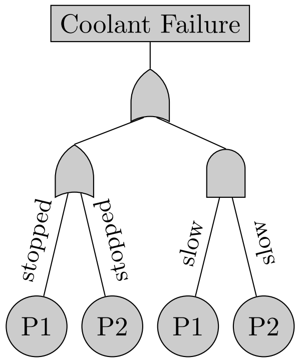
Finally, if the distributions of the BE probabilities can be bounded with certainty, for example based on manufacturer specifications, it is possible to use Interval Arithmetic to obtain exact bounds on the distribution of the TE probability, as shown by Carreras and Walker [41].

Analysis methods measures for fault trees with fuzzy numbers Since the structure of most fuzzy FT is the same as that of classic FT, qualitative analysis can be performed without change. Some extensions, such as the multistate FT by Ren and Kong [152], require different methods.

One of the first methods proposed to analyze a Fuzzy Fault Tree is to determine the minimal cut sets, and perform a standard quantitative analysis using the Extension Principle developed by Zadeh [190] to perform arithmetic on fuzzy numbers.

The use of the Extension Principle is computationally intensive for larger trees, and cannot be applied if repeated events are allowed in the tree. Soman and Misra [167] offer an alternative method to calculate the top event probability, called a 'resolution identity' using the 'α-cut' method, which does allow repeated events and has lower computational requirements.

Guimarẽes and Ebecken [88] present a computer program named FuzzyFTA that can calculate the FIM and FUIM of any gate using either the fuzzy logic approach using α-cut or a Monte Carlo Simulation. The results of these methods are in agreement, although the fuzzy approach provides more information and is quicker.



```
DEFINE FAILDEP pump1:
         CAUSE = P1.slow;
         EFFECT = RATECHANGES P2:*2;
END
DEFINE FAILDEP pump2:
         CAUSE = P2.slow;
         EFFECT = RATECHANGES P1:*2;
END
```

Figure 18: Example of an extended FT. Pumps P1 and P2 have failure modes 'stopped' and 'slow'. Either pump stopping or both pumps slowing leads to failure. Either pump slowing accelerates failure of the remaining pump.

Another approach described by Wang et al. [183, 184] is the conversion of the FT into a Bayesian Network and performing analysis using fuzzy numbers on the resulting BN. It is shown that this approach can give the same results as traditional FT analysis, but it also has the additional flexibility provided by BN.

4.2. Fault Trees with dependent events

Classic FT assume that the BE are all statistically independent. This is often not true in practice, as events can have common causes, or the failure of one component can accelerate the failure of another.

Dynamic gates in DFTs can model some dependencies, as was explained in Section 3.

Buchacker [38, 39] suggests to modify Fault Trees into 'extended Fault Trees' that allow components to have states other than fully operational and fully failed. This allows the modeling of gradual degradation of a component over time, as well as components that can fail in multiple ways that have different interactions with other failures. In addition, this model adds dependencies between components affecting failure and repair rates. Figure 18 shows an example of an extended FT with multi-state components and dependent failure rates.

Another approach for systems with multistate components is provided by Zang et al. [191]. This approach represents the overall system by multiple fault trees, each of which is a fault tree for a particular failure state of the overall system. These trees are then combined into a single multistate decision diagram with dependent nodes, and analyzed to determine the overall probability of the system reaching each failure state.

Twigg et al. [173] suggest a method to specify mutually exclusive events. An example of a model where this is useful, is a valve that can fail open or closed. Since these failure modes cannot occur at the same time, a traditional FT cannot correctly model this component.

Yet another design is provided by Vaurio [175], in which mutually dependent events are replaced by groups of independent events, such that a traditional analysis of the FT gives the correct results. A drawback of this approach is that each group of n dependent events is replaced by $2^n - 1$ independent events, which results in a combinatorial explosion if many events depend on each other.

For models with particularly complex interdependencies, Bouissou [33, 35] offers a formalism called *Boolean logic Driven Markov Processes* (BDMP) as an extension to fault trees. In this formalism, events are described by Markov Processes with designated failure states. Then, events in the FT can cause these events to switch to different processes, for example to increase the failure rate if another component fails.

In addition to analyzing the resulting Markov Chains to obtain reliability and availability, it is possible to extract cut sequences from a BDMP [44], and to construct a Finite State Automaton with equivalent behaviour to the BDMP [43].

Besides Markov Processes, Bouissou [34] also describes the option to replace BEs with Petri Nets, although no method is described for switching these due to external events. This method can improve the modeling of DFT spare gates with shared spare components.

4.3. Repairable Fault Trees

To analyze the reliability of a system over a long period of time, it is often useful to include the possibility of repairing or replacing failed components during this time. These repairs may extend the time before a system failure occurs, such as when a failed redundant part is replaced, or they may return a failed system to normal operation.

Sometimes the simple repair rate model presented in section 2.4.1 is not sufficient. Bobbio et al. [22] introduced *Repair Boxes* which can be connected to a gate, and begins repairs on the BEs of the subtree of that gate only when the gate fails. Raiteri et al. [52] extended these repair boxes to allow different repair policies to be used in the model. The resulting tree is called a *Repairable Fault Tree* (RFT). Figure 19 shows an example of an RFT.

In this formalism, each BE e has a failure rate $FR(e)$, which is the parameter of an exponential distribution that determines the time until the component fails.

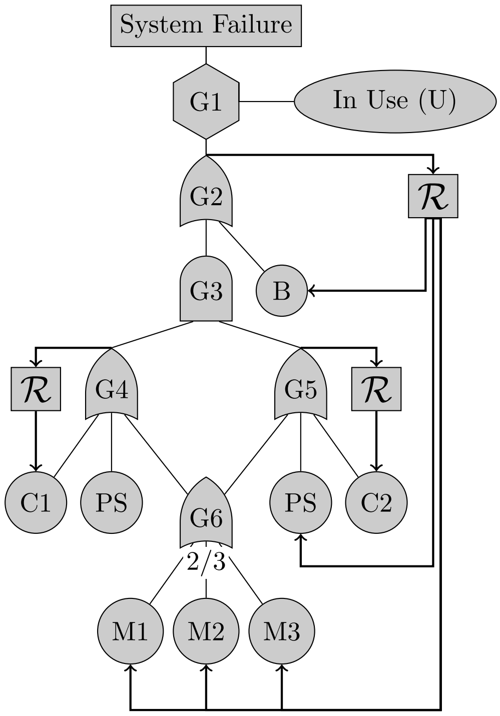


Figure 19: Example of an RFT, repairs on the shared components are only initiated when the entire system fails. CPUs 1 and 2 are repaired when their respective compute node fails.

Each RB is connected to one or more BE to repair, and one incoming BE or gate. When the incoming event occurs, the repair box is activated and begins repairs on the outgoing components according to the *repair policy.* Every component also has a *repair rate* that is the parameter of another exponential distribution modelling the time to repair the component.

Repair policies can be very simple, even equivalent to the simple repair rates model, or more complex, for example restricting the number of components that can be repaired simultaneously.

The major advantage of this approach is that it allows modelling of more realistic systems, and analysis of what repair strategies are best. A disadvantage is these trees cannot be quantitatively analyzed using combinatorial methods.

Flammini et al. [77] added the possibility of giving priority to the repair of certain components, based on the repair rate, failure rate, or level of redundancy of the components. Other priority schemes can also be implemented within this system.

A different extension is provided by Beccuti et al. [16, 18], which adds nondeterminism to the repair policies. This models cases where, for example, a mechanic individually decides which component to repair first. Conversion to Markov Decision Process allows optimal policies to be automatically derived from the FT when costs of unavailability, failures, and repairs are provided. A parametric version [17] of the formalism allows for more efficient modeling and analysis if the FT contains subtrees that differ only in the parameters of the BEs.

Leaving repair policies nondeterministic also allows the computation of an *optimal* repair policy, by associating costs with unavailability, failures and repairs. Becutti et al. [18] show that such an optimal policy can be computed by converting the FT into a Markov Decision Process.

Analysis RFTs can be analyzed to obtain the same measures that apply to classic FTs with repair rates.

Traditional qualitative analysis of an RFT is generally less useful, since such an analysis would ignore the repairability aspect.

Quantitative analysis is more useful, but also more difficult: Combinatorial methods are no longer sufficient, as the evolution of the system over time has to be considered.

For systems where each component can be individually and simultaneously repaired at a constant rate, Balakrishnan and Trivedi [13] proposed to convert the model into a Markov Chain, although this method uses an approximation to reduce computational requirements.

Another approximation is provided by Dutuit and Rauzy [68], although this approximation can also only be used in models with a constant repair rate. The approximation is shown to give results close to the exact solution and several other approximations.

The more general analysis method proposed by Raiteri et al. [52] is to convert the RFT into a Generalized Stochastic Petri Net, and then translate this into a Markov Model. Existing analysis tools for Markov Models can then be applied. Flammini et al. [76] show that this method can be used on parts of a system while the non-repairable parts can be analyzed using traditional methods.

