Special Issue Article

Institution of MECHANICAL ENGINEERS

Journal of RISK AND RELIABILITY

Proc IMechE Part O:
J Risk and Reliability
227(3) 315–326

DOI: 10.1177/1748006X13490260
pio.sagepub.com
SAGE

Fault tree linking versus event tree linking approaches: a reasoned comparison

Olivier Nusbaumer[1] and Antoine Rauzy[2]

Abstract

Two well-known modelling approaches are in use in probabilistic risk assessment: fault tree linking and event tree linking. The question of which modelling approach is most appropriate for specific applications has been extensively, if not emotionally, debated among experts in the past two decades, addressing both modelling and quantification issues. In this article, we determine their degree of equivalence and build 'methodological bridges' between the two approaches from a mathematical and algorithmic perspective. We show that, under certain conditions, both modelling approaches are equivalent. Since both fault tree linking and event tree linking approaches are subject to limitations and approximations, established bridges make it possible to formulate important recommendations for probabilistic risk assessment practitioners and quantification engine developers.

Keywords

Probabilistic risk assessment, methodology, modelling, quantification engines

Date received: 20 September 2012; accepted: 26 April 2013

Introduction

Two well-known modelling methodologies are in use in probabilistic safety assessment (PSA) and probabilistic risk assessment (PRA), see for example, Kumamoto and Henley[1] for an introduction: fault tree linking (FTL) and event tree linking (ETL). These two approaches are described and compared almost from the beginning of Nuclear PSA (see e.g. NUREG/CR-2300[2]). The question of which modelling methodology is most appropriate for specific applications has been extensively, if not emotionally, debated among experts in the past three decades, addressing both modelling and quantification issues. The already cited NUREG/CR-2300 (section 6.3) remarked that '*Overall, the basic conceptual difference between the methods is where the process quantification (conversion to from symbolic representation to numerical results takes places)*'.

This article aims to give a solid ground and to complement this assertion by comparing the two approaches from a mathematical and algorithmic perspective. As a starting point, we assume that both approaches start with the same virtual model. The FTL approach quantification algorithms work on that virtual model, or on a model that is close to that virtual model. On the contrary, the ETL approach first develops the virtual model so to make event tree sequences pair wisely disjoint. This development is supposed to preserve logical equivalence. In practice however, the two approaches give different results, because different modelling and algorithmic strategies are applied.

This situation is however not as bad as it may seem at first glance: it is actually possible to build 'methodological bridges' between the two approaches and to establish formally their degree of equivalence. A general assumption in sciences is that two objects under study can be considered as equivalent if they cannot be distinguished with the observation means at hand. In our case, objects are models and observation means are quantification algorithms.

Quantification algorithms can be classified into two categories: those computing a sum of disjoint products and those computing a set of minimal cutsets. The formers evaluate states, i.e. valuations of basic events to

[1]Leibstadt Nuclear Power Plant, Leibstadt, Switzerland
[2]Ecole Polytechnique, Palaiseau, France

Corresponding author:
Antoine Rauzy, Ecole Polytechnique, Palaiseau 91128, France.
Email: Antoine.Rauzy@lix.polytechnique.fr

either true or false, while the latter perform a coherent/monotone approximation of the models. In both cases, cutoffs can be applied. Quantification algorithms of RiskSprectrum[3,4] and CAFTA[5,6] are based on minimal cutsets calculation. The ETL approach can be seen as the partial application of a sum of disjoint products algorithm. Bryant's binary decision diagrams[7] adapted for PSA/PRA quantification (see e.g. Rauzy[8] for a survey) also belong to this category. With respect to this classification, we can define two natural equivalence relations on PSA/PRA models.

- Strong equivalence: two models are strongly equivalent if they agree on global states (minterms) whose probability (evaluated in a certain way) is bigger than the given cutoff, i.e. if they cannot be distinguished by means of a sum of disjoint products algorithm.
- Weak equivalence: two models are weakly equivalent if they agree on minimal cutsets whose probability is bigger than the given cutoff, i.e. if they cannot be distinguished by means of a minimal cutsets algorithm.

Weak equivalence is indeed coarser than strong equivalence or, to put it into another way, sum of disjoint products algorithms give finer results than minimal cutsets algorithms. This precision comes with a severe increase of necessary computing resources.

Strong and weak equivalence give a firm mathematical ground to compare FTL and ETL approaches. Basically, we show that the two approaches are weakly equivalent and that the approximations are in general accurate on coherent models. These results clarify Epstein and Rauzy's statements about PSA/PRA quantification.[9]

Since problems in quantification can arise mainly on non-coherent models, it is of interest to study carefully where non-coherence comes from. We discuss here this issue based on responses to a questionnaire provided by a large international panel of PSA/PRA analysts.

The remainder of the article is organized as follows. First we introduce the mathematical and algorithmic framework of PSA/PRA. Then, we present both FTL and ETL approaches by means of a small example. We establish formally their degree of equivalence, first for coherent models, then for non-coherent ones. Finally, we discuss where non-coherence comes from.

Mathematical preliminaries

In this section, we review the basics about Boolean logic involved in fault tree and event tree analysis. For a more thorough treatment of the subject, the reader is referred to Rauzy.[10]

Throughout this article we consider fault tree and event tree models. Fault trees and event trees are interpreted as Boolean formulas built over a finite set of variables, so-called basic events, and logical connectives '+' (or), '·' (and) and '−' (not). Other connectives, such as k-out-of-n and others can be easily derived from those.

A 'literal' is either a basic event or its negation. A 'product' is a conjunction of literals that does not contain both a basic event and its negation. A product is positive if it contains no negated basic event. A 'sum of products' is a set of products interpreted as their disjunction. Two products are disjoint if there is at least one basic event occurring positively in one of them and negatively in the other. A 'sum of disjoint products' (SDP) is a sum of products whose products are pair wisely disjoint. A 'minterm', relative to a set of basic events, is a product that contains a literal for each basic event in the set. By construction, two different minterms are disjoint. Minterms one-to-one correspond with truth assignments of basic events. For that reason, the following property holds.

Property 1

Any Boolean formula is equivalent to a unique sum of minterms.

Consider for instance the Boolean formula $F = A{\cdot}B + -A{\cdot}C$. The formula F can be expanded into a sum of minterms, relatively to the set of basic events $\{A, B, C\}$, as follows.

$$Minterms(F) = A \cdot B \cdot C + A \cdot B - C + - A \cdot B \cdot C + - A - B \cdot C$$

We shall say that a minterm π belongs to a formula F, and shall denote $\pi \in$ F, if π occurs in $Minterms(F)$. Similarly, we shall write $F \subseteq G$, when $Minterms(F) \subseteq Minterms(G)$ and $F \equiv G$ when $Minterms(F) = Minterms(G)$, i.e. when F and G are logically equivalent. Note that logical equivalence is the strongest equivalence relation over models. Two logically equivalent models are indistinguishable by any correct quantification algorithm.

