Mathematical Foundations of Minimal Cutsets

A. Rauzy

Abstract— **Since their introduction in the reliability field, Binary Decision Diagrams have proved to be the most efficient tool to assess Boolean models such as fault trees. Their success increases the need of sound mathematical foundations for the notions that are involved in reliability and dependability studies. The aim of this article is to clarify the mathematical status of the notion of minimal cutsets, that plays a central role in fault tree assessment. Algorithmic issues are also discussed.**

Keywords— **Boolean Algebrae, Fault Trees, Minimal Cutsets, Binary Decision Diagrams**

I. Introduction

Bryant's Binary Decision Diagrams (BDDs [1], [2]) are the state-of-the-art data structure to handle boolean functions. Since their introduction in the reliability field (the very first papers on this topics were [3] and [4]) they have proved to be the most efficient tool to assess Boolean models such as fault trees. First, BDDs make it possible to assess the top event probability in an exact and very efficient way [4]. Second, they can be used to compute and to encode very large sets of minimal cutsets [5], [6]. Their success increases the need of sound mathematical foundations for the notions that are involved in reliability and dependability studies: BDD algorithms are derived from recursive equations that apply on the Shannon decomposition of the formula under study. These recursive equations are obtained from the formal definitions of the quantities of interest (see, for instance, [6], [7]). They are, in general, different from those used to design more classical algorithms based on formula rewriting or on the Sylvester-Poincaré development. In other words, the design of BDD algorithms requires to separate clearly mathematical definitions from algorithmic issues.

The aim of this article is to clarify the mathematical status of the notion of minimal cutsets. Minimal cutsets play an important role in fault tree assessment. They are the basis for qualitative analyses for they represent minimal *scenarii* of failure. Moreover, they are used in several probabilistic computations. Although BDD encoding of fault trees makes them useless to assess the top event probability, there remains many probabilistic measures that rely on minimal cutsets: contribution of individual *scenarii* of failure, Fussel-Vesely importance factor [8], . . .

Intuitively, a minimal cutset is a minimal set of basic events of the tree that induces its top event. For coherent fault trees, this informal definition fits well with the formal notion of prime implicant. This is not the case for non coherent fault trees because prime implicants of such trees may contain negative literals. This raises the question whether minimal cutsets are a convenient, but definitely informal, approximation of prime implicants or a sound mathematical notion for their own. We advocate here that minimal cutsets have a well defined logical status, distinct from the notion of prime implicant. This status captures and generalises the above intuition. Minimal cutsets are actually prime implicants of a formula that is obtained from the formula under study by widening it with respect to a set of significant literals. Several structural theorems can be established that give insights about the actual nature of minimal cutsets. These theorems are used to design BDD algorithms.

Nowadays, fault trees with several hundred gates and basic events have to be assessed. Despite of their great efficiency, BDDs sometimes fail to handle such models because they cannot avoid the exponential blow up that results from manipulation of such large numbers of gates and events. Therefore, approximations have to be made. For instance, a commonly used approximation consists in considering short minimal cutsets only. This is justified by the fact that these cutsets capture, in general, the main part of the top-event probability. Indeed, this raises questions about the quality of the results. We clarify here the mathematical status of this kind of approximations through the introduction of a family of Boolean sub-algebrae. We show its interest from both a computational complexity and a practical point of views. We propose BDD algorithms to perform approximate computations.

In a word, the aim of this article is to revisit the notion of minimal cutsets in order to give it sound mathematical foundations and, by the way, to make it possible to design BDD algorithms. It is organized as follows. In section III, we recall basics about Boolean algebrae, that are the mathematical foundations of the notions we deal with. In section IV, we propose a formal definition of minimal cutsets and we discuss it. In section V, we present briefly BDDs and ZBDDs and we show the decomposition theorem that is used to design an algorithm to compute minimal cutsets. Finally, in section VI, we propose a formal framework to perform approximate computations of both the top event probability and the minimal cutsets of fault trees.

II. Notations

Boolean variables: a, b, c, v, v_1, v_2, . . .
Sets of Boolean variables: $\mathcal{V}$ (denumerable), V (finite)
Literals: v, $\bar{v}$, p, $\bar{p}$, . . .
Set of literals: L, . . .
Products & minterms: π, ρ, . . .
Sets of products: S, T, . . .
Boolean formulae: F, F_1, F_0, G, . . .
Logical operations: "." (and), "+" (or), "‾" (not)
Set operations: "$\cap$" (union), "$\cup$" (intersection), "$\complement$" (complement), $\setminus$ (set difference)
Power set of a set X: 2^X

III. Boolean Algebrae

This section recalls basics about Boolean algebrae, including the definitions of Boolean formulae, literals, products, minterms, assignments and Boolean algebrae.

A. Boolean Formulae

Boolean formulae are built over the two constants 0 (false) and 1 (true), a denumerable set of variables $\mathcal{V} = \{v_1, v_2, \dots\}$, and the usual logical connectives "." (and), "+" (or), "$^-$" (not). The set of Boolean formulae is the smallest set such that :

– 0, 1 are formulae,

– the variables of $\mathcal{V}$ are formulae,

– if F and G are formulae, then so are $F.G$, $F + G$, $\bar{F}$.

The set of variables that occur in the formula F is denoted by $var(F)$. The set of Boolean formulae that can be built over a finite subset V of $\mathcal{V}$ is denoted by $\mathcal{F}(V)$.

B. Literals, Products, Minterms, Assignments

A *literal* is either a Boolean variable v or its negation $\bar{v}$. v is a *positive literal*, $\bar{v}$ is a *negative literal*, they are said *opposite*. The opposite of a literal p is denoted by $\bar{p}$ ($\bar{\bar{p}} = p$).