If the FT contains subtrees that can be effectively parameterized, the method by Bobbio et al. [22] of converting the FT into a Stochastic Well-Formed Net (a coloured version of a Generalized Stochastic Petri Net) and then into a Markov Chain may be more efficient, although this formalism only allows relatively simple repair boxes. A later extension by Codetta-Raiteri [49, 51] combines Parametric, Dynamic, and Repairable FTs, allowing complex repair policies, and also performs quantitative analysis by conversion to a Stochastic Well-Formed Net.

A later alternative is offered by Portinale et al. [140], which translates an RFT into a Dynamic Bayesian Net for analysis. This method also allows complex repair policies, as well as components with several different failure modes and statistically dependent failure probabilities.

For performing very fast Monte Carlo simulations, Kara-Zaitri and Ever [103] developed a method for generating a hardware model of the system in a Field Programmable Gate Array, which can perform each Monte Carlo run many times faster than a conventional computer simulation.

4.4. Fault trees with temporal requirements

Dynamic fault trees allow for the inclusion of certain types of temporal information, but for some systems this is not enough. Several other ways have been proposed that offer more flexibility.

One way that has been proposed by Wijayarathna et al. [186] is to add an AND-THEN gate. This gate's output event occurs if the second input occurs immediately after the first. For example, a fire safety system might have backup systems that take time to deploy, so a primary system fault before a fire is not a failure, nor is a fault after a fire has already been extinguished. Only a fault immediately after a fire starts (perhaps caused by the fire) causes a system failure.

Walker and Papadopoulos [179, 180] have suggested extending static FTs with Priority-AND, Priority-OR, and Simultaneous-AND gates. These allow the same temporal relations to be enforced as a dynamic fault tree, but also allow a requirement for simultaneous faults. Such a simultaneous fault is most likely caused by a shared dependency. This method can model any system that can be modeled using the AND-THEN gate. A reduction procedure also described by Walker and Papadopoulos [181] can be used to simplify the analysis.

An advantage of this system is that it can still be qualitatively analyzed using algebraic methods, rather than needing to be converted into a Markov Model or other state-space system.

Another construction is described by Schellhorn et al. [159], which extends classic FTs with cause-consequence OR- and Inhibit-gates, and synchronous and asynchronous cause-consequence AND-gates. In this model, the classic (called decomposition gates or D-gates) are true if their condition is true at all times. The cause-consequence gates (or C-gates) are true for some indeterminate period after their condition is met.

This construction cannot be used for quantitative analysis, as the C-gates do not have well-defined times at which they are true. Qualitative analysis is possible, as it is proven that the prevention of at least one event from every cut set prevents the TE in this model, just like in a static FT.

If timing information is needed beyond the sequence of events, several other extensions can be used. Gluchowski [84] adds Duration Calculus [42] to FT. This formalism allows reasoning about situations where delays are important. Unfortunately, it has not yet been proven that the gate formulas are decidable, and automated analysis tools cannot currently analyze the dynamic portion of these trees.

Another formalism is the Temporal Fault Tree (TFT) by Palshikar [136]. This formalism adds several gates corresponding to operators in Propositional Linear Temporal Logic (PLTLP), such as PREV n, which is true if the input event has been true for the last n amount of time, and the SOMETIME-PAST, which is true if its input has ever been true.

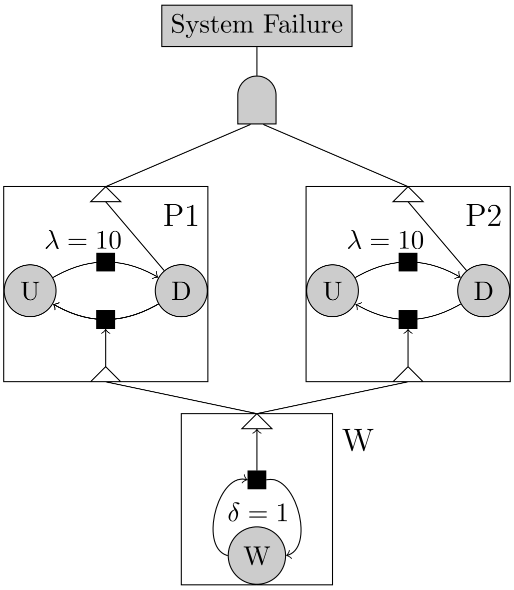


Figure 20: State-event fault tree example of two computer processes P1 and P2, which fail approximately once every 10 hours. The watchdog process W restarts any failed process once per hour. System failure occurs when both P1 and P2 are down.

TFT can impose many types of requirements on the event sequence, but have the disadvantage of requiring the user to understand the formalism of temporal logic.

Qualitative analysis of TFTs is performed by converting them into regular FTs with additional events for the PLTLP gates, and post-processing the resulting cut sets to recover the temporal requirements.

4.5. State-Event Fault Trees

Kaiser and Gramlich [101, 102] have proposed to extend Fault Trees by combining them with Finite State Machines. Such a State-Event Fault Tree (SEFT) allows for greater modularity, and keeps the diagram more readable than a traditional FT of a complex system. In addition, it can model systems and components that have different states with different failure modes. Computer programs are good examples of such systems.

SEFT have states and events. States describe conditions that last for some time, while events occur instantaneously. The two can be linked, as events can cause transitions between states, and a transition between states is an event. Like in an FT, gates can be used to require conditions before an event occurs. An SEFT distinguishes between a History-AND gate and a Sequential-AND or Priority-AND gate, in that the latter requires the input events to occur in a given order.

A later paper by Kaiser [100] adds delay gates, to model events and state transitions that occur some time after an initiating event, conditional probability gates, that cause the output event to occur with some probability every time the input event occurs, and a set of adapter gates that allow certain translations between states and events.

Analysis of SEFT can be performed by translating them into Deterministic and Stochastic Petri Nets, and using existing tools to analyze the resulting DSPN.

Förster and Kaiser [78] provide a more efficient way of performing this analysis, by dividing the SEFT into modules, and converting any static modules found into Component Fault Trees (CFT). A hybrid analysis can then be performed combining BDD for the CFT and DSPN for the dynamic submodules, which is more efficient than using a DSPN for the entire tree.

Xu et al. [188] introduce formal semantics for SEFT, and provide a method based on these semantics to determine MCS. This method extends Interface Automata [58] to Guarded Interface Automata, and translates an SEFT into a GIA Network. From this network the cut sequences can be determined and reduced into a minimal cut sequence set.

Another method for qualitative analysis is provided by Roth et al. [155], which converts the SEFT into an *extended Deterministic and Stochastic Petri-Net* (eDSPN), on which a reachability analysis can be performed to identify event sequences that result in failure.

4.6. Miscellaneous FT extensions

One particular extension that does not fit these categories was proposed by Fovino et al. [80], and integrates Attack Trees with FT. Attack Trees describe vulnerabilities in a system that an attacker could exploit, and countermeasures that could remedy these vulnerabilities.

Since an outside attack could cause a system failure, the combination of AT with FT may provide a better estimate of the system failure probability, assuming probabilities for attack scenarios can be provided.

The integration is performed by designating certain BEs as attack nodes, and decorating these BEs with attack trees. The attack trees are then individually and separately analyzed to determine the probability of a successful attack. Once this analysis is complete, the FT is analyzed by substituting the computed probabilities into the BEs.

Attack trees are sufficiently different from fault trees that we consider them beyond the scope of this paper. An overview of attack trees and related methodologies has been written by Kordy et al. [106].

4.7. Comparison

Tables 7 and 8 summarize the various extensions described above, with strong points denoted with a plus. The meaning of the headers is as follows:

Uncertainty How well the formalism can describe systems with uncertain probabilities and/or structure.

BE Dependence How well the method can model systems in which the basic events are not statistically independent.

Temporal Requirements How well the formalism can include requirements on the sequences or durations of events.

Repairable To what extent the method can include repairable components and descriptions of repair strategies.

Multi-state Whether the model can include components with more states than just failed or not.

BE Prob. distribution Whether the model can describe systems in which the basic events have failure distributions other than constant probability and inverse exponential failure rate.

5. Conclusions

We have given an extensive overview of fault tree analysis methods. Our exposé treated a wealth of available modelling techniques, being static fault trees and their extension; qualitative and quantitative analysis methods; and commercial and academic tools.

This overview lead to several observations and directions for future research.

First of all, as is often the case with modelling languages, fault tree analysis suffers — mildly — from the tower-of-babel-effect: whereas the (static) fault tree formalism was coined as a relatively simple and intuitive modeling language, a "wild jungle" of different formalisms and techniques nowadays exist: Therefore, it would be valuable to know which of the SFT extensions are most useful in practice. Similarly, it would be useful to identify which FT measures are most useful in practice. Also, a comparative case study that compares FT analysis with other risk analysis methods such as reliable block diagrams, AADL, UML/Marte provides useful insight in the capabilities and limitations for fault tree analysis. Thus, we suggest extensive field studies here.