Let π and ρ be two minterms. We say that $\pi \leqslant \rho$, if any basic event that occurs positively in π occurs positively in ρ as well. For instance, we have $A{\cdot}B{\cdot}-C \leqslant A{\cdot}B{\cdot}C$.

A Boolean formula F is coherent if for any two minterms π and ρ such that $\pi \in F$ and $\pi \leqslant \rho$, then $\rho \in F$. It is easy to verify that any formula built only over basic events and connectives '+' and '.' is coherent.

Let π be a positive product. We denote by π' the minterm built by completing π with the negative literals built over basic events that do not show up in π. For instance, in the above example, we have

$$(A \cdot B)' = A \cdot B \cdot -C \text{ and } C' = - A \cdot -B \cdot C$$

A cutset of a Boolean formula F is defined as a positive product π, such that $\pi' \in F$. A cutset π is minimal if no subproduct of π is a cutset of F. Consider again the above example, the formula F admits the following cutset and minimal cutsets (MCS)

$$Cutsets(F) = \{ A \cdot B, C, A \cdot B \cdot C\}, MCS(F) = \{A \cdot B, C\}$$

$MCS(F)$ can be interpreted as a sum of products, i.e. a Boolean formula. As illustrated by the above example, F and $MCS(F)$ may differ.

Model quantification

In PRA models, because of basic events that are repeated here and there, there is no efficient way to quantify the reliability indicators of interest, such as the top-event probability and importance factors. From a theoretical viewpoint, this impossibility has been formally established by Valiant.[11] Therefore, all of the algorithms known so far have two important characteristics.

1. Either they compute an approximation of the indicator of interest or they are subject to the exponential blow-up of computation resources, and most of the time they suffer from both drawbacks.
2. They compute an intermediate form of the model from which the reliability indicators can be calculated.

From a practical viewpoint, there are so far only two categories of algorithms.

1. Algorithms that compute minimal cutsets of the model and perform quantifications from these minimal cutsets. These algorithms are used mostly by the FTL approach.
2. Algorithms that compute a SDP and perform quantifications of this SDP. These algorithms are used mostly by the ETL approach.

Minimal cutsets algorithms

The first category is historically the first to have been developed and the most widely used. The algorithm implemented in RiskSprectrum,[4] which is derived from MOCUS[3] and the one implemented in CAFTA computation engine[5,6] belong to this category. The fact that CAFTA computation engine uses Minato's ZBDD[12] to encode MCS changes nothing to the matter: all quantifications are performed through (an encoding of) MCS. MOCUS, like algorithms, compute MCS top-down. ZBDD-based algorithms compute MCS bottom-up. Note that an important step forward has been made recently by Rauzy who proposed a branch and deduce algorithmic scheme to compute MCS.[13] This work makes it possible to implement very efficient MCS calculation engines and to better understand the algorithmic issues at stake.

The following property holds that characterizes the relationship between a formula and its minimal cutsets.

Property 2

Let F be a Boolean Formula. Then, $F \subseteq MCS(F)$. Moreover, $MCS(F)$ is the smallest monotone formula containing F.

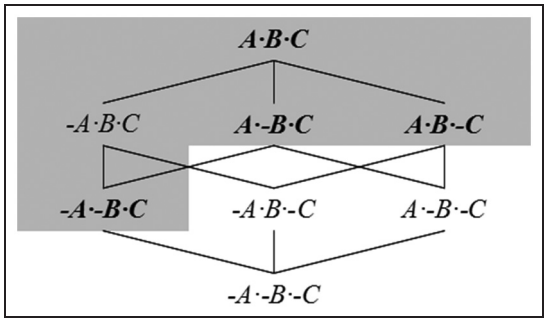

Figure 1. Minterm lattice over basic events A, B and C and formulas $F=A\cdot B + -A\cdot C$ and $MCS(F)=A\cdot B + C$.

As a consequence, $F = MCS(F)$ if, and only if, F is coherent.

This property has been stated and proved in Rauzy.[10] Property 2 ensures that by computing minimal cutsets, we make no logical approximation if the model under study is coherent and a conservative approximation if it is non-coherent. By conservative we mean here that the sum of minimal cutsets encode all failure states (minterms) of the original model, plus possibly some that are not in the original model.

As an illustration, consider again the formula $F = A\cdot B + -A\cdot C$. Recall that $MCS(F) = A\cdot B + C$. Both formulas are pictured Figure 1. In this figure, minterms are organized into a lattice according to the relation $\leqslant$ defined in the previous section. Minterms of F are drawn in bold font. Minterms of $MCS(F)$ are in the grey zone. $MCS(F)$ is obtained from F by adding to F all minterms that are bigger than a minterm in F. In our case, there is only one such a minterm: $-A\cdot B\cdot C$.

SDP algorithms

We shall see in the next section that the ETL approach can be seen as an incomplete SDP algorithm. Bryant's binary decision diagrams (BDD)[7] are the most prominent representative of this second category. Rauzy introduced them in the reliability engineering field in his seminal article[14] and many subsequent articles (see Rauzy[8] for a survey). All SDP algorithms make an extensive use of the so-called Shannon decomposition.

Let F be a Boolean formula, let G be a sub-formula of F and finally let v be a Boolean value (0 or 1). We denote by $F_{G=v}$ the Boolean formula obtained from F by substituting the constant v for the formula G. The following property, so-called Shannon decomposition, holds.

Property 3 (Shannon decomposition)

Let F be a Boolean formula, let G be a sub-formula of F and finally let v be a Boolean value (0 or 1). Then

$$F \equiv G \cdot F_{G=1} + - G \cdot F_{G=0}$$

G is called the pivot of the decomposition. In the case of BDD, pivots are always basic events. In the case of ETL development, it can be an intermediate event as well.

Consider for instance the formula $F = A{\cdot}B + A{\cdot}C + B{\cdot}C$, i.e. a 2-out-of-3 over basic events A, B and C. This sum of products can be rewritten into a logically equivalent SDP by successive applications of the Shannon decomposition

$$\begin{aligned} F &= A\cdot B + A\cdot C + B\cdot C \\ &\equiv A\cdot F_{A=1} + -A\cdot F_{A=0} = A\cdot(1\cdot B + 1\cdot C + B\cdot C) \\ &\quad + -A\cdot(0\cdot B + 0\cdot C + B\cdot C) = A(B + C + B\cdot C) \\ &\quad + -A\cdot B\cdot C \end{aligned}$$

$$\begin{aligned} &A\cdot(B(B + C + BC)_{B=1} + -B\cdot(B + C + B\cdot C)_{B=0} \\ &+ -A\cdot B\cdot C = A\cdot(B + -B\cdot C) + -A\cdot B\cdot C \\ &= A\cdot B + A - B\cdot C + -A\cdot B\cdot C \end{aligned}$$

In the sequel we shall denote by $SDP(F)$ the SDP obtained from F by some SDP algorithm. Indeed, $SDP(F)$ is by no means unique. But since the Shannon decomposition preserves the logical equivalence, $SDP(F) \equiv F$ for any correct algorithm.

