25

Binary Decision Diagrams for Reliability Studies

Antoine Rauzy

IML CNRS, 169, Avenue de Luminy, 13288 Marseille cedex 9, France

Abstract: Bryant's binary ecision diagrams are state-of-the-art data structures used to encode and to manipulate Boolean functions. Risk and dependability studies are heavy consumers of Boolean functions, for the most widely used modeling methods, namely fault trees and event trees, rely on them. The introduction of BDD in that field renewed its algorithmic framework. Moreover, several central mathematical definitions, like the notions of minimal cutsets and importance factors, were questioned. This article attempts to summarize fifteen years of active research on those topics.

25.1 Introduction

Binary decision diagrams (BDD for short) are state-of-the-art data structures used to encode and to manipulate Boolean functions. They were introduced in 1986 by R. Bryant [1] to handle logical circuits and further improved by Bryant and others in the early 1990s [2, 3]. Since then, they have been used successfully in a wide variety of applications (see [4] for a glance). Reliability and dependability studies are heavy consumers of Boolean functions, for the most widely used modeling methods, namely fault trees and event trees, rely on them. The introduction of BDD in that field [5, 6] renewed completely its algorithmic framework. Moreover, several central mathematical definitions, like the notions of minimal cutsets and importance factors, were questioned. This article attempts to summarize fifteen years of active research on those topics.

Fault tree and event tree analyses are classically performed in two steps: first, the minimal cutsets (MCS for short) of the model are determined by some top-down or bottom-up algorithm; second, some probabilistic quantities of interest are assessed from MCS (see, *e.g.*, [7, 8, 9]). Usually, not all of the MCS are considered: cut-offs are applied to keep only the most relevant ones, *i.e.* those with the highest probabilities. With BDD, the methodology is different: first, the BDD of the model is constructed. Then, a second BDD, technically a ZBDD [10], is built from the first one to encode MCS. Probabilistic quantities are assessed either from the BDD or from the ZBDD. From an engineering viewpoint, this new approach changes significantly the process: first, MCS are now used mainly for verification purposes since they are not necessary to assess probabilistic quantities. Second, probabilistic assessments provide exact results, for no cut-off needs to be applied. Last, both coherent (monotone) and non-coherent models can be handled, which is not possible, at least accurately, with the classical approach. However, the new approach has a drawback: the construction of the BDD must be feasible and, as with many Boolean problems, it is

of exponential worst case complexity. The design of strategies and heuristics to construct BDD for large reliability models is still challenging.
In this article, we focus on three issues: the computation of minimal cutsets, the assessment of various probabilistic quantities, and finally the treatment of large models.

A minimal cutset of a fault tree is a minimal set of basic events that induces the top event. The notion minimal cutset is thus related to the logical notion of prime implicant. For a coherent model the two notions actually correspond. For non-coherent models however, they differ. Until recently, the reliability engineering literature stayed at a very intuitive level on this subject. With the introduction of BDD, not only new efficient algorithms to compute MCS have been designed, but the mathematical foundations of MCS have been more firmly established [11].

Beyond the top event probability, a fault tree study involves in general the computation of various quantities like importance factors or equivalent failure rates. The BDD approach makes it possible (and necessary) to revisit these notions and to improve the computations in terms of both efficiency and accuracy [12, 13, 14].

Event trees of the nuclear industry involve up to thousands of basic events. The calculation of BDD for these models is still challenging, while the classical approach (based on the extraction of MCS) is always able to give results, approximate results of course, but still results. Recent studies showed however that, conversely to what was commonly admitted, these results may be not very accurate [15]. The design of heuristics and strategies to assess huge reliability models is therefore of a great scientific and technological interest.

The remainder of this article is organized as follows. Section 25.2 recalls basics about fault trees, event trees, and BDD. Section 25.3 presents the notion of minimal cutsets. Section 25.4 shows the various probabilistic quantities of interest that can be computed by means of BDD. Section 25.5 discusses the assessment of large models. Section 25.6 concludes the article.

25.2 Fault Trees, Event Trees and Binary Decision Diagrams

This section recalls basics about fault trees, event trees and BDD.

25.2.1 Fault Trees and Event Trees

A fault tree is a Boolean formula built over variables, so called basic events that represent failures of basic components (the e_i of Figure 25.1 , and gates (AND, OR, *k*-out-of-*n*). Gates are usually given a name, like the G_i's of Figure 25.1 and called intermediate events. The formula is rooted by an event, called the top event (T in Figure 25.1). The top event encodes the various combinations of elementary failures, *i.e.,* of basic events, that induce a failure of the system under study.

Fault tree studies are twofold. Qualitative analyses consist of determining the minimal cutsets, *i.e.* the minimal combinations of basic events that induce the top event. Quantitative analyses consist of computing various probabilistic quantities of interest, given the probabilities of occurrence of the basic events. Both problems are hard–counting the number of minimal solution of a Boolean formula in assessing the probability of the formula–and fall into the P-hard complexity class [16]. It is worth noting that these complexity results stand even in the case where the underlying function is monotone (coherent), which is in general the case (usually, fault trees do not involve negations).

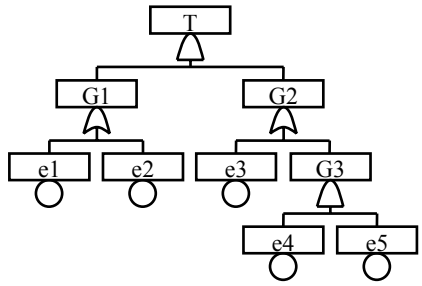

Figure 25.1. A fault tree

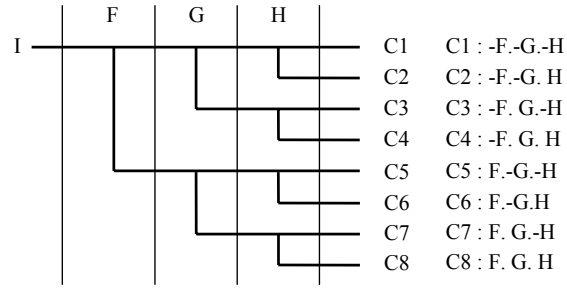

Figure 25.2. An event tree

Depending on the success or the failure of the system, the upper or the lower branch is taken. At the end, a consequence is reached (one of the C_i's on). Failures of safety systems are described by means of fault trees. Sequences are compiled into conjunctions of the initiating event and top events or negations of top events, as illustrated on the right hand side of Figure 25.2. The assessment of sequences, or groups of sequences, is similar to the assessment of fault trees. However, it is worth noting that, due to success branches, formulae may involve negations.

Fault trees are the most widely used method for risk analyses. Virtually all industries that present a risk for the environment use it. Event trees are used mainly in the nuclear industry. A good introduction to both techniques can be found in reference [9].