In terms of tool support, an overarching and industry-strength tool, that combines the most common SFT extensions, with the most common FT analysis measures is a valuable addition.

Acknowledgements

This work has been supported by the STW-ProRail partnership program ExploRail under the project ArRangeer (122238) with participation by Movares. We thank Ed Bouwman, Daniele Codetta-Raiteri, Joris ter Heijne, Joost-Pieter Katoen, Sipke van Manen, Luigi Portinale, Judi Romijn, and the anonymous reviewer for their useful comments on earlier versions of this paper.

References

[1] Directive 2006/42/EC of 17 May 2006 on machinery (2006). *(page 3)*

[2] Council directive 89/391/EEC of 12 June 1989 on the introduction of measures to encourage improvements in the safety and health of workers at work (1989). *(page 3)*

[3] S. B. Akers, Binary decision diagrams, IEEE Trans. Comput. C-27 (6) (1978) 509–516. `doi:10.1109/TC.1978.1675141`. *(page 6)*

[4] H. Aliee, H. R. Zarandi, Fault tree analysis using stochastic logic: A reliable and high speed computing, in: Proc. Reliability and Maintainability Symposium (RAMS), 2011, pp. 1–6. `doi:10.1109/RAMS.2011.5754466`. *(pages 23 and 24)*

[5] H. Aliee, H. R. Zarandi, A fast and accurate fault tree analysis based on stochastic logic implemented on field-programmable gate arrays, IEEE Trans. Rel. 62 (2013) 13–22. `doi:10.1109/TR.2012.2221012`. *(pages 9 and 13)*

[6] S. Amari, D. Glenn, H. Eileen, A new approach to solve dynamic fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), 2003, pp. 374–379. `doi:10.1109/RAMS.2003.1182018`. *(pages 21, 22, and 24)*

[7] S. V. Amari, J. B. Akers, Reliability analysis of large fault trees using the vesely failure rate, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2004, pp. 391–396. `doi:10.1109/RAMS.2004.1285481`. *(pages 9 and 14)*

[8] J. D. Andrews, S. Beeson, Birnbaum's measure of component importance for noncoherent systems, IEEE Trans. Rel. 52 (2003) 213–219. `doi:10.1109/TR.2003.809656`. *(page 15)*

[9] M. J. Armstrong, Joint reliability-importance of components, IEEE Trans. Rel. 44 (1995) 408–412. `doi:10.1109/24.406574`. *(page 15)*

[10] F. Arnold, A. Belinfante, D. G. Freark van der Berg, M. Stoelinga, DFTCalc: A tool for efficient fault tree analysis, in: Proc. 32nd Int. Conf. Computer Safety, Reliability and Security (SAFECOMP), Lecture Notes in Computer Science, Springer Berlin Heidelberg, Toulouse, Fance, 2013, pp. 293–301. `doi:10.1007/978-3-642-40793-2_27`. *(pages 21 and 24)*

[11] Automotive Industry Action Group, Potential Failure Mode & Effects Analysis (2008). *(page 2)*

[12] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008. *(page 21)*

[13] M. Balakrishnan, K. Trivedi, Componentwise decomposition for an efficient reliability computation of systems with repairable components, in: 25th Int. Symp. Fault-Tolerant Computing (FTCS), Digest of Papers, IEEE, 1995, pp. 259–268. `doi:10.1109/FTCS.1995.466972`. *(page 28)*

[14] K. Bänsch, A. Hein, M. Malhotra, K. Trivedi, Comment/correction: Dependability modeling using Petri nets, IEEE Trans. Rel. 45 (2) (1996) 272–273. `doi:10.1109/24.510814`. *(page 5)*

[15] R. E. Barlow, F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart, & Winstron, 1975. *(pages 8, 9, and 13)*

[16] M. Beccuti, D. Codetta-Raiteri, G. Franceschinis, S. Hadded, Non deterministic repairable fault trees for computing optimal repair strategy, in: Proc. 3rd Int. Conf. Performance Evaluation, Methodologies and Tools, 2008. *(pages 27, 31, and 32)*

[17] M. Beccuti, G. Franceschinis, D. Codetta-Raiteri, S. Haddad, Parametric NdRFT for the derivation of optimal repair strategies, in: Proc. Int. Conf. Dependable Systems and Networks (DSN), 2009, pp. 399–408. `doi:10.1109/DSN.2009.5270312`. *(pages 28, 31, and 32)*

[18] M. Beccuti, G. Franceschinis, D. Codetta-Raiteri, S. Haddad, Computing optimal repair strategies by means of NdRFT modeling and analysis, The Computer Journal 57 (12) (2014) 1870–1892. `doi:10.1093/comjnl/bxt134`. *(pages 27, 28, 31, and 32)*

[19] I. Ben-Gal, Bayesian networks, Encyclopedia of Statistics in Quality and Reliability I. `doi:10.1002/9780470061572.eqr089`. *(page 11)*

[20] S. Bernardi, S. Donatelli, J. Merseguer, From UML sequence diagrams and statecharts to analysable petri net models,

	Modeling						Remarks
	Uncertainty	**BE Dependence**	**Temporal requirements**	**Repairable**	**Multi-state**	**BE Prob. distribution**	
DFT with repair boxes (Bobbio et al. [22])		+	+	+			Parametric, simple repair policy
Repairable FT (Codetta-Raiteri et al. [52])				+			Complex repair policies
Combined FT (Codetta-Raiteri [49, 51])		+	+	+			Dynamic, complex repair policies
Nondeterministic RFT (Beccuti et al. [16, 17, 18])				+			Allows nondeterministic repair choices
FT with Attack Tree (Fovino et al. [80])						+	Models deliberate attacks
Fuzzy FT (Tanaka et al. [171])	+						Models uncertain BE prob.
Fuzzy FT (Singer [163])	+					+	Special membership functions
Fuzzy FT (Lin et al. [113])	+						Linguistic description
Fuzzy FT (Kim et al. [104])	+						Multiple expert estimates
Fuzzy FT (Shu et al. [162, 111])	+					+	Uncertain membership functions
Fuzzy FT (Ren and Kong [152])	+				+	+	Multi-state BEs
Fuzzy FT (El-Iraki and Odoom [69])	+			+		+	Uncertain repair rates
Fuzzy FT (Huang et al. [94])	+					+	For large uncertainties
Fuzzy FT (Page and Perry [135])	+					+	Normally distributed rates
Extended FT (Buchacker [38, 39])		+		+	+		Multi-state BEs
Multistate FT (Zang et al. [191])		+			+		Multi-state FT
FT with mutual exclusion (Twigg et al. [173])		+					Mutually exclusive events
FT with dependent events (Vaurio [175])		+					Stat. dependent events
BDMP (Bouissou [33, 34, 35])[a]		+	+	+	+	+	Complex dependencies
FT with AND-THEN (Wijayarathna et al. [186])			+				Requirement of immediacy
DFT with simultaneity (Walker et al. [179, 180])[b]			+				Requirement of simultaneity
Formal FT Semantics (Schellhorn et al. [159])	+		+	+			Includes delays
FT with Duration Calculus (Gluchowski [84])	+		+	+			Complex temp. requirements
Temporal FT (Pakshikar [136])			+	+			Includes temporal logic
State-Event FT (Kaiser et al. [101, 100])		+	+	+	+		Combines FT and FSM

Table 7: Comparison of fault tree extensions

[a] Analysis tool: BDMP [32]

[b] Analysis tool: Pandora [180]

	Measures				
	Cut sets	Reliability	Availability	Other	Methods
DFT with repair boxes (Bobbio et al. [22])		+	+	+	[22]
Repairable FT (Codetta-Raiteri et al. [52])		+	+	+	[52]
Combined FT (Codetta-Raiteri [49, 51])		+	+	+	[51]
Nondeterministic RFT (Beccuti et al. [16, 17, 18])		+	+	+	[17, 18]
FT with Attack Tree (Fovino et al. [80])	+	+			[80]
Fuzzy FT (Tanaka et al. [171])	+	+			[171, 184, 88]
Fuzzy FT (Singer [163])	+	+			[163]
Fuzzy FT (Lin et al. [113])	+			+	[113]
Fuzzy FT (Kim et al. [104])	+	+			[104]
Fuzzy FT (Shu et al. [162, 111])	+	+			[162, 111]
Fuzzy FT (Ren and Kong [152])	+	+			[152]
Fuzzy FT (El-Iraki and Odoom [69])	+	+			[69]
Fuzzy FT (Huang et al. [94])	+	+			[94]
Fuzzy FT (Page and Perry [135])	+	+			[135]
Extended FT (Buchacker [38, 39])		+	+	+	[39]
Multistate FT (Zang et al. [191])	+	+			[191]
FT with mutual exclusion (Twigg et al. [173])	+	+			[173]
FT with dependent events (Vaurio [175])			+		[175]
BDMP (Bouissou [33, 34, 35])		+	+	+	[35, 34]
FT with AND-THEN (Wijayarathna et al. [186])	+				[186]
DFT with simultaneity (Walker er al. [179, 180])	+				[179]
Formal FT Semantics (Schellhorn et al. [159])					[159]
FT with Duration Calculus (Gluchowski [84])				+	[84]
Temporal FT (Pakshikar [136])	+				[136]
Combined FT (Codetta-Raiteri [51])		+	+		[51]
State-Event FT (Kaiser et al. [101, 100])		+	+		[188, 101]