A *product* is a set of literals that does not contain both a literal and its opposite. A product is assimilated with the conjunction of its elements. We denote by $|\pi|$ the *order*, i.e. number of literals, of a product π.

Let V be a finite set of variables. A product that contains a literal built over each variable of V is called a *minterm* of V. We denote by $minterms(V)$ the set of minterms that can be built over V.

An *assignment* over $\mathcal{V}$ is any mapping from $\mathcal{V}$ to $\{0, 1\}$. Assignments are extended inductively into mappings from Boolean formulae into $\{0, 1\}$. Let σ be an assignment and let F and G be two formulae, then $\sigma[F.G] = min(\sigma[F], \sigma[G])$, $\sigma[F + G] = max(\sigma[F], \sigma[G])$ and $\sigma[\bar{F}] = 1 - \sigma[F]$.

There is a one to one correspondance between minterms over a finite set of variables V and assignments (restricted to V): a variable occurs positively in a minterm if and only if it receives the value 1 in the corresponding assignment. An assignment (or equivalently a minterm) σ satisfies a formula F if $\sigma[F] = 1$, otherwise it falsifies F.

Let F and G be two formulae. If any assignment satisfying F satisfies G as well, we say that F *implies* G, which is denoted by $F \models G$.

A formula F is monotone if for any minterm $\bar{v}.\pi$ that satisfies F, the minterm $v.\pi$ satisfies F as well. Coherent fault trees are monotone formulae since they are built just with and gates, or gates and k-out-of-n gates.

C. Boolean Algebrae

It is often convenient to consider Boolean formulae as sets of minterms and therefore to consider logical operations as set operations. This relies on the well known Stone's theorem about Boolean algebrae.

Let V be a finite set of Boolean variables. The set $\mathcal{F}(V)$ together with the constants 0 and 1 and the operations ., + and $^-$ forms a *Boolean algebra*. The power set $2^{minterms(V)}$ together with the constants $\emptyset$ and $minterms(V)$ and the set operations $\cap$ (intersection), $\cup$ (union) and $\complement$ (complementation) forms also a Boolean algebra, which is actually isomorphic (by the Stone's theorem) to the previous one: a Boolean formula corresponds to the set of minterms that "satisfy" it. The formula $\bar{F}$ corresponds to the complement of the set of minterms that corresponds to F. The formulae $F.G$ and $F + G$ correspond to respectively the intersection and the union of the sets of minterms that correspond to F and G.

Let, for instance, F be the formula $ab + \bar{a}c$. F and its subformulae are satisfied by the following minterms (over $\{a, b, c\}$).

formulae	minterms
a	$abc + ab\bar{c} + a\bar{b}c + a\bar{b}\bar{c}$
b	$abc + ab\bar{c} + \bar{a}bc + \bar{a}b\bar{c}$
c	$abc + a\bar{b}c + \bar{a}bc + \bar{a}\bar{b}c$
$\bar{a}$	$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}\bar{c}$
ab	$abc + ab\bar{c}$
$\bar{a}c$	$\bar{a}bc + \bar{a}\bar{b}c$
$ab + \bar{a}c$	$abc + ab\bar{c} + \bar{a}bc + \bar{a}\bar{b}c$

IV. Minimal Cutsets

This section presents the mathematical foundations of the notion of minimal cutsets.

A. Preliminaries

The notion of minimal cutsets seems clear to everybody who works with fault trees. A minimal cutset represents a minimal *scenario* of failure, i.e. a set (a conjunction) of basic events that induces the top event and that is minimal w.r.t. set inclusion. This definition is correct when applied to coherent fault trees. However, it is not correct when applied to non-coherent fault trees. For this later case, the notion of prime implicant should be substituted for the notion of minimal cutset. Prime implicants are sets of literals, i.e. they may contain negated variables.

To illustrate the above discussion, let us consider again the formula $F = ab + \bar{a}c$. As we shall see, the prime implicants of F are ab, $\bar{a}c$ and bc. This does correspond to the notion of minimal solution of F, but this does not correspond to the intuitive notion of minimal cutsets. The expected minimal cutsets are ab and c.

In reliability models, there is a foundamental asymetry between positive and negative literals. Since positive literals represent unexpected (and often undesirable and rare) events such as failures, they are in some sense the only ones of interest. This is the reason why most of the fault tree assessment tools (including electronic simulators such as ESCAF [9] and softwares such as Risk-Spectrum [10]) never produce minimal cutsets with negative literals. They produce only something which is related to positive parts of prime implicants.

This raises the question whether minimal cutsets are a convenient, but definitely informal, approximation of prime implicants or a sound mathematical notion for their own. We advocate here the later answer. To start with, let us recall the notion of prime implicant.

B. *Prime Implicants*

Let F be a Boolean formula and π be a product. π is an *implicant* of F if $\pi \models F$. Let F be a Boolean formula and π be an implicant of F. Such an implicant π is said *prime* if there is no implicant ρ of F such that $\rho \subset \pi$. We denote by $PI[F]$ the set of prime implicants of the formula F.

Consider for instance the formula $F = ab+\bar{a}c$. F admits 7 implicants ab, abc, $ab\bar{c}$, $\bar{a}bc$, $\bar{a}c$, $\bar{a}\bar{b}c$ and bc and 3 prime implicants ab and $\bar{a}c$, bc.

C. *Minimal Custsets*

Let V be a finite set of variables, let L be a subset of the literals that may built over V, finally let F be a formula built over V.