25.2.2 Binary Decision Diagrams

Binary Decision Diagrams are a compact encoding of the truth tables of Boolean formulae [1, 2]. The BDD representation is based on the Shannon decomposition: Let F be a Boolean function that depends on the variable v, then the following equality holds.

$$F = v.F[v \leftarrow 1] + \overline{v}.F[v \leftarrow 0]$$

By choosing a total order over the variables and applying recursively the Shannon decomposition, the truth table of any formula can be graphically represented as a binary tree. The nodes are labeled with variables and have two outedges (a *then*-outedge, pointing to the node that encodes F[v←1], and an *else*-outedge, pointing to the node that encodes F[v←0]). The leaves are labeled with either 0 or 1. The value of the formula for a given variable assignment is obtained by descending along the corresponding branch of the tree. The Shannon tree for the formula $F = ab + \overline{a}c$ and the lexicographic order is pictured Figure 25.3 (dashed lines represent *else*-outedges).

Indeed, such a representation is very space consuming. It is possible, however, to shrink it by means of the following two reduction rules.

- Isomorphic subtrees merging. Since two isomorphic subtrees encode the same formula, at least one is useless.
- Useless nodes deletion. A node with two equal sons is useless since it is equivalent to its son ($F = v.F + \overline{v}.F$).

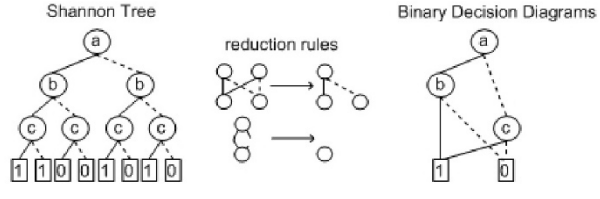

Figure 25.3. From the Shannon tree to the BDD

By applying these two rules as far as possible, one gets the BDD associated with the formula. A BDD is therefore a directed acyclic graph. It is unique, up to an isomorphism. This process is illustrated Figure 25.3.

25.2.3 Logical Operations

Logical operations (and, or, xor, ...) can be directly performed on BDD. This results from the orthogonality of the Shannon decomposition with usual connectives:

$$(v.F_1 + \overline{v}.F_0) \oplus (v.G_1 + \overline{v}.G_0)$$
$$=$$
$$v.(F_1 \oplus G_1) + \overline{v}.(F_0 \oplus G_0)$$

where $\oplus$ stands for any binary connective.

Among other consequences, this means that the complete binary tree is never built and then shrunk: the BDD encoding a formula is obtained by composing the BDD encoding its subformulae. Hash tables are used to store nodes and to ensure, by construction, that each node represents a different function. Moreover, a caching principle is used to store intermediate results of computations. This makes the usual logical operations (conjunction, disjunction) polynomial in the sizes of their operands. The complete implementation of a BDD package is described in reference [2].

25.2.4 Variable Orderings and Complexity Issues

It has been known since the very first uses of BDD that the chosen variable ordering has a great impact on the size of BDD and therefore on the efficiency of the whole methodology [1]. Finding the best ordering, or even a reasonably good one, is a hard problem (see *e.g.*, [17, 18]). Two kinds of heuristics are used to determine which variable ordering to apply. Static heuristics are based on topological considerations and select the variable ordering once for all (see, *e.g.*, [19, 20, 21]). Dynamic heuristics change the variable ordering at some points of the computation (see, *e.g.*, [3, 22]). They are thus more versatile than the former, but the price to pay is a serious increase of running time. Sifting is the most widely used dynamic heuristics [3]. We shall come back to this topic in Section 25.5.

25.2.5 Zero-suppressed Binary Decision Diagrams

Minato's zero-suppressed binary decision diagrams are BDD with a different semantics for nodes (and slightly different reduction rules) [10]. They are used to encode sets of minimal cutsets and prime implicants. Nodes are labeled with literals (and not just by variables). Let p be a literal and U be a set of products. By abuse, we denote p.U the set $\{\{p\} \cup \pi;\ \pi \in U\}$. The semantics of ZBDD is as follows.

The leaf 0 encodes the empty set: Set[0] = $\varnothing$.

The leaf 1 encodes the set that contains only the empty product: Set[1] = $\{\{\}\}$.

A node Δ(p,S1,S0), where p is a literal and S1 and S0 are two ZBDD encodes the following set of products.

Set[Δ(p,S1,S0)] = p.Set[S1] $\cup$ Set[S0]

It is possible to encode huge sets of MCS with relatively small ZBDD, provided MCS have some regularity that can be captured by the sharing of nodes.

25.3 Minimal Cutsets

Minimalcutsets (MCS for short) are the keystone of reliability studies. A minimal cutset is a minimal set of basic events that induces the realization of the top event. For coherent models, this informal definition is sufficient. It corresponds to the formal notion of prime implicant (PI for short). For non-coherent models, MCS and PI differ, for the latter contain negative literals while the former do not. A full understanding of both notions requires some algebraic developments (borrowed mainly to reference [11]).

25.3.1 Preliminary Definitions

A *literal* is either a variable v (positive literal), or its negation $\neg v$ (negative literal). v and $\neg v$ are said to be *opposite*. We write $\overline{p}$ the opposite of the

literal p. A *product* is a set of literals interpreted as the conjunction of its elements. Products are often written like words. For instance, the product {a, b, ¬c} is written $ab\bar{c}$. A *minterm* on a set of variables V = {v_1,...,v_n} is a product which contains exactly one literal built over each variable of V. We write minterms(V) for the set of minterms built on V. If V contains n variables, minterms(V) has 2^n elements.

An *assignment* of a set of variables V = {v_1,...,v_n} is a function σ from V to {0, 1} that assigns a value (true or false) to each variable of V. Using the truth tables of connectives, assignments are extended into functions from formulae built over V to {0, 1}. An assignment σ *satisfies* a formula F if σ(F) = 1. It falsifies F if σ(F) = 0.

There is a one-to-one correspondence between the assignments of V and the minterms built on V: a variable v occurs positively in the minterm π iff and only if σ(v)=1 in the corresponding assignment σ. For instance, the minterm $ab\bar{c}$ corresponds to the function σ such that σ(a)=σ(b)=1 and σ(c)=0, and vice-versa. Similarly, a formula F can be interpreted as the set of minterms (built on the set var(F) of its variables) that satisfy it. For instance, the formula $F = ab + \bar{a}c$ can be interpreted as the set $\{abc, ab\bar{c}, \bar{a}bc, \bar{a}b\bar{c}\}$. For the sake of convenience, we use set notations for formulae and minterms, *e.g.,* we note σ∈ F when σ(F)=1.

There exists a natural order over literals: ¬v < v. This order can be extended to minterms: $\pi \le \rho$ iff for each variable v, π(v)≤ ρ(v), *e.g.*, $\bar{a}\bar{b}c \le a\bar{b}c$ because a occurs negatively in $\bar{a}\bar{b}c$ and positively in $a\bar{b}c$. A physical interpretation of the inequality π≤ρ is that π contains less information than ρ for it realizes fewer basic events. From an algebraic viewpoint, the set minterms(V) equipped with the above partial order forms a lattice, as illustrated in Figure 25.4 (left). The order relation is represented by lines (bottom-up). For the sake of the simplicity, transitive relations are not pictured.

