Algebraic Determination of the Structure Function of Dynamic Fault Trees

Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage

Algebraic determination of the structure function of Dynamic Fault Trees

G. Merle*, J.-M. Roussel, J.-J. Lesage

LURPA - ENS Cachan, 61 avenue du Président Wilson, Cachan 94230, France

Abstract

This paper presents an algebraic framework allowing to algebraically model dynamic gates and determine the structure function of any Dynamic Fault Tree (DFT). This structure function can then be exploited to perform both the qualitative and quantitative analysis of DFTs directly, even though this latter aspect is not detailed in this paper. We illustrate our approach on a DFT example from the literature.

Keywords: Dynamic fault tree, algebraic modeling, structure function.

1. Introduction

The structure function of a Static Fault Tree (SFT) – a fault tree (FT) which only contains gates OR, AND, and K-out-of-N – is a Boolean function which represents the failure of the top event (*TE*) according to the failure of the basic events (*BEs*) of the FT. This algebraic model is classically used to perform both the qualitative and quantitative analysis of SFTs directly. For complex systems, these analyses are most often performed thanks to BDD-based methods [9, 19] or other combinatorial techniques [1, 17].

The introduction of dynamic gates – gates PAND, FDEP, and Spare – in FTs has changed the nature of the relation between the *TE* and the *BEs*. In a Dynamic Fault Tree (DFT), the failure of the *TE* depends not only on the failure of the *BEs* but also on the order of occurrence of these failures. As this last aspect is not taken into account in the Boolean model of failures (which only expresses whether a *BE* has occurred or not), a classical Boolean function cannot represent the dynamic relations between the *TE* and the *BEs* that exist in a DFT.

In a previous article, we presented the basics of an algebraic framework allowing to algebraically model dynamic gates PAND and FDEP, and determine the structure function of any Dynamic Fault Tree (DFT) containing these gates [13]. In this paper, we extend our previous work to Spare gates in order to be able to determine the structure function of any DFT. This structure function is based on a specific algebraic model of failures which allows to take into account the order of occurrence of failures. As this algebraic model is an extension of the Boolean model used for SFTs, all the results previously obtained for SFTs are preserved.

This paper is organised as follows. The most common approaches used to perform the analysis of DFTs are presented in Section 2. The algebraic framework that we introduce to model DFTs is detailed in Section 3, and the algebraic model of dynamic gates which can be determined from it is presented in Section 4. This algebraic model allows to determine the canonical form of the structure function of any DFT, as shown in Section 5, and our approach is illustrated on a DFT example in Section 6. Finally, we show how the qualitative analysis of DFTs can be performed directly from the canonical form of the structure function in Section 7.

2. State of the art

Several approaches have been used to avoid the problem of the determination of the structure function of DFTs. These approaches can be either modular or global.

Global approaches consist in solving the whole DFT directly, whereas modular approaches consist in:

- dividing the DFT into independent static and dynamic subtrees (or modules) prior to analysis: if a substree contains static gates only, it is considered as static; if a subtree contains at least one dynamic gate, it is considered as dynamic;
- solving the modules separately; and
- combining the results of the various modules to get the overall result for the entire tree.

Various methods exist to analyze the static and dynamic modules of DFTs. On the one hand, solving static modules can be done by using Binary Decision Diagrams (BDDs), other combinatorial techniques, or even some DFT Analysis models such as Markov Chains. On the other

*Corresponding author: Tel.: +33 6 23 79 54 51; fax: +33 1 47 40 22 20

Email addresses: guillaume.merle@lurpa.ens-cachan.fr (G. Merle), jean-marc.roussel@lurpa.ens-cachan.fr (J.-M. Roussel), jean-jacques.lesage@lurpa.ens-cachan.fr (J.-J. Lesage)

hand, solving dynamic modules is generally done using Input/Output Interactive Markov Chains [4, 5, 6], Stochastic Petri Nets (SPN) [1, 7], or Temporal Bayesian Networks [2, 3, 15]. On the one hand, Markov Chains provide the cut sequences of the (sub)tree, which are the failure sequences that lead to the states of the Markov Chain in which the *TE* of the (sub)tree fails. They also provide the failure probability of the *TE* of the (sub)tree by solving the set of differential equations which is equivalent to the Markov Chain. On the other hand, the reachability graph of SPNs provides the cut sequences of the (sub)tree, and the failure probability of the *TE* can be computed after converting the SPN into its corresponding Markov Chain. However, in both cases, the failure of the components of the system is most often modeled by exponential time-to-failure distributions. Temporal Bayesian Networks allow to address this limit by allowing to consider other distributions. However, Bayesian Networks only allow to perform the quantitative analysis of the dynamic (sub)tree, and the inference algorithms used limit the distributions considered to Gaussian distributions [10] and mixtures of truncated exponentials [16].

An analytic approach was introduced in [23] to analyze DFTs by modelling the dynamic gate PAND and by determining simplification theorems. The authors focus on three temporal gates: gates PAND, Simultaneous-AND (SAND), and Priority-OR (POR). Gate SAND was created to address the ambiguity encountered in the definition of gate PAND regarding the simultaneity of input events, whereas gate POR was created from the definition of the Exclusive-OR gate found in [22]. Each event of the FT is assigned a sequence value which allows to know the order in which events occur. The authors propose an extension of truth tables, denoted as Temporal Truth Tables and based on these sequence values, to prove theorems allowing to simplify the FTs which contain the three temporal gates considered. Nevertheless, this approach allows to perform the qualitative analysis of FTs, only, and the only dynamic gate considered is gate PAND.

We have not found in the literature any attempt to provide an algebraic model for all dynamic gates allowing to determine the structure function of a DFT explicitly as it is currently the case for SFTs. The goal of the algebraic determination of the structure function of DFTs is to be able to perform their analysis directly whatever the distribution considered for basic events. We present such an algebraic framework in Section 3.

3. Algebraic framework for the modeling of Dynamic Fault Trees

3.1. Temporal model of non-repairable events

The structure function of SFTs is based on a Boolean model of events, and of basic events in particular. With this simple model, the only aspect which is taken into account is the presence or absence of failure. However, this Boolean model cannot render the order of occurrence of events which is necessary for the modeling of dynamic gates. To count on the temporal aspect of events, we consider the top event, the intermediate events, and the basic events as *temporal functions*, which are piecewise right-continuous on $\mathbb{R}^+ \cup \{+\infty\}$, whose range is $\mathbb{B} = \{0,1\}$. In accordance with [22], we consider all events as non-repairable, each of them being perfectly defined by its unique date of occurrence–noted $d(a)$ for an event a. A generic timing diagram of such an event a is given in Fig. 1. In this paper, we denote by $\mathcal{E}_{nr}$ the set of these temporal functions, which corresponds to the set of non-repairable events.

With this temporal model, non-repairable events can be ordered according to their date of occurrence. This characteristics is the cornerstone of our approach, and it is used to model each of the operators which are needed to model both the static and dynamic gates of FTs.

However, specific attention must be paid to the simultaneity of events when modeling the order of occurrence of events, as it is explained below.

3.2. Simultaneity in DFTs

In a FT, simultaneity among events may arise in two ways. Independent basic events can occur simultaneously if they have a discrete probability distribution with a non-null probability mass exactly at the same time. Because the failure probability distributions are usually considered as continuous functions with infinite support, the simultaneous occurrence has null probability, and can be neglected. A second case of simultaneity may arise at any level of a FT when there are repeated basic events. FTs with repeated events represent the most powerful combinatorial model in dependability [11], and require ad hoc analysis techniques. Nevertheless, the presence of repeated events across modules of dynamic gates has not yet been explored in its full generality. In [24], repeated events are allowed, but the paper does not provide any algorithm to derive the list of the cut sequences.

Let us consider the DFT in Fig. 2, in which event A is a repeated basic event.

If basic events A, B, C, and D occur according to sequences $[B, C, A]$, $[C, B, A]$, or $[D, A]$, intermediate events G and H occur simultaneously at the same time as A occurs. This example shows that intermediate nodes of a FT can occur simultaneously because of the presence of repeated basic events. The simultaneity problem has been briefly addressed in [4], and has been solved by resorting to the concept of "non-determinism", a concept that is not

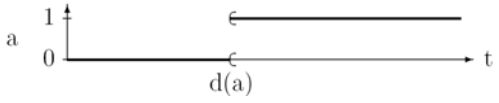

Figure 1: A non-repairable event.