We shall define minimal cutsets of F as minimal solutions from which literals outside L are remove because they "do not matter". Intuitively, a cutset is a product that contains only literals from L and that can be completed with literals not in L in order to give a implicant of F.

L is typically the set of all of the positive literals for, as discussed above, these literals are often the only ones that actually bring an information. Note however that some negative literals may be of interest as well, for instance because the corresponding basic events are of high probabilities. These negative literals must be kept in L. In other words, L is the set of significant literals.

From now, we shall say that a literal p is *significant* if it belongs to L and that it is *critical* if it is significant and while its opposite is not.

A first way to define minimal cutsets is derived straight from the idea to keep only significant parts of prime implicants. Let $PI_L[F]$ be the set of products obtained first by removing from products of $PI[F]$ the literals not in L and second by removing from the resulting set the non minimal products. Formally, $PI_L[F]$ is defined as follows.

$$PI_L[F] \overset{\text{def}}{=} \{\pi \cap L;\ \begin{array}{c} \pi \in PI[F] \\ \wedge\ \nexists \rho \in PI[F] \text{ s.t. } \rho \cap L \subset \pi \cap L \end{array}\}$$

This first definition captures actually 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 has the drawback to be based on the definition of prime implicants. This makes it not suitable to design an algorithm to compute minimal cutsets without computing prime implicants.

A second way to define minimal cutsets that avoids this drawback is as follows.

Let $\sqsubseteq_L$ be the binary relation among opposite literals defined as follows. $p \sqsubseteq_L \bar{p}$ if $p \notin L$. $\sqsubseteq_L$ is extended into a binary relation over $minterms(V)$ as follows. $\pi \sqsubseteq_L \rho$ if for any variable v, $\pi[v] \sqsubseteq_L \rho[v]$, where $\pi[v]$ (resp. $\rho[v]$) denotes the literal built over v that belongs to π (resp. to ρ). Intuitively, $\pi \sqsubseteq_L \rho$ when π is less significant than ρ.

$\sqsubseteq_L$ is both reflexive ($\pi \sqsubseteq_L \pi$, for any π) and transitive ($\pi \sqsubseteq_L \sigma$ and $\sigma \sqsubseteq_L \rho$ implies $\pi \sqsubseteq_L \rho$, for any π, σ and ρ). Therefore, $\sqsubseteq_L$ is a pre-order.

Let π be a product over V that contains significant literals only. π is a *cutset* of F w.r.t. L if it fulfils the first of the two following requirements, it is *minimal* if it fulfils the second one.

1. For any minterm σ such that $\pi \subseteq \sigma$, there exists a minterm δ such that $\delta \sqsubseteq_L \sigma$ and $\delta \models F$.
2. There is no cutset ρ such that $\rho \subset \pi$.

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 pairwisely incomparable. Therefore, $MC_L[F] = PI[F]$.
- If $L = \{a, b, c\}$, $ab\bar{c} \sqsubseteq_L abc$ and $\bar{a}\bar{b}c \sqsubseteq_L abc, a\bar{b}c, \bar{a}bc$, therefore the cutsets of F w.r.t. L are abc, ab, ac, bc and c and $MC_L[F] = \{ab, c\}$.
- 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 minimal cutsets are actually equivalent, as asserted by the following theorem.

Theorem 1 (Equivalence of minimal cutsets definitions) Let F be a Boolean formula and let L be a set of literals built over $var(F)$. Then, the following equality holds.

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

The proof is a straight application of the definitions.

Note finally that if $L = V$, a positive product π is a cutset if and only if the minterm $\pi \cup \{\bar{v}; v \in V \wedge v \notin \pi\}$ is an implicant of F. In this case, minimal cutsets of F are what we called minimal p-cuts in [6], [11].

D. *The widening operator ω_L*

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 set L of literals.

ω_L is an endomorphism of the Boolean algebra $(minterms(V), \cap, \cup, \complement)$ that associates to each set of minterms (formula) F the set of minterms $\omega_L(F)$ defined as follows.

$$\omega_L(F) \overset{\text{def}}{=} \{\pi; \exists \rho \text{ s.t. } \rho \sqsubseteq_L \pi \wedge \rho \models F\}$$

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

Let us consider again the formula $F = ab + \bar{a}c$.

- If $L = \{a, \bar{a}, b, \bar{b}, c, \bar{c}\}$, then $\omega_L(F) = F$.
- If $L = \{a, b, c\}$, then $\omega_L[F] = abc+ab\bar{c}+a\bar{b}c+\bar{a}bc+\bar{a}\bar{b}c$.
- If $L = \{b, \bar{b}, c, \bar{c}\}$, then $\omega_L[F] = abc + ab\bar{c} + a\bar{b}c + \bar{a}bc + \bar{a}\bar{b}c + \bar{a}\bar{b}c$.

The operator ω_L has a number of interesting properties, that are summarized by the following theorem.

Theorem 2 (The widening operator $\omega_{\sqsubseteq}$) Let F be a Boolean formula and let L be a set of literals built over $var(F)$. Then, the following facts hold.

- ω_L is indempotent: $\omega_L(\omega_L(F)) = \omega_L(F)$.
- $PI[\omega_L(F)] = MC_L[F]$.

The theorem 2 shows 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.

The theorem 2 gives also a third way to define minimal cutsets: the minimal cutsets of a formula F are the prime implicants of a pessimistic 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.

V. Binary Decision Diagrams

Due to space limitation, we recall in this section only basics about BDDs. The reader interested by a more detailed presentation should see for instance (see for instance [12], [13], [14].

A. Regular BDDs

The BDD associated with a formulae is a compact encoding of the truth table of this formula. This representation is based on the Shannon decomposition.