Probability calculation

Let F be a Boolean formula. We say that F has a probability structure if it comes with a function p(.) that associates a probability with each basic event of F. By extension, we denote by $p(F)$ the probability of the formula F. The well-known Sylvester-Poincaré development, also called inclusion-exclusion principle, makes it possible to compute the probability of a sum of products from the probabilities of basic events.

Property 4 (Sylvester-Poincaré development)

Let $F = \sum_{i=1,\ldots,n}\pi_i$ be a sum of products. Then, the following equality holds

$$\begin{aligned} p(F) = {} & \Sigma_{1\leqslant i\leqslant n}p(\pi_i) - \Sigma_{1\leqslant i_1\leqslant i_2\leqslant n}p(\pi_{i_1}\pi_{i_2}) \\ & + \ldots + -1^p\Sigma_{1\leqslant i_1\leqslant \ldots\leqslant i_p\leqslant n}p(\pi_{i_1}\ldots\pi_{i_p}) \\ & + \ldots + -1^n p(\pi_1\ldots\pi_n) \end{aligned}$$

We say that a model is solved exactly (but not obligatorily completely) when the full Sylvester-Poincaré development is applied. Unfortunately, applying this development fully is exponential in the number of products. For this reason, most (FTL) quantification engines approximate it by computing only the first term of this development (rare event approximation). This is a problem if individual probabilities are large or if the model is not coherent. Otherwise, the rare event approximation is, in general, quite accurate.

Another approximation, the so-called 'min cut upper bound' (MCUB), which is also first order and is often used in practice, gives better approximations than the rare event approximation. The MCUB formula is as follows

$$p(F) \cong 1 - \prod_i (1 - p(\pi_i))$$

As compared with the rare event approximation, the MCUB prevents the top event probability exceed 1. On the other hand, as a first-order technique, it cannot give exact results in most cases (if the formula is coherent, it always gives conservative results though).

In the case where the formula is a SDP, the rare event approximation gives even the exact probability, for all subsequent terms are cancelled (because they involved at least two products and products are pair wisely disjoint). The following property holds.

Property 5

Let $F = \sum_{i=1,\ldots,n}\pi_i$ be a sum of products. The following inequality holds

$$p(F) \leqslant \Sigma_{i=1,\ldots,n}p(\pi_i)$$

Moreover, if for each basic event E, $p(E) \ll 1$, then the approximation is accurate

$$p(F) \approx \Sigma_{i=1,\ldots,n}p(\pi_i)$$

If F is a SDP, then the inequality becomes equality

$$p(F) = \Sigma_{i=1,\ldots,n}p(\pi_i)$$

Let us summarize what we have obtained so far. Let F be a coherent Boolean formula. Then, to get the exact probability of F we can do any of the following operations.

- Build $SDP(F)$, then calculate the first term of the Sylvester-Poincaré development on $SDP(F)$.
- Build $MCS(F)$, then calculate the full Sylvester-Poincaré development on $MCS(F)$.
- Build $MCS(F)$, then $SDP(MCS(F))$, then calculate the first term of the Sylvester-Poincaré development on $SDP(MCS(F))$.

To get a conservative, and in many practical cases accurate, approximation of the probability of F, we can do the following operations.

- Build $MCS(F)$ then calculate the first term of the Sylvester-Poincaré development on $MCS(F)$.

As already said, calculating the full Sylvester-Poincaré development (to get an exact probability) is of exponential complexity in the number of minimal cutsets. So, calculating a SDP should (and actually is) costly as well. To illustrate this phenomenon, consider the following parametric formula (sum of minimal cutsets) and its transformation into a SDP

$$F_n = A_1\cdot B_1 + \ldots + A_n\cdot B_n$$

$$\equiv (A_1 \cdot B_1 + \ldots + A_{n-1} \cdot B_{n-1}) - A_n \cdot B_n$$
$$+ (A_1 \cdot B_{1s} + \ldots + A_{n-1} \cdot B_{n-1}) \cdot -B_n + A_n \cdot B_n$$
$$\equiv F_{n-1} \cdot -A_n \cdot B_n + F_{n-1} \cdot -B_n + A_n \cdot B_n$$

It follows that $SDP(F_n)$ is more than twice as big as $SDP(F_{n-1})$ so that the size of $SDP(F_n)$ is exponential in the number of MCS. Of course, $SDP(F_n)$ can be encoded with a linear size BDD, but there are simple formulas with a polynomial number of MCS, which admit no polynomial size BDD representation (see e.g. Nikolskaia and Nikolskaia[15]).

If F is non-coherent, we can get a conservative approximation of the probability of F by means of any of the above sequences of operations. Whether this approximation is accurate depends on F. For some F, it is quite good, for some others it is awfully bad. Consider for instance, the formula $F = -E$, where E is a basic event such that $p(E) \approx 1$. In that case, $p(F) = 1 - p(E) \approx 0$, while $p(MCS(F)) = p(1) = 1$.

Cutoffs

Most of the real-life models have a huge number of minimal cutsets (and even more disjoint products). So many that it is definitely impossible to encode all of them, even using symbolic techniques such as ZBDD. Therefore, cutoffs have to be applied to keep only the most probable minimal cutsets: all MCS whose probability is below a given threshold are discarded. This approximation is indeed not conservative, for it 'ignores' some of the failure situations.

Let F be a Boolean formula with a probability structure. We denote by $MCS_{\geqslant \chi}(F)$ denote the sum of minimal cutsets of F whose probability is bigger than the cutoff value χ. Formally

$$MCS_{\geqslant \chi}(F) = \{\pi \text{ in } MCS(F);\ p(\pi) \geqslant \chi\}$$

The question whether $p(MCS_{\geqslant \chi}(F))$ is an accurate approximation is of $p(MCS(F))$, or in other words whether this approximation can be controlled, is sometimes not very well understood. The following facts may clarify the situation.

- It is, in general, possible to tune MCS algorithms so that they give, while computing $MCS_{\geqslant \chi}(F)$, an upper bound of their error, i.e. of the quantity $p(MCS(F)) - p(MCS_{\geqslant \chi}(F))$, but…
- It is not possible, in general, to estimate this error without running the algorithm. Therefore, the calculation of the upper bound on the error is an a posteriori verification.

To understand where the problem comes from, it suffices to consider a formula F with say a million of minimal cutsets whose probability is just below the cutoff value. In this case, $p(MCS_{\geqslant \chi}(F)) = 0$ while $p(MCS(F)) \approx \chi \times 10^6$. One may argue that such a case has little chance to occur in practice. This is partly true, at least for what concerns the evaluation of the probability of the formula. Based on experiences they made on a Japanese PSA, Epstein and Rauzy[9] showed that calculations of so-called importance factors (indicators that aim to rank basic events according to their contribution of the overall risk) can be strongly impacted by this phenomenon. Their results have been confirmed by Duflot et al.[16] on the reference French PSA.