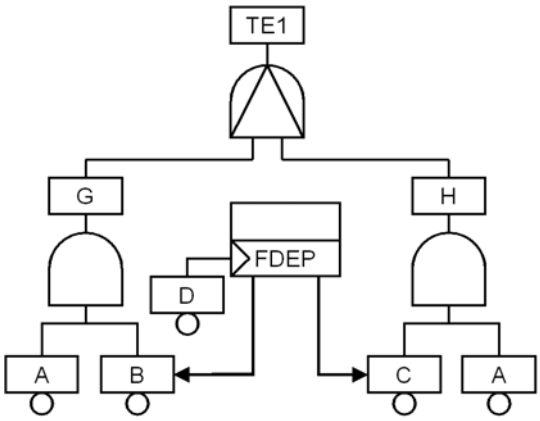

Figure 2: An example of DFT with one repeated basic event.

easy to accept in engineering practice because many engineers believe that the behavior of technical systems, and in particular control systems, must necessarily be deterministic. We assert that a choice must be made regarding the semantics of simultaneous events, and dynamic gates. For instance, in the case of simultaneous events in input to a PAND gate, two choices are possible (Fig. 2):

- if the order relation is considered strictly, when intermediate events G and H occur simultaneously, *TE1* does not occur, and gate PAND would then be considered as being "non-inclusive"; and
- if the order relation is not considered strictly, when intermediate events G and H occur simultaneously, *TE1* occurs at the same time as G or H, and gate PAND would then be considered as being "inclusive".

Both interpretations of the order relation can be taken into account, and algebraically modeled.

3.3. Boolean operators

Any elements of $\mathcal{E}_{nr}$ can be composed thanks to a rewriting of classical Boolean operators. The temporal definition of Boolean operators OR and AND, based on the dates of occurrence of a and b (which are denoted by $d(a)$ and $d(b)$, respectively), is

$$d(a+b) = \begin{cases} d(a) & \text{if } d(a) < d(b) \\ d(a) & \text{if } d(a) = d(b) \\ d(b) & \text{if } d(a) > d(b) \end{cases}$$

$$d(a \cdot b) = \begin{cases} d(b) & \text{if } d(a) < d(b) \\ d(a) & \text{if } d(a) = d(b) \\ d(a) & \text{if } d(a) > d(b) \end{cases}$$

Indeed, $a + b$ occurs as soon as a or b occurs, and $a \cdot b$ occurs as soon as a and b have occurred. It can be noted that operators OR and AND are commutative.

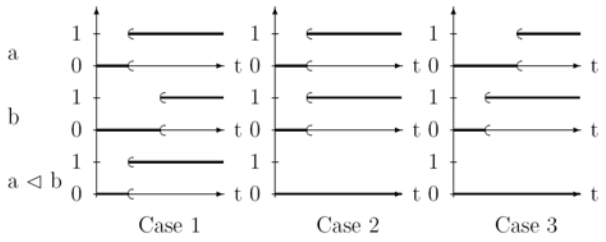

Figure 3: Timing diagrams of operator non-inclusive BEFORE (BF).

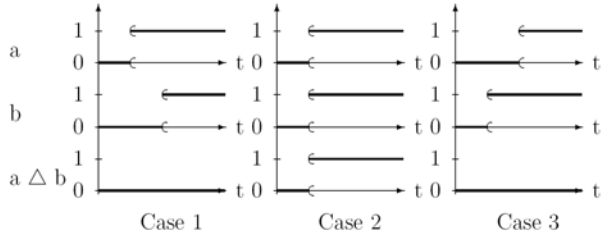

Figure 4: Timing diagrams of operator SIMULTANEOUS (SM).

The identity elements of operators OR and AND in $\mathcal{E}_{nr}$ are denoted by $\perp$, and $\top$, respectively, to which these dates can be assigned:

$$d(\perp) = +\infty, \qquad d(\top) = 0$$

$\perp$ is the never-occurring event whereas $\top$ is the always-occurring event.

3.4. Temporal operators

To model the order of occurrence of events, we introduce an operator non-inclusive BEFORE (BF, with symbol $\lhd$), and an operator SIMULTANEOUS (SM, with symbol $\triangle$), whose formal definitions, based on the dates of occurrence of a and b, are

$$d(a \lhd b) = \begin{cases} d(a) & \text{if } d(a) < d(b) \\ +\infty & \text{if } d(a) = d(b) \\ +\infty & \text{if } d(a) > d(b) \end{cases}$$

$$d(a \triangle b) = \begin{cases} +\infty & \text{if } d(a) < d(b) \\ d(a) & \text{if } d(a) = d(b) \\ +\infty & \text{if } d(a) > d(b) \end{cases}$$

The result of the composition of two events a and b by operators BF and SM is illustrated by the timing diagrams in Figs. 3 and 4, respectively, in three cases: Case 1: $d(a) < d(b)$, Case 2: $d(a) = d(b)$, Case 3: $d(a) > d(b)$.

Based on the previous two operators, we can introduce a non-strict or INCLUSIVE BEFORE (IBF, with symbol $\trianglelefteq$) operator

$$a \trianglelefteq b = a \lhd b + a \triangle b \tag{1}$$

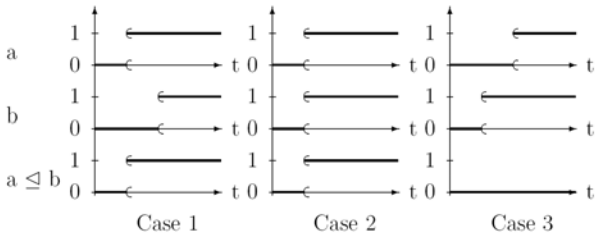

Figure 5: Timing diagrams of operator INCLUSIVE BEFORE (IBF).

whose definition, based on the dates of occurrence of a and b, is

$$d(a \trianglelefteq b) = \begin{cases} d(a) & \text{if } d(a) < d(b) \\ d(a) & \text{if } d(a) = d(b) \\ +\infty & \text{if } d(a) > d(b) \end{cases}$$

The result of the composition of two events a and b by operator IBF is illustrated by the timing diagrams in Fig. 5 in three cases: Case 1: $d(a) < d(b)$, Case 2: $d(a) = d(b)$, Case 3: $d(a) > d(b)$.

According to these timing diagrams, and to (1), $a \trianglelefteq b$ occurs in two cases: when a occurs strictly before b, Case 1 (which corresponds to $a \triangleleft b$); and when a occurs at the same time as b, Case 2 (which corresponds to $a \triangle b$).

3.5. Algebraic structure of $\mathcal{E}_{nr}$

Static Fault Tree Analysis is mainly based on the structure function of SFTs, which is determined and simplified thanks to a substructure of the Boolean algebra of Boolean values: the Abelian dioid $\{\{0,1\},+,\cdot,0,1\}$[1].

We have demonstrated in [14] that by providing a new temporal definition of events (Section 3.1) and by rewriting the Boolean operators + and · (Section 3.3), the framework obtained still has an Abelian dioid structure. The properties that are commonly used for the simplification of SFTs can hence still be applied with our model, and their structure functions can be determined as usual. In particular, operators OR and AND satisfy the four additional following theorems, which are theorems which hold on Boolean algebras:

$$a + (b \cdot c) = (a + b) \cdot (a + c) \tag{2}$$

$$a + \top = \top \tag{3}$$

$$a + (a \cdot b) = a \tag{4}$$

$$a \cdot (a + b) = a \tag{5}$$

$(\mathcal{E}_{nr}, +, \cdot)$ thus has an algebraic structure which allows to express gates OR, AND, and K-out-of-N, and to determine the structure function of SFTs as it is commonly done by using the classical Boolean algebra of Boolean variables. Temporal operators non-inclusive BEFORE and Inclusive BEFORE satisfy several theorems which are useful for calculation, some of which are presented in Appendix A. The proofs of these theorems can be found in [12]. Furthermore, the temporal operators introduced in this section allow to determine an algebraic model of dynamic gates, as shown in Section 4.

[1] $\{\{0,1\},+,\cdot,0,1\}$ is an Abelian dioid because + and · are commutative, idempotent, and, respectively, allow 0 and 1 as their identity element, · is distributive over +, and · allows 0 as an absorbing element.

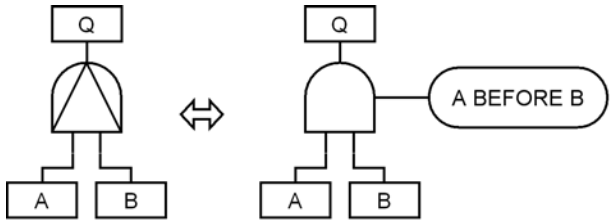

Figure 6: Definition of gate PAND from [20].