Let F be a Boolean formula that depends on the variable v. There exists two formulae F_1 and F_0 not depending on v such that:

$$F \quad = \quad v.F_1 + \bar{v}.F_0 \tag{1}$$

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 that points to the node that encodes F_1, and a *else*-outedge that points to the node that encodes F_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 $ab + \bar{a}c$ and the lexicographic order is pictured Fig. 1 (dashed lines represent *else*-outedges).

Indeed, such a representation is very expensive. It is however possible 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 unique son ($v.F + \bar{v}.F = F$).

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 [1]. This process is illustrated on Fig. 1.

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

$$\begin{array}{cc} & (v.F_1 + \bar{v}.F_0) \\ \odot & (v.G_1 + \bar{v}.G_0) \end{array} \quad = \quad \begin{array}{cc} & v.(F_1 \odot G_1) \\ + & \bar{v}.(F_0 \odot G_0) \end{array} \tag{2}$$

where $\odot$ is any binary connective. In order to compute $F \odot G$, one computes first $F_1 \odot G_1$ and $F_0 \odot G_0$ and then one composes the results. Terminal cases of this recursive principle are given by Boolean simplification rules, e.g. $F + 1 = 1$, $F.0 = 0$, $F.F = F$, ...

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 BDDs encoding its subformulae. 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. Negation is even more efficiently performed for it suffices to add flags on edges to get it in constant time. A complete implementation of a BDD package is described in [2]. The reader interested in details should thus refer to this article.

To end this short presentation, it should be noticed that almost all of the BDD algorithms are fully described by means of equations such as eq. 2 that recurse over the Shannon decomposition (eq. 1). For instance, the algorithm that computes the probability of the top-event from basic event probabilities is described as follows.

$$p(v.F_1 + \bar{v}.F_0) \quad = \quad p(v).p(F_1) + [1 - p(v)].p(F_0) \tag{3}$$

B. Zero-Suppressed BDDs

Working with minimal cutsets requires to encode sets of products. BDDs are not sufficient to encode such sets. This is because a product may contain a variable v either positively (v), or negatively ($\bar{v}$) or not at all. Therefore, ternary decision diagrams would be necessary to encode sets of products (this was actually proposed by Sasao in [15]).

Another (and better) solution consists in using BDDs, but with another semantics. This is the notion of Zero-suppressed BDD (ZBDD) introduced by Minato in [16]. The idea is to label nodes with literals and to decompose sets of products according to the presence of a given literal:

- The leaves **1** and **0** encode respectively the sets $\{\emptyset\}$ and $\emptyset$.
- A node $n = (p, n_1, n_0)$ encodes the set $S = \{\{p\} \cup \pi; \pi \in S_1\} \cup S_0$, where p is the literal that labels the node and S_1 and S_0 are respectively the sets encoded by its two sons n_1 and n_0.

This representation for sets of products is also canonical, although it requires to change the second reduction rule: useless nodes are those of the form $(p, \mathbf{0}, n)$.

As logical operations on BDDs, set operations (union, intersection, set difference) are of polynomial worst case complexity on ZBDDs (thanks to the caching principle already mentioned).

Thanks to the sharing of isomorphic sub-trees, BDDs and ZBDDs make it possible to encode huge functions

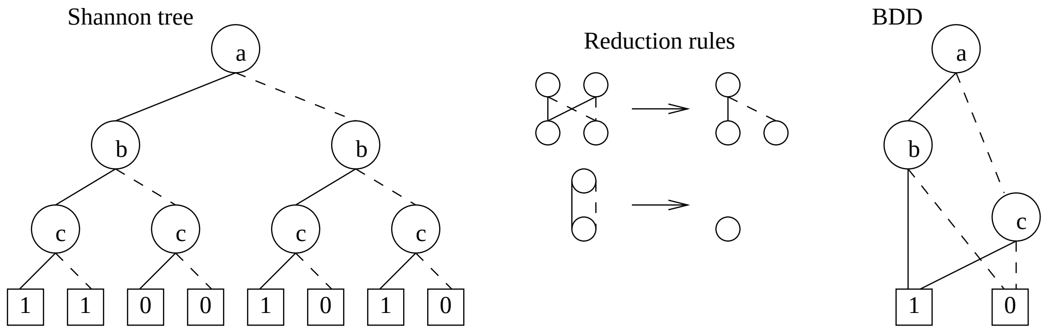

Fig. 1. From the Shannon tree to the BDD encoding $ab + \bar{a}c$.

(resp. sets of products) with relatively few nodes. Indeed, this does not work always (it is even easy to show that this does work for most of the functions), but this does work often in practice.

C. Decomposition Theorem

Let us present now the decomposition theorem which is the core of BDD algorithms to compute minimal cutsets. The principle of this algorithm is to traverse the BDD that encodes the formula F in a depth-first way and to build the ZBDD that encodes the minimal cutsets of F in a bottom-up way, by collecting the result of the BDD traversal.

In the sequel, $p.S$, where p is a literal and S is a set of products in which p and its opposite do not occur, denotes the following set of products.

$$p.S \stackrel{\text{def}}{=} \{\{p\} \cup \pi; \pi \in S\}$$

The following decomposition theorem generalizes both the theorem for prime implicant by Morreale [17] and our theorem for minimal p-cuts [6]. It can be stated as follows.