Table 8: Analysis and tool support for fault tree extensions

in: Proc. 3rd Int. Workshop on Software and Performance (WOSP), 2002, pp. 35 – 45. doi:10.1145/584369.584376. *(page 3)*

[21] Z. W. Birnbaum, On the importance of different components in a multicomponent system, Tech. rep., Department of Mathematics, University of Washington (1968). *(page 15)*

[22] A. Bobbio, D. Codetta-Raiteri, Parametric fault trees with dynamic gates and repair boxes, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2004, pp. 459–465. doi:10.1109/RAMS.2004.1285491. *(pages 18, 27, 28, 31, and 32)*

[23] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, Improving the analysis of dependable systems by mapping fault trees into Bayesian networks, Reliability Engineering & System Safety 71 (3) (2001) 249–260. doi:10.1016/S0951-8320(00)00077-6. *(pages 9, 11, 22, and 25)*

[24] B. Bollig, I. Wegener, Improving the variable ordering of OBDDs is NP-complete, IEEE Trans. Comput. 45 (9) (1996) 993–1002. doi:10.1109/12.537122. *(page 7)*

[25] A. Bondavalli, I. Majzik, I. Mura, Automatic dependability analysis for supporting design decisions in UML, in: Proc. 4th Int. Symp. High Assurance Systems Engineering (HASE), 1999, pp. 64–71. doi:10.1109/HASE.1999.809476. *(page 3)*

[26] H. Boudali, A. P. Nijmeijer, M. I. A. Stoelinga, DFTSim: A simulation tool for extended dynamic fault trees, in: Proc. 42nd Annual Simulation Symposium (ANSS), San Diego, California, USA, 2009. *(pages 23 and 24)*

[27] H. Boudali, P. Crouzen, M. Stoelinga, A compositional semantics for dynamic fault trees in terms of interactive Markov chains, in: Proc. 5th Int. Conf. Automated Technology for Verification and Analysis (ATVA), Vol. 4762 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 441–456. doi:10.1007/978-3-540-75596-8_31. *(pages 18, 19, 21, and 24)*

[28] H. Boudali, P. Crouzen, M. Stoelinga, CORAL - a tool for compositional reliability and availability analysis, in: ARTIST workshop, presented at the 19th Int. Conf. Computer Aided Verification, 2007. *(page 24)*

[29] H. Boudali, P. Crouzen, M. Stoelinga, Dynamic fault tree analysis using input/output interactive Markov chains, in: Proc. 37th Int. Conf. Dependable Systems and Networks (DSN), IEEE, 2007, pp. 708–717. doi:10.1109/DSN.2007.37. *(pages 17 and 21)*

[30] H. Boudali, P. Crouzen, M. Stoelinga, A rigorous, compositional, and extensible framework for dynamic fault tree analysis, IEEE Trans. Dependable Secure Comput. 7 (2) (2010) 128–143. doi:10.1109/TDSC.2009.45. *(page 17)*

[31] H. Boudali, J. B. Dugan, A new Bayesian network approach to solve dynamic fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), 2005, pp. 451–456. doi:10.1109/RAMS.2005.1408404. *(pages 22 and 24)*

[32] M. Bouissou, BDMP knowledge base for KB3, http://sourceforge.net/projects/visualfigaro/files/Doc_and_examples/English/ (2012). *(page 31)*

[33] M. Bouissou, Boolean logic Driven Markov Processes: a powerful new formalism for specifying and solving very large Markov models, in: Proc. 6th Int. Conf. Probabilistic Safety Assessment and Management (PSAM), San Juan, Puerto Rico, USA, 2002. *(pages 27, 31, and 32)*

[34] M. Bouissou, A generalization of dynamic fault trees through boolean logic driven Markov processes (BDMP)®, in: Proc. 16th European Safety and Reliability Conf. (ESREL), Stavanger, Norway, 2007. *(pages 27, 31, and 32)*

[35] M. Bouissou, BDMP (Boolean logic Driven Markov Processes)® as an alternative to Event Trees, in: Proc. European Safety and Reliability Conf. (ESREL), 2008. *(pages 27, 31, and 32)*

[36] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, M. Roveri, Safety, dependability and performance analysis of extended AADL models, The Computer Journal 54 (2011) 754–775. doi:10.1093/comjnl/bxq024. *(pages 2 and 3)*

[37] M. Bozzano, A. Villafiorita, Design and Safety Assessment of Critical Systems, CRC Press, 2010. *(page 2)*

[38] K. Buchacker, Combining fault trees and Petri nets to model safety-critical systems, in: Proc. High Performance Computing Symposium (HPC), The Society for Computer Simulation International, 1999, pp. 439–444. *(pages 26, 31, and 32)*

[39] K. Buchacker, Modeling with extended fault trees, in: Proc. 5th Int. Symp. High Assurance Systems Engineering (HASE), 2000, pp. 238–246. doi:10.1109/HASE.2000.895468. *(pages 26, 31, and 32)*

[40] J. A. Carrasco, V. Suñé, An algorithm to find minimal cuts of coherent fault-trees with event-classes using a decision tree, IEEE Trans. Rel. 48 (1999) 31–41. doi:10.1109/24.765925. *(page 7)*

[41] C. Carreras, I. D. Walker, Interval methods for fault-tree analysis in robotics, IEEE Trans. Rel. 50 (2001) 3–11. doi:10.1109/24.935010. *(page 26)*

[42] Z. Chaochen, C. A. R. Hoare, A. P. Ravn, A calculus of durations, Information Processing Letters 40 (5) (1991) 269–276. doi:10.1016/0020-0190(91)90122-X. *(page 28)*

[43] P.-Y. Chaux, J.-M. Roussel, J.-J. Lesage, G. Deleuze, M. Bouissou, Qualitative analysis of a BDMP by finite automaton, in: Proc. European Safety and Reliability Conf. (ESREL), 2011, pp. 2050–2057. *(page 27)*

[44] P.-Y. Chaux, J.-M. Roussel, J.-J. Lesage, G. Deleuze, M. Bouissou, Systematic extraction of minimal cut sequences from a BDMP model, in: Proc. European Safety and Reliability Conf. (ESREL), Vol. 4, 2012, pp. 3344–3351. *(page 27)*

[45] P.-Y. Chaux, J.-M. Roussel, J.-J. Lesage, G. Deleuze, M. Bouissou, Towards a unified definition of minimal cut sequences, in: Proc. 4th IFAC Workshop on Dependable Control of Discrete Systems (DCDS), Vol. 4, 2013, pp. 1–6. doi:10.3182/20130904-3-UK-4041.00013. *(pages 20 and 25)*

[46] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, Stochastic well-formed colored nets and symmetric modeling applications, IEEE Trans. Comput. 42 (1993) 1343–1360. *(page 21)*

[47] E. M. Clarke, O. Grumberg, D. Peled, Model checking, MIT Press, 1999. *(page 6)*

[48] D. Codetta-Raiteri, The conversion of dynamic fault trees to stochastic petri nets, as a case of graph transformation, in: Proc. Workshop on Petri Nets and Graph Transformations (PNGT 2004), Vol. 127, 2005, pp. 45 – 60. doi:10.1016/j.entcs.2005.02.005. *(pages 21 and 24)*

[49] D. Codetta-Raiteri, Extended fault trees analysis supported by stochastic petri nets, Ph.D. thesis, Università degli Studi di Torino (2005). *(pages 28, 31, and 32)*

[50] D. Codetta-Raiteri, BDD based analysis of parametric fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2006, pp. 442–449. doi:10.1109/RAMS.2006.1677414. *(page 7)*

[51] D. Codetta-Raiteri, Integrating several formalisms in order to increase fault trees' modeling power, Reliability Engineering & System Safety 96 (5) (2011) 534–544. doi:10.1016/j.ress.2010.12.027. *(pages 18, 28, 31, and 32)*

[52] D. Codetta-Raiteri, G. Franceschinis, M. Iacono, V. Vittorini, Repairable fault tree for the automatic evaluation of repair policies, in: Proc. Int. Conf. Dependable Systems and Networks (DSN), IEEE, 2004, pp. 659–668. *(pages 5, 24, 27, 28, 31, and 32)*

[53] L. R. Consulting, RiskSpectrum, www.riskspectrum.com/en/risk. *(page 16)*

[54] S. Contini, V. Matuzas, New methods to determine the importance measures of initiating and enabling events in fault tree analysis, Reliability Engineering & System Safety 96 (7) (2011) 775–784. doi:10.1016/j.ress.2011.02.001. *(page 15)*