Cutoffs can be also applied by SDP algorithms. This application is however trickier than for the minimal cutsets case. A first idea would be, given a SDP F and a cutoff value χ to discard all of the products of F whose probability is lower than χ. The problem with this approach is that it is sensitive to the way the formula is written. As an illustration, consider the formula $F = E_1 + E_2 + \ldots + E_{1000}$, where the E_i's are basic events. There are different ways to rewrite F as SDP. For instance

$$F \equiv G = E_1 + -E_1 \cdot E_2 + -E_1 \cdot -E_2 \cdot E_3$$
$$+ \ldots + -E_1 \cdot -E_2 \cdot \ldots - E_{999}\, E_{1000}$$
$$F \equiv H = E_1 \cdot -E_2 \cdot \ldots - E_{999} \cdot -E_{1000} + E_2 \cdot -E_3 \cdot \ldots$$
$$- E_{999} \cdot -E_{1000} + \ldots + E_{1000}$$

Assume that the probabilities of the E_i's are close to 10^{-4}. Then the probability of the terms $-E_1 \cdot -E_2 \cdot \ldots - E_{999}$ and $-E_2 \cdot \ldots - E_{999} \cdot -E_{1000}$ are close to $(1-10^{-4})^{1000} \approx 1\text{-}1000 \times 10^{-4} = 0.9$. Now, if we set up the χ a bit below 10^{-4}, which is reasonable to keep all of the MCS of F, we will discard the last terms of G and the first terms of H, resulting into two different approximated models and two different approximated set of MCS. This phenomenon could be a problem in the ETL approach if one would develop the model to such a level of detail that too many cutsets would fall under the cutoff level. Our experience is that it is definitely a problem on a large PSA model with the BDD technology. As a consequence, and it is all what Rauzy's seminal article[10] is about, the truncation should concern only the positive part of disjoint products (this is what typical MCS engines do).

Let F be a Boolean formula with a probability structure and χ be a cutoff value. We denote by $SDP_{\geqslant \chi}(F)$ the SDP obtained from $SDP(F)$ by discarding products whose probability of the positive part is smaller than χ. More formally, let π be a product. We denote by $|\pi|$ the sub-product of made of positive literals of π. Then we define $SDP_{\geqslant \chi}(F)$ as follows

$$SDP_{\geqslant \chi}(F) = \{\pi \in SDP(F);\ p(|\pi|) \geqslant \chi\}$$

Again, $SDP_{\geqslant \chi}(F)$ is not unique because $SDP(F)$ is not.

It is worth noting that $SDP_{\geqslant \chi}(F)$ has little chance to be coherent, even if F and therefore $SDP(F)$ are. As an illustration, consider the formula $F = A \cdot B + C$. Let $p(A) = p(B) = p(C) = 10^{-3}$. Assume that $SDP(F)$ is as follows
$SDP(F) = A \cdot B \cdot C + A \cdot B \cdot -C + -A \cdot C$
Finally, let $\chi = 10^{-7}$. We have

$$SDP_{\geqslant \chi}(F) = A \cdot B - C + -A \cdot C$$

Clearly, $SDP_{\geqslant\chi}(F)$ is not coherent for the biggest minterm $A{\cdot}B{\cdot}C$ has been discarded and there is no way to reintroduce it from the remaining products.

This makes a big difference between $MCS_{\geqslant\chi}(F)$ and $SDP_{\geqslant\chi}(F)$. $MCS_{\geqslant\chi}(F)$ is always coherent. The application of a cutoff just discards some of the minimal cutsets. The application of cutoffs to SDP gives much less intuitive results. Conversely to the former, it is a calculation artifact. Nevertheless, the following property holds.

Property 6

Let F be a Boolean formula with a probability structure and let χ be a cutoff value. Then the following logical inequalities hold.

$$SDP_{\square\square}(F) \subseteq MCS_{\square\square}(F)\,\square MCS(SDP_{\square\square}(F))$$

This property is again a consequence of results presented in Rauzy.[10]

Wrap-up

Let us summarize the practical consequences of Property 1 to Property 6. Let F be a Boolean formula with a probability structure and let χ be a cutoff value.

If F is coherent, then:

- The calculation of $MCS(F)$ preserves the logical equivalence with F.
- The calculation of $MCS_{\geqslant\chi}(F)$ underestimates F (for it discards failure scenarios): $p(MCS_{\geqslant\chi}(F)) \leqslant p(F)$. The lower the threshold, the better the approximation.
- The application of the rare-event approximation overestimates $p(MCS_{\chi}(F))$: $\textit{rare-events}(MCS_{\chi}(F)) \geqslant p(MCS_{\geqslant\chi}(F))$. The lower the probabilities of basic events, the better the approximation.
- The calculation of $SDP(F)$ preserves the logical equivalence with F.
- The calculation of $SDP_{\geqslant\chi}(F)$ underestimates F: $p(SDP_{\geqslant\chi}(F)) \leqslant p(F)$. The lower the threshold, the better the approximation.
- The calculation of $SDP_{\geqslant\chi}(F)$ is more optimistic than the calculation of $MCS_{\geqslant\chi}(F)$

$$p(SDP_{\geqslant\chi}(F)) \leqslant p(MCS_{\geqslant\chi}(F)) \leqslant \textit{rare-events}(MCS_{\chi}(F)).$$

- $MCS_{\geqslant\chi}(F)$ and $SDP_{\geqslant\chi}(F)$ agree on minimal failure scenarios: $MCS_{\geqslant\chi}(F) \equiv MCS(SDP_{\geqslant\chi}(F))$.

In some sense, the rare events approximation 'compensates' the application of a cutoff on minimal cutsets, but indeed one cannot rely on this fact, especially when probabilities of basic events are high.

If F is non-coherent, then:

- The calculation of $MCS(F)$ overestimates F: $p(MCS(F)) \geqslant p(F)$. It may be the case that this approximation gives meaningless results.
- The calculation of $MCS_{\geqslant\chi}(F)$ underestimates $MCS(F)$, but keeps the potential to overestimate dramatically $p(F)$.
- The application of the rare-event approximation overestimates $p(MCS_{\chi}(F))$.
- The calculation of $SDP(F)$ preserves the logical equivalence with F.
- The calculation of $SDP_{\geqslant\chi}(F)$ underestimates F: $p(SDP_{\geqslant\chi}(F)) \leqslant p(F)$. The lower the threshold, the better the approximation.
- $MCS_{\geqslant\chi}(F)$ and $SDP_{\geqslant\chi}(F)$ agree on minimal failure scenarios: $MCS_{\geqslant\chi}(F) \equiv MCS(SDP_{\geqslant\chi}(F))$.

The accuracy of SDP algorithms comes with a price in terms of calculation resources. There is no free lunch. In his PhD study, Nusbaumer,[17] using dedicated heuristics, solved a complete accident sequence of a Swiss PSA (which was a very big coherent model) using BDD. Despite of this success, large PSA models seem still out of the reach of the pure BDD technology and therefore probably of any SDP algorithm. When cutoffs have to be applied, which is always the case so far for big PRA models, results given by MCS algorithms and SDP algorithms may be quite different. They agree, however, on one very important point: they produce the same failure scenarios.

