Proceedings, 16th International ISSAT on Reliability and Quality in Design, 5-7 August, 2010, International Society of Science and Applied Technologies, 324-328.

Formalizing Fault Trees for Remote Ocean Systems

John C. Sloan
Taghi M. Khoshgoftaar
Computer and Electrical Engineering and Computer Science
Florida Atlantic University
777 Glades Rd., Boca Raton, FL 33431
Email: taghi@cse.fau.edu

Howard Hanson
Center for Ocean Energy Technology
Florida Atlantic University
777 Glades Rd., Boca Raton, FL 33431
Email: hphanson@fau.edu

Keywords: fault trees, condition-based monitoring, ocean systems, satisfiability, BDD.

***Abstract*—Real time condition-based evaluation of system health must not only be efficient, but also produce usable and expressive results. To this end, this paper presents a set-theoretic formulation of fault trees. Such a formulation provides a usable scaffolding on which ensembles of machine health measurement techniques may eventually operate. Initially, we present a negation-free formulation of a forest of fault trees as a set of 2-level sum-of-products expressions. Given this formulation, we express measures for *certainty* and *specificity*, both of which further qualify the various well-studied measures for *severity*. This formulation is subsequently refined to represent multi-state systems, non-coherent valuations, and node sharing – all necessary for practical monitoring solutions. Finally, we present an evaluation rule embodying these refinements and analyze its complexity. Examples pertaining to unattended ocean systems illustrate these concepts.**

I. Introduction

Fault tree analysis uses individual state snapshots emanating from the machines being monitored for condition-based reliability assessment. This analysis requires well-conditioned and denoised data from a machine's data acquisition/manipulation system. As a type of *logical fault model*, fault trees require data captured at some point in time and bundled into a vector V^n of n state variables indexed by $\{i : 0 < i \leq n\}$. Such data initially[1] comprises the state snapshot. Prior to conducting fault tree analysis, each state variable $v_i \in V^n$ had been coarsely discretized based on a variety of state detection algorithms. Furthermore, each variable constitutes the head element of its own timed data stream, where all such streams are subject to some form of barrier synchronization [17].

In this work, we will assume that each snapshot V^n had been made *temporally coherent*. That is, all values in V^n have been acquired within some acceptable time window using data fusion strategies surveyed in [6]. Later work will parameterize each state variable v_i with time (i.e., v_{ti}) to represent some state variable in a snapshot at time t. *Transition fault models*, like Markov chains or Petri nets, can then be applied to sequences of these time-parameterized snapshots.

This paper focuses on fault trees used for continuous monitoring and diagnosis of unattended ocean machinery. For an overview of reliability issues associated with such machinery, see [16]. Since responding to false alarms (i.e., false positives) for such machinery incurs a high expeditionary cost, this paper considers logical fault models that minimize non-determinism.

[1]In a later section we will be augmenting V^n with events derived from combinations in V^n.

Property:	Description:
indexing	$\{i : 0 < i \leq n\}$
membership	$v_i \in V^n$
valuation	$\Phi : V^n \rightarrow B^n$
extension	$V^+ = \{v_i : [v_i] = 1\}$

TABLE I
PROPERTIES OF STATE SNAPSHOTS

A unifying formalism is required to express the capabilities of successively more robust classes of fault trees. Section II defines an initial class of fault trees based on simple sum-of-products expressions. Section III augments this basic structure for multi-state systems, (non-)coherent valuation functions, and shared nodes. Section IV presents a real time fault tree evaluation rule and examines its complexity, distinguishing this result from complexity results for various aspects of fault tree construction. Section V presents a summary and lists future work.

II. Structure

A physical machine produces a *valuation* of vector V^n of n state variables using Boolean function $\Phi : V^n \rightarrow \{0, 1\}^n$, where for state variable v_i a fault is indicated by its value $[v_i] = 1$. The collection of all such faults, or more generally *events*, are referred to as the *extension* of V^n, namely $V^+ = \{v_i : [v_i] = 1\}$. Extension V^+ provides the input instance to an analysis involving fault trees described in the next paragraph, while Table I summarizes parameters and properties of the state snapshot.