A formula F is said to be *monotone* if for any pair of minterms π and ρ such that $\pi \le \rho$, then ρ∈ F implies that π∈ F. The formula $F = ab + \bar{a}c$ is not monotone because, $\bar{a}\bar{b}c \in F$, $\bar{a}\bar{b}c \le a\bar{b}c$ but $a\bar{b}c \notin F$. This is graphically illustrated Figure 25.4 (right), where the minterms not in F are grayed. Coherent fault trees that are built over variables, and-gates, or-gates and *k*-out-of-*n* connectives are monotone formulae. Non-monotony is introduced by negations.

25.3.2 Prime Implicants and Minimal Cutsets

25.3.2.1 Prime Implicants

We can now introduce the notion of prime implicant. A *product* π is an implicant of a formula F if for all minterms ρ containing π, ρ∈ F. An implicant π of F is *prime* if no proper subset of π is an implicant of F. The set of prime implicants of F is denoted PI[F].

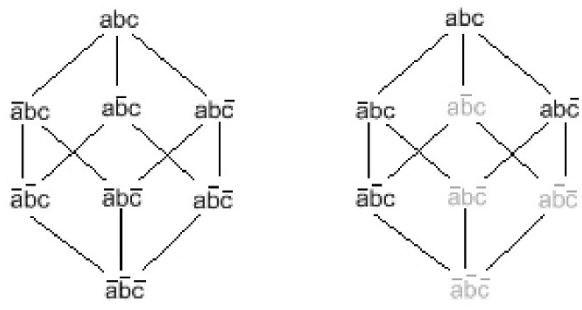

Figure 25.4. The lattice of minterms for {a,b,c}

For instance, the formula $F = ab + \bar{a}c$ admits the following set of prime implicants: $PI[F] = \{ab, \bar{a}c, bc\}$. Note that $\bar{a}b$ is an implicant of F because both $\bar{a}b\bar{c}$ and $\bar{a}bc$ satisfy F. It is prime because neither $\bar{a}$ nor b are implicants of F.

25.3.2.2 Minimal Cutsets

In reliability models, there exists a fundamental asymmetry between positive and negative literals. Positive literals represent undesirable and rare events such as failures. Negative literals represent thus the non occurrence of these events. Positive literals are the only ones that convey relevant information. This is the reason why most of the fault tree assessment tools never produce minimal cutsets with negative literals. To illustrate this idea, consider again the formula $F = ab + \bar{a}c$. We have $PI[F] = \{ab, \bar{a}c, bc\}$. This does correspond to the notion of minimal solutions of F, but this does not correspond to the intuitive notion of minimal cutsets. The expected minimal cutsets are ab and c which are the "positive parts" of prime implicants. There are cases however where negative literals are informative as well. This is typically the case when the values of some physical parameters have to be taken into account (see, *e.g.*, [23]). Thus, we have to consider the set L of literals that convey interesting information. L is typically the set of all positive literals plus possibly some negative literals. A literal p is *significant* if it belongs to L. It is *critical* if it is significant while its opposite is not.

Let V be a finite set of variables. Let L be a subset of significant literals built over V. Finally, let F be a formula built over V. We shall define minimal cutsets of F as minimal solutions (prime implicants) from which literals outside L are removed because they "do not matter". Let $PI_L[F]$ be the set of products obtained first by removing from products of PI[F] literals not in L and second by removing from the resulting set the non minimal products. Formally, $PI_L[F]$ is defined as follows.

$PI_L[F] = \{\pi \cap L;\ \pi \in PI[F]$ and there is no ρ in PI[F] such that $\rho \cap L \subset \pi \cap L\}$

This first definition captures the intuitive notion of minimal cutsets. For instance, it is easy to verify that $PI_{\{a,b,c\}}[ab + \bar{a}c] = \{ab, c\}$. However, it relies on the definition of prime implicants. This makes it not suitable for the design of an algorithm to compute MCS without computing prime implicants. The second way to define MCS which avoids this drawback is as follows. Let $\leq_L$ be the binary relation among opposite literals defined as follows.

$$p \leq_L \bar{p} \text{ if } p \notin L$$

The comparator $\leq_L$ is extended into a binary relation over minterms(V) as follows:

$$\sigma \leq_L \rho \text{ if for any variable v, } \sigma[v] \leq \rho[v],$$

where $\sigma[v]$ (or $\rho[v]$) denotes the literal built over v that belongs to σ (or. to ρ). Intuitively, $\sigma \leq_L \rho$ when σ is less significant than ρ. For instance, $ab\bar{c} \leq_{\{a,b,c\}} abc$ because $ab\bar{c}$ contains fewer positive literals than abc. The comparator $\leq_L$ is both reflexive ($\sigma \leq_L \sigma$ for any σ) and transitive ($\pi \leq_L \sigma$ and $\sigma \leq_L \rho$ implies $\pi \leq_L \rho$, for any π, σ and ρ). Therefore, $\leq_L$ is a pre-order.

A product π over V is a *cutset* of F w.r.t. L if $\pi \subset L$ and for all minterms σ containing π there exists a minterm $\delta \in F$ such that $\delta \leq_L \sigma$. A cutset π is *minimal* if no proper subset of π is a cutset. We denote by $MC_L[F]$ the set of minimal cutsets w.r.t. L of F.

Consider again the formula $F = ab + \bar{a}c$. If $L = \{a, \bar{a}, b, \bar{b}, c, \bar{c}\}$, minterms are pairwise incomparable. Therefore, MCS are products π such that all minterms σ containing π belong to F, $MC_L[F] = PI[F]$.

If $L = \{a,b,c\}$, $ab\bar{c} \leq_L abc$, and $\bar{a}\bar{b}c \leq_L abc$, $a\bar{b}c$, $\bar{a}bc$, therefore the cutsets of F w.r.t. L are abc, ab, ac, bc and c and $MC_L[F] = \{ab, c\}$. As an illustration, consider the product c. Four minterms contain it: abc, $\bar{a}bc$, $a\bar{b}c$, $\bar{a}\bar{b}c$. Except $a\bar{b}c$, they all belong to F. $\bar{a}\bar{b}c \leq_L a\bar{b}c$, therefore there is a minterm of F smaller than $\bar{a}\bar{b}c$. So, all the minterms containing c are covered and c is a cutset.

If $L = \{b, \bar{b}, c, \bar{c}\}$, the cutsets of F w.r.t. L are bc, b and c and $MC_L[F] = \{b,c\}$.

The two definitions of MCS are actually equivalent. Let F be a Boolean formula and let L be a set of literals built over var(F). Then, the following equality holds [11].

$$PI_L[F] = MC_L[F]$$

Note finally that if L=V, a positive product π is a cutset if and only if the minterm $\pi \cup \{\overline{v}; v \in V \text{ and } v \notin \pi\}$ is an implicant of F.