[55] O. Coudert, J. C. Madre, Fault tree analysis: 10^{20} Prime implicants and beyond, in: Proc. Reliability and Maintainability Symposium (RAMS), 1993, pp. 240–245. doi:10.1109/RAMS.1993.296849. *(pages 6 and 7)*

[56] O. Coudert, J. C. Madre, MetaPrime: An interactive fault-tree analyzer, IEEE Trans. Rel. 43 (1994) 121–127. doi:10.1109/24.285125. *(page 7)*

[57] P. A. Crosetti, Fault tree analysis with probability evaluation, IEEE Trans. Nucl. Sci. 18 (1) (1971) 465–471. doi:10.1109/TNS.1971.4325911. *(page 11)*

[58] L. de Alfaro, T. A. Henzinger, Interface automata, in: Proc. Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT Int. Symp. Foundations of Software Engineering, ACM Press, 2001, pp. 109–120. doi:10.1145/503209.503226. *(page 29)*

[59] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, P. G. Webster, The Möbius framework and its implementation, IEEE Trans. Softw. Eng. 28 (10) (2002) 956–969. doi:10.1109/TSE.2002.1041052. *(page 3)*

[60] A. L. Development, Fault Tree Analysis (FTA) Software, http://aldservice.com/en/reliability-products/fta.html. *(page 16)*

[61] S. Distefano, A. Puliafito, Dynamic reliability block diagrams: Overview of a methodology, in: Proc. European Safety and Reliability Conf. (ESREL), Vol. 7, 2007, pp. 1059–1068. *(pages 3 and 16)*

[62] J. B. Dugan, S. J. Bavuso, M. A. Boyd, Fault trees and sequence dependencies, in: Proc. 1990 Annual Reliability and MaintainabilitySymp., IEEE, 1990, pp. 286–293. doi:10.1109/ARMS.1990.67971. *(pages 17, 18, 20, and 24)*

[63] J. B. Dugan, S. J. Bavuso, M. A. Boyd, Dynamic fault-tree models for fault-tolerant computer systems, IEEE Trans. Rel. (1992) 363–377doi:10.1109/24.159800. *(page 20)*

[64] J. B. Dugan, B. Venkataraman, R. Gulati, DIFtree: A software package for the analysis of dynamic fault tree models, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 1997, pp. 64–70. doi:10.1109/RAMS.1997.571666. *(pages 7 and 24)*

[65] K. Durga Rao, V. Gopika, V. V. S. Sanyasi Rao, H. S. Kushwaha, A. K. Verma, A. Srividya, Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment, Reliability Engineering & System Safety 94 (4) (2009) 872–883. doi:10.1016/j.ress.2008.09.007. *(pages 9, 11, 13, 23, and 24)*

[66] Y. Dutuit, A. B. Rauzy, Efficient algorithms to assess component and gate importance in fault tree analysis, Reliability Engineering & System Safety 72 (2) (2001) 213–222. doi:10.1016/S0951-8320(01)00004-7. *(page 15)*

[67] Y. Dutuit, A. B. Rauzy, A linear-time algorithm to find modules of fault trees, IEEE Trans. Rel. 45 (1996) 422–425. doi:10.1109/24.537011. *(page 7)*

[68] Y. Dutuit, A. B. Rauzy, Approximate estimation of system reliability via fault trees, Reliability Engineering & System Safety 87 (2) (2005) 163–172. doi:10.1016/j.ress.2004.02.008. *(page 28)*

[69] A. El-Iraki, E. R. Odoom, Fuzzy probist reliability assessment of repairable systems, in: Proc. Conference of the North American Fuzzy Information Processing Society (NAFIPS), 1998, pp. 96–100. *(pages 25, 31, and 32)*

[70] EPRI, CAFTA, http://www.epri.com/abstracts/Pages/ProductAbstract.aspx?ProductId=000000000001015514. *(page 16)*

[71] C. A. Ericson, Fault Tree Analysis – a history, in: Proc. 17th International System Safety Conference, Orlando, Florida, USA, 1999, pp. 1–9. *(page 3)*

[72] B. Ern, V. Y. Nguyen, T. Noll, Characterization of failure effects on AADL models, in: Computer Safety, Reliability, and Security, Vol. 8153 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 241–252. doi:10.1007/978-3-642-40793-2_22. *(page 3)*

[73] U. D. o. T. Federal Aviation Administration, Reusable launch and reeintry vehicle system safery process (2005). *(page 2)*

[74] Federal Aviation Administration, U.S. Department of Transportation, FAA Order 8040.4: Safety Risk Management (1998). *(page 3)*

[75] Federal Aviation Administration, U.S. Department of Transportation, System Safety Handbook (2000). *(page 3)*

[76] F. Flammini, S. Marrone, M. Iacono, N. Mazzocca, V. Vittorini, A multiformalism modular approach to ERTMS/ETCS failure modelling, Int. J. Reliability, Quality and Safety Engineering 21. doi:10.1142/S0218539314500016. *(page 28)*

[77] F. Flammini, N. Mazzocca, M. Iacono, S. Marrone, Using repairable fault trees for the evaluation of design choices for critical repairable systems, in: Proc. Int. Symp. High Assurance Systems Engineering (HASE), IEEE, 2005, pp. 163–172. doi:10.1109/HASE.2005.26. *(page 27)*

[78] M. Förster, B. Kaiser, Increased efficiency in the quantitative evaluation of state/event fault trees, in: Information Control Problems in Manufacturing, Vol. 12 of Proc. 12th IFAC Symp., Elsevier Science Ltd, 2006, pp. 255–260. doi:10.3182/20060517-3-FR-2903.00143. *(page 29)*

[79] M. Forster, M. Trapp, Fault tree analysis of software-controlled component systems based on second-order probabilities, in: Proc. 20th Int. Symp. on Software Reliability Engineering (ISSRE), IEEE, 2009, pp. 146–154. doi:10.1109/ISSRE.2009.22. *(page 26)*

[80] I. N. Fovino, M. Masera, A. D. Cian, Integrating cyber attacks within fault trees, Reliability Engineering & System Safety 94 (9) (2009) 1394–1402. doi:10.1016/j.ress.2009.02.020. *(pages 29, 31, and 32)*

[81] H. Furuta, N. Shiraishi, Fuzzy importance in fault tree analysis, Fuzzy Sets and Systems 12 (3) (1984) 205–213. doi:10.1016/0165-0114(84)90068-X. *(page 26)*

[82] J. B. Fussell, How to hand-calculate system reliability and safety characteristics, IEEE Trans. Rel. R-24 (3) (1975) 169–174. doi:10.1109/TR.1975.5215142. *(page 15)*

[83] J. B. Fussell, E. B. Henry, N. H. Marshall, MOCUS: a computer program to obtain minimal sets from fault trees, Tech. rep., Aerojet Nuclear Co., Idaho Falls (1974). *(page 7)*

[84] P. Gluchowski, Duration calculus for analysis of fault trees with time dependencies, in: Proc. 2nd Int. Conf. on Dependability of Computer Systems (DepCoS-RELCOMEX), 2007, pp. 107–114. doi:10.1109/DEPCOS-RELCOMEX.2007.19. *(pages 28, 31, and 32)*

[85] A. Goyal, W. C. Carter, E. de Souza e Silva, S. S. Lavenberg, K. S. Trivedi, The system availability estimator, in: Proc. 25th Int. Symp. Fault-Tolerant Computing (FTCS), Highlights from Twenty-Five Years, 1995, pp. 182–187. doi:10.1109/FTCSH.1995.532632. *(page 3)*

[86] T. L. Graves, M. S. Hamada, R. Klamann, A. Koehler, H. F. Martz, A fully Bayesian approach for combining multi-level information in multi-state fault tree quantification, Reliability Engineering & System Safety 92 (10) (2007) 1476–1483. doi:10.1016/j.ress.2006.11.001. *(pages 22 and 24)*

[87] D. Guck, J.-P. Katoen, M. Stoelinga, T. Luiten, J. Romijn, Smart railroad maintenance engineering with stochastic model checking, in: Proc. 2nd Int. Conf. Railway Technology: Reseach, Development and Maintenance (Railways), Saxe-Coburg Publications, Ajaccio, Corsica, France, 2014. doi:10.4203/ccp.104.299. *(pages 18 and 21)*

[88] A. C. F. Guimarẽes, N. F. F. Ebecken, FuzzyFTA: A fuzzy fault tree system for uncertainty analysis, Annals of Nuclear Energy 26 (6) (1999) 523–532. doi:10.1016/S0306-4549(98)00070-X. *(pages 26 and 32)*

[89] W. Han, W. Guo, Z. Hou, Research on the method of dynamic fault tree analysis, in: Proc. 9th Int. Conf. Reliability, Maintainability and Safety (ICRMS), IEEE, 2011, pp. 950–953. doi:10.1109/ICRMS.2011.5979422. *(pages 21 and 24)*