Fault trees are used to assess the machine's health given V^+. Consider a forest of p 2-level fault trees for p possible event types. Each tree represents a logical sum-of-products expression over $v_i \in V^n$ mapping to some event type x_l. In particular, a fault (sub)tree l is a mapping $\Psi_l : E^q \rightarrow X^1$ defining the disjunction of q_l *cut sets* corresponding to exactly one event type $x_l \in X^p$ among p possible fault types. The

Property:	Description:
indexing	$\{j : 0 < j \leq m\}$
membership	$e_j \in E^m$
valuation	$E^m = \{e_j : e_j \subseteq V^n\}$
extension	$E^+ = \{e_j : e_j \subseteq V^+\}$

TABLE II
PROPERTIES OF CUT SETS

Property:	Description:
indexing	$\{l : 0 < l \leq p\}$
membership	$x_l \in X^p$
valuation	$\Psi : E^m \rightarrow X^p$
extension	$X^+ = \{(x_l, j) : x_l = \Psi(e_j), e_j \in E^+\}$

TABLE III
PROPERTIES OF FAULT TREES

resulting event type x_l may be interpreted as a type of low-level failure or higher-level fault.

Each cut set $e_j \in E^q$ is itself a conjunction or product over r_j state variables indexed by k, where each such variable is denoted as u_{jk}. Cut sets are manually specified by a domain expert, with additional cut sets generated using a variety of imputation techniques. The resulting m cut sets indexed by $\{j : 0 < j \leq m\}$ spanning all p fault trees are nonetheless many orders of magnitude smaller than the cardinality the power set over V^n. Already known at run time, the total number of cut sets m is expected not to exceed several million. Table II summarizes parameters and properties of cut sets. Formula 1 computes the value $[x_l]$ as a Boolean function in disjunctive normal form (DNF).

$$[x_l] = \bigvee_{j}^{q_l} \bigwedge_{k}^{r_j} u_{jk} \tag{1}$$

Formula 1 can be made concrete by the following *evaluation* condition:

$$[x_l] = 1 \Leftrightarrow \{k : (\exists j)(u_{jk} \in e_j, x_l = \Psi(e_j), e_j \subseteq V^+)\} \tag{2}$$

Formula 2 stipulates that a fault tree rooted in x_l evaluates to true (i.e., $[x_l] = 1$) iff every variable u_{jk} in at least one cut set e_j corresponding to x_l is wholly contained in extension V^+. This firing condition provides a scaffold on which diagnostic measures for *certainty*, *severity*, and *specificity* can be computed. First, consider the certainty measure. For some fault, two cut sets evaluating to true will result in a higher certainty measure for an event than if only one cut set were to fire. Likewise, if the same number of two or more cut sets were to fire for one event as it did for some other event, then the event having the more dissimilar cut sets would have the higher certainty score. Next, consider the severity measure. An extensive literature exists for the computation of such scores based on the severities of individual events making up the cut set and on the combination of events comprising the cut set. Finally, consider the specificity measure. Given two distinct firings $x_{l_0} = \Psi(e_{j_0})$ and $x_{l_1} = \Psi(e_{j_1})$, if $e_{j_0} \subset e_{j_1}$, then x_{l_1} will have a higher specificity value than would x_{l_0}.

Although Formula 2 may be satisfactory for a single 2-level fault tree, real time fault tree evaluation requires listing *all* such fault trees evaluating to true given V^+. This entails propagating extension V^+, including its associated metrics, to extension E^+ for cut sets and thence to extension X^+ for fault trees in the forest. Extension $E^+ = \{e_j : e_j \subseteq V^+\}$ comprises all cut sets e_j that are wholly contained in V^+. Based on E^+, the list of all fault trees evaluating to true entails computing X^+ described in Table III, which also summarizes the parameters and properties of fault trees. Composing extension properties from Tables I - III results in Formula 3.

$$X^+ = \{(x_l, j) : (\exists e_j)(x_l = \Psi(e_j), e_j \subseteq V^+)\} \tag{3}$$

Among other things, the following section augments this 2-level structure into a hierarchy so that X^+ and its metrics may thence be propagated up to some Top-Level Event (TLE).

III. REFINEMENTS

The structure in Section II can neither represent multi-valued state variables, nor composition of fault trees into a hierarchy ultimately rooted in some Top-Level Event (TLE). The negation-free notion of extension developed thus far has limitations, which we extend to multi-state systems in Section III-A. Section III-B introduces a means of specifying hierarchies of events using the notion of shared nodes. Shared nodes reduce the need to replicate fault trees for multiple types of events common to specific combinations of lower level events. As alluded to earlier, shared nodes provide placeholders for certainty, severity, and specificity measures, as well as a framework for propagating these values to the TLE for overall machine health assessment. These values will furthermore depend on whether the valuation of the system's state is coherent. Closely related to multi-state systems, the coherence property is examined in Section III-C.