4. Algebraic model of dynamic gates

Based on the algebraic framework introduced in Section 3, the algebraic model of dynamic gates is now going to be developed. The temporal operators introduced in Section 3.4 allow to take into account and algebraically model both a strict and a non-strict order relation. However, a non-strict inclusive interpretation of dynamic gates seems more coherent with the designers' expectations. For this reason, in the remainder of this paper, we define an algebraic model of dynamic gates by means of operator IBF ($\trianglelefteq$), only, even though it is easy to define an algebraic model of dynamic gates by means of operator BF ($\triangleleft$) as well.

Furthermore, in accordance with [13], we assume that basic events are statistically independent, and have a continuous failure time distribution, so that they cannot occur simultaneously. Hence, for any two basic events a and b with the above characteristics, the following relation holds:

$$a \triangle b = \bot \tag{6}$$

The algebraic models of gates PAND and FDEP are presented in Sections 4.1 and 4.2, respectively. The algebraic model of Spare gates is presented in an increasing order of complexity: Spare gates with 2 and 3 input events are studied in Sections 4.3 and 4.4, respectively. Besides, we consider that there is only one type of Spare gate which is the Warm Spare gate and that Cold and Hot Spare gates [20] are particular cases of Warm Spare gates. Both of them are studied in Section 4.6.

4.1. Algebraic model of gate PAND

According to [20], gate PAND is defined in Fig. 6.

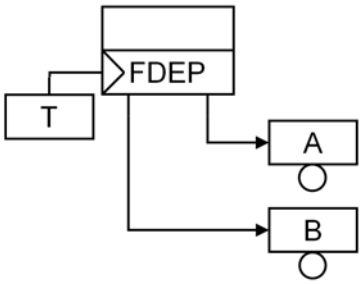

Figure 7: An FDEP gate with two dependent basic events A and B.

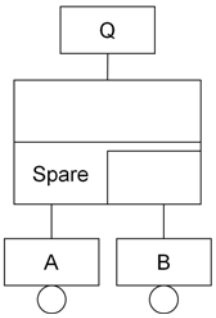

Figure 8: A single Spare gate with one primary event A and one spare event B.

An algebraic model of gate PAND can hence be determined as

$$Q = (A \cdot B) \cdot (A \trianglelefteq B)$$

This model can be simplified, thanks to the theorems of Appendix A, as follows[2]:

$$\begin{array}{lcl} Q & = & (A \cdot B) \cdot (A \trianglelefteq B) \\ Q & \overset{(A.30)}{=} & B \cdot (A \trianglelefteq B) \end{array}$$

4.2. Algebraic model of gate FDEP

According to [4, 8, 20], the FDEP gate – Functional Dependency gate – is a dynamic gate comprised of a trigger input event – either a basic event or the output of another gate of the tree – and a set of dependent basic events. Fig. 7 provides a pictorial depiction of an FDEP gate with two dependent basic events A and B, T representing the trigger event. When the trigger event occurs, the dependent basic events are forced to occur.

In Fig. 7, basic events A and B can fail by themselves, or can be forced to fail by the trigger event T. In accordance with [3], we choose to denote the global failure of basic events A, and B by the substituted variables A_T, and B_T to explicitly indicate the effect of trigger T: basic event A fails (A_T) if it is forced to fail by the trigger event (T) or if it fails by itself before the trigger event fails ($A \trianglelefteq T$). The algebraic model of gate FDEP thus is

$$\left\{ \begin{array}{lcl} \mathrm{A_T} & = & \mathrm{T} + (\mathrm{A} \trianglelefteq \mathrm{T}) \\ \mathrm{B_T} & = & \mathrm{T} + (\mathrm{B} \trianglelefteq \mathrm{T}) \end{array} \right.$$

This model can be simplified, thanks to the theorems of Appendix A, as follows:

$$\left\{ \begin{array}{lcl} \mathrm{A_T} & = & \mathrm{T} + (\mathrm{A} \trianglelefteq \mathrm{T}) \overset{(A.29)}{=} \mathrm{A} + \mathrm{T} \\ \mathrm{B_T} & = & \mathrm{T} + (\mathrm{B} \trianglelefteq \mathrm{T}) \overset{(A.29)}{=} \mathrm{B} + \mathrm{T} \end{array} \right.$$

[2]In the equation below, the notation $\overset{(A.30)}{=}$ indicates that the expression $B \cdot (A \trianglelefteq B)$ is obtained from the expression $(A \cdot B) \cdot (A \trianglelefteq B)$ by applying theorem $(A.30)$ from the Appendix. This notation will be used in the remainder of this paper.

Thanks to this simplification, it appears that the behavior of dynamic gate FDEP is equivalent to the behavior of static gate OR, as it has been suggested by some authors [20].

4.3. Algebraic model of Spare gates with two input events

In this section, we completely detail the algebraic model of Spare gates with two input events in the main configurations which may be encountered in DFTs. The different cases are treated in an increasing order of complexity, from a single Spare gate in Section 4.3.1 to two Spare gates sharing a spare event in Section 4.3.2, and even to the generalization to n Spare gates sharing a spare event in Section 4.3.3.

4.3.1. Algebraic model of a single Spare gate

Let us consider a Spare gate with two input events – the primary event A and one spare event B – as shown in Fig. 8.

As stated in [20], the output Q of the gate occurs when the primary and all spares have failed, so when A and B have failed, in this case. A and B are basic events and cannot fail simultaneously – $A \triangle B = \perp$ – so Q will occur if A and B fail according to sequences $[A, B]$ or $[B, A]$. It is important to note that in sequence $[A, B]$, B fails while in its active mode (denoted as B_a), whereas in sequence $[B, A]$, B fails while in its dormant mode (denoted as B_d). It is essential to distinguish both failure modes by using two different variables, for quantitative analysis purposes. Indeed, B does not have the same failure distribution when it fails during its dormant mode ($B \equiv B_d$) or during its active mode ($B \equiv B_a$). As we aim at making possible the quantitative analysis of DFTs from their structure function, this structure function must hence provide sufficient information to know whether spare events are in their dormant or active mode.

The algebraic model of gate Spare can hence be expressed as

$$Q = B_a \cdot (A \lhd B_a) + A \cdot (B_d \lhd A)$$

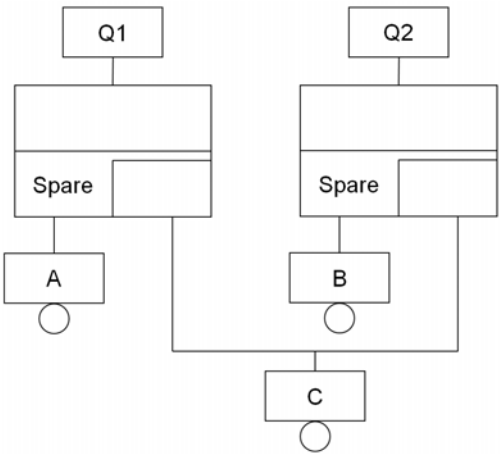

Figure 9: Two Spare gates sharing a spare event C.

Furthermore, as B cannot be both in an active state and in a dormant state, we have

$$B_d \cdot B_a = \bot$$

4.3.2. Algebraic model of two Spare gates sharing a spare event

Let us now consider two Spare gates with two input events – with primary events A and B – sharing a spare event C, as shown in Fig. 9.

If we focus on the Spare gate on the left side, $Q1$ will occur as soon as A and C have failed – as stated in Section 4.3.1 – or if A fails and C is made unavailable because B has failed before A. As a consequence, the algebraic model of this Spare gate is

$$\begin{cases} Q1 = C_a \cdot (A \lhd C_a) + A \cdot (C_d \lhd A) + A \cdot (B \lhd A) \\ C_d \cdot C_a = \bot \end{cases}$$

The algebraic expression for the Spare gate on the right side can be determined in the same way by symmetry. Consequently, the final algebraic model of any of two Spare gates sharing a spare event is

$$\begin{cases} Q1 = C_a \cdot (A \lhd C_a) + A \cdot (C_d \lhd A) + A \cdot (B \lhd A) \\ Q2 = C_a \cdot (B \lhd C_a) + B \cdot (C_d \lhd B) + B \cdot (A \lhd B) \\ C_d \cdot C_a = \bot \end{cases}$$

4.3.3. Algebraic model of n Spare gates sharing a spare event

Let us consider n Spare gates with 1 output event Q_i and two input events: a primary event P_i – $i \in \{1, \cdots, n\}$ – and a spare event S.