FTL versus ETL approaches

After these mathematical and algorithmic preliminaries, we go back to the heart of our matter by presenting both FTL and ETL approaches and establish their degree of equivalence.

The FTL approach

The FTL approach uses relatively small event trees to represent the combinations of functional events following an initiating event. All such event combinations together make up a set of accident sequences.

System fault tree models are developed to represent the combination of basic event failures that would result in the failure of each function event in the event trees. System fault trees usually require some adaptation for applicability at different event tree branches, representing the direct effects of different initiating events, success criteria, or other conditions imposed by the status of earlier events in the event tree (e.g. the failure to perform an operator action).

Quantification tools typically assess FTL models in three steps: first, the event tree sequence (or set of sequences) is translated into a Boolean formula, so-called the master fault tree; second, MCS of the master fault tree are computed; third, MCS are quantified to get the reliability indicator of interest.

As seen in the previous section, most FTL quantification engines approximate the exact result by computing only the first term of the Sylvester-Poincaré

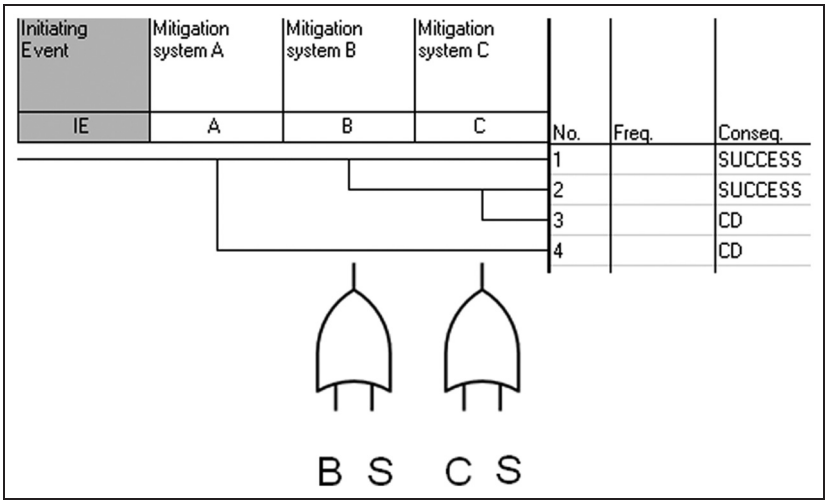

Figure 2. FTL model for a simplified nuclear power plant.

development. This is often a problem when probabilities are large.

The master fault tree is obtained in two steps. First, a formula is associated with each sequence of the event tree. This formula is the product of the functional events labelling the failure branches of the sequence and the negation of the functional events labelling the success branches of the sequence. Second, a formula is associated with each consequence, i.e. group of sequences. This formula is the sum of the formulas associated with the sequences of the group. Let M be an event tree model and C be a consequence. We denote by $FTL(M,C)$ the master fault tree associated with the consequence C in the model M.

Consider a simplified nuclear power plant model M consisting of three safety systems A, B and C. If A fails, then it goes straight to a core damage state (CD). Any of the safety systems B and C may prevent CD, but both share a common support system S. The FTL model for that power plant is pictured Figure 2.

The master fault tree for the core damage is

$$FTL(M,C) = -A(B+S)\cdot(C+S) + A$$

Despite appearances, $FTL(M,C)$ is coherent (it is easy to see that it is equivalent to the formula $(B+S)\cdot(C+S) + A$). In that case, the minimal cut-sets are easily determined

$$MSC(FTL(M,C)) = A + S + B\cdot C.$$

On large models, success branches are often ignored, i.e. products associated with sequences embed just functional events of failure branches. Let M be an event tree model and C be a consequence. We denote by $FTL^+(M,C)$ the master fault tree associated in this way with the consequence C in the model M when ignoring success branches. In our example, we have

$$FTL^+(M,C) = (B+S)\cdot(C+S) + A$$

It is important to note at this stage that $FTL^+(M,C)$ is always coherent.

The ETL approach

The ETL approach uses relatively large event trees to represent accident sequences. Typically, several event trees are linked together. The ETL approach accounts for dependencies between plant systems differently than the FTL approach does: the fault trees associated with functional events are developed so that they do not share any basic event. At least conceptually, the ETL approach is obtained from the initial model by successive applications of the Shannon decomposition during model development. Each application corresponds to the creation of a new functional event and aims to make function events independent one another. The process stops when all functional events are independent.

Let M be an event tree model and C be a consequence (e.g. core damage, CD). We denote by $ETL(M,C)$ the master fault tree associated with the consequence C in the model M, once developed according to the ETL approach by adding all CD sequences. In the example of Figure 3 (our simplified nuclear power plant model), this would be

$$ETL(M,C) = A + -A\cdot S + -A\cdot -S\cdot B\cdot C$$

Note that $ETL(M,C)$ can also be obtained, at least conceptually, from $FTL(M,C)$, by successive applications of the Shannon decomposition and Boolean simplifications

$$\begin{aligned} FTL(M,C) &= -A\cdot(B+S)\cdot(C+S) + A \\ &= -S\cdot(-A\cdot(B+0)\cdot(C+0)) \\ &\quad + S\cdot(-A\cdot(B+1)\cdot(C+1)) + A \\ &= -S\cdot -A\cdot B\cdot C + S\cdot -A + A \\ &= ETL(M,C) \end{aligned}$$

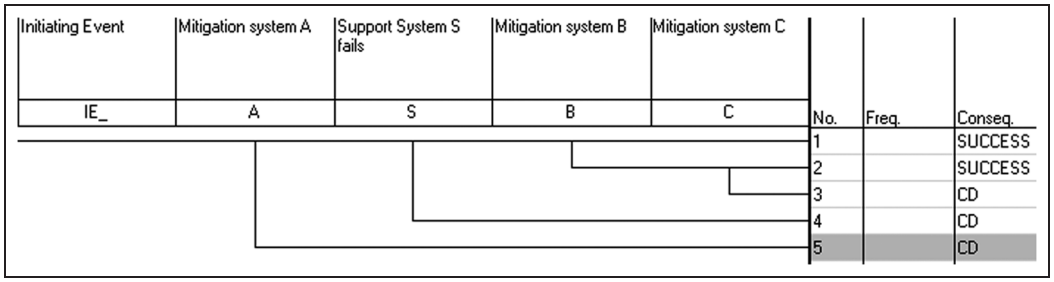

Figure 3. ETL model for the simplified nuclear power plant.