A. Multistate systems

The extension expressed in Formula 3 can express the *presence* but not the *absence* of events. That is, negation is not supported. Unfortunately, the inability to affirmatively test for the *absence* of events prevents use of diagnostic procedures involving the *ruling out* of certain other events. Modeling explicitly binary-valued variables will enable *differential diagnosis* – a process by which one event can be distinguished from some other event based on the absence of certain other events. A fault tree that supports negation can make explanation of an event easier by stipulating inside the contents of cut sets which events are expressly absent.

Suppose by V^{n_1}, we mean vector V^n of negation-free states defined in Section II. Introducing the notion of negation entails augmenting V^n with a set V^{n_2} of binary-valued state variables. Applying the technique used to reduce the Satisfiability

Problem (SAT) to Monotone SAT refered to in Appendix A9 of [7], we can augment V^n with distinct and indivisible state variables $\neg v_i$ subject to the *separation* condition in Formula 4.

$$\{i, j : v_i \in e_j, \neg v_i \in e_j\} = \oslash \tag{4}$$

A parsimonious state V^{n_2} would include only $\neg v_i$ variables for which there exists at least one cut set. Subjecting V^{n_2} to this *relevance* condition – necessary for *coherence* – results in Formula 5.

$$V^{n_2} = \{\neg v_i : \exists(e_j)(\neg v_i \in e_j)\} \tag{5}$$

Three-valued variables, like operating temperature, are useful for detecting departures from some *interior* optimum. Notions like: 'depressed', 'normal', and 'elevated' can be expressed in a manner similar to the negation case by including state variables like $\{-v_i, \neg v_i, +v_i\}$ respectively into V^{n_3}. By a similar construction one may define the four-valued notion: 'normal', 'alert', 'warning', and 'emergency' into V^{n_4}. Hence, the refinement for multi-state systems involves computing the state vector in Formula 6, and redefining n accordingly.

$$V^n = V^{n_1} \cup V^{n_2} \cup V^{n_3} \cup V^{n_4} \tag{6}$$

Fault trees for multi-state systems entail more intricate computation of severity scores like those described in [3], [18]. Such fault trees require construction of a larger number of cut sets, or alternatively, incorporating a notion of ordering when computing extension X^+. Subjecting their construction to separation and relevance conditions partially mitigates these tractability problems.

B. Node commonality

As a two-level sum of products expression, Formula 1 cannot represent an arbitrary Boolean expression without replicating a potentially large number of subtrees. The same event x_l may be in common with more than one fault subtree. Furthermore, x_l may be a member of more than one cut set within a fault tree. Finally, fault trees are rarely balanced so that a (sub)tree rooted in event x_a may include cut sets comprised of faults v_i and child events x_l. By a technique similar to that in Section III-A we may further augment V^n with $x_l \in X^p$ to result in W^{n+p}. Inclusion of x_l into W^{n+p} is subject to a separation condition like that in Formula 4. If all events $x_l \in X^p$ are *relevant* to the TLE – by a condition similar to Formula 5 – then the augmented state is computed by Formula 7.

$$W^{n+p} = V^n \cup X^p \tag{7}$$

Defining W^+ in a manner similar to V^+, extension X^+ can be redefined by Formula 8.

$$X^+ = \{(x_l, j) : (\exists e_j)(x_l = \Psi(e_j), e_j \subseteq W^+)\} \tag{8}$$

C. Coherence

The hazard scoring and propagation techniques described in [3], [10], [18] assume the multi-state system being modeled is *coherent*, namely the valuation of its state does not improve with an increasing number of component faults. We adapt the definition of coherence in [4] to specifically highlight the problem of transient effects causing false positives.

Valuation function $\Phi : W^{n+p} \rightarrow B^{n+p}$ of augmented state snapshot W^{n+p} is said to be *coherent* if (i) all variables are relevant, and (ii) $\Phi(W^{n+p})$ is monotonically non-decreasing.[2] For condition (i), every state variable $v_i \in W^{n+p}$ and failure type $x_l \in W^{n+p}$ are part of at least one cut set, so all variables in W^{n+p} are *relevant*. For condition (ii), consider two successive state snapshots, $W^{n+p}_{t_0}$ and $W^{n+p}_{t_1}$ at times t_0 and t_1 respectively. If $W^{n+p}_{t_0} \subseteq W^{n+p}_{t_1}$ then valuation function Φ is coherent.