25.3.3 What Do Minimal Cutsets Characterize?

Any formula is equivalent to the disjunction of its prime implicants. A formula is not in general equivalent to the disjunction of its minimal cutsets. The widening operator ω_L gives some insights about the relationship between a formula F, its prime implicants and its minimal cutsets w.r.t. the subset L of the significant literals. The operator ω_L is an endomorphism of the Boolean algebra (minterms(V), $\cap$, $\cup$, $\overline{\ }$) that associates to each set of minterms (formula) F the set of minterms ω_L defined as follows.

$\omega_L = \{\pi$; there exists ρ s.t. $\rho \leq_L \pi$ and $\rho \in F\}$

Intuitively, ω_L enlarges F with all of the minterms that are more significant than a minterm already in F.

Consider again the formula $F = ab + \overline{a}c$.

If $L=\{a, \overline{a}, b, \overline{b}, c, \overline{c}\}$, then, $\omega_L[F] = F$.

If L={a,b,c}, then

$\omega_L[F] = abc + ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}c$.

If L={b, $\overline{b}$,c, $\overline{c}$ }, then

$\omega_L[F] = abc + ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}c$

The operator ω_L has the following properties.

ω_L is idempotent; $\omega_L(\omega_L(F)) = \omega_L(F)$.

$PI[\omega_L(F)] = MC_L[F]$.

The above facts show that ω_L acts as a projection. Therefore, the formulae F such that $PI[F]=MC_L[F]$ are the fixpoints of ω_L, *i.e.* the formulae such that $\omega_L(F) = F$. If L=V, fixpoints are monotone formulae. They also give a third way to define MCS: MCS of a formula F are the prime implicants of the least monotone approximation of F. This approximation is obtained by widening F with all of the minterms that are more significant, and therefore less expected, than a minterm already in F.

25.3.4 Decomposition Theorems

The recursive algorithms to compute prime implicants and minimal cutsets from BDD rely on so-called decomposition theorems. These theorems use the Shannon decomposition as a basis. They are as follows.

Decomposition Theorem for Prime Implicants [24]
Let $F = v.F_1 + \overline{v}.F_0$ be a formula (such that F_1 and F_0 don't depend on v). Then the set of prime implicants of F is the as follows.

$$PI[F] = PI_n \cup PI_1 \cup PI_0$$

where

$PI_n = PI[F_1.F_0]$
$PI_1 = \{v.\pi ; \pi \in PI[F_1]/PI_n\}$
$PI_0 = \{\overline{v} .\pi ; \pi \in PI[F_0]/PI_n\}$

and "/" stands for the set difference.

Decomposition Theorem for Minimal Cutsets [11]:
Let $F = v.F_1 + \overline{v}.F_0$ be a formula (such that F_1 and F_0 don't depend on v). Let L be the set of relevant literals. Then, the set of minimal cutsets F is the as follows.

Case 1: Both v and its negation belong to L. In that case, the decomposition theorem is the same as for prime implicants.

Case 2: v belongs to L, its negation does not. In that case, there are two ways to compute MCS[F].

First Decomposition:
$MCS[F] = MCS_1 \cup MCS_0$,
where

$MCS_0 = MCS[F_0]$
$MCS_1 = \{v.\pi; \pi \in MCS[F_1+F_0]/MCS_0\}$

Second Decomposition:
$MCS[F] = MCS_1 \cup MCS_0$,
where

$MCS_0 = MCS[F_0]$

$MCS_1 = \{v.\pi; \pi \in MCS[F_1] \div MCS_0\}$

and $P \div Q = \{\pi \in P ; \rho \in Q, \rho$ is not included in $\pi\}$

Case 3: Neither v nor $\overline{v}$ belong to L. In that case,

$$MCS[F] = MCS[F_1] \cup MCS[F0]$$

25.3.5 Cutoffs, p-BDD and Direct Computations

The number of prime implicants of a Boolean function involving *n* variables is in $O(3^n)$, and $O(2^n)$ if the function is monotone [25]. There is no direct relationship between the size of the BDD encoding the function and the ZBDD encoding its prime implicants. Not much progress has been done in establishing this relationship (see however [26]). In practice, it is sometimes the case that building the BDD is tractable, while building the ZBDD is not. Non-coherent models are more specifically subject to this phenomenon.

When the ZBDD takes too much time or space to build, it is still possible to apply cutoffs to keep only PI or MCS whose size (or equivalently probability) is lower than a given threshold. To do so, it suffices to introduce the threshold into the decomposition theorem (see [11] for more details).

Moreover, rather than computing the BDD and then the ZBDD encoding MCS, it is possible either to compute a truncated BDD and then the ZBDD from this truncated BDD [11] or to compute directly the ZBDD encoding (the most important) MCS [27] (see Section 25.5.1).

25.4 Probabilistic Assessments

One of the main advantages, if not the main, of the BDD technology for reliability and dependability analyses stands in the accuracy and the efficiency of the assessment of probabilistic quantities. In this section, we present algorithms to compute top event probabilities, importance factors, and to perform time dependent analyses.

25.4.1 Probabilities of Top (and Intermediate) Events

Top event probabilities can be assessed either from BDD or from ZBDD. Since these data structures are based on different decomposition principles, algorithms used in each case are different. Exact computations are performed with BDD. Rare events approximation is applied on ZBDD similarly to what is done with an explicit encoding of MCS, but with a better efficiency, thanks to sharing. The algorithm to compute the probability of a gate from a BDD is based on the Shannon Decomposition. It is defined by the following recursive equations [6].

$$\text{BDD-Pr}(0) = 0.0$$
$$\text{BDD-Pr}(1) = 1.0$$
$$\text{BDD-Pr}(v.F_1 + \overline{v}.F_0) = p(v).\text{BDD-Pr}(F_1) + (1-p(v)).\text{BDD-Pr}(F_0)$$

As a consequence, the result is exact. Moreover, a caching mechanism is used to store intermediate results. Therefore, the algorithm is linear in the size of the BDD.

The algorithm to compute the top event probability from a ZBDD is based on the rare events approximation, i.e: $p(S) \approx \sum_{\pi \in MCS[S]} p(\pi)$. It is defined by the following recursive equations.

$$\text{ZBDD-Pr}(0) = 0.0$$
$$\text{ZBDD-Pr}(1) = 1.0$$
$$\text{ZBDD-Pr}(v.S_1 \cup S_0) = p(v).\text{ZBDD-Pr}(S_1) + \text{ZBDD-Pr}(S_0)$$

The corresponding algorithm is also linear in the size of the ZBDD which is in general much smaller than the size of the set of MCS. Therefore, even when the BDD technology just mimics MCS calculations, it improves significantly the efficiency of the approach.

The assessment of importance factors rely on the computation conditional probabilities $p(S|e)$ and $p(S|\overline{e})$, where S is a gate and e is a basic event. BDD makes it possible to compute both (thanks again to the Shannon decomposition). Recursive equations are as follows.