Once the model expansion is completed, individual sequence frequencies can be easily computed in a two-steps process. First, the top event branch failure probabilities are computed, mostly using fault trees. In the second step, each of the accident sequence frequencies is obtained by multiplying the initiating event frequency by the product of all subsequent event tree branch probabilities along the respective sequence path. Here, both success and failure branch probabilities can be easily calculated in the sequence frequency product (because functional events have been made independent of each other). In our example, the exact probability $p(ETL(M,C))$ is calculated as

$$p(ETL(M,C)) = (1-p(A)) * (1-p(S)) * p(B) * p(C) + 1 - p(A) * p(S) + p(A)$$

It would be also possible to apply a minimal cutsets algorithm to quantify the master fault tree of ETL models. However, it is much more efficient to take advantage of the independence of sequences. Moreover, since fault trees associated with functional events are now smaller (and pair wisely independent), it is, in general, possible to assess them exactly with a SDP algorithm, typically with BDD, as in Riskman™.[18] This is why advocates of the ETL approach often claim that ETL is superior to FTL when it comes to PSA/PRA quantification.

In general, there are too many sequences to make it practical for the analyst to assign in advance end states for every sequence. Instead, analysts develop logic rules for assigning the end states so that the software can make the assignments during quantification. Here also, truncation prevails: prefixed sequences with a too low probability are discarded.

Although exact, the ETL approach requires much more manual work from the analyst to make the functional events independent and usually yields less intuitive models. The more shared events there are, the bigger the effort. ETL supporters advocate that this extra workload comes with a great benefit in terms of system and model understanding. We do not enter this debate here.

Strong and weak equivalences between models

As pointed out in the introduction, a general assumption in sciences is that two objects under study can be considered as equivalent if they cannot be distinguished with the observation means at hand. In our case, objects are models, and observation means are quantification algorithms. This leads to the definition of two degrees of equivalence between models.

Let F and G be two Boolean formulas built over the same set of basic events and χ be a cutoff value

- F strongly entails G at precision χ if for any minterm π such that $p(|\pi|) \geqslant \chi$, if $\pi \in F$ then $\pi \in G$. F and G are strongly equivalent at precision χ, if both F strongly entails G and G strongly entails F at precision χ.
- F weakly entails G at precision χ if for any minterm π such that $p(|\pi|) \geqslant \chi$, if $\pi \in G$ then there exists a minterm $\rho \leqslant \pi$ such that $p(|\rho|) \geqslant \chi$ and $\rho \in F$. F and G are weakly equivalent at precision χ, if both F weakly entails G and G weakly entails F at precision χ.

Strong equivalence means that models agree on all failure scenarios. Weak equivalence means that they agree at least on minimal failure scenarios. The following property holds, which defines the degree of equivalence of two models.

Property 7

Let F and G be two Boolean formulas and χ be a cutoff value. Then:

- If F strongly entails G at precision χ, then F weakly entails G at precision χ (weak entailment is coarser the strong entailment).
- If F and G are strongly equivalent at precision χ, then they cannot be distinguished by any correct SDP algorithm: $SDP_{\geqslant\chi}(F) \equiv SDP_{\geqslant\chi}(G)$.
- If F and G are weakly equivalent at precision χ, then they cannot be distinguished by any correct minimal cutsets algorithm: $MCS_{\geqslant\chi}(F) \equiv MCS_{\geqslant\chi}(G)$.

The first two points are direct consequences of the definitions. For the third one, consider a minterm π such that $|\pi|$ is a minimal cutset of F and assume that $p(|\pi|) \geqslant \chi$. Since F weakly entails G, there must be a minterm ρ such that $\rho \leqslant \pi$, $p(|\rho|) \geqslant \chi$ and $\rho \in G$. Now,

the same reasoning applies to ρ. Therefore, there must exist a minterm σ such that $\sigma \leqslant \rho$, $p(|\sigma|) \geqslant \chi$ and $\sigma \in F$. Since $|\pi|$ is a minimal cutset, we must have $\pi = \rho = \sigma$. Therefore, F and G agree on their minimal cutsets at precision χ.

We are now at the heart of our matter and can compare FTL and ETL approaches.

A first, nearly obvious, fact is that by discarding success branches, we make a conservative approximation, which formally established by the following property.

Property 8

Let M be a PSA/PRA model and C be a consequence. Then, $FTL(M,C) \subseteq FTL^+(M,C)$.

Proof. Let π be a minterm such that $\pi \in FTL(M,C)$. Then, there must be a sequence S of M that leads to the consequence C such that π satisfies $FTL(S,C)$. By construction, π must satisfy $FTL^+(S,C)$ as well and therefore $FTL^+(M,C)$.

Now what is really interesting to compare is the FTL and ETL evaluation processes, i.e. the two formulas: $SDP(ETL\chi(M,C))$ and $MCS\chi(FTL(M,C))$. The fundamental result is the following.

Property 9

Let M be a PSA/PRA model, C be a consequence and χ be a threshold value. Then $ETL\chi(M,C)$ and $FTL(M,C)$ are weakly equivalent at precision χ.

Proof. $ETL\chi(M,C)$ is obtained from $FTL(M,C)$ by means iterative applications of two operations: Shannon decomposition and truncation at precision χ. Both operations preserve the weak equivalence, hence the first result.

In other words, although they may give different results, the FTL and ETL approaches agree eventually on what is definitely the most important thing: the most probable accident scenarios. Moreover, thanks to what has been established in the previous section, when the model is coherent and probabilities of basic events not too high, the two approaches should give close quantitative results.

Corollary. Let M be a PSA/PRA model, C be a consequence and χ be a threshold value. If $FTL(M,C)$ is coherent and probabilities of basic events are not too high then

$$p(SDP(ETL\chi(M,C)) \approx rare-events(MCS\chi(FTL(M,C))))$$

Note that one can get rid of imprecision owing to rare events approximation in several ways. For instance, it is possible to calculate a SDP (a BDD) from the minimal cutsets, possibly applying the cutoff χ. It is also possible to apply the Sylvester-Poincaré development using χ to prune products with a too low probability. In both cases, the use of cutoffs is probably necessary in practice to avoid the exponential blow-up of calculation resources.

So, problems, if any, stand in non-coherent models. In the next section, we will discuss where non-coherence may come from and how to cope with it.

Non-coherent models

In this section, we extend those concepts for the treatment of non-coherence model. Non-coherent models typically arise when non-monotonic connectives (e.g. negation) are implemented. In this case, a success may cause the top event to occur, indicating how the non-occurrence of an event can cause the top event. In practice, non-coherent models are much more complex to quantify. In fact, large models are, in many cases, impossible to solve using complete negative logic treatment.

In an ETL approach, there is no problem to solve incoherent models exactly. In FTL however, there is no way the exact top event probability can be calculated from the MCS of an incoherent equation. The strategies developed in the previous sections break down and an appropriate alternative treatment should be looked for. For example, assume the following top event $F = A \cdot B + -A \cdot C$, with cutsets $\{AB, C\}$. There is absolutely no way the original formula can be retrieved from its MCS, i.e. the previous algorithm would fail.