The succession of states in the previous paragraph indicates either the presence of a spreading fault or transient effects. To rule out transient effects, suppose the only difference between the two states is the change of one or more variables w_i from zero to one. If at some later time t', $W^{n+p}_{t'} = W^{n+p}_{t_0}$ and there were no repair actions, then intermediate state $W^{n+p}_{t_1}$ reflects transient effects.

The notion of what constitutes *transient effects* is open to interpretation. Few would argue that a machine experienced transient effects when it broke down but was subsequently repaired. Considering events requiring repair as *stuck-at* faults, excludes these situations from the notion of transient effects.

Less obvious are transient effects stemming from routine operation and automated control. Such effects are expected to occur often as ocean turbines right themselves in the presence of turbulent waters. Data streaming from the attitude sensor at 10 Hz may be used for self-righting. The fault tree, however, must depend on these pitch yaw and roll measurements conditioned over longer time intervals. The intention is to detect failures in the control system, rather than registering an abundance of false positives that have been automatically corrected by movement of any rudders or fins. Other sensor types noted for displaying transient effects include the five vibration sensors located at various points on the turbine's drive train.

A recent study [4] surveyed other scenarios in which non-coherence apparently arises, and how machine health assessment procedures can be adapted. One such scenario in [4] anticipated our need to operate ocean turbines at reduced output as its state gradually degrades. For example, operating the turbine closer to the surface maximizes momentum flux, and hence output, but at the expense of both turbulence and accelerated rates of fouling. As its state degrades, due either to bad weather or biofilm formation, the turbine can operate at reduced output further down the water column. Such degraded operation comes at the expense of increased bathymetric pressure on seals. Hence, formulating an appropriate response to degraded states due to one set of variables like turbulence and

[2] $\Phi()$ is said to be *strictly coherent* if it is monotonically increasing.

fouling, will often depend on other variables like bathymetric pressure.

IV. Algorithm Design and Analysis

Fault tree evaluation ultimately entails evaluation of Boolean functions. A variety of implementations of Boolean function manipulation and evaluation involve binary decision diagrams (BDD)'s, hypergraphs, and SAT solvers. In addition to work already cited, implementations of fault trees using BDD's or their variants also appear in [1], [2], [12], [13], [14]. Visualizing Satisfiability (SAT) instances and Boolean functions that include the use of hypergraphs were reported in [15]. Use of hypergraphs that relate vertices to state predicates w_i and hyperedges to clauses e_j were reported in [9], providing a context for the complexity analysis in the following paragraphs.

A. Evaluation

Recall that to solve the Satisfiability Problem (SAT) requires finding a satisfiable truth assignment, given some Boolean formula [8]. That entails first *guessing* a truth assignment, then *checking* if that assignment is indeed satisfied. Although both operations can be done in low-order polynomial time, what makes SAT NP-Complete (NP-C) in the number of variables $n + p$ is the intractably large number of guesses required to identify a satisfying truth assignment.

The *checking* phase of SAT trivially reduces to computing extension X^+ in Formula 8. To see this, negate the problem instance for SAT originally expressed by Cooke in Conjunctive Normal Form (CNF) to obtain the fault tree in DNF shown in Formula 1, and then reverse the sense of the truth assignment.

In condition-based monitoring, we are already given truth assignment V^+ so *checking* whether X^+ is non-empty can be done in polynomial time. Event explanation and possibly localization, however, requires listing *all* event types generated ultimately from extension V^+. Known as the SAT Evaluation Problem, implementations of this checking procedure are defined for most variants of BDD's surveyed in [5]. One state-of-the-art data structure known as Zero-suppressed Decision Diagrams (ZBDD) is adapted to efficient identification of sets of subsets [11]. Run time comparison of these and other data structures and techniques for computation of X^+ is left for future work. As a reification of Formula 8, Formula 9 provides an evaluation rule for computing extension set X^+.

$$((\{i, j : w_i \in W^+, w_i \in e_j\} \Rightarrow c_j \leftarrow c_j + 1) = |e_j|) \Rightarrow$$
$$X^+ \leftarrow (\Psi(e_j), j) \cup X^+; W^+ \leftarrow \Psi(e_j) \cup W^+; c_j \leftarrow 0 \quad (9)$$

This rule supposes that for each cut set e_j, we maintain a count c_j, initially 0, that gets incremented for each $w_i \in W^+$ whenever $w_i \in e_j$. Once count c_j equals $|e_j|$, clause e_j is fully satisfied, so we append the event associated with e_j and cut set identifier j to extension X^+. Including j in the solution facilitates fault localization and explanation, Finally, we include the event associated with e_j into set W^+ and re-initialize c_j to zero. Implied in this rule is some stepwise algorithm which we will describe and implement in future work.