BDD-Pr(0|e) = 0.0
BDD-Pr(1|e) = 1.0
BDD-Pr($v.F_1 + \overline{v}.F_0$ |e) = // if (v<e)
p(v).BDD-Pr(F_1|e) + (1-p(v)).BDD-Pr(F_0|e)
BDD-Pr($v.F_1 + \overline{v}.F_0$ |e) = // if (v>e)
BDD-Pr($v.F_1 + \overline{v}.F_0$)
BDD-Pr($v.F_1 + \overline{v}.F_0$ |v) = BDD-Pr(F_1|v)
BDD-Pr($v.F_1 + \overline{v}.F_0 | \overline{v}$) = BDD-Pr(F_0| $\overline{v}$)

Again, the corresponding algorithm gives exact results and is linear in the size of the BDD. The ZBDD algorithm is very similar and does not deserve a further presentation.

25.4.2 Importance Factors

One of the principal activities of risk assessment is expected to be either the ranking or the categorization of structures, systems and components with respect to their risk-significance or their safety significance. Several measures of such significance have been proposed for the case where the support model is a fault tree. These measures are grouped under the generic designation of "importance factors". Many articles and book chapters have been devoted to their mathematical expressions, their physical interpretations, and the various ways they can be evaluated by computer programs (see, *e.g.*, [7, 8]). Seminal work on the use of BDD to assess importance factors has been done by J. Andrews and his students [28 and 12], followed by Dutuit and Rauzy [14]. In this section, we review the main importance factors and we discuss the BDD and ZBDD algorithms to assess them. Our presentation follows mainly the reference [14].

25.4.2.1 Marginal Importance Factor

The first importance factor to consider is the marginal importance factor, denoted by MIF(S,e) which is defined as follows.

$$\mathrm{MIF}(S,e) = \frac{\partial(p(S))}{\partial(p(e))}.$$

MIF is often called the Birnbaum importance factor in the literature [29]. It can be interpreted, when S is a monotone function, as the conditional probability that, given that e occurred, the system S has failed and e is critical, *i.e.,* a repair of e makes the system work. The following equalities hold.

$$\mathrm{MIF}(S,e) = \frac{\partial(p(S))}{\partial(p(e))}$$

$$= p(S_{[e\leftarrow 1]}.\overline{S_{[e\leftarrow 0]}}) \qquad \text{(i)}$$

$$= p(S/e) - p(S/\overline{e}) \qquad \text{(ii)}$$

Equality (i) holds only in the case of monotone functions. Equality (ii) can be used to compute MIF(S,e), by means of two conditional probability calculations. However, MIF(S,e) can also be computed in a single BDD traversal using the following recursive equations.

BDD-MIF(0|e) = 0.0
BDD-MIF(1|e) = 0.0
BDD-MIF($v.S_1 + \overline{v}.S_0$ |e) = // if e < v
p(v).BDD-MIF(S_1|e) + (1-p(v)).BDD-MIF(S_0|e)
BDD-MIF($v.S_1 + \overline{v}.S_0$ |e) = 0.0 // if e > v
BDD-MIF($e.S_1 + \overline{e}.S_0$ |e) =
BDD-Pr(S_1|e) - BDD-Pr(S_0| $\overline{e}$)

The corresponding algorithm is linear in the size of the BDD. A last way to compute MIF(S,e) is to use a numerical differentiation. The ZBDD algorithm to compute MIF(S,e) is very similar and is also linear in the size of the ZBDD.

25.4.2.2 Other Classical Importance Factors: CIF, DIF, RAW and RRW

Beyond the MIF, the most diffuse importance factors and their definitions are reported in Table 25.1. We refer to [14, 30, 31] for a thorough discussion of the definitions of CIF, DIF, RAW and RRW. BDD and ZBDD algorithms to compute these importance factors are derived straight from their definition and the rare event approximation, respectively.

It is worth noting that there are many real-life models for which the ranking obtained with BDD and ZBDD algorithms do not coincide (see, *e.g.*, [15]). The effects of such a discrepancy are ignored

by most of the practitioners and by regulation authorities.

25.4.2.3 Discussion

The reliability engineering literature often refers to another importance factor called Fussel–Vesely which is defined as follows.

$$\mathrm{FV}(S,e) = \frac{p\left(\bigcup_{e.\pi \in \mathrm{MCS}[S]} e.\pi\right)}{p(S)}$$

If the rare event approximation is applied and the numerator is approximated by $\sum_{e.\pi \in \mathrm{MCS}[S]} p(e.\pi)$, we have FV(S,e) = CIF(S,e). Many authors consider thus that these two measures are actually the same. However, FV(S,e) as defined above has no interpretation in terms of system state. One of the merits of the introduction of BDD algorithm has been to show this kind of incoherence in the mathematical foundations of reliability engineering.

Recently,E. Borgonovo and G.E. Apostolakis introduced a new importance measure, called the differential importance measure (DIM), which is defined as follows [32].

$$\mathrm{DIM}(S,e) = \frac{\frac{\partial p(S)}{\partial p(e)} \mathrm{d}\,p(e)}{\sum_v \frac{\partial p(S)}{\partial p(v)} \mathrm{d}\,p(v)}$$

DIM(S,e) has a number of interesting properties. It is additive, so the DIM of a group of basic events is the sum of their DIM. It is shown in [32] that if the dp(v) are all the same then DIM(S,e) is proportional to MIF(S,e) and if the dp(v) are proportional to the p(v), then DIM(S,e) is proportional to CIF(S,e). As noted by the authors, the calculations of DIM(S,e) can be done at almost no cost once either the MIF(S,e)'s or the CIF(S,e)'s have been computed (depending of the chosen type of variations).

Table 25.1. Importance factors

Importance factor	Symbol	Definition	Rare event approximation
Top event probability	p(S)	p(S)	$\sum_{\pi \in \mathrm{MCS}[S]} p(\pi)$
Marginal importance factor	MIF(S,e)	$\frac{\partial(p(S))}{\partial(p(e))}$	$\sum_{e.\pi \in \mathrm{MCS}[S]} p(\pi)$
Critical importance factor	CIF(S,e)	$\frac{p(e)}{p(S)} \times \mathrm{MIF}(S,e)$	$\frac{\sum_{e.\pi \in \mathrm{MCS}[S]} p(e.\pi)}{\sum_{\pi \in \mathrm{MCS}[S]} p(\pi)}$
Diagnostic importance factor	DIF(S,e)	$p(e\|S) = \frac{p(e) \times p(S\|e)}{p(S)}$	$p(e) \times \mathrm{RAW}(S,e)$
Risk achievement worth	RAW(S,e)	$\frac{p(S\|e)}{p(S)}$	$\frac{\sum_{\pi \in \mathrm{MCS}[S]} p(\pi[e \leftarrow 1])}{\sum_{\pi \in \mathrm{MCS}[S]} p(\pi)}$
Risk reduction worth	RRW(S,e)	$\frac{p(S)}{p(S\|\overline{e})}$	$\frac{\sum_{\pi \in \mathrm{MCS}[S]} p(\pi)}{\sum_{\pi \in \mathrm{MCS}[S]} p(\pi[e \leftarrow 0])}$

25.4.3 Time Dependent Analyses

Another important issue stands in so-called time dependent analyses. A fault tree model makes it possible to assess system availability, *i.e.*, the probability that the system works at time t. The system reliability, *i.e.*, the probability that the system worked without interruption from time 0 to time t, can only be approximated if the system involves repairable components.