If we focus on the first Spare gate, Q_1 will occur as soon as P_1 and S have failed – as stated in Section 4.3.1 – or if P_1 fails and S is made unavailable because the primary event of any of the other Spare gates has failed before P_1. As a consequence, the algebraic model of the first Spare gate is

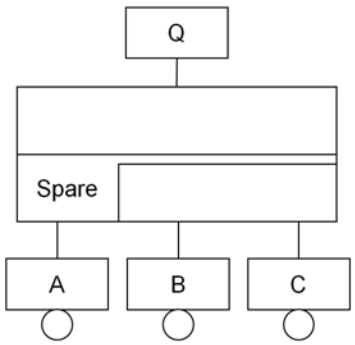

Figure 11: A single Spare gate with one primary event A and two spare events B and C.

$$\begin{cases} Q_1 = S_a \cdot (P_1 \lhd S_a) + P_1 \cdot (S_d \lhd P_1) \\ \qquad + \sum_{i \neq 1} P_1 \cdot (P_i \lhd P_1) \\ S_d \cdot S_a = \bot \end{cases}$$

The algebraic expression for Q_i, $i \in \{1, \ldots, n\}$, can be determined in the same way by symmetry. Consequently, the final algebraic model of any of n Spare gates sharing a spare event is

$$\begin{cases} Q_i = S_a \cdot (P_i \lhd S_a) + P_i \cdot (S_d \lhd P_i) \\ \qquad + \sum_{j \neq i} P_i \cdot (P_j \lhd P_i) \\ S_d \cdot S_a = \bot \end{cases}$$

4.4. Algebraic model of a single Spare gate with three input events

Let us consider a Spare gate with three input events – the primary event A and two spare events B and C – as shown in Fig. 11.

As stated in [20], the output Q of the gate occurs when the primary and all spares have failed, so when A, B, and C have failed. A, B, and C are basic events and cannot fail simultaneously so Q will occur if A, B, and C fail according to sequences $[A, B, C]$, $[A, C, B]$, $[B, A, C]$, $[B, C, A]$, $[C, A, B]$, or $[C, B, A]$. It is important to note that, when the quantitative analysis will be performed from the structure function, B and C will not have the same distribution function in the six sequences. For instance, in sequence $[A, B, C]$, both B and C fail during their active mode (denoted by B_a and C_a), whereas in sequence $[B, C, A]$, both B and C fail during their dormant mode (denoted by B_d and C_d). The algebraic model of gate Spare can hence be expressed as

$$\begin{aligned} Q \;=\; & C_a \cdot (A \lhd B_a) \cdot (B_a \lhd C_a) \\ & + B_a \cdot (A \lhd C_d) \cdot (C_d \lhd B_a) \\ & + C_a \cdot (B_d \lhd A) \cdot (A \lhd C_a) \\ & + A \cdot (B_d \lhd C_d) \cdot (C_d \lhd A) \\ & + B_a \cdot (C_d \lhd A) \cdot (A \lhd B_a) \\ & + A \cdot (C_d \lhd B_d) \cdot (B_d \lhd A) \end{aligned}$$

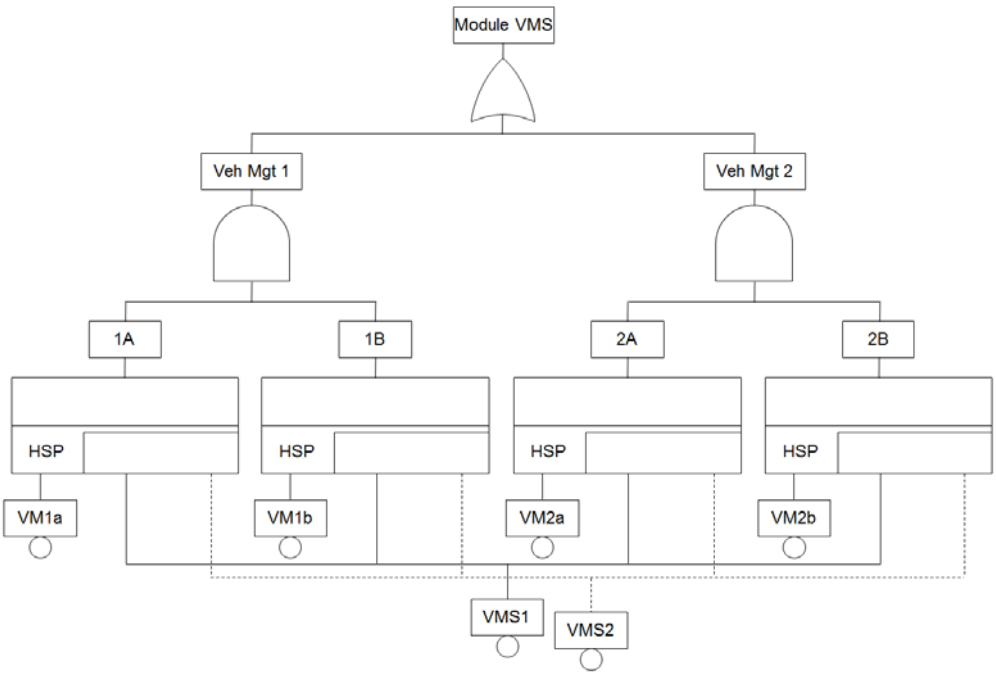

Figure 10: A benchmark from [25] with four Spare gates with three input events sharing two spare events.

As B and C cannot be both in an active state and in a dormant state, we have

$$\begin{cases} B_d \cdot B_a = \bot \\ C_d \cdot C_a = \bot \end{cases}$$

The algebraic model of many Spare gates with many common spare events can be deduced from these models by considering the same approach. It will not be detailed here though.

4.5. General case of n Spare gates with m input events sharing $p \leq (m-1)$ spare events

In the general case, the algebraic model of n Spare gates with m input events sharing $p \leq (m-1)$ spare events is complex. However, a few information can be provided regarding its complexity.

The algebraic model of each Spare gate can be divided into two parts:

- A first part which describes the failure of the gate 'by itself', i.e. without taking into account the influence of the other Spare gates. In the case of two Spare gates sharing a spare event described in Section 4.3.2, this first part, respectively, is $C_a \cdot (A \lhd C_a) + A \cdot (C_d \lhd A)$ and $C_a \cdot (B \lhd C_a) + B \cdot (C_d \lhd B)$ for $Q1$ and $Q2$; it can be noted that this first part does not depend on the main components of the other gates.
- A second part which describes the influence of the other Spare gates on the Spare gate considered. In the case of two Spare gates sharing a spare event described in Section 4.3.2, this second part, respectively, is $A \cdot (B \lhd A)$ and $B \cdot (A \lhd B)$ for $Q1$ and $Q2$.

To illustrate this complexity, if we consider the case of n Spare gates with three input events sharing two spare events, the algebraic model of each Spare gate contains two parts:

- the first part contains 3! terms
- the second part contains:
  - $6(n-1)+(n-1)!$ terms if there are three Spare gates or more ($n \geq 3$)
  - $6(n-1)$ terms if there are less than three Spare gates ($n < 3$)

If we consider Fig. 10 which shows a benchmark from [25] containing four Spare gates with three input events sharing two spare events ($n = 4$), the algebraic model of each Spare gate includes 30 terms:

- six terms which correspond to the first part, and which represent the 3! possible failure sequences of the inputs of the Spare gate
- $24 = 6 \times (4-1) + (4-1)!$ terms which correspond to the second part, and hence to the influence of the three other gates. These 24 terms are divided as follows:

– six terms which correspond to the 6 order-3 sequences in which the spare events $VMS1$ and $VMS2$ fail first

– six terms which correspond to the 6 order-3 sequences in which the spare events $VMS1$ and $VMS2$ fail second

– six terms which correspond to the 6 order-3 sequences in which the spare events $VMS1$ and $VMS2$ fail last

– six terms which correspond to the 3! order-3 sequences in which the failures of the main components only are sufficient to engender the failure of the gate

4.6. Specific case of Cold and Hot Spare events

The algebraic models presented in Sections 4.3 and 4.4 are models of Spare gates in the general case of Warm Spare events. These algebraic models can be simplified in the specific cases of Cold and Hot Spare events:

- if a spare event S is a Cold Spare event, it cannot fail while in a dormant state, so S_d will never occur and any expression containing S_d in the algebraic models can be removed;
- if a spare event S is a Hot Spare event, it will have the same distribution function when in an active and in a dormant state, so $S_a \equiv S_d \equiv S$ and the algebraic models can be simplified.

Thanks to the algebraic model of all dynamic gates, we are now going to show how the structure function of any DFT can be determined and simplified to a canonical form.