Theorem 3 (Decomposition Theorem) Let $F = v.F_1 + \bar{v}.F_0$ be the Shannon decomposition of a Boolean formula F w.r.t. a variable v and let L be a set of literals built over $var(F)$. Then, $MC_L[F]$ can be obtained as the union of several sets depending on the literals built over v that belong to L.

case 1: $v \in L$, $\bar{v} \in L$. In this case, $MC_L[F]$ is the union of three sets:

$$MC_L[F] = v.S_1 \cup \bar{v}.S_0 \cup S_2$$

where, S_1, S_0 and S_2 are defined as follows.

$$\begin{aligned} S_2 &\stackrel{\text{def}}{=} MC_L[F_1.F_0] \\ S_1 &\stackrel{\text{def}}{=} MC_L[F_1] \setminus S_2 \\ S_0 &\stackrel{\text{def}}{=} MC_L[F_0] \setminus S_2 \end{aligned}$$

case 2: $v \in L$, $\bar{v} \notin L$. In this case, $MC_L[F]$ is the union of two sets:

$$MC_L[F] = v.S_1 \cup S_0$$

where, S_1 and S_0 are defined as follows.

$$\begin{aligned} S_0 &\stackrel{\text{def}}{=} MC_L[F_0] \\ S_1 &\stackrel{\text{def}}{=} MC_L[F_1 + F_0] \setminus S_0 \\ \text{or } S_1 &\stackrel{\text{def}}{=} MC_L[F_1] \div S_0 \end{aligned}$$

where $S \div T$ denotes the following set.

$$S \div T \stackrel{\text{def}}{=} \{\pi \in S; \nexists \rho \in T, \rho \subseteq \pi\}$$

case 3: $v \notin L$, $\bar{v} \in L$. Similar to case 2.
case 4: $v \notin L$, $\bar{v} \notin L$. In this case, $MC_L[F]$ is as follows.

$$MC_L[F] = MC_L[F_1 + F_0]$$

The case 1 corresponds to Morreale's theorem [17]. The case 2 corresponds to our theorem [6]. More details about the principle of this algorithm and the operator $\div$ can be found in [3], [6].

Consider again the formula $F = ab + \bar{a}c$, which is already Shannon decomposed (for the lexicographic order).

- If $L = \{a, \bar{a}, b, \bar{b}, c, \bar{c}\}$ (case 1), then $MC_L[F] = a.S_1 \cup \bar{a}.S_0 \cup S_2$, where
  - $S_2 = MC_L[bc] = \{bc\}$
  - $S_1 = MC_L[b] \setminus S_2 = \{b\} \setminus \{bc\} = \{b\}$
  - $S_0 = MC_L[c] \setminus S_2 = \{c\} \setminus \{bc\} = \{c\}$

  Therefore, $MC_L[F] = \{ab, \bar{a}c, bc\}$.
- If $L = \{a, b, c\}$ (case 2), then $MC_L[F] = a.S_1 \cup S_0$, where
  - $S_0 = MC_L[c] = \{c\}$
  - $S_1 = MC_L[b + c] \setminus S_0 = \{b, c\} \setminus \{c\} = \{b\}$ or $S_1 = MC_L[b] \div S_0 = \{b\}$

  Therefore, $MC_L[F] = \{ab, c\}$.
- If $L = \{b, \bar{b}, c, \bar{c}\}$ (case 4), then $MC_L[F] = MC_L[b + c] = \{b, c\}$.

VI. Approximate computations

In this section, we propose an algebraic framework to perform approximate computations of both the top event probability and the minimal cutsets of fault tree (coherent or not). This framework is based on the notion of sub-algebra.

A. Sub-Algebrae

Let X be a subset of a set Y, we denote by C_X the unary operation from 2^Y to 2^Y defined as follows.

$$\mathsf{C}_X(Z) \overset{\text{def}}{=} \mathsf{C}(Z) \cap X$$

Let V be a finite set of variables and let A be a subset of $minterms(V)$. 2^A together with the constant $\emptyset$ and A and with the operations $\cup$, $\cap$ and C_A forms a Boolean algebra. It is actually a sub-algebra of the propositional calculus over V. There is a canonical homomorphism h_A from $2^{minterms(V)}$ into 2^A: $h_A(F) \overset{\text{def}}{=} F \cap A$. In the reverse direction, the identity is a homomorphism that comes back from 2^A to $2^{minterms(V)}$.

Assume that A contains most of the minterms of interest and that A is not too large, i.e. it is polynomially bounded in the number of variables. Then, it may be a safe approximation to interpret the functions under study in the sub-algebra $(A, \cap, \cup, \mathsf{C}_A)$ rather than in the regular algebra $(minterms(V), \cap, \cup, \mathsf{C})$. The point is that computations in the former algebra are far easier than in the later. This is because when A is not too large, it is always tractable to consider one by one its elements.

As an illustration, consider the problem of determining the probability $p(F)$ of a formula F from the probabilities of its variables. In $(A, \cap, \cup, \mathsf{C}_A)$, this problem is trivially of a polynomial worst case complexity, for it suffices to sum the probabilities of the minterms that satisfy F. In $(minterms(V), \cap, \cup, \mathsf{C})$, it is #P-hard [18], even if strong syntactic restrictions are set on the formula F under study [18], [19].

B. The Sub-Algebrae $\mathcal{A}_L^k$

We shall consider the sub-algebrae $\mathcal{A}_L^k$ induced by our formalisation of the notion of minimal cutsets. Let F and L be respectively a formula and a set of literals built over a set of variables V and let k be an integer such that $k \leq |V|$.

The set of minterms built over V that contain at most k critical literals is denoted by $minterms_L^k(V)$.