[90] M. Hecht, A. Lam, C. Vogl, C. Dimpfl, A tool set for generation of failure modes and effects analyses from AADL models, in: Presentation at Systems and Software Technology Conference 2012, 2012. *(page 3)*

[91] H. Hermanns, Interactive Markov chains: and the quest for quantified quality, Springer-Verlag Berlin, 2002. *(page 21)*

[92] J. S. Hong, C. H. Lie, Joint reliability-importance of two edges in an undirected network, IEEE Trans. Rel. 42 (1993) 17–23.

doi:10.1109/24.210266. *(page 15)*

[93] P. Hoogerkamp, Praktijkgids Risicobeoordeling Machinerichtlijn, Nederlands Normalisatie-instituur (2010). *(page 3)*

[94] H.-Z. Huang, X. Tong, M. J. Zuo, Posbist fault tree analysis of coherent systems, Reliability Engineering & System Safety 84 (2) (2004) 141–148. doi:10.1016/j.ress.2003.11.002. *(pages 25, 31, and 32)*

[95] S. ichi Minato, Zero-suppressed BDDs for set manipulation in combinatorial problems, in: Proc. 30th ACM/IEEE Design Automation Conf., ACM New York, 1993, pp. 272–277. doi:10.1145/157485.164890. *(page 20)*

[96] IEC 61025: Fault tree analysis (2006). *(page 3)*

[97] Isograph, FaultTree+, www.isograph.com/software/reliability-workbench/fault-tree-analysis/. *(page 16)*

[98] P. S. Jackson, On the s-importance of elements and prime implicants of non-coherent systems, IEEE Trans. Rel. R-32 (1) (1983) 21–25. doi:10.1109/TR.1983.5221464. *(page 15)*

[99] M. A. Johnson, M. R. Taaffe, The denseness of phase distributions, Tech. rep., School of Industrial Engineering Research Memoranda 88-20, Purdue University (1988). *(page 16)*

[100] B. Kaiser, Extending the expressive power of fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2005, pp. 468–474. doi:10.1109/RAMS.2005.1408407. *(pages 29, 31, and 32)*

[101] B. Kaiser, C. Gramlich, State-event-fault-trees - a safety analysis model for software controlled systems, in: Computer Safety, Reliability, and Security, Vol. 3219 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 195–209. doi:10.1007/978-3-540-30138-7_17. *(pages 29, 31, and 32)*

[102] B. Kaiser, C. Gramlich, M. Förster, State/event fault trees – a safety analysis model for software-controlled systems, Reliability Engineering & System Safety 92 (11) (2007) 1521–1537. doi:10.1016/j.ress.2006.10.010. *(page 29)*

[103] C. Kara-Zaitri, E. Ever, A hardware accelerated semi analytic approach for fault trees with repairable components, in: Proc. 11th Int. Conf. Computer Modelling and Simulation (UKSIM), IEEE, 2009, pp. 146–151. doi:10.1109/UKSIM.2009.83. *(page 28)*

[104] C. Kim, Y. Ju, M. Gens, Multilevel fault tree analysis using fuzzy numbers, Computers & Operations Research 23 (7) (1996) 695–703. doi:10.1016/0305-0548(95)00070-4. *(pages 25, 31, and 32)*

[105] T. A. Kletz, Hazop and Hazan, CRC Press, 1999. *(page 3)*

[106] B. Kordy, L. Piètre-Cambacèdés, P. Schweitzer, Dag-based attack and defense modeling: Don't miss the forest for the attack trees, Computer Science Review 13–14 (2014) 1–38. doi:10.1016/j.cosrev.2014.07.001. *(page 29)*

[107] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: Computer Aided Verification, Vol. 6806 of LNCS, Springer Berlin Heidelberg, 2011, pp. 585–591. doi:10.1007/978-3-642-22110-1_47. *(page 20)*

[108] M. Lampis, Application of bayesian belief networks to system fault diagnostics, Ph.D. thesis, Loughborough University (2010). *(page 11)*

[109] H. Langseth, L. Portinale, Bayesian networks in reliability, Reliability Engineering & System Safety 92 (1) (2007) 92 – 108. doi:doi:10.1016/j.ress.2005.11.037. *(page 11)*

[110] W.-S. Lee, D. L. Grosh, F. A. Tillman, C. H. Lie, Fault tree analysis, methods, and applications — A review, IEEE Trans. Rel. R-34 (3) (1985) 194–203. doi:10.1109/TR.1985.5222114. *(pages 4, 5, and 7)*

[111] D.-F. Li, A note on "using intuitionistic fuzzy sets for fault-tree analysis on printed circuit board assembly", Microelectronics Reliability 48 (10) (2008) 1741. doi:10.1016/j.microrel.2008.07.059. *(pages 25, 31, and 32)*

[112] X. Liang, H. Yi, Y. Zhang, D. Li, A numerical simulation approach for reliability analysis of fault-tolerant repairable system, in: Proc. 8th Int. Conf. Reliability, Maintainability and Safety (ICRMS), IEEE, 2009, pp. 191–196. doi:10.1109/ICRMS.2009.5270210. *(pages 23 and 24)*

[113] C.-T. Lin, M.-J. J. Wang, Hybrid fault tree analysis using fuzzy sets, Reliability Engineering & System Safety 58 (3) (1997) 205–213. doi:10.1016/S0951-8320(97)00072-0. *(pages 25, 31, and 32)*

[114] D. Liu, W. Xing, C. Zhang, R. Li, H. Li, Cut sequence set generation for fault tree analysis, in: Embedded Software and Systems, Vol. 4523 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 592–603. doi:10.1007/978-3-540-72685-2_55. *(pages 20 and 25)*

[115] D. Liu, L. Xiong, Z. Li, P. Wang, H. Zhang, The simplification of cut sequence set analysis for dynamic systems, in: Proc. 2nd Int. Conf. Computer and Automation Engineering (ICCAE), Vol. 3, 2010, pp. 140–144. doi:10.1109/ICCAE.2010.5451831. *(pages 21 and 24)*

[116] D. Liu, C. Zhang, W. Xing, R. Li, H. Li, Quantification of cut sequence set for fault tree analysis, in: High Performance Computing and Communications, Vol. 4782 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 755–765. doi:10.1007/978-3-540-75444-2_70. *(pages 20 and 25)*

[117] W. Long, Y. Sato, M. Horigome, Quantification of sequential failure logic for fault tree analysis, Reliability Engineering & System Safety 67 (3) (2000) 269–274. doi:10.1016/S0951-8320(99)00075-7. *(pages 21 and 24)*

[118] L. Lu, J. Jiang, Joint failure importance for noncoherent fault trees, IEEE Trans. Rel. (2007) 435–443doi:10.1109/TR.2007.898574. *(page 15)*

[119] Y. A. Mahmood, A. Ahmadi, A. K. Verma, A. Srividya, U. Kumar, Fuzzy fault tree analysis: a review of concept and application, Int. J. System Assurance Engineering and Management 4 (1) (2013) 19–32. doi:10.1007/s13198-013-0145-x. *(page 25)*

[120] M. Malhotra, K. S. Trivedi, Dependability modeling using Petri-nets, IEEE Trans. Rel. 44 (3) (1995) 428–440. doi:10.1109/24.406578. *(page 5)*

[121] G. Merle, Algebraic modelling of dynamic fault trees, contribution to qualitative and quantitative analysis, Ph.D. thesis, École normale supérieure de Cachan (2010). *(pages 20 and 25)*

[122] G. Merle, J.-M. Roussel, Algebraic modelling of fault trees with priority AND gates, in: Proc. 1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS), 2007, pp. 175–180. *(page 20)*

[123] G. Merle, J.-M. Roussel, J.-J. Lesage, Dynamic fault tree analysis based on the structure function, in: Proc. Reliability and Maintainability Symposium (RAMS), 2011, pp. 1–6. doi:10.1109/RAMS.2011.5754452. *(page 20)*

[124] G. Merle, J.-M. Roussel, J.-J. Lesage, A. Bobbio, Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events, IEEE Trans. Rel. 59 (1) (2010) 250–261. doi:10.1109/TR.2009.2035793. *(page 20)*

[125] Z. Miao, R. Niu, T. Tang, J. Liu, A new generation algorithm of fault tree minimal cut sets and its application in CBTC system, in: Proc. Int. Conf. Intelligent Rail Transportation (ICIRT), IEEE, 2013, pp. 221–226. doi:10.1109/ICIRT.2013.6696297. *(page 7)*

[126] Y. Mo, A multiple-valued decision-diagram-based approach to solve dynamic fault trees, IEEE Trans. Rel. 63 (1) (2014) 81–93. doi:10.1109/TR.2014.2299674. *(pages 22 and 24)*

[127] M. Modarres, M. Kaminskiy, V. Krivtsov, Reliability Engineering and Risk Analysis, CRC Press, 2009. *(page 3)*

[128] S. Montani, L. Portinale, A. Bobbio, Dynamic Bayesian networks for modeling advanced fault tree features in dependability analysis, in: Proc. European Safety and Reliability Conf. (ESREL), 2005, pp. 1415–1422. *(pages 22 and 24)*