There are industrial systems for which the latter notion is more relevant than the former, *e.g.*, the safety instrumented system with a high demand described in the norm CEI 61508. Other norms require the assessment of an equivalent failure rate for the system, which is strongly related to its reliability, as we shall see. The accuracy and efficiency of BDD calculations make it possible to design interesting algorithms to approximate both reliability and equivalent failure rate. The following presentation follows reference [33].

25.4.3.1 Assumptions, Definitions and Basic Properties

Throughout this section, we make the following assumptions.

Systems under study involve repairable and non-repairable components.

Each component e has two modes (working and failed), a failure rate λ_e and repair rate μ_e. If the component is non-repairable, μ_e is null. λ_e and μ_e are constant through time. The probability $Q_e(t)$ that the component has failed at time t is therefore obtained by the following equation [9].

$$Q_e(t) = \frac{\lambda_e}{\lambda_e + \mu_e} \times \left(1 - e^{-(\lambda_e + \mu_e).t}\right)$$

Components are independent, *i.e.*, both their failures and their repairs are statistically independent.

Components are as good as new after a repair. They are as good as new at time 0. Failures of systems under study are modeled by means of coherent fault trees. In the sequel, we assimilate systems with their fault trees.

As a consequence, systems under study can also be represented as Markov models. Let S denote the system under study. Let T denote the date of the first failure of S. T is a random variable. It is called the lifetime of S.

The *availability* $A_S(t)$ of S at t is the probability that S is working at t, given that all its components were working at 0. The *unavailability* $Q_S(t)$ is just the opposite.

$$A_S(t) \stackrel{def}{=} \Pr\{\text{S is working at t}\}$$

$$Q_S(t) \stackrel{def}{=} 1 - A_S(t)$$

The *reliability* $R_S(t)$ of S at t is the probability that S experiences no failure during time interval [0,t], given that all its components were working at 0. The *unreliability*, or cumulative distribution function $F_S(t)$, is just the opposite. Formally,

$$R_S(t) \stackrel{def}{=} \Pr\{t < T\}$$

$$F_S(t) \stackrel{def}{=} \Pr\{t \geq T\} = 1 - R_S(t)$$

The curve $R_S(t)$ is a survival distribution. This distribution is monotonically decreasing. Moreover, the following asymptotic properties hold.

$$\lim_{t \to 0} R_S(t) = 1$$

$$\lim_{t \to \infty} R_S(t) = 0$$

Note that $Q_S(t) \leq F_S(t)$, for general systems and that $Q_S(t) = F_S(t)$, for systems with only non-repairable components. The algorithms described in Section 25.4.1 compute $F_S(t)$.

The *failure density* $f_S(t)$ refers to the probability density function of the distribution of T. It is the derivative of $F_S(f)$:

$$f_S(t) \stackrel{def}{=} \frac{d\,F_S(t)}{dt}$$

For sufficiently small dt's, $f_S(t).dt$ expresses the probability that the system fails between t and t+dt, given its was working at time 0.

The *failure rate* or hazard rate $r_S(t)$ is the probability the system fails for the first time per unit of time at age t. Formally,

$$r_S(t) \stackrel{def}{=} \lim_{dt \to 0} \frac{\Pr\{\text{S fails btw. t and t + dt} / C\}}{dt}$$

where C denotes the event "the system experienced no failure during the time interval [0,t]". The following property is well known.

$$R_S(t) = \exp\left[-\int_0^t r_S(u)\,du\right]$$

The *conditional failure intensity* $\lambda_S(t)$ refers to the probability that the system fails per unit time at time t, given that it was working at time 0 and is working at time t. Formally,

$$\lambda_S(t) \stackrel{def}{=} \lim_{dt\to 0} \frac{\Pr\{\text{S fails btw. t and t + dt / D}\}}{dt}$$

where D denotes the event: "the system S was working at time 0 and is working at time t". The conditional failure intensity is sometimes called Vesely rate. $\lambda_S(t)$ is an indicator of how likely the system is to fail.

The *unconditional failure intensity* $w_S(t)$ refers to the probability that the system fails per unit of time at time t, given it was working at time 0. Formally,

$$w_S(t) \stackrel{def}{=} \lim_{dt\to 0} \frac{\Pr\{\text{S fails btw. t and t + dt / E}\}}{dt}$$

where E denotes the event "the system was working at time 0". Note that $w_S(t) = f_S(t)$ for systems with only non-repairable components. The following property is easily deduced from the definitions [33]:

$$\lambda_S(t) = \frac{w_S(t)}{A_S(t)}.$$

The unconditional failure intensity is sometimes called the "instantaneous equivalent failure rate". In some reliability studies, regulation authorities require the computation of its mean value through a period of time. This *mean equivalent failure rate* is defined as follows:

$$\lambda_S^{[Mean]}(t) = \frac{\int_0^t \lambda_S(t).dt}{t}.$$

25.4.3.2 Calculations

The set of states in which the system S has failed can be decomposed in three subsets: the set S_1 of states in which the repair of the component e repairs the system, the set S_0 of states in which the failure of e repairs the system, and the set S_2 of states in which the system has failed, whatever the state of the component e.

Since the system is assumed to be coherent, $S_0 = \varnothing$. It follows that $S[e\leftarrow 1]$ describes $S_1 \cup S_2$ and $S[c\leftarrow 0]$ describes S_2. Let $CRIT_{S,e}(t)$ denote the probability that system S is in a critical state w.r.t. the component e at time t, *i.e.* a state in which S has not failed and a failure of e induces a failure of S. From the above developments, the following equality holds:

$$CRIT_{S,e}(t) = A_e(t) \times MIF_{S,e}(t).$$

For a sufficiently small value of dt, the probability that the system fails between t and dt is as follows:

$$\Pr\{\text{the system fails between t and t + dt}\} \approx \sum_{e\in S} dt.\lambda_e.CRIT_{S,e}(t).$$

Therefore, assuming that the system was perfect at time 0, the following equality holds:

$$w_S(t) = \sum_{e\in S} MIF_{S,e}(t).w_e(t),$$

where $w_e(t) = \lambda_e.A_e(t)$. In reference [33], four approximations of the reliability are considered, each having its own merits and drawbacks. They are given (without justification) in Table 25.2 .