The algebra $(minterms_L^k(V), \cap, \cup, \mathsf{C}_{minterms_L^k(V)})$ is denoted by $\mathcal{A}_L^k$.

h_L^k denotes the following homomorphism from the regular propositional calculus to $\mathcal{A}_L^k$.

$$h_L^k(F) \overset{\text{def}}{=} F \cap minterms_L^k(V)$$

h_L^k is a *narrowing operator* for it removes minterms from the original formula.

Finally, $MC_L^k[F]$ denotes the set of minimal cutsets (of the formula F w.r.t. a set of literals L) that contain at most k critical literals.

It is clear that if L contains most of the variables of V and k is small, then $\mathcal{A}_L^k$ is a good candidate to be the framework of approximate computations: let n be the number of variables of V and let m be the number of critical literals, then the number of atoms of $\mathcal{A}_L^k$ is as follows.

$$\begin{aligned} |minterms_L^k(V)| &= 2^{n-m}.\sum_{i=0}^{k}\binom{m}{i} \\ &= \mathcal{O}(2^{n-m}.m^k) \end{aligned} \tag{4}$$

Therefore, if m is close to n and k is small, $\mathcal{A}_L^k$ is not too large.

Moreover, due to the probabilistic asymetry between critical literals and their opposite, minterms of $minterms_V^k(V)$ concentrate in general most of the probability of the formula under study. Minterms outside this set have a neglictible probability, even considered altogether. The table given figure 2 summarizes — for some values of n, some values of k and some values of p, the probability of all of the basic events — the maximum error E_L^k that may be done by considering only $minterms_V^k(V)$. E_L^k is bounded as follows.

$$E_L^k \leq \sum_{\pi \notin minterms_L^k(V)} p(\pi) \tag{5}$$

This illustrates that, even for large number of basic events and small values of k, the maximum error is not too large. In practice, the actual error is in general far less than the maximum error for not all the minterms not in $minterms_L^k(V)$ belong to the formula under study.

Note that it is possible to tune algorithms by considering different values of k.

Note also that upper approximations of the above bound can be used, that are possibly easier to compute than the exact bound.

Note finally that if no assumption is made on the probability of minterms, not only the exact value of $p(F)$ is hard to compute, but also approximations are hard to get, even again if strong syntactic restrictions are set on the formulae under study (see [20] for recent results on that topics).

C. BDD Implementation of Sub-Algebrae $\mathcal{A}_L^k$

In order to use BDDs to perform approximate computations, we have to consider the BDD implementation of the three operations $\cap$, $\cup$ and C in $\mathcal{A}_L^k$. A BDD encodes only minterms of $minterms_L^k(V)$ if and only if no path from its root node to a sink node contains more than k critical literals. The problem is thus to implement $\cap$, $\cup$ and C in such a way that this property is kept.

The negation over BDDs just inverts 0 and 1 leaves (or flips a flag). Therefore, the BDD that encodes $\bar{F}$ contains only minterms of $minterms_L^k(V)$ if and only if this property is true for the BDD that encodes F. Hence, the operation C in $\mathcal{A}_L^k$ can be implemented by means of the regular negation.

The implementation of $\cap$ and $\cup$ requires to revisit equation (2) to include the parameter k. Let $F = (v, F_1, F_0)$ and $G = (v, G_1, G_0)$ be the Shannon decompositions of two formulae in $\mathcal{A}_L^k$, and assume that v is critical. Then, the

	$k =$	1	2	3	4	5	6	7
$n = 100$	$p = 10^{-3}$	$9.43\ 10^{-2}$	$4.55\ 10^{-3}$	$1.46\ 10^{-4}$	$3.49\ 10^{-6}$	$6.61\ 10^{-8}$	$1.03\ 10^{-9}$	$1.37\ 10^{-11}$
	$p = 10^{-4}$	$9.85\ 10^{-3}$	$4.82\ 10^{-5}$	$1.56\ 10^{-7}$	$3.74\ 10^{-10}$	$7.09\ 10^{-13}$	$4.44\ 10^{-16}$	≈ 0
$n = 200$	$p = 10^{-3}$	$1.80\ 10^{-1}$	$1.72\ 10^{-2}$	$1.11\ 10^{-3}$	$5.43\ 10^{-5}$	$2.10\ 10^{-6}$	$6.78\ 10^{-8}$	$1.86\ 10^{-9}$
	$p = 10^{-4}$	$1.97\ 10^{-2}$	$1.94\ 10^{-4}$	$1.27\ 10^{-6}$	$6.24\ 10^{-9}$	$2.43\ 10^{-11}$	$7.79\ 10^{-14}$	≈ 0
$n = 500$	$p = 10^{-3}$	$3.93\ 10^{-1}$	$8.98\ 10^{-2}$	$1.42\ 10^{-2}$	$1.72\ 10^{-3}$	$1.68\ 10^{-4}$	$1.37\ 10^{-5}$	$9.54\ 10^{-7}$
	$p = 10^{-4}$	$4.87\ 10^{-2}$	$1.20\ 10^{-3}$	$1.98\ 10^{-5}$	$2.45\ 10^{-7}$	$2.42\ 10^{-9}$	$1.99\ 10^{-11}$	$1.39\ 10^{-13}$

Fig. 2. $\sum_{\pi \notin minterms_V^k(V)} p(\pi)$ for different values of $n = |V|$, k and p (p is the probability of all basic events).

following equality holds.

$$\begin{array}{rlcrl} & (v.F_1 + \bar{v}.F_0) & & & v.(F_1 \odot^{k-1} G_1) \\ \odot^k & (v.G_1 + \bar{v}.G_0) & = & + & \bar{v}.(F_0 \odot^k G_0) \end{array} \qquad (6)$$

where $\odot$ denotes either $\cap$ or $\cup$. Indeed, the above equation is completed by the usual simplification rules and the additional terminal case $F \odot^0 G = 0$.