SAT evaluation entails evaluating a logic circuit like the one in Figure 1 (top). An equivalent schematic in Figure 1 (bottom) shows the evaluation rule for the same circuit, but in terms of increment/compare/output/reset operators suggested by Formula 9.

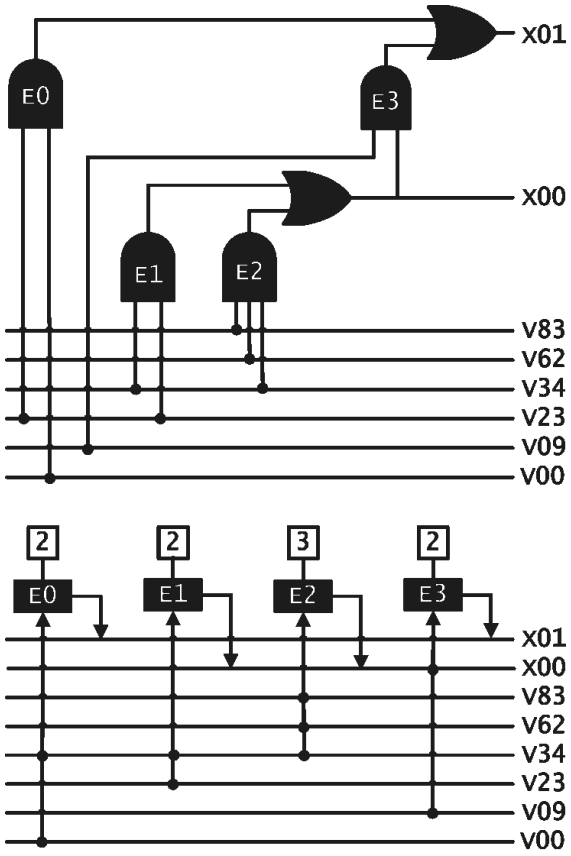

Fig. 1. SAT Evaluator example: (top) logic circuit, (bottom) rule schematic

Formula 9 solves the Fault Tree Evaluation Problem with a worst case complexity of $\mathcal{O}(m^2)$. To see this, contrive a problem instance comprised of m cutsets with $n + p = m$ elements in W^+. Such pathological cases are easy to spot at the time of fault tree construction by detecting violations of the bounds in Formula 10.

$$\{i, j : |W^+| \ll (n+p), |e_j| \ll (n+p), |Q_i| \ll m\} \quad (10)$$

Formula 10 asserts that the fault tree rooted in the TLE is comprised of a sparse set of cut sets wherein each cut set is itself sparse in the number of variables. Note that set Q_i represents the set of cutsets in which w_i is a member. The actual number of steps can be expressed by Formula 11 and can be predicted in $\mathcal{O}(|W^+|)$ steps.

$$\sum_{i}^{|W^+|} |Q_i| \tag{11}$$

B. Construction

Although this paper focuses on SAT Evaluation, a number of problems associated with the *construction* of fault trees remain NP-C. Unless P = NP, fault trees for machinery like ocean turbines will always remain not fully specified with imperfect fault coverage. This is unfortunate, since silent failures (i.e., false negatives) disrupt maintenance schedules, which are driven by the high expeditionary cost incurred prior to ocean equipment maintenance and repair. Silent failures too often mask unexpectedly severe damage, while providing insufficient or misleading diagnostic information to maintenance personnel.

An NP-C problem known as the Automatic Test Pattern Generation Problem (ATPG), from the field of electronic design automation, can be reduced to a corresponding problem in fault tree construction. ATPG requires the listing of all test patterns (cut sets) that can lead to failure. Not only is ATPG NP-C, but it is also PSPACE-Complete – requiring intractably large storage. Under a similar guise, any imputation technique by which one must infer a complete set of cut sets given some 'starter' set, also appears to be PSPACE-Complete. Still another intractable problem reducible from SAT involves the computation of prime implicants in a Boolean expression – important for identifying the cut sets having the least number of conditions that can prompt some given event. Additional NP-C problems associated with BDD's and hence fault trees were identified in [5]. Fault trees generated by imputation procedures will have its number of cut sets vastly exceeding the number of variables. Partially mitigating this, we observed cut sets rarely exceeding four terms when formulating the starter set for ocean turbines.