Table 25.2. Approximations of reliability

Name	Approximation of $F_S(t)$
Murchland	$\int_0^t w_S(t).dt$
Barlow–Proschan [34]	$t.\lambda_S(\infty)$
Vesely	$1-\exp\left[-\int_0^t \lambda_S(u)\,du\right]$
asymptotic Vesely	$1-e^{-\lambda_S(\infty).t}$

The assessment of both the reliability and of equivalent failure rate rely on the evaluation of $w_S(t)$. Moreover, lots of calculations are required for numerical integrations. That is the reason why the BDD technology improves significantly, here again, the engineering process. In order to implement these computations, we need basically two algorithms: an algorithm to assess $Q_S(t)$ and an

algorithm to assess $MIF_{S,e}(t)$ or equivalently $w_S(t)$. Integrals are computed numerically using in general triangular approximation. Sections 25.4.1 and 25.4.2 present the algorithms to evaluate $Q_S(t)$ and $MIF_{S,e}(t)$. However, $w_S(t)$ can be obtained by means of two traversals of the BDD which avoids performing a computation of $MIF_{S,e}(t)$ for each basic event e. The corresponding recursive equations are as follows (for the sake of simplicity, the time t is omitted).

$$\text{BDD-w}(1) = 0$$
$$\text{BDD-w}(0) = 0$$
$$\text{BDD-w}(e.S_1 + \bar{e}.S_0) =$$
$$w_e \times [\text{BDD-Pr}(S_1) - \text{BDD-Pr}(S_0)] + p(e) \times \text{BDD-w}(S_1) + (1-p(e)) \times \text{BDD-w}(S_0)$$

The algorithm derived from the above equations is linear in the size of the BDD and therefore does not depend on the number of components, conversely to the algorithm that calculates the MIF of each component.

25.5 Assessment of Large Models

The assessment with BDD of large models, especially large event trees coming from the nuclear industry, is still challenging. Two different approaches can be tried to handle this problem: the first one consists of using ZBDD to implement classical MCS algorithms. The second approach consists of designing heuristics and strategies to reduce the complexity of the BDD construction. In this section, we discuss both approaches.

25.5.1 The MCS/ZBDD Approach

Set operations (union, intersection, elimination of non-minimal cutsets…) can be performed on ZBDD in the same way logical operations are performed on BDD. It is therefore possible to design ZBDD algorithms to compute MCS, without constructing BDD at all. Moreover, cutoffs can be applied on each intermediate ZBDD. By tuning these cutoffs, one is always able to get a result. In practice, the obtained results are often accurate enough. Such algorithms can be seen as ZBDD implementations of classical MCS algorithms, with the advantage of a compact representation and efficient set operations. The algorithm proposed in [27] outperforms the classical MCS algorithm previously implemented in the same fault tree tool. This additional success of the BDD technology should not hide the problems inherent in this approach.

These problems come from three sources: first, the MCS approximation of the underlying function; second, the use of cutoffs; and third, the use of rare event approximations. Rare event approximation is not really an issue if basic event probabilities are low (say below 10^{-3}), which is in general the case. Moreover, it is conservative. The effect of the two other approximations is pictured in Figure 25.5.

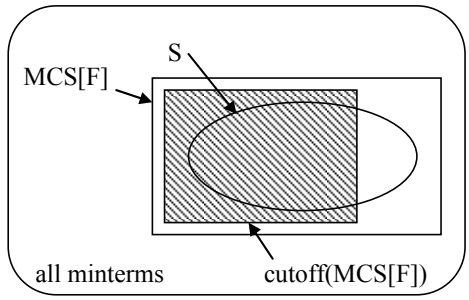

Figure 25.5. Effects of MCS approximation and cutoffs

Considering MCS[S] rather than the function S itself is an upper approximation of the function [11], *i.e.,* it is conservative. Note that MCS[S] = S if S is a coherent model. The use of cutoffs is an optimistic approximation of MCS[S]. It follows that if the model is not coherent, like success branches of event trees, what is computed and the actual function may be quite different. Epstein and Rauzy showed that the MCS approach may lead to erroneous results [15].

The determination of the "right" cutoffs requires some expertise and always results in a tradeoff between the complexity of calculations and the accuracy of the results [35, 36]. From a theoretical view point, nothing can prevent cutoffs from producing flawed results.

The MCS approach has another drawback: if the probability of a basic event varies (like in sensitivity analyses), the whole set of MCS should be recomputed which is indeed costly.

Despite all these problems, the MCS/ZBDD approach is the state-of-the-art technology used to assess large models of nuclear industry.

25.5.2 Heuristics and Strategies

Many techniques have been proposed to improve the construction of the BDD, including various preprocessing of the formula (rewritings, modularization), static and dynamic variable ordering heuristics, and strategies of construction. Dozens of articles have been published on those subjects. We give here only a few hints about their respective interest and limits.

Rewritings like those proposed in [37, 38] (coalescing, literal propagation, grouping of common factors, application of de Morgan's law) simplify formulae and must certainly be present in any fault tree assessment toolbox. However, they are not so easy to implement efficiently. Modularization [39, 40], which consists of isolating independent parts of the formulae, can also be helpful, but in our experience, large models have only few modules.

Most of the static variable ordering heuristics can be seen as two steps processes: first, arguments of gates are sorted according to some metrics and second, a depth first left most ordering is applied (*e.g.*, [20, 21]). This later ordering is interesting because it tends to keep close related variables. Static variable ordering heuristics can improve significantly the BDD construction. However, they are not versatile: a heuristic may be very efficient on one model and quite bad on another one. In [41], it is suggested to embed these heuristics into strategies. The idea is to rearrange more or less at random the arguments of gates and then to apply a heuristics. If the obtained formula is not tractable in a limited amount of time or space, then a new rearrangement is tried, and so on. The allowed time and space for each try may evolve based on the results of the first tries. This kind of strategy has been applied successfully to several large fault trees.

Dynamic variable reordering heuristics (see, *e.g.*, [3 and 22]) are considered as a significant improvement of the BDD technology when applied to logical circuits. Their application to large reliability models has been up to now quite deceptive: they are much too time consuming. It is often better to restart the calculation from scratch with a new candidate order, as suggested in [41].

Dramatic improvements can be obtained by combining the above ideas and there is room for innovation. Assessing large event trees of nuclear industry is still challenging.

25.6 Conclusions

Since their introduction in reliability and dependability studies, binary decision diagrams (BDD) have proved to be of great interest from both a practical and theoretical point of view. Not only does this technology provide more efficient and more accurate algorithms, but also mathematical foundations have been questioned and more firmly established.

In this chapter, we presented various aspects of the use of BDD for reliability and dependability studies. First, we discussed the mathematical basis of the notion of minimal cutsets (MCS). We described BDD algorithms to compute and to store MCS. Second, we reviewed the notion of importance factors. We gave recursive equations from which linear time BDD algorithms to compute MIF, CIF, DIF, RAW and RRW are easily derived. Third, we studied so-called time dependent analyses, which include approximations of reliability and computation of equivalent failure rates. Finally, we reviewed various techniques that can be used to handle large models.