The following theorem holds that asserts that the BDD implementation of the algebra $\mathcal{A}_L^k$ is efficient, i.e. of polynomial complexity if the number of non critical variables is not too large.

Theorem 4 (Approximate computation of BDDs) Let F be formula, let L be a set of literals built over $var(F)$, let $n = |var(F)|$ and m be the number of non critical literals of L, finally let k be a positive integer such that $k \ll n$. Then, the construction of the BDD that encodes $h_L^k(F)$ is in $\mathcal{O}(2^{(n-m).|F|}.n^{k.|F|})$,

This theorem generalizes our previous theorem [11].

D. Sub-Algebrae $\mathcal{A}_L^k$ and Minimal Cutsets

Now, let us consider the interest of the $\mathcal{A}_L^k$ sub-algebrae for the computation of minimal cutsets of small order. The question is whether it is possible to compute efficiently $MC_L^k[F]$, i.e. in polynomial time w.r.t. to the number n of variables (where $n = |var(F)|$).

First, it is easy to introduce the parameter k in the recursive equations given by the theorem 3. Let $F = v.F_1 + \bar{v}.F_0$ be the Shannon decomposition of a formula F, and assume that v is critical. Then, $MC_L^k[F]$ is the union of two sets:

$$MC_L^k[F] \quad = \quad v.MC_L^{k-1}[F_1 + F_0] \cup MC_L^k[F_0] \qquad (7)$$

This is basically what we proposed in [6] to compute short minimal p-cuts ($L = V$). However, this process requires to compute first the BDD that encodes F, which may lead to an exponential blow up. There exists actually monotone formulae that admit a polynomial number of short prime implicants and that do no admit polynomial polynomial size BDD whatever is the order over the variables [21]. Fortunately, there is a mean to avoid the full construction of the BDD.

Let F be a formula and π be a minterm of $minterms_L^k(V)$ such that $\pi \notin F$ and $\pi \in \omega_L(F)$. Then, there exists a minterm σ such that $\sigma \models F$ and $\sigma \sqsubseteq_L \pi$. By definition, σ belongs to $minterms_L^k(V)$. Therefore, the following theorem holds.

Theorem 5 (h_L^k and ω_L) Let F be any formula, let L be a set of literals built over $var(F)$, finally let k be a positive integer such that $k \ll n$. The following equality holds.

$$h_L^k(\omega_L(F)) \quad = \quad h_L^k(\omega_L(h_L^k(F))) \qquad (8)$$

As a consequence of the theorem 5, minimal cutsets with at most k critical literals come only from minterms with the same property. Therefore, in order to compute short minimal cutsets of a formula F, it suffices to compute the minimal cutsets of the narrowing of F, as asserted by the following corollary.

Corollary 6 (Computation of small minimal cutsets) Let F be any formula, let L be a set of literals built over $var(F)$, finally let k be a positive integer such that $k \ll n$. Then, the following equality holds.

$$MC_L^k[F] \quad = \quad MC_L[h_L^k(F)] \qquad (9)$$

E. Roadmap for the computation of $MC_L^k[F]$

The roadmap for the computation of the ZBDD that encodes $MC_L^k[F]$ is pictured figure 3.

The algorithms $TrBDD_L^k$ and $TrZBDD_L^k$ just consist in traversing the BDD and the ZBDD and in keeping only the products that contain less than k critical literals (see [11] for a very similar algorithm). The other algorithms are fully described by equations given throughout this article.

The figure 3 shows the three main ways to compute $MC_L^k[F]$. The first one, $build/\omega_L/PI/TrZBDD_L^k$ recalls the definition of minimal cutsets. The second one, $build/MC_L^k$ follows from the Morreale's theorem and the definition of minimal cutsets. Both compute the BDD encoding F and are therefore of exponential worst case complexities. The last one, $build - h_L^k/MC_L$ illustrates the interest of the mathematical foundations we gave for minimal cutsets. As shown by theorem 4, $build - h_L^k$ is of polynomial worst case complexity. It is easy to verify that, if a BDD encodes only minterms of $minterms_L^k(V)$, then MC_L is also of polynomial worst case complexity, at least if it is implemented by means of the operator $\div$ [6]. Therefore, the following theorem holds.

Theorem 7 (Complexity of MC_L^k) Let F be formula, let L be a set of literals built over $var(F)$, finally let k be a

$$\begin{tikzcd}
 & & \boxed{\text{BDD}} \arrow[dr, "PI"] & \\
\boxed{\text{Formula}} \arrow[r, "build"] \arrow[dr, "build - h_L^k"'] & \boxed{\text{BDD}} \arrow[ur, "\omega_L"] \arrow[rr, "MC_L"'] \arrow[d, "TrBDD_L^k"] \arrow[drr, "MC_L^k"] & & \boxed{\text{ZBDD}} \arrow[d, "TrZBDD_L^k"] \\
 & \boxed{\text{BDD}} \arrow[rr, "MC_L"'] & & \boxed{\text{ZBDD}}
\end{tikzcd}$$

Fig. 3. The roadmap for the computation of the ZBDD that encodes $MC_L^k[F]$.

positive integer such that $k \ll n$. Assume that the number of non-critical literals is small w.r.t. $|var(F)|$, i.e. in $\mathcal{O}(\log(|var(F)|))$. Then, the construction of the ZBDD that encodes $MC_L^k[F]$ is is of polynomial worst case complexity.