Origin of non-coherence

Negation arises naturally as success branches in event trees. In fault trees however, the use of negation is more questionable. It may be used for a variety of different reasons, and is subject to different algorithmic treatments. In fact, there is no reason for quantification engines to cover all the possible uses of negative logic that can occur. As will be shown later, such result will often have no real-life meaning when working in the failure space. As we shall see, only a few uses of negative logic are pertinent for PSA/PRA.

First, note that FTL models, as found in the nuclear industry, are largely of coherent nature. Typically, only a very small fraction of the logical gates makes use of negative logic. They may be introduced for a number of reasons, for example:

- 'If-Then-Else' (ITE) operations;
- exclude forbidden or impossible configurations;
- conditional adaptation of success criteria;
- taking credit of failures;
- taking credit of success branches;
- delete term;
- exchanging basic events (specific to CAFTA).

A typical use of negative logic in FTL is as follows: the fault tree model needs to re-test a hypothesis that was not queried previously in the attached event tree, leading to the following incoherent ITE operation

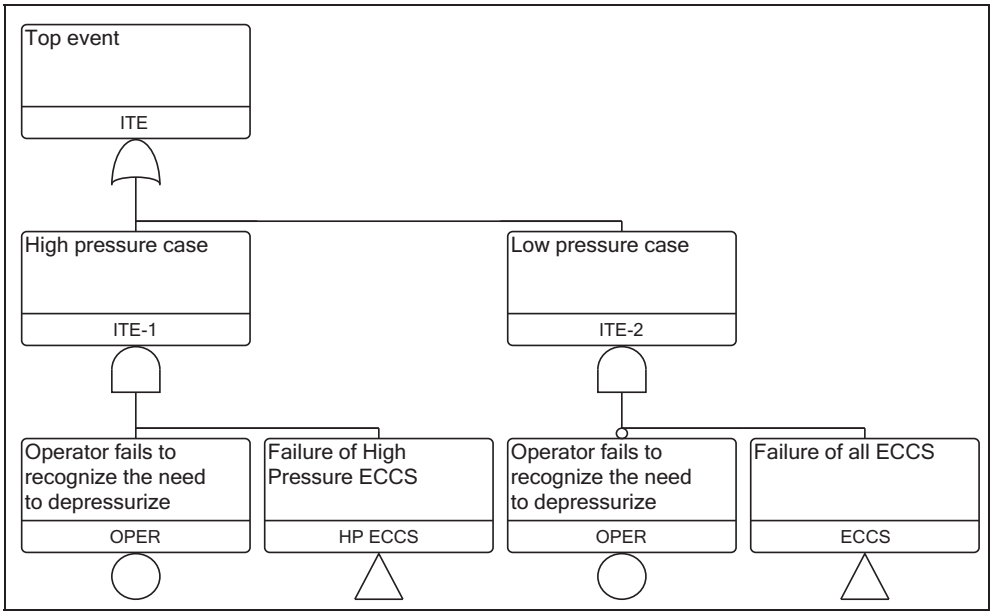

Figure 4. Example of ITE fault tree using negative logic.
ITE: If-Then-Else.

$$F = A \cdot B + -A \cdot C, \; with \, MCS(F) = \{AB, C\}$$

As preparatory task for this work, we conducted an international survey on the use of negative logic among practitioners. We distributed a questionnaire asking for the different uses of negative logic in nuclear PSA/PRA models. Participating countries included Sweden, Finland, France, Germany, Switzerland, Spain and USA. This survey allowed us to better appraise the numerous reasons for using negation (major ones are listed above). Based on the answers from the questionnaire, a categorization according to mathematical characteristics and treatment by quantification engines was undertaken. The following three categories were identified and will be discussed in the next sections.

- Exclusion of forbidden or impossible configurations.
- Conditional adaptation of success criteria (ITE operation).
- Delete terms.

Exclusion of forbidden or impossible configurations

A typical example of this category is when the analyst wants to exclude different configuration alignments. Assume we have a system consisting of two trains A and B. Each train is taken out of service one day every 30 days, with failure probabilities 10^{-3} when operating. As both trains are not allowed to be taken out of service at the same time by administrative rules, practitioners typically implement negative logic to exclude this particular state. In practice, this is modelled using two basic events per train, e.g. PA stands for 'failure of the train A' and UA stands for 'train A is out of service'. The probability of PA and UA are respectively 10^{-3} and 1/30. The unavailability of the whole system is then modelled as follows

$$\mathrm{F} = (\mathrm{PA} \cdot \mathrm{UB} \cdot -\mathrm{UA}) + (\mathrm{PB} \cdot \mathrm{UA} \cdot -\mathrm{UB}) + \mathrm{PA} \cdot \mathrm{PB}$$

The above equation is however an approximation, for PA and UA are not independent events. In fact, the described situation can be solved exactly neither by a FTL nor by an ETL approach, since it is a superposition of two distinct states.

Using the rare event approximation to solve such problems yields optimistic results in general, as combined failures of A and B when no system is in service is neglected. It is possibly to solve such problems exactly though. To achieve this, quantification engines need to quantify each system configuration individually and sum over all independent configurations so created. This feature is available in Riskman, which can manage up to 32 different configuration states. To the knowledge of the authors, no other code offers such a configuration management feature.

Conditional adaptation of success criteria (ITE operation)

The If-Then-Else (ITE) operation is by far the most common use of negative logic in fault trees, and is also the most difficult case of negative logic to solve by quantification engines, especially in an FTL framework. A typical example of this category is when the analyst wants to re-ask a question that, for some reason, was not properly queried and segregated out in the event tree model (see example in Figure 4).

The use of the ITE operation obviously yields a non-coherent Boolean equation that cannot be solved exactly (with the FTL approach). However, an easy

```
AssessSequence( S: sequence ) : real
if S contains only positive events
then  calculate MCS(S), evaluate p(MCS(S)) and return it
else  S = -Q.S'
      p1 = AssessSequence( S' )
      p2 = AssessSequence( Q.S' )
      return p1 − p2
```

Figure 5. An algorithm to assess sequences with success branches.

workaround exists that will permit solving the problem exactly: when working in FTL, the attached event tree can be rewritten so that the ITE condition (in our example OPER) can be tested in advance in the event tree. This makes it disjoint (e.g. by application of the Shannon decomposition), and coherency is regained (at least in the fault tree). The event tree 'absorbs' the non-coherent parts. The rewriting task can be either done by the analyst during model development (as in the ETL approach), or, more appealingly, automatically by the quantification engine during event tree pre-processing. This comes at a cost however; it generates new functional events in the event trees, whose non-coherent success branches need to be calculated, if possibly exactly.

Delete terms

A delete term operation is simply the task to remove one or many MCS or basic events from the overall result. A typical example is when the analyst wants to get rid of a specific basic event(s) in a fault tree. Instead of duplicating the entire fault tree, one typically re-uses an existing fault tree by multiplying (under an AND connective) with the negated basic event in question.

Similar to truncation on a FTL framework, deleting basic events or MCS does not yield incoherent results. The resulting model is, despite appearances, coherent, and the previous results hold.