[129] S. Montani, L. Portinale, A. Bobbio, C. Codetta-Raiteri, Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks, Reliability Engineering & System Safety 93 (7) (2008) 922–932. doi:10.1016/j.ress.2007.03.013. *(page 24)*

[130] S. Montani, L. Portinale, A. Bobbio, M. Varesio, D. Codetta-

Raiteri, DBNet, a tool to convert dynamic fault trees into dynamic Bayesian networks, Tech. rep., Dip. di Informatica, Univ. del Piemonte Orientale (Aug. 2005). *(page 9)*

[131] J. Ni, W. Tang, Y. Xing, A simple algebra for fault tree analysis of static and dynamic systems, IEEE Trans. Rel. 62 (2013) 846–861. doi:10.1109/TR.2013.2285035. *(pages 22 and 24)*

[132] Occupational Safety and Health Administration, U.S. Department of Labor, OSHA 3133: Process Safety Management Guidelines for Compliance (1994). *(page 3)*

[133] OpenFTA, www.openfta.com/. *(page 16)*

[134] Y. Ou, J. B. Dugan, Sensitivity analysis of modular dynamic fault trees, in: Proc. IEEE Int. Computer Performance and DependabilySymp. (IPDS), 2000, pp. 35–43. doi:10.1109/IPDS.2000.839462. *(page 23)*

[135] L. B. Page, J. E. Perry, Standard deviation as an alternative to fuzziness in fault tree models, IEEE Trans. Rel. 43 (3) (1994) 402–407. doi:10.1109/24.326434. *(pages 25, 31, and 32)*

[136] G. K. Palshikar, Temporal fault trees, Information and Software Technology 44 (3) (2002) 137–150. doi:10.1016/S0950-5849(01)00223-3. *(pages 28, 31, and 32)*

[137] P. K. Pande, M. E. Spector, P. Chatterjee, Computerized fault tree analysis: TREEL and MICSUP, Tech. rep., Operation Research Centre, University of California, Berkeley (1975). *(page 7)*

[138] P. Popic, D. Desovski, W. Abdelmoez, B. Cukic, Error propagation in the reliability analysis of component based systems, in: Proc. 16th Int. Symp. on Software Reliability Engineering (ISSRE), 2005, pp. 52 – 62. doi:10.1109/ISSRE.2005.18. *(page 3)*

[139] L. Portinale, A. Bobbio, D. Codetta-Raiteri, S. Montani, Compiling dynamic fault trees into dynamic Bayesian nets for reliability analysis: the RADYBAN tool, in: Proc. 5th UAI Bayesian Modeling Applications Workshop (UAI-AW), 2007. *(page 24)*

[140] L. Portinale, D. Codetta-Raiteri, S. Montani, Supporting reliability engineers in exploiting the power of dynamic Bayesian networks, International Journal of Approximate Reasoning 51 (2) (2010) 179–195. doi:10.1016/j.ijar.2009.05.009. *(pages 18, 22, and 28)*

[141] PTC, Windchill FTA, www.ptc.com/product/relex/fault-tree. *(page 16)*

[142] L. L. Pullum, J. B. Dugan, Fault tree models for the analysis of complex computer-based systems, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 1996, pp. 200–207. doi:10.1109/RAMS.1996.500663. *(pages 21 and 24)*

[143] A. Rajabzadeh, M. S. Jahangiry, Hardware-based reliability tree (HRT) for fault tree analysis, in: Proc. 15th CSI Int. Symp. Computer Architecture and Digital Systems (CADS), IEEE, 2010, pp. 171–172. doi:10.1109/CADS.2010.5623587. *(pages 23 and 24)*

[144] M. Rausand, A. Hoylan, System Reliability Theory. Models, Statistical Methods, and Applications, Wiley series in probability and statistics, Wiley, 2004. *(page 2)*

[145] A. Rauzy, Binary decision diagrams for reliability studies, Handbook of Performability Engineering (2008) 381–396doi:10.1007/978-1-84800-131-2_25. *(pages 10 and 11)*

[146] A. Rauzy, Y. Dutuit, Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within Aralia, Reliability Engineering & System Safety 58 (2) (1997) 127–144. doi:10.1016/S0951-8320(97)00034-3. *(page 7)*

[147] A. B. Rauzy, New algorithms for fault tree analysis, Reliability Engineering & System Safety 40 (3) (1993) 203–211. doi:10.1016/0951-8320(93)90060-C. *(pages 7, 9, and 10)*

[148] A. B. Rauzy, Sequence algebra, sequence decision diagrams and dynamic fault trees, Reliability Engineering & System Safety 96 (7) (2011) 785–792. doi:10.1016/j.ress.2011.02.005. *(pages 20 and 25)*

[149] ReliaSoft, BlockSim, www.reliasoft.com/BlockSim/index.html. *(page 16)*

[150] R. Remenyte, J. D. Andrews, A simple component connection approach for fault tree conversion to binary decision diagram, in: Proc. 1st Int. Conf. Availability, Reliability and Security (ARES), 2006, pp. 449–456. doi:10.1109/ARES.2006.17. *(page 7)*

[151] R. Remenyte-Prescott, J. Andrews, An enhanced component connection method for conversion of fault trees to binary decision diagrams, Reliability Engineering & System Safety 93 (10) (2008) 1543–1550. doi:10.1016/j.ress.2007.09.001. *(page 7)*

[152] Y. Ren, L. Kong, Fuzzy multi-state fault tree analysis based on fuzzy expert system, in: Proc. 9th Int. Conf. Reliability, Maintainability and Safety (ICRMS), IEEE, 2011, pp. 920–925. doi:10.1109/ICRMS.2011.5979415. *(pages 25, 26, 31, and 32)*

[153] D. Rongxing, W. Guochun, D. Decun, A new assessment method for system reliability based on dynamic fault tree, in: Proc. Int. Conf. Intelligent Computation Technology and Automation (ICICTA), IEEE, IEEE, 2010, pp. 219–222. doi:10.1109/ICICTA.2010.237. *(pages 22 and 24)*

[154] D. E. Ross, K. M. Butler, M. R. Mercer, Exact ordered binary decision diagram size when representing classes of symmetric functions, Journal of Electronic Testing 2 (3) (1991) 243–259. doi:10.1007/BF00135441. *(page 7)*

[155] M. Roth, P. Liggesmeyer, Qualitative analysis of state/event fault trees for supporting the certification process of software-intensive systems, in: Proc. Int. Symp. on Software Reliability Engineering Workshops(ISSREW), 2013, pp. 353–358. doi:10.1109/ISSREW.2013.6688920. *(page 29)*

[156] J. Rumbaugh, I. Jabobson, G. Booch, Unified Modeling Language Referance manual, The (2nd Edition), Pearson Higher Education, 2004. *(page 3)*

[157] A. M. Rushdi, Uncertainty analysis of fault-tree outputs, IEEE Trans. Rel. R-34 (5) (1985) 458–462. doi:10.1109/TR.1985.5222232. *(page 14)*

[158] W. H. Sanders, T. Courtney, D. Deavours, D. Daly, S. Derisavi, V. Lam, Multi-formalism and multi-solution-method modeling frameworks: The Möbius approach, in: Proc. Symp. Performance Evaluation - Stories and Perspectives, Vienna, Austria, 2003, pp. 241–256. *(page 3)*

[159] G. Schellhorn, A. Thums, W. Reif, Formal fault tree semantics, in: Proc. 6th World Conf. on Integrated Design and Process Technology, 2002. *(pages 28, 31, and 32)*

[160] W. Schneeweiss, SyRePa'89–a package of programs for systems reliability evaluations, Tech. rep., Informatik-Rep. 91, Fern Universität (1990). *(page 9)*

[161] W. G. Schneeweiss, On the polynomial form of boolean functions: Derivations and applications, IEEE Trans. Comput. 47 (1998) 217–221. doi:10.1109/12.663768. *(pages 9 and 14)*

[162] M.-H. Shu, C.-H. Cheng, J.-R. Chang, Using intuitionistic fuzzy sets for fault-tree analysis on printed circuit board assembly, Microelectronics Reliability 46 (12) (2006) 2139–2148. doi:10.1016/j.microrel.2006.01.007. *(pages 25, 31, and 32)*

[163] D. Singer, A fuzzy set approach to fault tree and reliability analysis, Fuzzy Sets and Systems 34 (2) (1990) 145–155. doi:10.1016/0165-0114(90)90154-X. *(pages 25, 31, and 32)*

[164] R. M. Sinnamon, J. D. Andrews, Fault tree analysis and binary decision diagrams, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 1996, pp. 215–222. doi:10.1109/RAMS.1996.500665. *(page 7)*

[165] Architecture, analysis and design language, aS5506 (2004). *(page 3)*

[166] I. Software, ITEM Toolkit: Fault Tree Analysis (FTA), www.itemsoft.com/fault_tree.html. *(page 16)*