This is major advantage of minimal cutsets over prime implicants: the existence of a prime implicant of length k or less is a hard problem. Namely, it is ΔP-complete, where ΔP is the class of problems or languages that can be defined as the intersection of a language in NP and a language in $coNP$ [22].

VII. Conclusion

In this article, we clarified the mathematical status of the notion of minimal cutsets, through the introduction the notion of significant and critical literal, the induced order relation over minterms $\sqsubseteq_L$ and widening operator ω_L. We show that minimal cutsets are distinct from prime implicants and that they have a great interest from both a computational complexity and practical point of views. We introduced the families of narrowing operators h_L^k and sub-algebrae $\mathcal{A}_L^k$ that provide a sound mathematical framework for approximate computations of both the top event probabilities of fault trees and of their minimal cutsets.

We discussed also the implementation of BDD algorithms. All of the algorithms evocated here are actually implemented in the Aralia software, which is now widely used by most of the french compagnies that are involved in risk assessment studies. These algorithms and their mathematical foundations were designed to enabled us to assess efficiently a very large non-coherent fault tree that models the emergency shutdown system of a nuclear reactor [23]. This gives evidences of the practical interest of the mathematical framework we introduced here.

References

[1] R. Bryant, "Graph Based Algorithms for Boolean Fonction Manipulation", *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, August 1986.

[2] K. Brace, R. Rudell, and R. Bryant, "Efficient Implementation of a BDD Package", in *Proceedings of the 27th ACM/IEEE Design Automation Conference*. 1990, pp. 40–45, IEEE 0738.

[3] O. Coudert and J.-C. Madre, "A New Method to Compute Prime and Essential Prime Implicants of Boolean Functions", in *Advanced Research in VLSI and Parallel Systems*, T. Knight and J. Savage, Eds., March 1992, pp. 113–128.

[4] A. Rauzy, "New Algorithms for Fault Trees Analysis", *Reliability Engineering & System Safety*, vol. 05, no. 59, pp. 203–211, 1993.

[5] O. Coudert and J.-C. Madre, "Fault Tree Analysis: 10^{20} Prime Implicants and Beyond", in *Proceedings of the Annual Reliability and Maintainability Symposium, ARMS'93*, January 1993, Atlanta NC, USA.

[6] Y. Dutuit and A. Rauzy, "Exact and Truncated Computations of Prime Implicants of Coherent and non-Coherent Fault Trees within Aralia", *Reliability Engineering and System Safety*, vol. 58, pp. 127–144, 1997.

[7] R.M. Sinnamon and J.D. Andrews, "Improved Accuracy in Qualitative Fault Tree Analysis", *Quality and Reliability Engineering International*, vol. 13, pp. 285–292, 1997.

[8] J.B. Fussel, "How to hand-calculate system reliability characteristics", *IEEE Transactions on Reliability*, vol. R-24, no. 3, 1975.

[9] A. Laviron, A. Carnino, and J.-C. Manarache, "Escaf, a new a cheap system for complex reliability analysis and computation", *IEEE Transactions on Reliability*, vol. R-31, pp. 339–348, 1982.

[10] U. Berg, *RISK SPECTRUM, Theory Manual*, RELCON Teknik AB, April 1994.

[11] Y. Dutuit and A. Rauzy, "Polynomial approximations of boolean functions by means of positive binary decision diagrams", in *Proceedings of European Safety and Reliability Association Conference, ESREL'98*, Lydersen, Hansen, and Sandtorv, Eds. 1998, pp. 1467–1472, Balkerna, Rotterdam, ISBN 90 54 10 966 1.

[12] R. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams", *ACM Computing Surveys*, vol. 24, pp. 293–318, September 1992.

[13] S.-I. Minato, *Binary Decision Diagrams and Applications to VLSI CAD*, Kluwer Academics Publishers, 1996, ISBN 0-7923-9652-9.

[14] C. Meinel and T. Theobald, *Algorithm and Data Structures in VLSI Design*, Springer Verlag, 1998, ISBN 3-540-64486-5.

[15] T. Sasao, "Ternary Decision Diagrams and their Applications", in *Representations of Discrete Functions*, chapter 12, pp. 269–292. Kluwer Academic Publisher, 1996, ISBN 0-7923-9720-7.

[16] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems", in *Proceedings of the 30th ACM/IEEE Design Automation Conference, DAC'93*, 1993, pp. 272–277.

[17] E. Morreale, "Recursive Operators for Prime Implicant and Irredundant Normal Form Determination", *IEEE Trans. Computers*, vol. C-19, no. 6, pp. 504–509, 1970.

[18] L.G. Valiant, "The complexity of enumeration and reliability problems", *SIAM Journal of Computing*, vol. 8, pp. 410–421, 1979.

[19] N. Linial, "Hard enumeration problems in geometry and combinatorics", *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, no. 2, pp. 331–335, April 1986.

[20] D. Roth, "On the hardness of approximate reasoning", *Artificial Intelligence*, vol. 82, pp. 273–302, 1996.

[21] K. Hayase and H. Imai, "OBDDs of a Monotone Function and Its Prime Implicants", *Theory of Computing Systems*, vol. 41, pp. 579–591, 1998.

[22] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994, ISBN 0-201-53082-1.

[23] S. Combacon, Y. Dutuit, A. Laviron, and A. Rauzy, "Comparison between two tools (Aralia and ESCAF) applied to the Study

of the Emergency Shutdown System of a Nuclear Reactor", in *Proceedings of the International Conference of Probabilistic Safety Assessment and Management, PSAM'4*, A. Mosleh and R.A. Bari, Eds., New-York, 1998, vol. 2, pp. 1019–1024, Springer Verlag.