V. Summary

The set-theoretic perspective on condition-based evaluation of fault trees enabled us to define a class of fault trees useful for health assessment of remote ocean machinery. This class can represent multi-state systems, non-coherent valuations, and node sharing – all of which are pre-requisites for current research into logical fault models like fault trees. Due to high expeditionary cost to service this machinery, we sought to minimize the number of false positives by formally characterizing one class of transients in terms of a state coherence condition.

Each successive refinement exposes variables to which we may attach health assessment indicators. In addition to the well-studied phenomenological measures for severity, the compositional framework also characterized epistemological measures of certainty and specificity. The proposed framework provides an effective fault tree evaluation rule having complexity bounds known at fault tree construction time.

Future work involves identifying fault tree evaluation tools included with SAT solvers and BDD packages. Run time and usability comparison of these tools to our software implementation of the SAT evaluation rule is anticipated. Technical documentation currently underway will be posted, along with the current version of the executable code for the SAT evaluator in Formula 9, its supporting tools, and sample fault trees.

Acknowledgment

The work discussed here grew from collaborations within the Prognostics and Health Monitoring (PHM) working group of the Center for Ocean Energy Technology (COET) at Florida Atlantic University and was funded through COET by the State of Florida.

References

[1] T. Assaf and J. Dugan. Diagnosis based on reliability analysis using monitors and sensors. *Reliability Engineering & System Safety*, 93(4):509 – 521, 2008.

[2] A. Bobbio, D. Codetta-Raiteri, M. D. Pierro, and G. Franceschinis. Efficient analysis algorithms for parametric fault trees. *Techniques, Methodologies and Tools for Performance Evaluation of Complex Systems, Workshop on*, 0:91–105, 2005.

[3] Y.-R. Chang, S. Amari, and S.-Y. Kuo. Obdd-based evaluation of reliability and importance measures for multistate systems subject to imperfect fault coverage. *Dependable and Secure Computing, IEEE Transactions on*, 2(4):336–347, Oct.-Dec. 2005.

[4] S. Contini, G. Cojazzi, and G. Renda. On the use of non-coherent fault trees in safety and security studies. *Reliability Engineering & System Safety*, 93(12):1886 – 1895, 2008. 17th European Safety and Reliability Conference.

[5] R. Drechsler and D. Sieling. Binary decision diagrams in theory and practice. *International Journal on Software Tools for Technology Transfer (STTT)*, 3(2):112–136, May 2001.

[6] J. Duhaney, T. M. Khoshgoftaar, A. Agarwal, and J. C. Sloan. Mining and storing data streams for reliability analysis. ISSAT, August 5-7 2010.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York City, 1979.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*, chapter 2.6, pages 38–44. W.H. Freeman and Company, New York City, 1979.

[9] D. Habet, L. Paris, and C. Terrioux. A tree decomposition based approach to solve structured sat instances. In *Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI'09)*, pages 115–122, Washington, DC, USA, 2009. IEEE Computer Society.

[10] W. S. Jung, S. H. Han, and J. Ha. A fast bdd algorithm for large coherent fault trees analysis. *Reliability Engineering & System Safety*, 83(3):369 – 374, 2004.

[11] S.-I. Minato. Zero-suppressed bdds and their applications. *International Journal on Software Tools for Technology Transfer (STTT)*, 3(2):156–170, May 2001.

[12] Y. Mo. Variable ordering to improve bdd analysis of phased-mission systems with multimode failures. *Reliability, IEEE Transactions on*, 58(1):53–57, March 2009.

[13] A. Myers and A. Rauzy. Efficient reliability assessment of redundant systems subject to imperfect fault coverage using binary decision diagrams. *IEEE Transactions on Reliability*, 57(2):336–348, June 2008.

[14] R. Remenyte-Prescott and J. Andrews. An enhanced component connection method for conversion of fault trees to binary decision diagrams. *Reliability Engineering & System Safety*, 93(10):1543 – 1550, 2008.

[15] C. Sinz. Visualizing sat instances and runs of the dpll algorithm. *Journal of Automated Reasoning*, 39(2):219–243, August 2007.

[16] J. C. Sloan, T. M. Khoshgoftaar, P.-P. Beaujean, and F. Driscoll. Ocean turbines – a reliability assessment. *International Journal of Reliability, Quality and Safety Engineering*, 16(5):413–433, 2009.

[17] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.

[18] L. Xing and Y. Dai. A new decision-diagram-based method for efficient analysis on multistate systems. *IEEE Transactions on Dependable and Secure Computing*, 6(3):161–174, 2009.