[167] K. P. Soman, K. B. Misra, Fuzzy fault tree analysis using resolution identity, J Fuzzy Math 1 (1993) 193–212. *(page 26)*

[168] M. Stamatelatos, W. Vesely, J. B. Dugan, J. Fragola, J. Minarick, J. Railsback, Fault Tree Handbook with Aerospace Applications, Office of safety and mission assurance NASA headquarters, 2002. *(pages 2 and 10)*

[169] K. Stecher, Evaluation of large fault-trees with repeated events

using an efficient bottom-up algorithm, IEEE Trans. Rel. 35 (1986) 51–58. doi:10.1109/TR.1986.4335344. *(pages 9 and 11)*

[170] P. V. Suresh, A. K. Babar, V. V. Raj, Uncertainty in fault tree analysis: A fuzzy approach, Fuzzy Sets and Systems 83 (2) (1996) 135–141. doi:10.1016/0165-0114(95)00386-X. *(page 26)*

[171] H. Tanaka, L. Fan, F. Lai, K. Toguchi, Fault-tree analysis by fuzzy probability, IEEE Trans. Rel. 32 (5) (1983) 453–457. doi:10.1109/TR.1983.5221727. *(pages 23, 31, and 32)*

[172] Z. Tang, J. B. Dugan, Minimal cut set/sequence generation for dynamic fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2004, pp. 207–213. doi:10.1109/RAMS.2004.1285449. *(pages 7, 19, and 25)*

[173] D. W. Twigg, A. V. Ramesh, U. R. Sandadi, T. C. Sharma, Modeling mutually exclusive events in fault trees, in: Proc. Reliability and Maintainability Symposium (RAMS), IEEE, 2000, pp. 8 – 13. doi:10.1109/RAMS.2000.816276. *(pages 27, 31, and 32)*

[174] U.S. Department of Defense, Procedures for performing a failure mode, effects and criticality analysis (MIL-P-1629A) (1949). *(page 2)*

[175] J. K. Vaurio, Treatment of general dependencies in system fault-tree and risk analysis, IEEE Trans. Rel. 51 (2002) 278–287. doi:10.1109/TR.2002.801848. *(pages 27, 31, and 32)*

[176] W. E. Vesely, A time-dependent methodology for fault tree evaluation, Nuclear Engineering and Design 13 (2) (1970) 337–360. doi:10.1016/0029-5493(70)90167-6. *(pages 14 and 15)*

[177] W. E. Vesely, F. F. Goldberg, N. H. Roberts, D. F. Haasl, Fault Tree Handbook, Office of Nuclear Regulatory Reasearch, U.S. Nuclear Regulatory Commision, 1981. *(pages 3, 4, 6, 7, 9, and 14)*

[178] W. E. Vesely, R. E. Narum, PREP and KITT: computer codes for the automatic evaluation of a fault tree, Tech. rep., Idaho Nuclear Corp., Idaho Falls (1970). *(pages 7, 9, and 11)*

[179] M. Walker, L. Bottaci, Y. Papadopoulos, Compositional temporal fault tree analysis, in: Computer Safety, Reliability, and Security, Vol. 4680 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 106–119. doi:10.1007/978-3-540-75101-4_12. *(pages 28, 31, and 32)*

[180] M. Walker, Y. Papadopoulos, Qualitative temporal analysis: Towards a full implementation of the Fault Tree Handbook, Control Engineering Practice 17 (10) (2009) 1115–1125. doi:10.1016/j.conengprac.2008.10.003. *(pages 18, 28, 31, and 32)*

[181] M. Walker, Y. Papadopoulos, A hierarchical method for the reduction of temporal expressions in pandora, in: Proc. First Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS), ACM New York, 2010, pp. 7–12. doi:10.1145/1772630.1772634. *(page 28)*

[182] M. Walter, M. Siegle, A. Bode, Opensesame: the simple but extensive, structured availability modeling environment, Reliability Engineering & System Safety 93 (6) (2007) 857–873. doi:10.1016/j.ress.2007.03.034. *(page 3)*

[183] Y. F. Wang, M. Xie, K. M. Ng, Y. F. Meng, Quantitative risk analysis model of integrating fuzzy fault tree with Bayesian network, in: Proc. Int. Conf. Intelligence and Security Informatics (ISI), IEEE, 2011, pp. 267–271. doi:10.1109/ISI.2011.5984095. *(page 26)*

[184] Y. Wang, M. Xie, Approach to integrate fuzzy fault tree with Bayesian network, Procedia Engineering 45 (2012) 131–138. doi:10.1016/j.proeng.2012.08.133. *(pages 26 and 32)*

[185] Y.-S. Way, D.-Y. Hsia, A simple component-connection method for building binary decision diagrams encoding a fault tree, Reliability Engineering & System Safety 70 (1) (2000) 59–70. doi:10.1016/S0951-8320(00)00048-X. *(page 7)*

[186] P. G. Wijayarathna, M. Maekawa, Extending fault trees with an AND-THEN gate, in: Proc. 11th Int. Symp. on Software Reliability Engineering (ISSRE), 2000, pp. 283–292. doi:10.1109/ISSRE.2000.885879. *(pages 28, 31, and 32)*

[187] J. Xiang, K. Yanoo, Y. Maeno, K. Tadano, F. Machida, A. Kobayashi, T. Osaki, Efficient analysis of fault trees with voting gates, in: Proc. 22nd Int. Symp. on Software Reliability Engineering (ISSRE), 2011, pp. 230–239. doi:10.1109/ISSRE.2011.23. *(page 7)*

[188] B. Xu, Z. Huang, J. Hu, O. Wei, Y. Zhou, Minimal cut sequence generation for state/event fault trees, in: Proc. 2013 Middleware Doctoral Symposium, ACM New York, 2013, p. Article No. 3. doi:10.1145/2541534.2541592. *(pages 29 and 32)*

[189] O. Yevkin, An improved modular approach for dynamic fault tree analysis, in: Proc. Reliability and Maintainability Symposium (RAMS), 2011, pp. 1–5. doi:10.1109/RAMS.2011.5754437. *(pages 22 and 24)*

[190] L. A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, Information Sciences 8 (3) (1975) 199–249. doi:10.1016/0020-0255(75)90036-5. *(page 26)*

[191] X. Zang, D. Wang, H. Sun, K. S. Trivedi, A BDD-based algorithm for analysis of multistate systems with multistate components, IEEE Trans. Comput. 52 (12) (2003) 1608–1618. doi:10.1109/TC.2003.1252856. *(pages 26, 31, and 32)*

[192] H.-L. Zhang, C.-Y. Zhang, D. Liu, R. Li, A method of quantitative analysis for dynamic fault tree, in: Proc. Reliability and Maintainability Symposium (RAMS), 2011, pp. 1–6. doi:10.1109/RAMS.2011.5754471. *(pages 20 and 25)*

[193] X. Zhang, Q. Miao, X. Fan, D. Wang, Dynamic fault tree analysis based on Petri nets, in: Proc. 8th Int. Conf. Reliability, Maintainability and Safety (ICRMS), IEEE, IEEE, 2009, pp. 138–142. doi:10.1109/ICRMS.2009.5270223. *(pages 23 and 24)*

Appendix A. Glossary and notation

Mathematical notations

- $\emptyset$: Empty set
- $\mathcal{P}(X)$: Power set of X
- $\mathbb{P}(X)$: Probability of X
- $\mathbb{E}(X)$: Expected value of X
- BE: Set of BEs
- G: Set of gates
- E: Set of elements ($BE \cup G$)
- $T(g)$: Type of gate g
- $I(g)$: Inputs of gate g
- $Re(F)$: Reliability of F.

Definition of terms and abbreviations

Availability Fraction of time a system is in a functioning state.

BE Basic Event; leaf node of an FT, typically denoting a component or a specific failure mode of one component.

BDD Binary Decision Diagram

Coherent system System where the failure of a component never prevents a system failure.

CCF Common Cause Failure; event where a single cause results in multiple BEs failing.

Cut Set Set of BEs such that, if all events in a cut set occur, the top event will occur.

DAG Directed Acyclic Graph.

DFT Dynamic Fault Tree; FT with additional gates for dynamic behaviour.

FT Fault Tree; graphical model describing failure propagation behaviour through a system.

FTA Fault Tree Analysis; the computation of measures of interest from an FT.

Gate Intermediate node in an FT, describing how failures of its children combine.

Intermediate Event Event caused by one or more BEs, see also ‘Gate’.

MCS Minimal Cut Set.

MTBF Mean Time Between Failures.

MTTF Mean Time To Failure.

MTTFF Mean TIme To First Failure.

MTTR Mean TIme To Repair.

RB See *R*epair Box

SFT Static (or Standard) Fault Tree; fault tree with only boolean gates.

Shared subtree Element of a fault tree, together with its descendants, which is an input to multiple gates.

TE Top Event; root node of an FT, representing the failure of the system being analyzed.