The introduction of BDD in the reliability engineering framework has been successful. However, large event tree models coming from the nuclear industry are still out of the reach of an exact evaluation. It can be argued that these models are anyway too large to be mastered and that approximate computations are good enough. Nevertheless, the design of heuristics and strategies to handle these models and the integration of these

techniques into user friendly toolboxes would be a real accomplishment.

References

[1] Bryant R. Graph based algorithms for boolean function manipulation. IEEE Transactions on Computers 1986; 35(8):677–691.

[2] Brace K, Rudell R, Bryant R. Efficient implementation of a BDD package. In Proceedings of the 27th ACM/IEEE Design Automation Conference, IEEE 1990; 0738.

[3] Rudell R. Dynamic Variable ordering for ordered binary decision diagrams. In Proceedings of IEEE International Conference on Computer Aided Design, ICCAD 1993; Nov.:42–47.

[4] Bryant R. Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys 1992; Sept. 24:293–318.

[5] Coudert O, Madre J.-C. Fault tree analysis: 10^{20} prime implicants and beyond. In Proceedings of the Annual Reliability and Maintainability Symposium, ARMS'93, Atlanta NC, USA. 1993; January.

[6] Rauzy A. New algorithms for fault trees analysis. Reliability Engineering and System Safety 1993; 59(2):203–211.

[7] Vesely WE, Goldberg FF, Robert NH, Haasl DF. Fault tree handbook. Technical report NUREG 0492, U.S. Nuclear Regulatory Commission 1981.

[8] Høyland A, Rausand M. System reliability theory. John Wiley & Sons, 1994; ISBN 0–471-59397.

[9] Kumamoto H, Henley EJ. Probabilistic risk assessment and management for engineers and scientists. IEEE Press, 1996; ISBN 0–7803-6017-6.

[10] Minato S.-I. Binary decision diagrams and applications to VLSI CAD. Kluwer, Dordrecht, 1996; ISBN 0–7923-9652-9.

[11] Rauzy A. Mathematical foundation of minimal cutsets. IEEE Transactions on Reliability 2001; 50(4):389–396.

[12] Sinnamon RM, Andrews JD. Improved accuracy in qualitative fault tree analysis. Quality and Reliability Engineering International 1997; 13:285–292.

[13] Sinnamon RM, Andrews JD. Improved efficiency in qualitative fault tree analysis. Quality and Reliability Engineering International 1997; 13:293–298.

[14] Dutuit Y, Rauzy A. Efficient algorithms to assess components and gates importance in fault tree analysis. Reliability Engineering and System Safety 2000; 72(2):213–222.

[15] Epstein S, Rauzy A. Can we trust PRA? Reliability Engineering and System Safety 2005; 88(3):195–205.

[16] Papadimitriou CH. Computational complexity. Addison Wesley, Reading, MA, 1994; ISBN 0-201-53082-1.

[17] Friedman SJ, Supowit KJ. Finding the optimal variable ordering for binary decision diagrams. IEEE Transactions on Computers 1990; 39(5):710–713.

[18] Bollig B, Wegener I. Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. on Software Engineering 1996; 45(9):993–1001.

[19] Aloul FA, Markov IL, Sakallah KA. FORCE: A fast and easy-to-implement variable-ordering heuristic. Proceedings of GLVLSI 2003.

[20] Fujita M, Fujisawa H, Kawato N. Evaluation and improvements of Boolean comparison method based on binary decision diagrams. In Proceedings of IEEE International Conference on Computer Aided Design, ICCAD 1988; 2–5.

[21] Fujita M, Fujisawa H, and Matsugana Y. Variable ordering algorithm for ordered binary decision diagrams and their evaluation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 1993; 2(1):6–12.

[22] Panda S, Somenzi F. Who are the variables in your neighborhood. In Proceedings of IEEE International Conference on Computer Aided Design, ICCAD 1995; 74–77.

[23] Yau M, Apostolakis G, Guarro S. The use of prime implicants in dependability analysis of software controlled systems. Reliability Engineering and System Safety 1998; 62:23–32.

[24] Morreale E. Recursive operators for prime implicant and irredundant normal form determination. IEEE Transactions on Computers 1970; C-19(6):504–509.

[25] Chandra AK, Markowsky G. On the number of prime implicants. Discrete Mathematics 1978; 24:7–11.

[26] Hayase K, Imai H. OBDDs of a monotone function and its prime implicants. Theory of Computing Systems 1998; 41:579–591.

[27] Jung WS, Han SH, Ha J. A fast BDD algorithm for large coherent fault trees analysis. Reliability Engineering and System Safety 2004; 83:69–-374.

[28] Sinnamon RM, Andrews JD. Quantitative fault tree analysis using binary decision diagrams. Journal Européen des Systèmes Automatisés, RAIRO-APII-JESA, Special Issue on Binary Decision Diagrams 1996; 30:1051–1072.

[29] Birnbaum ZW. On the importance of different components and a multicomponent system. In: Korishnaiah PR, editor. Multivariable analysis II. Academic Press, New York, 1969.

[30] Cheok MC, Parry GW, Sherry RR. Use of importance measures in risk informed regulatory applications. Reliability Engineering and System Safety 1998; 60:213–226.

[31] Vesely WE. Supplemental viewpoints on the use of importance measures in risk informed regulatory applications. Reliability Engineering and System Safety 1998; 60:257–259.

[32] Borgonovo E, Apostolakis GE. A new importance measure for risk-informed decision making. Reliability Engineering and System Safety 2001; 72(2):193–212.

[33] Dutuit Y, Rauzy A. Approximate estimation of system reliability via fault trees. Reliability Engineering and System Safety 2005; 87(2):163–172.

[34] Barlow RE, Proschan F. Theory for maintained system: Distribution of time to first failure. Mathematics of Operation Research 1976; 1:32–42.

[35] Čepin M. Analysis of truncation limit in probabilistic safety assessment. Reliability Engineering and System Safety 2005; 87(3):395–403.

[36] Jung WS, Han SH. Development of an analytical method to break logical loops at the system level**.** Reliability Engineering and System Safety 2005; 90(1):37–44.

[37] Camarinopoulos L, Yllera J. An improved top-down algorithm combined with modularization as highly efficient method for fault tree analysis. Reliability Engineering and System Safety 1985; 11:93–108.

[38] Niemelä I. On simplification of large fault trees. Reliability Engineering and System Safety 1994; 44:135–138.

[39] Chatterjee P. Modularization of fault trees: A method to reduce the cost of analysis. Reliability and Fault Tree Analysis, SIAM 1975; 101–137.

[40] Dutuit Y, Rauzy A. A linear time algorithm to find modules of fault trees. IEEE Transactions on Reliability 1996; 45(3):422–425.

[41] Bouissou M, Bruyère F, Rauzy A. BDD based fault-tree processing: A comparison of variable ordering heuristics. In: Soares C Guedes, editor. Proceedings of European Safety and Reliability Association Conference, ESREL, Pergamon, London, 1997; 3(ISBN 0–08–042835–5):2045–2052.