5. Determination of the canonical form of the structure function of Dynamic Fault Trees

The algebraic models of dynamic gates presented in Section 4 allow to determine the structure function of any DFT. It can be interesting to manipulate this structure function to obtain a sum-of-product canonical form since such a canonical form of the structure function can be much useful to perform both the qualitative and the quantitative analysis of the DFT.

On the one hand, each product term of this canonical form is an algebraic expression which engenders the occurrence of the *TE* of the DFT and which holds for a given number of failure sequences. The cut sequences of each product term, and hence of the whole DFT, can hence be determined from this canonical form. On the other hand, the failure probability of the *TE* can be determined from this canonical form thanks to the standard inclusion–exclusion formula [21].

Given a DFT with n basic events $\{b_i, i \in (1, ..., n)\}$, the structure function for the *TE* becomes an expression containing at most the n basic events, and operators $+$, $\cdot$, $\lhd$, $\triangle$, and $\trianglelefteq$. In [13], we showed that the structure function of any DFT with gates PAND and FDEP could be developed and simplified, thanks to the theorems presented in Appendix A, to arrive to a standardized sum-of-product *canonical form* where each product term contains operator $\cdot$, and ordered pairs of variables linked by operator $\lhd$ only, as in (7):

$$TE = \sum \left(\prod b_i \cdot \prod (b_j \lhd b_k) \right), j \notin \{i, k\} \tag{7}$$

However, in this paper, we extend the work presented in [13] to Spare gates, and the algebraic model of Spare gates defined in Sections 4.3 to 4.6 involves the same temporal operator that is used to model gates PAND and FDEP, so the expression (7) still holds in the case of a DFT with Spare gates.

In this canonical form of the structure function, each product term $\prod b_i \cdot \prod (b_j \lhd b_k)$ is not a single cut sequence, but an algebraic expression providing a sufficient condition on the order of basic event failures that leads to the *TE* which may contain more than one cut sequence, and actually is a *cut sequence set (CSS)*.

If we suppose that there are n product terms in (7), the canonical form can be rewritten in the compact form

$$TE = \sum_{i=1}^{n} CSS_i \tag{8}$$

Nevertheless, a CSS may be included in one or more CSSs. CSS_i is included in one of the CSS_j if it satisfies the criterion [18]

$$CSS_i \cdot \sum_{j \neq i} CSS_j \; = \; CSS_i \tag{9}$$

If CSS_i is included in one of the CSS_j, it is redundant, and can be removed from the structure function (8). Iterative application of the criterion (9) removes all the redundant CSSs, and returns the minimal set $\mathbb{S}_{min}$ of non-redundant CSSs.

If $\mathbb{S}_{min}$ contains $(m \leq n)$ cut sequence sets, the minimal canonical form of the structure function can be expressed as

$$TE = \sum_{i=1}^{m} CSS_i, CSS_i \in \mathbb{S}_{min} \tag{10}$$

This minimal canonical form of the structure function can be exploited to perform the qualitative analysis – as it will be shown through an example in the next section – and the quantitative analysis of any DFT (even though this latter aspect will not be detailed in this paper).

6. Determination of the structure function of a DFT example

We propose to determine the structure function of a DFT example extracted from [2] which is depicted in Fig. 12.

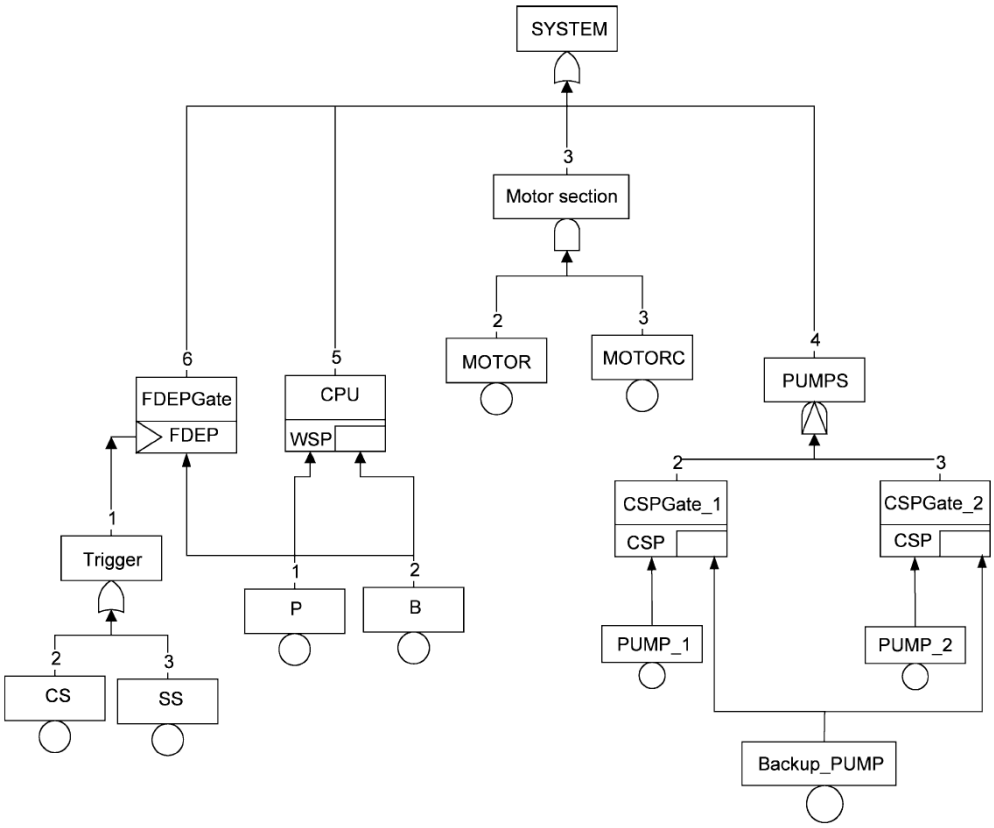

Figure 12: The HCAS Dynamic Fault Tree from [2].

This DFT models the failure of a cardiac assist system (HCAS) which is divided into four modules: Trigger, CPU unit, motor section, and pumps. The Trigger consists of a crossbar switch (CS) and a system supervisor (SS). The failure of either CS or SS triggers the failure of both CPUs. The CPU unit is a Warm Spare, which has a primary P and a spare unit B having a dormancy of 0.5. For the motor section to function, either MOTOR or MOTORC need to be working. The pumps unit is comprised of two Cold Spares, each having a primary pump (PUMP_1 and PUMP_2), and sharing a common spare pump (Backup_PUMP). In order for the pumps unit to fail, all three pumps need to fail and CSP_1 needs to fail before (or at the same time as) CSP_2, i.e. PAND gate.

The structure function of this DFT could be determined without any further simplification. However, in an educational purpose, we propose to divide this DFT into three subtrees whose structure functions will be successively determined. These three subtrees are as follows:

- subtree 1, which corresponds to the failure of the CPU unit: this subtree contains one OR gate, one FDEP gate, and one Warm Spare gate, and is hence dynamic;
- subtree 2, which corresponds to the failure of the motor section: this subtree contains a single AND gate and is hence static;
- subtree 3, which corresponds to the failure of the pumps unit: this subtree contains one PAND gate and two Cold Spare gates, and is hence dynamic.

If we denote by TE_1, TE_2, and TE_3 the top events of these three subtrees, the structure function of the DFT in Fig. 12 can be expressed as

$$TE = TE_1 + TE_2 + TE_3$$

The structure function of each one of these three subtrees can now be determined thanks to the algebraic model of dynamic gates presented in Section 4.

6.1. Determination of the structure function for TE_2

Subtree 2 is static since it contains a single AND gate. Its structure function can thus be determined directly as

$$TE_2 = MOTOR \cdot MOTORC$$

6.2. Determination of the structure function for TE_3

Subtree 3 is dynamic since it contains gates PAND and Cold Spare. The algebraic model of gate PAND presented

in Section 4.1 allows to express TE_3 as

$$TE_3 = CSP2 \cdot (CSP1 \trianglelefteq CSP2)$$

According to the algebraic model of two Spare gates sharing a spare event presented in Section 4.3.2,

$$\begin{cases} CSP1 = BP \cdot (P1 \triangleleft BP) + P1 \cdot (P2 \triangleleft P1) \\ CSP2 = BP \cdot (P2 \triangleleft BP) + P2 \cdot (P1 \triangleleft P2) \end{cases}$$

where BP denotes the active state BP_a of the spare pump since it cannot fail while in a dormant state, and $P1$ and $P2$ denote $PUMP\_1$ and $PUMP\_2$, respectively, for the sake of clarity.