Exact quantification of success paths in event trees

It may be the case that one wants to assess exactly sequences of events trees that contain success branches. There are different ways of handling these branches (besides calculating a SDP).

Let $S = P_1 \cdot \ldots \cdot P_m \cdot -Q_1 \cdot \ldots \cdot -Q_n$ be a sequence of functional events.

A first way to handle this sequence, as implemented for instance in RiskSpectrum,[4] consists in calculating the MCS of the formula $P_1 \cdot \ldots \cdot P_m$, then those of the formula $Q_1 + \ldots + Q_n$ and finally to remove from the former the minimal cutsets π such that there exists a MCS ρ of the latter's such that $\rho \subseteq \pi$.

The MCS algorithm proposed recently by Rauzy[13] is able directly to handle negative logic. However, in both cases the quantification passes by the calculation of minimal cutsets and is subject to the problems raised in the section devoted to algorithmic issues.

The algorithm presented in Figure 5 makes it possible to assess the probability of a sequence with success branches, while calculating only MCS of sequences with no success branches. This algorithm relies on the following property.[11]

Property 10 (valiant)

Let P and Q be two coherent formulas. Then, $p(P \cdot -Q) = p(P) - p(P \cdot Q)$.

For instance, let $S = A \cdot -B \cdot -C$. To assess S, we proceed as follows

$$p(A \cdot -B \cdot -C) = p(A \cdot -C) - p(A \cdot B \cdot -C)$$
$$p(A \cdot -C) = p(A) - p(A \cdot C)$$
$$p(A \cdot B \cdot -C) = p(A \cdot B) - p(A \cdot B \cdot C)$$

Note that it is easy to add a caching mechanism to the algorithm in order not to redo the same work twice, especially in the case where several sequences should be assessed. Note also that the calculation of an upper bound of the error owing to cutoffs is affected by this way of calculating success branches.

Conclusion

In this article, we proposed a formal framework to compare the FTL and the ETL approach of PSA/PRA. We showed that these approaches are equivalent in the sense that they agree on most probable failure scenarios. As a consequence, if the model under study is coherent and probabilities of basic events are not too high, quantitative results given by both approaches should be close. We discussed also from where non-coherence comes from and how to cope in practice with it.

Both FTL and ETL approaches clearly satisfy the PSA/PRA needs when applied correctly, i.e. according to the algorithmic recommendations of the preceding sections. Generally speaking, the following observations can be made.

- The strength of the FTL approach lies in the fact that the model is (i) complete and (ii) built in an intuitive manner, using immediate cause concept in fault trees, and causality chains in event trees.
- The strength of the ETL approach lies in the fact that the model has been initially decomposed in disjoint functions and that model can easily be solved exactly.
- The shortcomings of the FTL approach lie in the difficulty to solve it exactly (e.g. without approximation).
- The shortcomings of the ETL approach lie in the fact that sequences are made pair-wisely disjoint, leading to (i) *a priori* truncated models (owing to cutoffs), and (ii) making the model less easy to construct and review because it is more distant from the architecture of the plant under study than an FTL model would be.

To a large extent, the choice of one or the other methodology is a matter of taste. Clearly, some operations that are manually performed, such as the creation of a functional event from an internal event of a fault tree, could be advantageously automated or semi-automated.

Conflict of interest statement

The author declares that there is no conflict of interest.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

1. Kumamoto H and Henley EJ. *Probabilistic risk assessment and management for engineers and scientists*. IEEE Press, 1996. New York USA.
2. NUREG/CR-2300. A guide to performance of probabilistic risk assessments for nuclear power plants. Washington DC: US Nuclear Regulatory Commission, 1983.
3. Fussell JB and Vesely WE. A new methodology for obtaining cut sets for fault trees. *Trans Am Nucl Soc* 1972; 15: 262–263.
4. Berg U. *RISK SPECTRUM, Theory Manual*, 1994.
5. Jung WS, Han SH and Ha J. A fast BDD algorithm for large coherent fault trees analysis. *Reliab Engng Sys Saf* 2004; 83(3): 369–374.
6. Jung WS. ZBDD algorithm features for an efficient probabilistic safety assessment. *Nuclear Engng Des* 2009; 239(10): 2085–2092.
7. Brace K, Rudell R and Bryant R. Efficient implementation of a BDD package. In: *Proceedings of the 27th ACM/IEEE design automation conference*, 1990. IEEE. pp.40–45. of the event, publisher, and location of the publisher. Orlando, Florida, USA, June 24–28, 1990. IEEE Computer Society Press 1990, ISBN 0-8186-9650-X, New York, USA.
8. Rauzy A. BDD for reliability studies. In: Misra KB (ed.) *Handbook of performability engineering*. Elsevier, 2008, pp.381–396. London, England.
9. Epstein S and Rauzy A. Can we trust PRA? *Reliab Engng Sys Saf* 2005; 88(3): 195–205.
10. Rauzy A. Mathematical foundation of minimal cutsets. *IEEE Trans Reliab* 2001; 50(4): 389–396.
11. Valiant LG. The complexity of enumeration and reliability problems. *SIAM J Comput* 1979; 8: 410–421.
12. Minato S. Zero-suppressed BDDs for set manipulation in combinatorial problems. In: *Proceedings of the 30th ACM/IEEE Design automation conference*, 1993, pp.272–277. Dallas, Texas, USA, June 14–18, 1993. ACM Press, New York, NY, USA 1993 ISBN 0-89791-577-1 Reference 13. June 23–27, Helsinki Finland, IAPSAM, USA, ISBN: 978-1-62276-436-5.
13. Rauzy A. Anatomy of an efficient fault tree assessment engine. In: Virolainen R (ed.) *Proceedings of international joint conference PSAM'11/ESREL'12*, 2012, Helsinki.
14. Rauzy A. New algorithms for fault trees analysis. *Reliab Engng Sys Saf* 1993; 59(2): 203–211.
15. Nikolskaia M and Nikolskaia L. Size of OBDD representation of 2-level redundancies functions. *Theoret Comput Sci* 2001; 255(1–2): 615–625.
16. Duflot N, Berenguer C, Dieulle L, et al. How to build an adequate set of minimal cut sets for PSA importance measures calculation. In: Stamatelatos MG and Blackman HS (eds) *Proceedings of the 8th international conference on probabilistic safety assessment and management*, 2006. May 14–18, 2006, New Orleans, Louisiana, ISBN-10: 0791802442| ISBN-13: 978-0791802441, Cdr
17. Nusbaumer O. *Analytical solutions of linked fault tree probabilistic risk assessments using binary decision diagrams with emphasis on nuclear safety applications*. Zurich: ETH Zurich; 2007.
18. Wakefield DJ, Epstein SA, Xiong Y, et al. Riskman™, celebrating 20+ years of excellence! In: *Proceedings of PSAM'10 conference*, 2010, Seattle. June 7–11, Seattle, USA, IAPSAM, USA