The following result will be exploited to determine the structure function:

$$\begin{aligned} A \trianglelefteq ((A \cdot B) + C) &\overset{(A.20)}{=} (A \trianglelefteq (A \cdot B)) \cdot (A \trianglelefteq C) \\ &\overset{(A.21)}{=} ((A \trianglelefteq A) + (A \trianglelefteq B)) \\ &\quad \cdot (A \trianglelefteq C) \\ &\overset{(A.17)}{=} (A + (A \trianglelefteq B)) \cdot (A \trianglelefteq C) \\ &\overset{(A.28)}{=} A \cdot (A \trianglelefteq C) \\ &\overset{(A.30)}{=} A \trianglelefteq C \end{aligned} \tag{11}$$

$CSP1 \trianglelefteq CSP2$ can now be expressed as

$$\begin{aligned} CSP1 \trianglelefteq CSP2 &= [BP \cdot (P1 \triangleleft BP) + P1 \cdot (P2 \triangleleft P1)] \\ &\quad \trianglelefteq CSP2 \\ &\overset{(A.24)}{=} (BP \cdot (P1 \triangleleft BP)) \trianglelefteq CSP2 \\ &\quad + (P1 \cdot (P2 \triangleleft P1)) \trianglelefteq CSP2 \\ &\overset{(A.25)}{=} (BP \trianglelefteq CSP2) \cdot ((P1 \triangleleft BP) \trianglelefteq CSP2) \\ &\quad + (P1 \trianglelefteq CSP2) \cdot ((P2 \triangleleft P1) \trianglelefteq CSP2) \\ &\overset{(A.26)}{=} (BP \trianglelefteq CSP2) \cdot (P1 \triangleleft BP) \cdot (P1 \trianglelefteq CSP2) \\ &\quad + (P1 \trianglelefteq CSP2) \cdot (P2 \triangleleft P1) \cdot (P2 \trianglelefteq CSP2) \\ &= (P1 \triangleleft BP) \cdot (BP \trianglelefteq CSP2) \cdot (P1 \trianglelefteq CSP2) \\ &\quad + (P2 \triangleleft P1) \cdot (P1 \trianglelefteq CSP2) \cdot (P2 \trianglelefteq CSP2) \\ &\overset{(1),(6)}{=} (P1 \trianglelefteq BP) \cdot (BP \trianglelefteq CSP2) \cdot (P1 \trianglelefteq CSP2) \\ &\quad + (P2 \trianglelefteq P1) \cdot (P1 \trianglelefteq CSP2) \cdot (P2 \trianglelefteq CSP2) \\ &\overset{(A.31)}{=} (P1 \trianglelefteq BP) \cdot (BP \trianglelefteq CSP2) \\ &\quad + (P2 \trianglelefteq P1) \cdot (P1 \trianglelefteq CSP2) \\ &\overset{(1),(6)}{=} (P1 \triangleleft BP) \cdot (BP \trianglelefteq CSP2) \\ &\quad + (P2 \triangleleft P1) \cdot (P1 \trianglelefteq CSP2) \end{aligned}$$

On the one hand,

$$\begin{aligned} BP \trianglelefteq CSP2 &= BP \trianglelefteq [BP \cdot (P2 \triangleleft BP) + P2 \cdot (P1 \triangleleft P2)] \\ &\overset{(11)}{=} BP \trianglelefteq [P2 \cdot (P1 \triangleleft P2)] \\ &\overset{(A.21)}{=} (BP \trianglelefteq P2) + (BP \trianglelefteq (P1 \triangleleft P2)) \\ &\overset{(6),(A.22)}{=} (BP \trianglelefteq P2) + (BP \triangleleft P1) + BP \cdot P1 \cdot (P2 \trianglelefteq P1) \\ &\overset{(1),(6)}{=} (BP \triangleleft P2) + (BP \triangleleft P1) + BP \cdot P1 \cdot (P2 \triangleleft P1) \end{aligned}$$

On the other hand,

$$\begin{aligned} P1 \trianglelefteq CSP2 &= P1 \trianglelefteq [BP \cdot (P2 \triangleleft BP) + P2 \cdot (P1 \triangleleft P2)] \\ &\overset{(A.14)}{=} P1 \trianglelefteq [BP \cdot (P2 \triangleleft BP) + P1 \cdot P2 \cdot (P1 \triangleleft P2)] \\ &\overset{(11)}{=} P1 \trianglelefteq [BP \cdot (P2 \triangleleft BP)] \\ &\overset{(A.21)}{=} (P1 \trianglelefteq BP) + (P1 \trianglelefteq (P2 \triangleleft BP)) \\ &\overset{(6),(A.22)}{=} (P1 \trianglelefteq BP) + (P1 \triangleleft P2) + P1 \cdot P2 \cdot (BP \trianglelefteq P2) \\ &\overset{(1),(6)}{=} (P1 \triangleleft BP) + (P1 \triangleleft P2) + P1 \cdot P2 \cdot (BP \triangleleft P2) \end{aligned}$$

Consequently,

$$\begin{aligned} CSP1 \trianglelefteq CSP2 &= (P1 \triangleleft BP) \cdot [(BP \triangleleft P2) + (BP \triangleleft P1) \\ &\quad + BP \cdot P1 \cdot (P2 \triangleleft P1)] + (P2 \triangleleft P1) \cdot [(P1 \triangleleft BP) \\ &\quad + (P1 \triangleleft P2) + P1 \cdot P2 \cdot (BP \triangleleft P2)] \\ &\overset{(A.15)}{=} (P1 \triangleleft BP) \cdot [(BP \triangleleft P2) + BP \cdot P1 \cdot (P2 \triangleleft P1)] \\ &\quad + (P2 \triangleleft P1) \cdot [(P1 \triangleleft BP) + P1 \cdot P2 \cdot (BP \triangleleft P2)] \\ &= (P1 \triangleleft BP) \cdot (BP \triangleleft P2) \\ &\quad + BP \cdot P1 \cdot (P1 \triangleleft BP) \cdot (P2 \triangleleft P1) \\ &\quad + (P2 \triangleleft P1) \cdot (P1 \triangleleft BP) \\ &\quad + P1 \cdot P2 \cdot (P2 \triangleleft P1) \cdot (BP \triangleleft P2) \\ &\overset{(4),(A.14)}{=} (P1 \triangleleft BP) \cdot (BP \triangleleft P2) + (P2 \triangleleft P1) \\ &\quad \cdot (P1 \triangleleft BP) + P1 \cdot (BP \triangleleft P2) \cdot (P2 \triangleleft P1) \end{aligned}$$

Since $BP \equiv BP_a$ cannot fail before $P1$ and $P2$ as it can only fail in its active mode,

$$\begin{aligned} CSP1 \trianglelefteq CSP2 &= (P1 \triangleleft BP) \cdot (BP \triangleleft P2) \\ &\quad + (P2 \triangleleft P1) \cdot (P1 \triangleleft BP) \end{aligned}$$

Finally,

$$\begin{aligned} TE_3 &= [BP \cdot (P2 \triangleleft BP) + P2 \cdot (P1 \triangleleft P2)] \\ &\quad \cdot [(P1 \triangleleft BP) \cdot (BP \triangleleft P2) + (P2 \triangleleft P1) \cdot (P1 \triangleleft BP)] \\ &\overset{(A.15)}{=} P2 \cdot (P1 \triangleleft P2) \cdot (P1 \triangleleft BP) \cdot (BP \triangleleft P2) \\ &\quad + BP \cdot (P2 \triangleleft BP) \cdot (P2 \triangleleft P1) \cdot (P1 \triangleleft BP) \\ &\overset{(A.16)}{=} P2 \cdot (P1 \triangleleft BP) \cdot (BP \triangleleft P2) \\ &\quad + BP \cdot (P2 \triangleleft P1) \cdot (P1 \triangleleft BP) \end{aligned}$$

6.3. Determination of the structure function for TE_1

Subtree 1 is dynamic since it contains gates FDEP and Warm Spare. The model of the Spare gate presented in Section 4.3.1 is valid when the input events of the Spare gate are independent basic events which can consequently not occur simultaneously. However, in the case of subtree 1, basic events P and B are basic events which have a common cause failure represented by the Trigger, and they can hence occur simultaneously when the trigger occurs. Nevertheless, this particular aspect can be taken into account in our model by introducing an additional term related to the simultaneous occurrence of P_T and B_T – $P_T \triangle B_T$ – in the algebraic model of the Spare gate. TE_1 can hence first be expressed as

$$TE_1 = B_{a_T} \cdot (P_T \triangleleft B_{a_T}) + P_T \cdot (B_{d_T} \triangleleft P_T) + P_T \triangle B_T$$

where B_a and B_d denote the active and dormant state of the spare unit B, according to the algebraic model of the Warm Spare gate presented in Section 4.3.1. As explained in Section 4.2, the substituted variables B_{a_T}, B_{d_T}, B_T, and P_T explicitly indicate the effect of trigger T and denote the global failure of basic events B and P. Thus we have:

$$\begin{cases} B_T = B + T \\ B_{a_T} = B_a + T \\ B_{d_T} = B_d + T \\ P_T = P + T \end{cases}$$

The additional term $P_T \triangle B_T$ can first be determined since $T = CS + SS$:

$$\begin{aligned}P_T \triangle B_T &= (P + CS + SS) \triangle (B + CS + SS)\\ &= ((CS + SS) \lhd B) \cdot ((CS + SS) \lhd P)\\ &= ((CS + SS) \lhd B_d) \cdot ((CS + SS) \lhd P)\\ &+ ((CS + SS) \lhd P) \cdot ((CS + SS) \lhd B_a)\end{aligned}$$

Consequently,

$$\begin{aligned}TE_1 &= (B_a + CS + SS) \cdot ((P + CS + SS) \lhd (B_a + CS + SS))\\ &+ (P + CS + SS) \cdot ((B_d + CS + SS) \lhd (P + CS + SS))\\ &+ ((CS + SS) \lhd B_d) \cdot ((CS + SS) \lhd P)\\ &+ ((CS + SS) \lhd P) \cdot ((CS + SS) \lhd B_a)\end{aligned}$$

Finally, this structure function can be developed thanks to the use of theorems (A.4) and (A.8), and simplified to the following form:

$$\begin{aligned}TE_1 = \; & (P \lhd B_a) \cdot (B_a \lhd (CS + SS))\\ &+ (P \lhd (CS + SS)) \cdot ((CS + SS) \lhd B_a)\\ &+ (B_d \lhd P) \cdot (P \lhd (CS + SS))\\ &+ (B_d \lhd (CS + SS)) \cdot ((CS + SS) \lhd P)\\ &+ ((CS + SS) \lhd B_d) \cdot ((CS + SS) \lhd P)\\ &+ ((CS + SS) \lhd P) \cdot ((CS + SS) \lhd B_a)\end{aligned}$$

Some of the terms of this structure function can be grouped to obtain the final simplified following form:

$$TE_1 = CS + SS + P \cdot (B_d \lhd P) + B_a \cdot (P \lhd B_a)$$

6.4. Determination of the structure function of the whole DFT in Fig. 12

The structure function of the DFT in Fig. 12 can finally be determined as

$$\begin{aligned}TE = \; & CS + SS + MOTOR \cdot MOTORC\\ &+ P \cdot (B_d \lhd P) + B_a \cdot (P \lhd B_a)\\ &+ BP \cdot (P2 \lhd P1) \cdot (P1 \lhd BP)\\ &+ P2 \cdot (P1 \lhd BP) \cdot (BP \lhd P2)\end{aligned}$$

This structure function can be used to perform the qualitative analysis of the DFT in Fig. 12 directly, as explained in Section 7.

7. Qualitative analysis of DFTs based on the structure function

The canonical form of the structure function of the DFT in Fig. 12, which was determined previously, is a sum-of-product form whose each product term can provide *minimal cut sets* or *minimal cut sequences*. Two cases may happen:

- if a product term does not contain the temporal operator BF ($\lhd$), it is static and provides *minimal cut sets* for the DFT;
- if a product term contains the temporal operator BF ($\lhd$), it is dynamic and provides *minimal cut sequences* for the DFT. In some cases, a set of minimal cut sequences may represent all the possible sequences which correspond to a minimal cut set and can hence be reduced to this minimal cut set.

The structure function of the DFT in Fig. 12 contains seven terms. On the one hand, three terms do not contain the temporal operator BF ($\lhd$). They are static and can hence provide three minimal cut sets for the DFT:

$$CS, SS, (MOTOR \cdot MOTORC)$$

On the other hand, four terms contain the temporal operator BF ($\lhd$). They are dynamic and can hence provide the minimal cut sequences of the DFT:

$$[B_d, P], [P, B_a], [P2, P1, BP], [P1, BP, P2]$$

The minimal cut sets and sequences of the DFT in Fig. 12 can then be determined as

$$\begin{gathered}CS, SS, (MOTOR \cdot MOTORC),\\ [B_d, P], [P, B_a], [P2, P1, BP], [P1, BP, P2]\end{gathered}$$

In this case, it can be noted that the two minimal cut sequences $[B_d, P]$ and $[P, B_a]$ are logically equivalent to the single minimal cut set $P \cdot B$. However, the minimal cut set does not render the two states of the basic event B which will be needed to perform the quantitative analysis of the DFT. This is the reason why these two minimal cut sequences were not reduced to the equivalent minimal cut set $B \cdot P$.

This canonical form of the structure function thus provides a hybrid result for the qualitative analysis of DFTs by allowing to determine both minimal cut sets and minimal cut sequences. Furthermore, as it was shown above, a set of minimal cut sequences may sometimes be equivalent to a single minimal cut set. Two cases may happen:

- if these minimal cut sequences contain spare events, they must not be reduced to their equivalent minimal cut set since the knowledge of the state in which spare events fail will be needed to perform the quantitative analysis of the DFT;
- if these minimal cut sequences do not contain spare events, they can be reduced to their equivalent minimal cut set. Indeed, even though both results are equivalent, minimal cut sets represent a more concise – and hence more useful – result to the practitioner than the corresponding set of minimal cut sequences.

8. Conclusion

In this paper, we presented an algebraic framework allowing to determine the structure function of any DFT, as it is commonly the case for SFTs. Furthermore, we

showed that this structure function can be simplified to a canonical form for any DFT. Starting from this canonical form of the structure function, the qualitative analysis of the DFT can be performed directly by determining both the minimal cut sets and sequences of the DFT.

Regarding the quantitative analysis of DFTs, which has not been developed in this paper, it can also be performed from the structure function of DFTs thanks to appropriate probabilistic models of all dynamic gates (PAND, FDEP, and Spare). We have already determined such probabilistic models, which do not depend on the failure distribution considered for basic events, but they could not be presented in this paper.

Even though cut sequences can be extracted quite easily from the structure function, ongoing work is currently addressed to the systematic determination of the *minimal set* of *minimal cut sequences*. Besides, the work presented in this paper allowed to propose a formal background for the determination of the structure function of DFTs, and future work will be dedicated to the elaboration of efficient algorithms allowing to automatically perform the calculation of this structure function and the qualitative analysis of DFTs.

Appendix A. Development and simplification theorems

The temporal operators non-inclusive BEFORE and Inclusive BEFORE introduced in Section 3.4 satisfy the following theorems (their proofs can be found in [12]), for any non-repairable events a, b, and c. These theorems will allow to calculate and simplify the structure function of DFTs.

Appendix A.1. Theorems satisfied by operator non-inclusive BEFORE

Operator non-inclusive BEFORE satisfies the following theorems, for all $a, b, c \in \mathcal{E}_{nr}$:

$$a \lhd a = \bot \tag{A.1}$$

$$\bot \lhd a = \bot \tag{A.2}$$

$$a \lhd \bot = a \tag{A.3}$$

$$a \lhd (b + c) = (a \lhd b) \cdot (a \lhd c) \tag{A.4}$$

$$a \lhd (b \cdot c) = (a \lhd b) + (a \lhd c) \tag{A.5}$$

$$a \lhd (b \lhd c) = (a \lhd b) + (a \cdot b \cdot ((c \lhd b) + (c \triangle b))) \tag{A.6}$$

$$a \lhd (b \trianglelefteq c) = (a \lhd b) + (a \cdot b \cdot (c \lhd b)) \tag{A.7}$$

$$(a + b) \lhd c = (a \lhd c) + (b \lhd c) \tag{A.8}$$

$$(a \cdot b) \lhd c = (a \lhd c) \cdot (b \lhd c) \tag{A.9}$$

$$(a \lhd b) \lhd c = (a \lhd b) \cdot (a \lhd c) \tag{A.10}$$

$$(a \trianglelefteq b) \lhd c = (a \trianglelefteq b) \cdot (a \lhd c) \tag{A.11}$$

$$a + (a \lhd b) = a \tag{A.12}$$

$$(a \lhd b) + b = a + b \tag{A.13}$$

$$a \cdot (a \lhd b) = a \lhd b \tag{A.14}$$

$$(a \lhd b) \cdot (b \lhd a) = \bot \tag{A.15}$$

$$(a \lhd b) \cdot (b \lhd c) \cdot (a \lhd c) = (a \lhd b) \cdot (b \lhd c) \tag{A.16}$$

Appendix A.2. Theorems satisfied by operator Inclusive BEFORE

Operator Inclusive BEFORE satisfies the following theorems, for all $a, b, c \in \mathcal{E}_{nr}$:

$$a \trianglelefteq a = a \tag{A.17}$$

$$\bot \trianglelefteq a = \bot \tag{A.18}$$

$$a \trianglelefteq \bot = a \tag{A.19}$$

$$a \trianglelefteq (b + c) = (a \trianglelefteq b) \cdot (a \trianglelefteq c) \tag{A.20}$$

$$a \trianglelefteq (b \cdot c) = (a \trianglelefteq b) + (a \trianglelefteq c) \tag{A.21}$$

$$a \trianglelefteq (b \lhd c) = (a \lhd b) + (a \cdot b \cdot (c \trianglelefteq b)) + (a \triangle b) \cdot (b \lhd c) \tag{A.22}$$

$$a \trianglelefteq (b \trianglelefteq c) = (a \lhd b) + (a \cdot b \cdot (c \lhd b)) + (a \triangle b) \cdot (b \trianglelefteq c) \tag{A.23}$$

$$(a + b) \trianglelefteq c = (a \trianglelefteq c) + (b \trianglelefteq c) \tag{A.24}$$

$$(a \cdot b) \trianglelefteq c = (a \trianglelefteq c) \cdot (b \trianglelefteq c) \tag{A.25}$$

$$(a \lhd b) \trianglelefteq c = (a \lhd b) \cdot (a \trianglelefteq c) \tag{A.26}$$

$$(a \trianglelefteq b) \trianglelefteq c = (a \trianglelefteq b) \cdot (a \trianglelefteq c) \tag{A.27}$$

$$a + (a \trianglelefteq b) = a \tag{A.28}$$

$$b + (a \trianglelefteq b) = a + b \tag{A.29}$$

$$a \cdot (a \trianglelefteq b) = a \trianglelefteq b \tag{A.30}$$

$$(a \trianglelefteq b) \cdot (b \trianglelefteq c) \cdot (a \trianglelefteq c) = (a \trianglelefteq b) \cdot (b \trianglelefteq c) \tag{A.31}$$

Appendix A.3. Simplification theorems

Temporal operators satisfy the following theorems, for all $a, b, c \in \mathcal{E}_{nr}$:

$$(a \trianglelefteq b) + (a \lhd b) = a \trianglelefteq b \tag{A.32}$$

$$(a \lhd b) \cdot (a \triangle b) = \bot \tag{A.33}$$

$$(a \trianglelefteq b) \cdot (a \lhd b) = a \lhd b \tag{A.34}$$

$$(a \lhd b) \cdot (b \trianglelefteq a) = \bot \tag{A.35}$$

$$(a \trianglelefteq b) \cdot (a \triangle b) = a \triangle b \tag{A.36}$$

$$(a \trianglelefteq b) \cdot (b \trianglelefteq a) = a \triangle b \tag{A.37}$$

$$(a \lhd b) + (a \triangle b) + (b \lhd a) = a + b \tag{A.38}$$

$$(a \cdot (b \lhd a)) + (a \triangle b) + (b \cdot (a \lhd b)) = a \cdot b \tag{A.39}$$

$$(a \lhd b) + (a \triangle b) + (a \cdot (b \lhd a)) = a \tag{A.40}$$

$$(a \trianglelefteq b) + (b \trianglelefteq a) = a + b \tag{A.41}$$

$$(a \cdot (b \trianglelefteq a)) + (b \cdot (a \trianglelefteq b)) = a \cdot b \tag{A.42}$$

$$(a \trianglelefteq b) + (a \cdot (b \trianglelefteq a)) = a \tag{A.43}$$

$$(a \lhd b) \cdot (b \lhd c) \cdot (a \trianglelefteq c) = (a \lhd b) \cdot (b \lhd c) \tag{A.44}$$

References

[1] Bobbio A, Codetta Raiteri D. Parametric fault trees with dynamic gates and repair boxes. In: Proceedings of the annual reliability and maintainability symposium (RAMS 2004), Los Angeles, CA, USA, 2004. p. 459-65.

[2] Boudali H, Dugan JB. A discrete-time Bayesian network reliability modeling and analysis framework. Reliability Engineering and System Safety 2005;87(3):337-49.

[3] Boudali H, Dugan JB. A continuous-time Bayesian network reliability modeling, and analysis framework. IEEE Transactions on Reliability 2006;55(1):86-97.

[4] Boudali H, Crouzen P, Stoelinga M. Dynamic fault tree analysis through input/output interactive Markov chains. In: Proceedings of the international conference on dependable systems and networks (DSN 2007). IEEE Computer Society; 2007. p. 25-38.

[5] Boudali H, Crouzen P, Stoelinga M. A compositional semantics for dynamic fault tree in terms of interactive Markov chains. In: Proceedings of the international symposium on automated technology for verification and analysis (ATVA'07). Lecture notes in computer science, vol. 4762. Springer Verlag; 2007. p. 441-56.

[6] Boudali H, Crouzen P, Stoelinga M. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Transactions on Dependable and Secure Computing 2010;7(2):128-43.

[7] Codetta Raiteri D. The conversion of dynamic fault trees to stochastic petri nets, as a case of graph transformation. Electronic Notes on Theoretical Computer Science 2005;127(2):45-60.

[8] Dugan JB, Bavuso SJ, Boyd MA. Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability 1992;41(3):363-77.

[9] Dugan JB, Sullivan KJ, Coppit D. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. IEEE Transactions on Reliability 2000;49(1):49-59.

[10] Lauritzen SL, Jensen F. Stable local computation with conditional Gaussian distributions. Statistics and Computing 2001;11(2):191-203.

[11] Malhotra M, Trivedi K. Power-hierarchy among dependability model types. IEEE Transactions on Reliability 1994;43(3):493-502.

[12] Merle G, Roussel JM, Lesage JJ. Algebraic framework for the modelling of priority dynamic fault trees. Internal report; 2008. Available: <http://www.lurpa.ens-cachan.fr/isa/aadft/documents/LURPA-2008-Framework.pdf>.

[13] Merle G, Roussel JM, Lesage JJ, Bobbio A. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. IEEE Transactions on Reliability 2010;59(1):250-61.

[14] Merle G. Algebraic modelling of dynamic fault trees, contribution to qualitative and quantitative analysis. PhD thesis, École Normale Supérieure de Cachan, France, 2010.

[15] Montani S, Portinale L, Bobbio A, Varesio M, Codetta-Raiteri D. DBNet, a tool to convert dynamic fault trees into dynamic Bayesian networks. Università del Piemonte Orientale, Technical Report TR-INF-2005-08-02-UNIPMN; 2005.

[16] Moral S, Rumí R, Salmerón A. Mixtures of truncated exponentials in hybrid Bayesian networks. In: Proceedings of the 6th European conference on symbolic and quantitative approaches to reasoning with uncertainty. Lecture notes in artificial intelligence, vol. 2143; 2001. p. 145-67.

[17] Ortmeier F, Schellhorn G, Thums A, Reif W, Hering B, Trappschuh H. Safety analysis of the height control system for the Elbtunnel. Reliability Engineering and System Safety 2003;81(3):259-68.

[18] Rauzy A. Mathematical foundations of minimal cutsets. IEEE Transactions on Reliability 2001;50(4):389-96.

[19] Reay KA, Andrews JD. A fault tree analysis strategy using binary decision diagrams. Reliability Engineering and System Safety 2002;78(1):45-56.

[20] Stamatelatos M, Vesely W. Fault tree handbook with aerospace applications. NASA Office of Safety and Mission Assurance, vol. 1.1, 2002. p. 1-205.

[21] Trivedi K. Probability & statistics with reliability, queueing & computer science applications. 2nd ed.. Wiley; 2001.

[22] Vesely WE, Goldberg FF, Roberts NH, Haasl DF. Fault tree handbook. Washington, DC, USA: US Nuclear Regulatory Commission; 1981.

[23] Walker M, Papadopoulos Y. Qualitative temporal analysis: towards a full implementation of the fault tree handbook. Control Engineering Practice 2009;17(10):1115-25.

[24] Yuge T, Yanagi S. Quantitative analysis of a fault tree with priority AND gates. Reliability Engineering and System Safety 2008;93(11):1577-83.

[25] Zhu H, Zhou S, Dugan JB, Sullivan KJ. A benchmark for quantitative fault tree reliability analysis. In: Proceedings of the annual reliability and maintainability symposium 2001 (RAMS 2001), Philadelphia, PA, USA, 2001. p. 86-93.